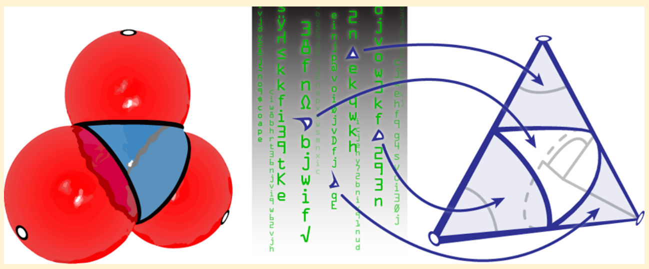
Journal of Chemical Theory and Computation
Article
pubs.acs.org/JCTC
Terms of Use

TRIFORCE: Tessellated Semianalytical Solvent Exposed Surface Areas and Derivatives

Nils J. D. Drechsel,*[†,‡,§] Christopher J. Fennell,[§] Ken A. Dill,[∥] and Jordi Villà-Freixa*[†,⊥]

[†]Computational Biochemistry and Biophysics Laboratory, Research Unit on Biomedical Informatics, Universitat Pompeu Fabra, C/Doctor Aiguader, 88, 08003 Barcelona, Catalunya, Spain
[‡]Laufer Center for Physical and Quantitative Biology, Stony Brook University, Stony Brook, New York 11794-5252, United States
[§]Department of Chemistry, Oklahoma State University, Stillwater, Oklahoma 74078, United States
[∥]Laufer Center for Physical and Quantitative Biology and Departments of Physics and Chemistry, Stony Brook University, Stony Brook, New York 11794-5252, United States
[⊥]Escola Politècnica Superior, Universitat de Vic—Universitat Central de Catalunya, C/de la Laura, 13, 08500 Vic, Catalunya, Spain



ABSTRACT: We present a new approach to the calculation of solvent-accessible surface areas of molecules with potential application to surface area based methods for determination of solvation free energies. As in traditional analytical and statistical approaches, this new algorithm, called TRIFORCE, reports both component areas and derivatives as a function of the atomic coordinates and radii. Unique to TRIFORCE are the rapid and scalable approaches for the determination of sphere intersection points and numerical estimation of the surface areas, derivatives, and other properties that can be associated with the surface area facets. The algorithm performs a special tessellation and semianalytical integration that uses a precomputed look-up table. This provides a simple way to balance numerical accuracy and memory usage. TRIFORCE calculates derivatives in the same manner, enabling application in force-dependent activities such as molecular geometry minimization. TRIFORCE is available free of charge for academic purposes as both a C++ library, which can be directly interfaced to existing molecular simulation packages, and a web-accessible application.

1. INTRODUCTION

Molecular simulations often involve balancing between complexity and computational tractability. One common approximation that makes macromolecular simulations, such as protein folding and association, tractable is to use implicit rather than explicit solvent. Implicit solvents trade the detailed accounting of water, lipid, or other surrounding solvent molecules for an averaged effective solvent representation. To do this, the many explicit degrees of solvent freedom are integrated out of the system and replaced with free energies of solvation. Free energies of solvation have been observed to be well-correlated with the interfacial or solvent-accessible surface area (SASA) in the case of nonpolar solutes.[1−4] Many methods for implicit solvation have been developed which take advantage of this observation to provide quantitative predictions for solvation free energies using the SASA,[5−14] though it should be noted that surface area based approaches are not the only avenue for estimating such quantities.[15−17] To perform simulations with implicit solvents that depend upon the SASA, we need both per-atom SASAs and their derivatives. Several algorithms and methods have been developed which can compute these quantities spanning from analytically exact approaches[18−23] to statistical or numerically approximate approaches.[24−27]

We are interested in performing molecular simulations in implicit solvents that report solvation free energies with a high degree of accuracy. Much of our motivation for the work presented within comes as a result of our experiences applying numerical surface area methods to these problems and encountering performance limitations with subsequent numerical derivatives that make them, while accurate, undesirable in general use. Analytical approaches available are also often performance limiting, and potentially faster statistical approaches are often marked by substantial nonuniform errors. Additionally, many methods are physically limited in some way, such as the inability to treat hydrogen atom contributions to the SASA. Thus, despite the large number of surface area algorithms in the scientific literature, we found there to be a surprising lack of readily accessible, working, and robust approaches that provide both accurate surface areas and accurate derivatives.

Here, we present an alternative semianalytical approach for computing Lee−Richards SASAs and their derivatives from coarse-grained or all-atom representations of molecular systems.[28] This method, which we refer to as TRIFORCE,

Received: April 2, 2014
Published: July 23, 2014

employs a precomputed look-up table that enables accelerated determination of component areas and derivatives as a function of the principal angles defining boundary interfaces. We refer to this as a semianalytical method because, while the process for determining these principal angles is analytical, we perform a numerical interpolation look-up on a discretized grid of surface areas and their derivatives as a function of these angles. We evaluate the correctness of this approach with comparisons to analytical methods and show that the numerical accuracy is determined by the density of this look-up table, with the primary cost being one of memory size and memory access rather than compute cycles. Our intent is for the TRIFORCE method to fill the apparent gap between the theoretical landscape of surface area algorithms and the realm of practical usage.

2. AREA CALCULATION

In classical molecular simulations, molecules are modeled as a set of intersecting spheres S_l with radii corresponding to their extended van der Waals (vdW) surface. The radii for this extended vdW surface often include a solvent probe radius term (traditionally 1.4 Å) to best align this extended surface with the average distance to the surrounding solvent. This extended surface is also called the Lee–Richards solvent-accessible surface.[28] The solvent-accessible surface area A of a molecule is then calculated by adding up all solvent exposed areas of the set of spheres $A^{(l)}$ using

$$A = \sum_l A^{(l)} \tag{1}$$

$A^{(l)}$ is a complex shape that we represent as a sum of simpler component areas. These component areas are triangular patches which are constructed by a special tessellation of the exposed area of a sphere, as depicted in Figure 1. Here, the exposed area of central sphere S_0 is composed of six triangular patches, each having two sides which are great circle segments and a third side which is a spherical arc.

The boundary of the exposed area, which we will refer to as the exposed boundary, is composed of these spherical arcs, which are segments of circular interfaces I between spheres S that intersect with S_l. For example, circular interface I_j is located in the plane of intersection between S_l and S_j where the hulls touch each other. An intersection point $\boldsymbol{p}$ exists between circular interfaces whenever two interfaces intersect on the exposed boundary. Two consecutive intersection points $\boldsymbol{p}_{ij}$ and $\boldsymbol{p}_{jk}$ on an exposed boundary, in conjunction with a tessellation point $\dot{\chi}$, form the three points of a triangular patch $A_{ijk}^{(l)}$. $\dot{\chi}$ is the point on S_l where the tessellation axis $\boldsymbol{\chi}$ intersects. $\boldsymbol{\chi}$ is an arbitrary vector for each atom fixed for the duration of the calculations.

Areas $A_{ijk}^{(l)}$ are calculated on a unit sphere and therefore have to be multiplied with the squared radius r_l of sphere S_l:

$$\hat{A}^{(l)} = \sum_{(ijk)\in \mathrm{Seg}_l} A_{ijk}^{(l)} r_l^2 \tag{2}$$

which results in area $\hat{A}^{(l)}$. Here, Seg_l denotes the set of all interface segments of the exposed boundaries on S_l.

Patches $A_{ijk}^{(l)}$ can be either fully or fractionally part of the SASA, or the solvent excluded surface area (SESA), which is complement to the SASA, i.e., the area that is buried by neighbor spheres. Figure 2 shows that, over the course of the algorithm, triangular patches contribute positively if they are fully or fractionally part of the SASA (blue) or contribute negatively if they are fully part of the SESA (red).

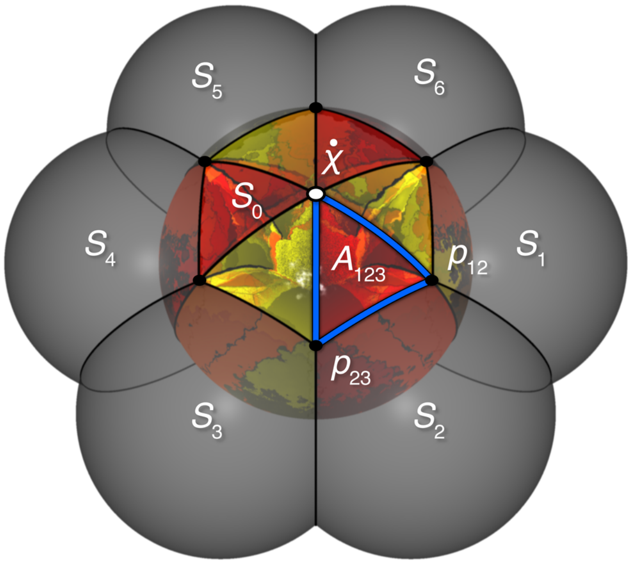


Figure 1. Tessellation of the exposed area of sphere S_0 by component triangles. The corners of the triangles are two consecutive intersection points, denoted by $\boldsymbol{p}$, and the tessellation point, $\dot{\chi}$. $\boldsymbol{p}$ are intersections of circular interfaces I on the surface of S_0.

In addition, the interface segments of a patch $A_{ijk}^{(l)}$ can be either part of a convex or a concave interface. Figure 3 illustrates the distinction between these two types of interfaces. When a sphere with a smaller radius moves toward and begins to intersect a sphere with a larger radius, the sphere–sphere intersection which results is a convex interface. Once half the small sphere is occluded, this interface transitions to a concave interface. Concave interfaces can be seen as inverted convex interfaces, and an area computed using a concave interface will have opposite sign. Once patches $A_{ijk}^{(l)}$ are summed up to form $\hat{A}^{(l)}$, this area will either be positive or negative, depending on whether SASA or SESA has been counted, respectively. If the area is a SESA, we compute the complement to convert it into a SASA. This is done by adding the surface area of a sphere with radius r_l to this negative $\hat{A}^{(l)}$:

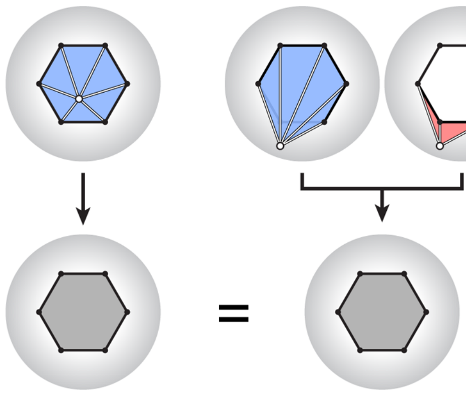


Figure 2. Illustration of the tessellation of an exposed boundary when the tessellation-point is (left) inside or (right) outside the exposed boundary. When the tessellation point is outside the boundary, we compute both positive (blue) and negative (red) areas, the sum of which is equal to the result when the intersection point is inside the boundary.

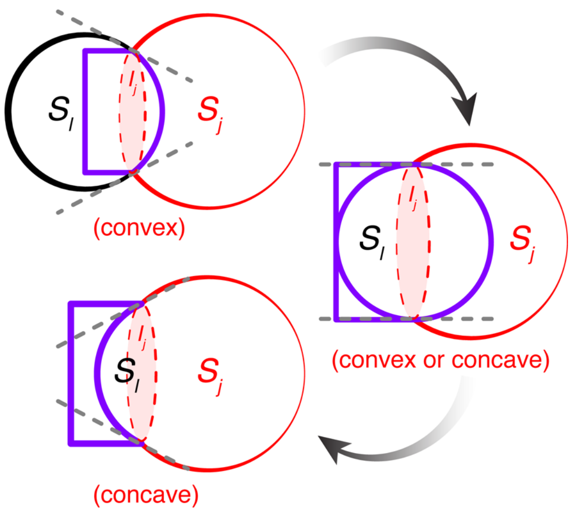


Figure 3. Illustration showing how an interface between two spheres transitions from convex to concave with increasing sphere overlap.

$$A^{(l)} = \begin{cases} \hat{A}^{(l)} & \hat{A}^{(l)} \geq 0 \\ 4\pi r_l^2 + \hat{A}^{(l)} & \text{else} \end{cases} \tag{3}$$

Figure 4 shows the principal angles Φ, ψ, and λ used in determining the areas of a triangular patch. Each triangular patch $A_{ijk}^{(l)}$ is assembled from two subpatches split by the so-called tessellation plane, i.e., the plane orthogonal to the cross product of χ and the intersphere vector. Here, Φ is the angle between the tessellation plane and a vector from the center of I_j to a $\boldsymbol{p}$. The angle ψ_j is between the tessellation axis and the center of I_j. Finally, λ_j is the opening angle of the cone constructed from I_j and the center of sphere S_l.

The area of each subpatch is looked-up and interpolated from a precomputed integration table $T(\Phi,\psi,\lambda)$. Φ can either be positive or negative depending on whether its corresponding intersection point is on the "right" or "left" side of the tessellation plane, which results in either a positive or negative area, respectively. The two subpatches are subtracted from each other to result in the full area of patch $A_{ijk}^{(l)}$.

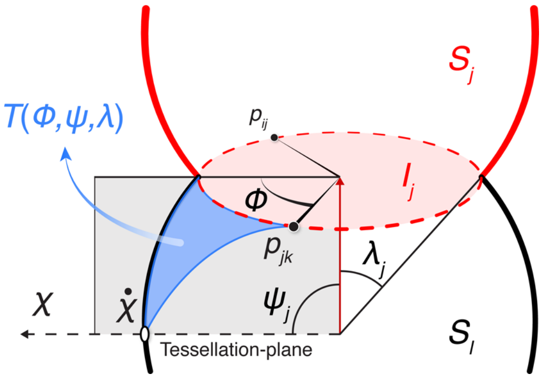


Figure 4. Triangular patch formed by points $\boldsymbol{p}_{ij}$, $\boldsymbol{p}_{jk}$, and $\dot{\chi}$, formed from two subpatches, defined by principal angles Φ, ψ, and λ. The area of the subpatch is acquired from a simple look-up table $T(\Phi,\psi,\lambda)$ dependent on these principal angles.

From now on, whenever unambiguous, we omit the index (l) for simplicity. Throughout the work we will use the symbol · to denote the dot product, symbol ⊗ to denote the outer vector product, and symbol × to denote the cross-product. Whenever the cross-product is used between a matrix and a vector, a columnwise cross-product is assumed.

Figure 5 highlights the key angles and vectors that factor into the calculation of the surface area by illustrating both concave

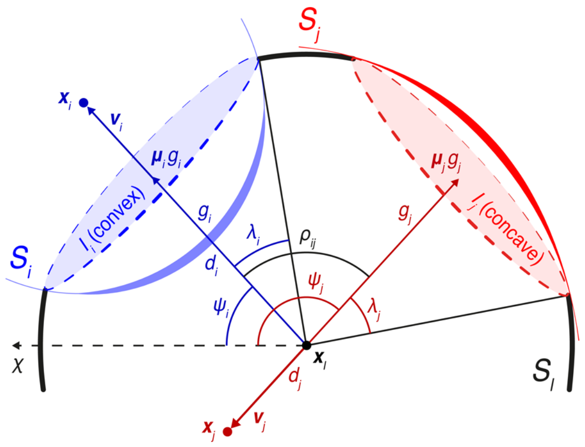


Figure 5. Sphere S_l intersects with two neighboring spheres S_i and S_j. The intersections result in one concave (red) and one convex (blue) circular interface, respectively, to which normalized vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ point, with ρ as the angle between these normalized vectors. Vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ connect S_l to the respective sphere centers $\mathbf{x}_i$ and $\mathbf{x}_j$, and these vectors have magnitude d_i and d_j. g_i and g_j are simply the distances from the S_l center to interfaces I_i and I_j.

and convex sphere intersection interfaces. An interface I_i is determined by two quantities: (1) the amount of penetration g_i^* of S_i into S_l normalized to a unit sphere:

$$\boldsymbol{v}_i = \mathbf{x}_i - \mathbf{x}_l \tag{4}$$

$$d_i = |\boldsymbol{v}_i| \tag{5}$$

$$g_i^* = \frac{d_i^2 + r_l^2 - r_i^2}{2d_i r_l} \tag{6}$$

where $\boldsymbol{v}_i$ is a vector of length d_i between Cartesian coordinates $\mathbf{x}_l$ and $\mathbf{x}_i$ of spheres S_l and S_i. (2) Interface type f_i, which is either positive for convex or negative for concave interfaces, is given by

$$f_i = \mathrm{sign}(g_i^*) \tag{7}$$

Previously we stated that concave interfaces are treated as inverted convex interfaces. Concave interfaces will have a negative g_i^* which needs to be made positive:

$$g_i = |g_i^*| \tag{8}$$

Angular calculations are performed on unit vectors:

$$\boldsymbol{\mu}_i = f_i \frac{\boldsymbol{v}_i}{d_i} \tag{9}$$

which include the radial distances between two interfaces I_i and I_j,

$$\rho_{ij} = \arccos(\boldsymbol{\mu}_i \cdot \boldsymbol{\mu}_j) \tag{10}$$

as well as the opening angle to the interface,

$$\lambda_i = \arccos(g_i) \tag{11}$$

and the angle to the tessellation axis,

$$\psi_i = \arccos(\boldsymbol{\chi} \cdot \boldsymbol{\mu}_i) \tag{12}$$

Multiple circular interfaces will intersect if the sum of their conical opening angles λ_i and λ_j fall below their radial distance ρ_{ij}, and if one does not entirely contain the other. The intersection between these circular interfaces will then result in two intersection points $\boldsymbol{p}_{ij}$ and $\boldsymbol{p}_{ji}$. The cases in which opening angles exactly match their radial distance, or two interfaces are identical, are treated as if there were no intersection at all.

The three look-up parameters are derived from these circular interfaces. Φ angles are calculated with respect to a specific interface, which is illustrated in Figure 6. Here, Φ angles for $\boldsymbol{p}_{ij}$ differ with respect to interface I_i or I_j. The curved arrows over the Φ indices represent the direction of the intersection. Seen from I_j, $\boldsymbol{p}_{ij}$ is viewed as an incoming point (connecting I_i with I_j) and $\boldsymbol{p}_{jk}$ as an outgoing point (connecting I_j with I_k). The respective angles for interface I_j are an incoming angle $\Phi_{\overset{\frown}{ji}}$ and an outgoing angle $\Phi_{\overset{\frown}{jk}}$.

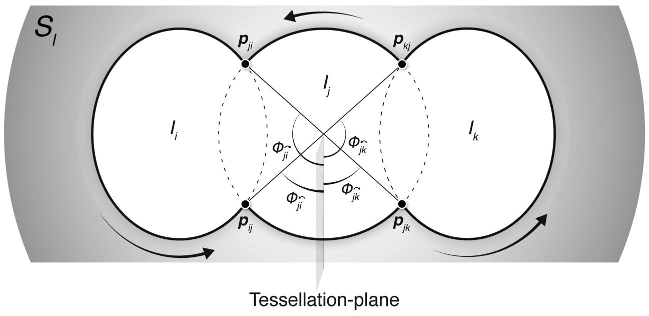


Figure 6. Intersections between circular interfaces I_i, I_j, and I_k on sphere S_l resulting in intersection points $\boldsymbol{p}_{ij}$, $\boldsymbol{p}_{jk}$, $\boldsymbol{p}_{kj}$, and $\boldsymbol{p}_{ji}$. Φ angles are calculated with respect to each interface's tessellation plane. Here, we show all of the Φ angles for interface I_j. Following the path indicated by the arrows, the SASA will be on the right-hand side. This path direction makes $\boldsymbol{p}_{ij}$ an incoming intersection point of I_j and $\boldsymbol{p}_{jk}$ an outgoing intersection point.

Figure 7 illustrates how these incoming and outgoing angles are calculated utilizing the previously introduced quantities for circular interfaces. First, η_{ij} is derived from the two opening angles λ_i and λ_j, and the angle between the two interface vectors ρ_{ij}:

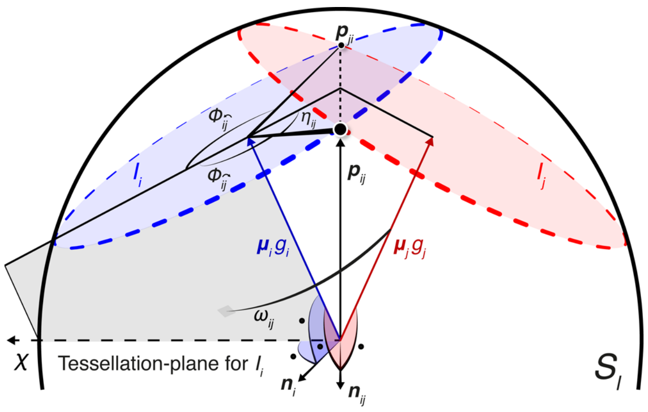


Figure 7. Intersections between circular interfaces I_i and I_j resulting in $\boldsymbol{p}_{ij}$ and $\boldsymbol{p}_{ji}$. A tessellation plane is formed between $\boldsymbol{\mu}_i$ and $\boldsymbol{\chi}$. The calculation of Φ can be split into the calculation of angles ω and η. ω_{ij} denotes the rotation of $\boldsymbol{\mu}_j$ around $\boldsymbol{\mu}_i$ with the tessellation plane as reference. η_{ij} is the opening angle of the intersection of interfaces I_i and I_j.

$$\eta_{ij} = \arccos(\cot(\lambda_i)\cot(\rho_{ij}) - \cos(\lambda_j)\csc(\lambda_i)\csc(\rho_{ij})) \tag{13}$$

η_{ij} is the opening angle of the intersection between the interfaces. Next, a normal vector $\mathbf{n}_i$ to the tessellation plane is established:

$$\boldsymbol{\nu}_i = \boldsymbol{\chi} \times \boldsymbol{\mu}_i \tag{14}$$

$$\mathbf{n}_i = \frac{\boldsymbol{\nu}_i}{|\boldsymbol{\nu}_i|} \tag{15}$$

in which $\boldsymbol{\nu}_i$ is the unnormalized vector. This calculation is followed by the computation of a vector normal to $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$, which we will denote as the normal to the interfacial plane of I_i and I_j:

$$\boldsymbol{\nu}_{ij} = \boldsymbol{\mu}_i \times \boldsymbol{\mu}_j \tag{16}$$

$$\mathbf{n}_{ij} = \frac{\boldsymbol{\nu}_{ij}}{|\boldsymbol{\nu}_{ij}|} \tag{17}$$

in which again $\boldsymbol{\nu}_{ij}$ describes the unnormalized vector. The angle between these two normal vectors determines the rotation of I_j around I_i:

$$\varpi_{ij} = \arccos(\mathbf{n}_i \cdot \mathbf{n}_{ij}) \tag{18}$$

Whether this rotation is to the left or to the right depends on the orientation of $\boldsymbol{\chi}$ with respect to the interfacial plane of I_i and I_j:

$$q_{ij} = -\,\mathrm{sign}(\mathbf{n}_{ij} \cdot \boldsymbol{\chi}) \tag{19}$$

resulting in the signed rotation:

$$\omega_{ij} = q_{ij}\varpi_{ij} \tag{20}$$

η_{ij} and ω_{ij} need to be combined into both outgoing $\Phi_{\overset{\frown}{ij}}$ and incoming $\Phi_{\overset{\frown}{ij}}$ angles:

$$\Phi_{\overset{\frown}{ij}} = \gamma_{\overset{\frown}{ij}} + \omega_{ij} + f_i f_j \eta_{ij} \tag{21}$$

$$\Phi_{ij}^{\curvearrowleft} = \gamma_{ij}^{\curvearrowleft} + \omega_{ij} - f_i f_j \eta_{ij} \tag{22}$$

with $\gamma_{ij}^{\curvearrowright}$ being either zero or π and $\gamma_{ij}^{\curvearrowleft}$ being either zero or $-\pi$, indicating whether the complement of ω needs to be calculated. Both quantities depend on the rotation of I_j around I_i and the magnitude of η_{ij}:

$$\gamma_{ij}^{\curvearrowright} = \pi(\text{sign}(\pi - \omega_{ij} - \eta_{ij}) - 1) \tag{23}$$

$$\gamma_{ij}^{\curvearrowleft} = -\pi(\text{sign}(-\pi - \omega_{ij} + \eta_{ij}) - 1) \tag{24}$$

Utilizing the preceding equations, we can look-up the areas of two subpatches of a triangular patch $A_{ijk}^{(l)}$ from precomputed tables $T(\Phi_{jk}^{\curvearrowright},\psi_j,\lambda_j)$ and $T(\Phi_{jk}^{\curvearrowleft},\psi_j,\lambda_j)$. As previously stated, the two subareas need to be subtracted from each other. Furthermore, if the outgoing angle is smaller than the incoming angle (indicated by q_{ijk}), the complementary area needs to be calculated:

$$q_{ijk} = \text{sign}(\Phi_{jk}^{\curvearrowright} - \Phi_{ji}^{\curvearrowleft}) \tag{25}$$

$$M_j = T(\pi, \psi_j, \lambda_j) \tag{26}$$

$$A_{ijk}^{(l)} = -f_j (M_j(q_{ijk} - 1) - T(f_j \Phi_{jk}^{\curvearrowright}, \psi_j, \lambda_j) + T(f_j \Phi_{ji}^{\curvearrowleft}, \psi_j, \lambda_j)) \tag{27}$$

where M_j is the maximal area of a triangular patch for a given interface I_j specified by ψ_j and λ_j.

3. EXPOSED BOUNDARY SEGMENTS

Each sphere can possibly have multiple contributions to $A^{(l)}$ in the form of multiple discontinuous segments of its circular interface. Each segment, described by a tuple (i,j,k), can be uniquely identified by its corresponding intersection points $\boldsymbol{p}_{ij}$ and $\boldsymbol{p}_{jk}$. Segments containing $\boldsymbol{p}_{ij}$ belong to the exposed boundary if, generally speaking, they are not covered by any other interface.

To calculate the three principal angles Φ, ψ, and λ for each intersection point of the boundary of the exposed area, it is necessary to find all segments (i,j,k) that contain these points. Each interface can possibly contain multiple segments for which the calculation is performed separately.

For each interface a cyclic sorted list is established. In the course of the algorithm, Φ angles for intersection points of all intersecting interfaces are added to the list. After each addition, the list is pruned by removing occluded intersection points (and their Φ angles). Figure 8 illustrates this pruning process. Here, every node between two corresponding intersection points is removed. Corresponding intersection points are either the two points of the newly added interface (blue double-dashed area) or two points already present in the cyclic list (red single-dashed). At the conclusion of the algorithm, the list contains only intersection points that are on the boundary of the exposed surface.

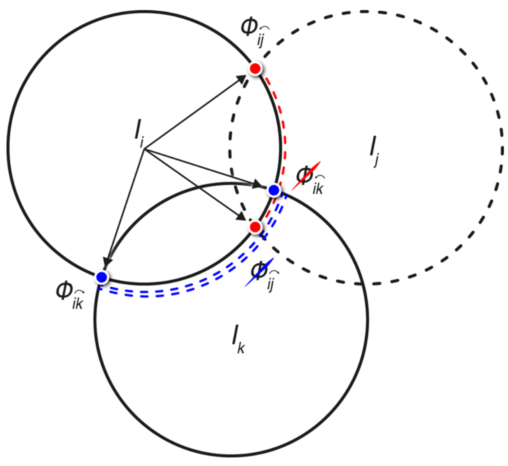


Figure 8. Addition of interface I_k to the cyclic list of interface I_i, which already contains data from an intersection with I_j. Intersections $\Phi_{ij}^{\curvearrowright}$ and $\Phi_{ij}^{\curvearrowleft}$ cause newly added intersection $\Phi_{ik}^{\curvearrowleft}$ to be deleted (red single-dashed area). Intersections $\Phi_{ik}^{\curvearrowright}$ and $\Phi_{ik}^{\curvearrowleft}$ cause old intersection $\Phi_{ij}^{\curvearrowleft}$ to be deleted (blue double-dashed area). Only intersections $\Phi_{ik}^{\curvearrowright}$ and $\Phi_{ij}^{\curvearrowleft}$ remain.

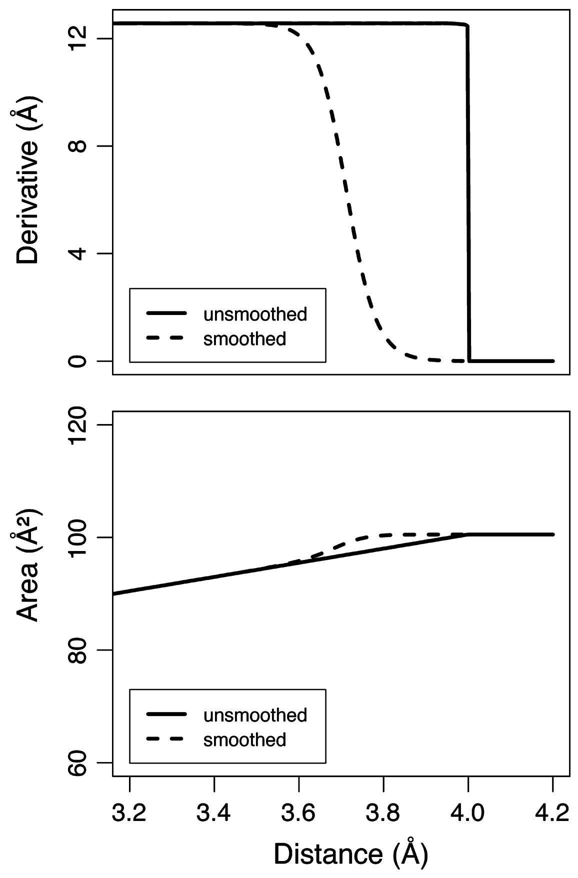


Figure 9. Illustration of the derivative (top) and area (bottom) calculated as a function of separation distance for two 2.0 Å radius spheres, with and without a distance-dependent logistic smoother. Without the smoother, derivatives rapidly jump from zero to some nonzero value upon contact. With the smoother, this discontinuity is mitigated in force space at the cost of adding a small hill in area space.

4. DISCONTINUITIES AND SINGULARITIES

Some constellations of spheres express extreme properties, or singular behaviors, which can cause instabilities during force calculations. These result in artificially large forces due to numerical limits and force discontinuities due to the discrete nature of soft-sphere and soft-circle intersections.

4.1. Contact Point between Two Spheres.

Imagine the following scenario for which key quantities are shown in Figure 9: two nonintersecting atoms S_l and S_i with radii of 2.0 Å approach one another. In the moment when the hulls just touch, i.e., when their distance is at precisely 4.0 Å, the derivative in the direction of the axis of separation abruptly jumps from zero to some nonzero value (illustrated by the continuous lines). Discontinuities of this type will act as large force barriers during a minimization or lead to numerical instabilities in the time-dependent evolution of the surface.[29] To compensate, we smooth the force and area around the contact point with a logistic function $\Lambda(t_j)$. Figure 9 illustrates the effect of $\Lambda(t_j)$. The dashed line shows a smooth transition in force space around the contact point. However, it also introduces a small hill in the corresponding region in area space. The change in area in this region is quite small, leading to only a minor perturbation of the computed surface area values. We calculate $\Lambda(t_j)$ using a term t_j which depends on the distance between the spheres:

$$\tilde{A}^{(l)}_{(ijk)} = \Lambda(t_j) A^{(l)}_{(ijk)} \tag{28}$$

$$t_j = p_0\left(1 - \frac{d_j}{r_j + r_l}\right) \tag{29}$$

$$\Lambda(x) = \frac{1}{1 + \exp(-2xp_1 - p_1)} \tag{30}$$

Here, parameters p_0 and p_1 adjust the shape of the smoothing function. We set $p_0 = 0.15$ and $p_1 = 8.0$ in this implementation.

4.2. Interfacial Angular Singularities.

The derivatives of the principal angles ψ, λ, and Φ (which involves η and ω) lead to the calculation of fractional terms in which the denominator vanishes once the angles approach either 0 or π. This gives rise to numerically anomalous very large (or infinite) forces. λ and η are two angles that can describe extreme geometrical constellations: a very small λ refers to two spheres that just barely touch, a very small η refers to two interfaces that just barely touch, and a very large η refers to interfaces of identical size with nearly perfect overlap. In all cases, we can use a threshold to circumvent calculation in these extreme angular regions. Doing so creates a small discontinuity which could be removed using a smoothing strategy similar to that used for sphere–sphere contact-point discontinuities.

ψ angle singularities are entirely artificial due to the arbitrary nature of the tessellation axis χ. Singularities can be prevented by bringing χ into general position, i.e., choosing a χ which is not degenerate with respect to any ψ angle.

ω angle singularities can be handled in the same way as ψ angle singularities, i.e., by choosing a general position with respect to ψ and ω. However, we have found this approach to be computationally intensive. Instead, we remove the singularity via a tessellation-axis transformation. If a degenerate ω angle is detected for a given derivative, a new χ^* is chosen arbitrarily such that it is in a general position. We then calculate a normal vector $\mathbf{n}_i^*$ to the new tessellation plane using this new axis. We calculate a transformed ω_{ij}^* with respect to the new axis using eqs 14–20. ω_{ij} and ω_{ij}^* are related by the angle between the new and old tessellation plane τ_i:

$$\tau_i = \arccos(\mathbf{n}_i^* \cdot \mathbf{n}_i) \tag{31}$$

The original ω_{ij} can be reconstructed:

$$\omega_{ij} = \frac{1}{2}\pi(q_1 - q_0 q_1 + q_2 - q_0 q_2) + q_0 q_1 \tau_i + q_0 q_2 \omega_{ij}^* \tag{32}$$

where the relationship between the new and old tessellation planes depend on the geometrical constellation of χ^* and $\mathbf{n}_{ij}$. q_0 indicates if the tessellation plane and the interfacial plane of I_i and I_j are similarly orientated with respect to the tessellation axis:

$$q_0 = \operatorname{sign}(\chi \cdot \mathbf{n}_{ij}) \operatorname{sign}(\chi \cdot \mathbf{n}_i) \tag{33}$$

q_1 and q_2 determine the angular relationship between τ_i and ω_{ij}^*, and whether the complement of ω_{ij}^* needs to be computed:

$$q_1 = \begin{cases} -1 & (q_0\tau_i > \omega_{ij}) \wedge (q_0\tau_i \geq q_0\omega_{ij}^*) \\ 1 & \text{else} \end{cases} \tag{34}$$

$$q_2 = \begin{cases} -1 & (q_0\tau_i > \omega_{ij}) \wedge (q_0\tau_i < q_0\omega_{ij}^*) \\ 1 & \text{else} \end{cases} \tag{35}$$

The advantage of eq 32 is that it does not depend on the possibly singular angle between $\mathbf{n}_i$ and $\mathbf{n}_{ij}$. Instead, it depends on the guaranteed nonsingular angles ω_{ij}^* and τ_i for which partial derivatives can be robustly calculated.

A different kind of singularity arises if an interface contains the mirror point of the tessellation point on the opposite side of the sphere. This would be the case if the sum of λ and ψ is equal to π or if it exceeds π. During the calculation of the integration tables, we map Φ angles to ϕ angles, which represent longitudinal lines that pass through the same intersection point (see the Appendix for a more detailed explanation of ϕ angles). The former case would map one Φ angle to an infinite number of ϕ angles. The latter would map Φ angles to ϕ angles in such a way that close to the singularity the look-up tables would be too sparse. This gives rise to imprecise areas and derivatives. To handle this, we split each sphere into two independently tessellated hemispheres. As a result, no Φ angles close to the singularity need be used.

5. RESULTS

5.1. Correctness of Areas and Derivatives.

We show the correctness of the area and derivatives by comparing to both exact and approximate values for a set of proteins and integration tables of different grid sizes. Here, we compare against the exact output of http://curie.utmb.edu/getarea.html and the approximate LCPO algorithm.[26] The set of proteins we chose includes ubiquitin, five structures of increasing size that were mentioned in ref 21, 14 structures from Decoys 'R' Us[30] that we have observed to be challenging for statistically derived area calculation algorithms, and 22 structures from ref 31 that represent a set of different folds. We list PDB codes for these structures in Table 1 and Table 2. We note that all of these protein structures lack hydrogen atoms due to limitations of both the exact GETAREA and approximate LCPO methods. TRIFORCE suffers from no such limitation.

Table 1 and Table 2 show the relative unsigned error (RUE) of per-atom areas and derivatives, respectively, for this set of 42 proteins. At the bottom of these tables we also report the average RUE and average root-mean-square deviation (RMSD) over this set. In general, TRIFORCE shows significantly improved agreement with the exact per-atom areas and derivatives over the approximate LCPO technique in these comparisons. Note that while the listed proteins are sorted by the number of heavy atoms, TRIFORCE is not affected by protein size. LCPO is also not greatly affected by the number of heavy atoms, though there appears to be a greater degree of variability of the predicted per-atom areas when using this approach. When we use an interpolation grid detail of 64 rather than 16 with TRIFORCE, we observe a roughly 15 times

Table 1. Relative Unsigned Error (RUE) of the Per-Atom Surface Areas Calculated Using TRIFORCE, with Grid Dimensions of 16 and 64, and LCPO Relative to Exact (GETAREA) Surface Areas for 42 Proteins

		RUE of area (Å^2)		
		LCPO	TRIFORCE	
PDB code	heavy atoms		grid 16	grid 64
1PLX	40	0.9602	0.0070	0.0001
1CBH	260	1.8208	0.0148	0.0021
1SP2	269	1.6250	0.0183	0.0018
5RXN	422	3.8093	0.0372	0.0020
1I6F	436	3.8105	0.0486	0.0003
4PTI	454	4.2465	0.0344	0.0012
1FAS	468	3.7621	0.0333	0.0058
1SN3	492	3.4444	0.0390	0.0008
1CSP	505	1.8924	0.0241	0.0012
2CRO	520	5.6744	0.0239	0.0006
1FVQ	545	5.7127	0.0465	0.0022
1SDF	550	5.5041	0.0374	0.0034
1UBQ	602	2.6297	0.0178	0.0008
1HIP	617	3.9922	0.0266	0.0011
1PHT	666	4.1957	0.0460	0.0017
1J5D	721	4.9532	0.0614	0.0031
2CDV	801	5.4789	0.0316	0.0020
1OPC	805	5.0891	0.0361	0.0039
1KTE	818	5.4651	0.0401	0.0021
1NSO	858	4.2843	0.0279	0.0019
2PAZ	932	2.6954	0.0303	0.0018
1CHN	966	3.7557	0.0273	0.0017
1K40	976	3.4175	0.0206	0.0004
1OOI	986	5.0779	0.0348	0.0019
1PDO	988	5.8762	0.0361	0.0025
6LYZ	1001	4.8848	0.0342	0.0026
1LIT	1045	3.3452	0.0219	0.0012
1BJ7	1208	3.1121	0.0254	0.0015
2I1B	1219	8.8207	0.0349	0.0023
1MBS	1223	7.6788	0.0372	0.0025
1EMR	1232	6.8879	0.0289	0.0037
1CZT	1311	5.4416	0.0344	0.0012
2PTN	1629	4.6474	0.0286	0.0013
5PAD	1655	4.8401	0.0304	0.0007
1SUR	1739	3.1552	0.0304	0.0020
2HVM	2087	3.2928	0.0287	0.0022
2CYP	2299	5.3554	0.0478	0.0021
1RHD	2319	7.9767	0.0455	0.0034
2TMN	2432	7.6176	0.0414	0.0031
2TS1	2457	3.8035	0.0333	0.0026
1FRG	3361	3.7585	0.0313	0.0015
1MCP	3401	4.0348	0.0284	0.0020
av RUE		4.5577	0.0331	0.0020
av RMSD		9.352	0.045	0.004

Table 2. Relative Unsigned Error (RUE) of the Per-Atom Derivatives Calculated Using TRIFORCE, with Grid Dimensions of 16 and 64, and LCPO Relative to Exact (GETAREA) Derivatives for 42 Proteins

		RUE for grad (Å)		
		LCPO	TRIFORCE	
PDB code	heavy atoms		grid 16	grid 64
1PLX	40	9.030	0.164	0.005
1CBH	260	4.558	0.113	0.013
1SP2	269	6.743	0.181	0.026
5RXN	422	5.543	0.108	0.011
1I6F	436	6.408	0.138	0.043
4PTI	454	7.469	0.154	0.009
1FAS	468	6.032	0.114	0.013
1SN3	492	7.871	0.125	0.031
1CSP	505	5.817	0.114	0.015
2CRO	520	5.209	0.067	0.009
1FVQ	545	4.990	0.068	0.006
1SDF	550	6.004	0.122	0.007
1UBQ	602	6.049	0.107	0.008
1HIP	617	7.415	0.101	0.009
1PHT	666	6.425	0.172	0.014
1J5D	721	6.140	0.101	0.010
2CDV	801	6.614	0.108	0.014
1OPC	805	6.407	0.104	0.019
1KTE	818	5.425	0.080	0.010
1NSO	858	5.330	0.117	0.027
2PAZ	932	5.677	0.097	0.010
1CHN	966	5.060	0.071	0.008
1K40	976	4.625	0.071	0.012
1OOI	986	5.930	0.088	0.009
1PDO	988	6.853	0.124	0.013
6LYZ	1001	5.903	0.096	0.010
1LIT	1045	5.047	0.081	0.012
1BJ7	1208	5.741	0.104	0.014
2I1B	1219	5.988	0.109	0.018
1MBS	1223	6.590	0.174	0.059
1EMR	1232	6.636	0.232	0.059
1CZT	1311	4.568	0.079	0.009
2PTN	1629	5.307	0.120	0.031
5PAD	1655	5.750	0.103	0.009
1SUR	1739	6.108	0.179	0.020
2HVM	2087	5.081	0.088	0.008
2CYP	2299	5.640	0.108	0.016
1RHD	2319	5.015	0.092	0.011
2TMN	2432	5.050	0.087	0.014
2TS1	2457	5.207	0.114	0.009
1FRG	3361	4.780	0.083	0.013
1MCP	3401	5.409	0.086	0.008
av RUE		5.815	0.112	0.016
av RMSD		4.367	0.166	0.071

reduction in area RUE and a 7 times reduction in derivative RUE. These calculated per-atom areas and derivatives are numerically invariant upon rotation of the molecules and choice of the tesselation axis, as the measured fluctuations when rotating the molecule and/or the tesselation axis are smaller than the errors reported in Table 1.

We note that the real differences in the derivatives are better than those shown in Table 2. The reason is not imprecise calculations by TRIFORCE but differences between the algorithms of GETAREA and TRIFORCE due to limited numerical precision: both algorithms utilize different methods to determine whether a sphere contributes to the exposed boundary or not. As an example, observe the large difference in the derivatives of molecule “1MBS”. Here, a single atomistic constellation expresses extreme properties. In this constellation there are two interfaces in which one is buried almost completely in the other. This leads to an ambiguous situation in which GETAREA sees no intersection between the interfaces while TRIFORCE sees the intersection. Consequently, TRIFORCE will calculate forces for the extra interface while it is invisible for GETAREA. Due to the ambiguous nature of these cases, there is no right or wrong result. Both are correct within the choices made in algorithm implementation.

Figure 10 shows the average per-atom RUE of both TRIFORCE and LCPO areas and derivatives relative to exact

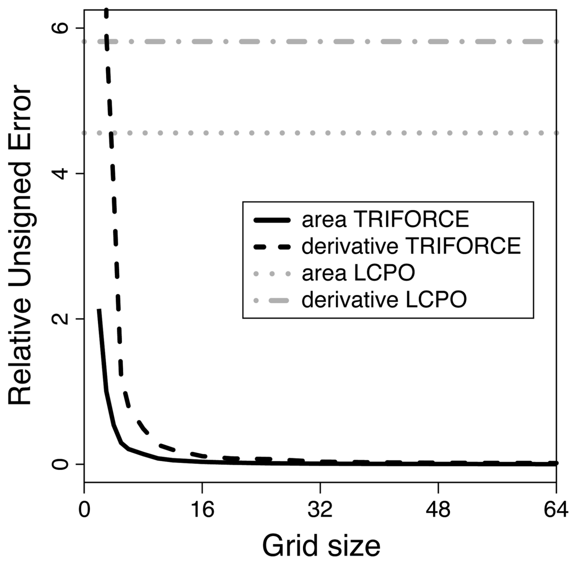


Figure 10. Relative unsigned error of per-atom areas and derivatives over a set of 42 proteins calculated using TRIFORCE and LCPO relative to the analytical GETAREA method. We see that TRIFORCE error decreases with increasing interpolation grid density.

(GETAREA) values. We show the TRIFORCE results as a function of interpolation grid detail. With an extremely coarse grid of 3 ($3 \times 3 \times 3$), the TRIFORCE areas and derivatives are already closer to the exact values than those reported by the statistically derived LCPO method. With finer grids, the TRIFORCE results converge on the exact values. Aside from the time required to load a grid in the initialization of TRIFORCE, there is, in principle, no computational performance penalty in choosing a finer grid. The primary cost as a function of grid detail is the memory required for storing the grid values. To test this, we performed a series of TRIFORCE surface area calculations on egg white lysozyme (6LYZ: 1001 heavy atoms, 129 residues) on a 2.6 GHz Intel Core i7 processor with 6.25 MB of accessible cache. For grid details up to 32, the averaged real world performance of the TRIFORCE integration time was 60 ms. Grid details of 64 and 80 have average integration times of 70 ms, a greater than 15% performance penalty. This penalty is due to cache overflow, which forces an increasing number of system memory calls. All grids with details smaller than 32 will fit entirely on this processor's cache while larger grids do not. For example, a quadratic interpolation scheme with a grid detail of 16 uses 328 KB of memory while a grid detail of 64 uses 21 MB. We can modulate these memory requirements to varying degrees by implementing different interpolation schemes; however, we expect increasingly large grids that overflow the cache to suffer from an increasing number of system memory calls. This will lead to an increasing performance penalty up to a hardware specific maximum, where nearly all of the grid access steps of integration are done with system memory calls rather than cache look-ups.

We note that performance analyses such as those stated above are very much hardware- and implementation-dependent. This makes it difficult to perform rigorous performance comparisons between TRIFORCE, analytical, and even numerical approaches. Instead of head-to-head timings, we attempt to provide performance guidance relative to the GETAREA analytical method through operation counts from the published equations. Based on the relative numbers of floating point addition, multiplication, division, and square root operations, TRIFORCE integration will exhibit an approximately 15% reduced latency over GETAREA analytical integration on current Intel Core i7 processors. It should be noted that *integration* does not consider the construction of the Gauss–Bonnet path, which has an identical cost for all analytical surface area methods that require it and which is approximately an order of magnitude more costly than TRIFORCE integration. We are currently preparing a manuscript detailing how to accelerate Gauss–Bonnet path constructions with little loss in numerical accuracy, bringing performance in-line with TRIFORCE integration.

Finally, while not overly apparent in Figure 10, there is a slight bump in the per-atom derivative RUE for a grid detail of 25. This bump comes not from the TRIFORCE algorithm but from a numerical artifact in the construction of this specific grid. The integration routine used in its construction reached numerically unstable regions, preventing calculation of this grid to the same precision as the others. As the performance differences are minimal when using neighboring grid details, we recommend not using this specific interpolation grid.

5.2. Minimization Test of Particle Surface Areas.

We performed both steepest descent and Broyden–Fletcher–Goldfarb–Shanno (BFGS) minimizations on a fluorene molecule ($C_{13}H_{10}$) to test correctness of the derivatives. No atomic bonding, repulsive, torsional, or angular forces were included; as such the minimization was expected to reduce the molecule into a single sphere. While this is somewhat unphysical, it provides a stringent test of the algorithm as it has a known target surface area, that of an inflated sphere (vdW + water distance) of one single carbon with radius of 3.09 Å. Figure 11 shows the averaged course of the steepest descent minimization as a function of TRIFORCE grid detail, each over 1000 randomized runs. After 80 steepest descent steps, the averaged total area is mostly level and the system is nearly completely collapsed. The BFGS minimizations (not shown) tended to converge somewhat sooner at around 50 steps. Complete collapse is not observed with either minimizer due to how the minimization algorithms treat buried spheres. In the late stages of minimization, all spheres with radii smaller than 3.09 Å are entirely buried inside the larger spheres, and all spheres with radii of 3.09 Å are mostly covered by each other. Because of their lack of SASA, the completely buried spheres do not experience forces and are therefore invisible to the minimizer. As such, these spheres are prone to resurface constantly when their enveloping spheres shift positions. Similarly, the principal angles become numerically less well defined when spheres nearly overlap, exaggerating this resurfacing behavior.

Coarser grid details have less accurate derivatives, leading to greater numerical variability in this minimization process, denoted by the shaded standard deviation envelop regions in Figure 11. We attempted even lower resolution grid details, but

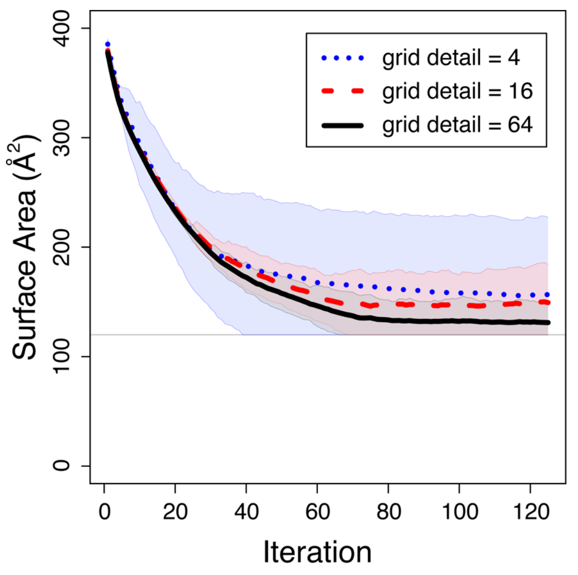


Figure 11. Solvent-accessible surface area of a fluorene molecule ($C_{13}H_{10}$) over the course of a steepest descent minimization without bonding or repulsive forces between atoms when using TRIFORCE with different grid details. The molecule minimizes down to a single sphere of approximately radius 3.09 Å, corresponding to the radius of the largest component atom and illustrated here with the gray line. The shaded regions are the standard deviation envelopes over 1000 randomized runs. While all shown grid details minimize at much the same rate, coarser grids exhibit more numerical noise as spheres begin to overlap and the principal angles become less well defined.

the derivatives become so poor at a grid detail of 3 (RUE of derivatives, 5.9) that the minimizations would fail to converge. Here, the atoms would move in mostly random directions, separating the molecule into individual atomic spheres. It appears that we need a grid detail of 4, which has an RUE of the per-atom derivatives of 3.8, to ensure convergence. It is unclear from this test if the derivatives from approximate surface area methods such as LCPO (RUE of derivatives, 5.8) are themselves sufficiently accurate to ensure convergence to the expected minimum structure.

6. DISCUSSION

TRIFORCE has several features which distinguish it from algorithms that perform equivalent tasks. It has the ability to segment the exposed surface efficiently into triangular patches. This ability allows TRIFORCE to accumulate not just surface areas but any precomputed quantity. For such a purpose, the algorithm seamlessly allows for two adjustments: (1) the selection of an arbitrary tessellation axis χ, to adjust for geometrical requirements and (2) the option to add additional dimensions to the integration tables, to allow the modeling of a functional surface with higher complexity.

In contrast to some algorithms that rely on united atom molecular representations, TRIFORCE is capable of handling all topologies, including concave circular interfaces. This is crucial because such interfaces frequently occur in molecules where hydrogen atoms are buried in large van der Waals radii of bonded atoms.

Its tabular integration procedure can be efficiently implemented onto GPUs (development is in progress) in which this task becomes a simple texture look-up. The whole algorithm has been designed to allow for parallel processing on the level of circular interfaces, in contrast to an atomic level. The nature of the algorithm structure gives TRIFORCE the potential to utilize the GPUs or other massively parallel architectures to full capacity. Integration tables are stored in an architecture-independent way, enabling usage of the library on different platforms right away.

All calculations can be made arbitrarily precise by increasing the grid detail, as Figure 10 shows. Doing this will require additional memory but will not slow down the algorithm, aside from the initial loading procedure, if the grid is still small enough to remain in the processing unit's cache. TRIFORCE can be accessed and run on the web from http://dillgroup.io. A downloadable version of the source code is also available on this Web site, free of charge for academic purposes.

7. CONCLUSION

We have presented a novel algorithm for the calculation of the SASA and its derivatives for arbitrary molecules. Although the method is partly numerical, errors in both of these quantities are marginal, and we see performance and flexibility gains over fully analytical approaches. We tested its correctness through an extensive set of calculations on a database of 42 proteins of various sizes and folds. We evaluated its robustness by successfully applying TRIFORCE in minimization calculations of the SASA. To compensate for discontinuities in the derivatives upon sphere contact, we have introduced a logistical smoothing function which is able to bridge the gap in force space between two distant spheres and their intersecting counterparts. Future direction for the development of TRIFORCE involves complete parallelization of the algorithm through implementation on GPUs and other massively parallel architectures.

8. APPENDIX

8.1. Integration Tables

Integration tables are computed for a predefined grid of Φ, ψ, and λ values. ψ and λ are parametrizations which correspond to a virtual circular interface that has the two properties. The integration over these simplistic triangular surface areas is formally done by considering a vanishingly small patch Ω and integrating over θ and φ:

$$A = \int_{\varphi_i}^{\varphi_j} \int_0^{\Theta(\varphi,\psi,\lambda)} \mathrm{d}\psi \; \mathrm{d}\theta \; \sin(\theta) \left| \frac{\partial \Omega}{\partial \theta} \times \frac{\partial \Omega}{\partial \psi} \right| \quad (36)$$

θ is the radial length of an arc drawn between the tessellation point and the boundary of a virtual interface. φ describes the rotation of this radial line around the tessellation axis, whereas a value of 0 corresponds to a radial line that intersects with the center of the interface. Consequently, integration limits for θ depend on the parametrization of the interface, as well as the rotation φ, and is given by

$$\Theta(\varphi, \psi, \lambda) = \arccos\left(\frac{\cos(\lambda)\cos(\psi) \pm \iota}{\kappa} \right) \quad (37)$$

with $\iota = [\kappa - \cos(\psi)^2\,(-\cos(\lambda)^2 + \kappa)]^{1/2}$ and $\kappa = \cos(\psi)^2 + \cos(\varphi)^2\,\sin(\psi)^2$.

8.2. Derivatives

Forces are calculated with respect to the change in atomic coordinates and its effect on surface area. We have to take into account each atom $\mathbf{x}_l$ and the change of coordinates of intersecting atoms $\mathbf{x}_\delta^{(l)}$. For the sake of readability, we omit index (l) where possible. We denote the set of interface indices of S_l by Int_l.

8.2.1. Area. The derivative of the area of a molecule is composed of the derivatives of the areas of its atoms:

$$\frac{\partial A}{\partial \mathbf{x}_\delta} = \sum_l \frac{\partial A^{(l)}}{\partial \mathbf{x}_\delta} \tag{38}$$

If $\delta \notin \mathrm{Int}_l$ or $\delta \neq l$, the derivative will be zero. The derivatives of each atom are comprised by derivatives from triangular patches:

$$\frac{\partial A^{(l)}}{\partial \mathbf{x}_\delta} = r_l^{\,2} \sum_{(ijk)\in \mathrm{Seg}_l} \frac{\partial A^{(l)}_{(ijk)}}{\partial \mathbf{x}_\delta} \tag{39}$$

If $\delta \notin \{i,j,k\}$ or $\delta \neq l$, the derivative will be zero. Each derivative from a triangular patch is divided into two subpatches:

$$\frac{\partial A^{(l)}_{(ijk)}}{\partial \mathbf{x}_\delta} = -(q_{ijk} - 1)\frac{\partial M_j}{\partial x_\delta} + \left(\frac{\partial T(P_{jk})}{\partial \mathbf{x}_\delta} - \frac{\partial T(P_{ji})}{\partial \mathbf{x}_\delta}\right) \tag{40}$$

With $P_{jk} = (\Phi_{jk}^{\frown}, \psi_j, \lambda_j)$ and $P_{ji} = (\Phi_{ji}^{\frown}, \psi_j, \lambda_j)$.

8.2.2. Integration Tables. The integration tables are functions of ψ, λ, and Φ; their derivatives are split accordingly:

$$\frac{\partial T(P_{ij})}{\partial \mathbf{x}_\delta} = \frac{\partial T(P_{ij})}{\partial \Phi_{ij}}\frac{\partial \Phi_{ij}}{\partial \mathbf{x}_\delta} + \frac{\partial T(P_{ij})}{\partial \psi_j}\frac{\partial \psi_j}{\partial \mathbf{x}_\delta} + \frac{\partial T(P_{ij})}{\partial \lambda_j}\frac{\partial \lambda_j}{\partial \mathbf{x}_\delta} \tag{41}$$

The derivatives of the integration table with respect to the look-up parameters are themselves drawn and interpolated from three different look-up tables $G^{(\partial\Phi)}$, $G^{(\partial\psi)}$, and $G^{(\partial\lambda)}$:

$$\frac{\partial T(P_{ij})}{\partial \Phi_{ij}} = G^{(\partial\Phi)}(P_{ij}) \tag{42}$$

$$\frac{\partial T(P_{ij})}{\partial \psi_j} = G^{(\partial\psi)}(P_{ij}) \tag{43}$$

$$\frac{\partial T(P_{ij})}{\partial \lambda_j} = G^{(\partial\lambda)}(P_{ij}) \tag{44}$$

Derivatives of the look-up parameters with respect to atomic coordinates are calculated analytically as follows.

8.2.3. Angle ψ. Derivatives for angle ψ, illustrated in Figure 4 and eq 12, are

$$\frac{\partial \psi_a}{\partial \mathbf{x}_\delta} = \frac{\partial \psi_a}{\partial \boldsymbol{\mu}_a}\frac{\partial \boldsymbol{\mu}_a}{\partial \mathbf{x}_\delta} \tag{45}$$

$$\frac{\partial \psi_a}{\partial \boldsymbol{\mu}_a} = -\frac{\chi}{\sqrt{1 - (\boldsymbol{\mu}_a \cdot \chi)^2}} \tag{46}$$

$$\frac{\partial \boldsymbol{\mu}_a}{\partial \mathbf{x}_a} = \frac{1}{d_a}(\mathbf{I} - \boldsymbol{\mu}_a \otimes \boldsymbol{\mu}_a)^{\mathrm{T}} \tag{47}$$

$$\frac{\partial \boldsymbol{\mu}_a}{\partial \mathbf{x}_l} = -\frac{1}{d_a}(\mathbf{I} - \boldsymbol{\mu}_a \otimes \boldsymbol{\mu}_a)^{\mathrm{T}} \tag{48}$$

8.2.4. Angle λ. Derivatives for angle λ, as illustrated in Figure 4 and eq 11, are

$$\frac{\partial \lambda_a}{\partial \mathbf{x}_\delta} = \frac{\partial \lambda_a}{\partial g_a}\frac{\partial g_a}{\partial \mathbf{x}_\delta} \tag{49}$$

$$\frac{\partial \lambda_a}{\partial g_a} = -\frac{1}{\sqrt{1 - g_a^{\,2}}} \tag{50}$$

$$\frac{\partial g_a}{\partial \mathbf{x}_\delta} = \frac{\partial g_a}{\partial d_a}\frac{\partial d_a}{\partial \mathbf{x}_\delta} \tag{51}$$

$$\frac{\partial g_a}{\partial d_a} = -\frac{g_a}{d_a} + \frac{f_a}{r_l} \tag{52}$$

$$\frac{\partial d_a}{\partial \mathbf{x}_a} = \boldsymbol{\mu}_a \tag{53}$$

$$\frac{\partial d_a}{\partial \mathbf{x}_l} = -\boldsymbol{\mu}_a \tag{54}$$

8.2.5. Angle Φ. Derivatives for angle Φ, as illustrated in Figure 4 and eqs 21–22, are

$$\frac{\partial \Phi_{ab}^{\frown}}{\partial \mathbf{x}_\delta} = \frac{\partial \eta_{ab}}{\partial \mathbf{x}_\delta} + \frac{\partial \omega_{ab}}{\partial \mathbf{x}_\delta} \tag{55}$$

$$\frac{\partial \Phi_{ab}^{\frown}}{\partial \mathbf{x}_\delta} = -\frac{\partial \eta_{ab}}{\partial \mathbf{x}_\delta} + \frac{\partial \omega_{ab}}{\partial \mathbf{x}_\delta} \tag{56}$$

8.2.6. Angle η. Derivatives for angle η, as illustrated in Figure 7 and eq 13, are

$$\frac{\partial \eta_{ab}}{\partial \mathbf{x}_\delta} = \frac{\partial \eta_{ab}}{\partial \lambda_a}\frac{\partial \lambda_a}{\partial \mathbf{x}_\delta} + \frac{\partial \eta_{ab}}{\partial \lambda_b}\frac{\partial \lambda_b}{\partial \mathbf{x}_\delta} + \frac{\partial \eta_{ab}}{\partial \rho_{ab}}\frac{\partial \rho_{ab}}{\partial \mathbf{x}_\delta} \tag{57}$$

Let $\rho_{ab} = \cot(\lambda_a)\cot(\rho_{ab})$, $\sigma_{ab} = \cos(\lambda_b)\csc(\lambda_a)\csc(\rho_{ab})$, and $\varsigma_{ab} = \rho_{ab} - \sigma_{ab}$:

$$\frac{\partial \eta_{ab}}{\partial \lambda_a} = \left[\frac{\csc(\lambda_a)\rho_{ab}}{\cos(\lambda_a)} - \sigma_{ab}\tan(\lambda_a)\right] / \sqrt{1 - \varsigma_{ab}^{\,2}} \tag{58}$$

$$\frac{\partial \eta_{ab}}{\partial \lambda_b} = -[\sigma_{ab}\tan(\lambda_b)] / \sqrt{1 - \varsigma_{ab}^{\,2}} \tag{59}$$

$$\frac{\partial \eta_{ab}}{\partial \rho_{ab}} = \left[\frac{\csc(\rho_{ab})\rho_{ab}}{\cos(\rho_{ab})} - \sigma_{ab}\tan(\rho_{ab})\right] / \sqrt{1 - \varsigma_{ab}^{\,2}} \tag{60}$$

8.2.7. Angle ρ. Derivatives for angle ρ, as illustrated in Figure 5 and eq 10, are

$$\frac{\partial \rho_{ab}}{\partial \mathbf{x}_a} = \frac{\partial \rho_{ab}}{\partial \boldsymbol{\mu}_a}\frac{\partial \boldsymbol{\mu}_a}{\partial x_a} \tag{61}$$

$$\frac{\partial \rho_{ab}}{\partial \boldsymbol{\mu}_a} = -\frac{\boldsymbol{\mu}_b}{\sqrt{1 - (\boldsymbol{\mu}_a \cdot \boldsymbol{\mu}_b)^2}} \tag{62}$$

The following identity applies: $\rho_{ba} = \rho_{ab}$.

8.2.8. Angle ω. Derivatives for angle ω, as illustrated in Figure 7 and eq 20, are

$$\frac{\partial \omega_{ab}}{\partial \mathbf{x}_\delta} = \frac{\partial \omega_{ab}}{\partial \varpi_{ab}}\frac{\partial \varpi_{ab}}{\partial \mathbf{x}_\delta} \tag{63}$$

$$\frac{\partial \omega_{ab}}{\partial \varpi_{ab}} = q_{ab} \quad (64)$$

$$\frac{\partial \varpi_{ab}}{\partial \mathbf{x}_\delta} = \frac{\varpi_{ab}}{\partial \mathbf{n}_a}\frac{\partial \mathbf{n}_a}{\partial \mathbf{x}_\delta} + \frac{\partial \varpi_{ab}}{\partial \mathbf{n}_{ab}}\frac{\partial \mathbf{n}_{ab}}{\partial \mathbf{x}_\delta} \quad (65)$$

$$\frac{\partial \varpi_{ab}}{\partial \mathbf{n}_a} = \frac{\mathbf{n}_{ab}}{\sqrt{1 - (\mathbf{n}_a \cdot \mathbf{n}_{ab})^2}} \quad (66)$$

$$\frac{\partial \varpi_{ab}}{\partial \mathbf{n}_{ab}} = \frac{\mathbf{n}_a}{\sqrt{1 - (\mathbf{n}_a \cdot \mathbf{n}_{ab})^2}} \quad (67)$$

$$\frac{\partial \mathbf{n}_a}{\partial \mathbf{x}_\delta} = \frac{\partial \mathbf{n}_a}{\partial \mathbf{v}_a}\frac{\partial \mathbf{v}_a}{\partial \boldsymbol{\mu}_a}\frac{\partial \boldsymbol{\mu}_a}{\partial \mathbf{x}_\delta} \quad (68)$$

$$\frac{\partial \mathbf{n}_a}{\partial \boldsymbol{\nu}_a} = -\frac{1}{d_a}(\mathbf{I} - \mathbf{n}_a \otimes \mathbf{n}_a)^{\mathrm{T}} \quad (69)$$

$$\frac{\partial \boldsymbol{\nu}_a}{\partial \boldsymbol{\mu}_a} = \mathbf{I} \times \boldsymbol{\chi} \quad (70)$$

$$\frac{\partial \mathbf{n}_{ab}}{\partial \mathbf{x}_\delta} = \frac{\partial \mathbf{n}_{ab}}{\partial \boldsymbol{\nu}_{ab}}\frac{\partial \boldsymbol{\nu}_{ab}}{\partial \boldsymbol{\mu}_a}\frac{\partial \boldsymbol{\mu}_a}{\partial \mathbf{x}_\delta} + \frac{\partial \mathbf{n}_{ab}}{\partial \boldsymbol{\nu}_{ab}}\frac{\partial \boldsymbol{\nu}_{ab}}{\partial \boldsymbol{\mu}_b}\frac{\partial \boldsymbol{\mu}_b}{\partial \mathbf{x}_\delta} \quad (71)$$

$$\frac{\partial \mathbf{n}_{ab}}{\partial \boldsymbol{\nu}_{ab}} = -\frac{1}{d_{ab}}(\mathbf{I} - \mathbf{n}_{ab} \otimes \mathbf{n}_{ab})^{\mathrm{T}} \quad (72)$$

$$\frac{\partial \boldsymbol{\nu}_{ab}}{\partial \boldsymbol{\mu}_a} = \mathbf{I} \times \boldsymbol{\mu}_b \quad (73)$$

$$\frac{\partial \boldsymbol{\nu}_{ab}}{\partial \boldsymbol{\mu}_b} = -\mathbf{I} \times \boldsymbol{\mu}_a \quad (74)$$

8.2.9. Logistic Smoothing. The logistic smoother is applied to the area term by multiplication which creates the following additional partial derivatives in force space. Generally, the smoothing has impact on all derivatives that depend on interface I_j:

$$\frac{\partial \tilde{A}^{(l)}_{(ijk)}}{\partial \mathbf{x}_\delta} = \Lambda(t_j)\frac{\partial A^{(l)}_{(ijk)}}{\partial \mathbf{x}_\delta} + \frac{\partial \Lambda(t_j)}{\partial \mathbf{x}_\delta}A^{(l)}_{(ijk)} \quad (75)$$

$$\frac{\partial \Lambda(t_j)}{\partial \mathbf{x}_\delta} = \frac{\partial \Lambda(t_j)}{\partial \mathbf{t}_j}\frac{\partial t_j}{\partial \mathbf{x}_\delta} \quad (76)$$

$$\frac{\partial \Lambda(t_j)}{\partial \mathbf{t}_j} = \frac{p_1}{1 + \cosh(p_1 - 2xp_1)} \quad (77)$$

$$\frac{\partial t_j}{\partial \mathbf{x}_j} = -\frac{\boldsymbol{\mu}_j p_0}{r_j + r_l} \quad (78)$$

$$\frac{\partial t_j}{\partial \mathbf{x}_l} = -\frac{\partial t_j}{\partial \mathbf{x}_j} \quad (79)$$

8.2.10. Tessellation-Axis Transformation. Applying a tessellation-axis transformation as discussed in the section describing singularities involves the calculation of additional partial derivatives:

$$\frac{\partial \omega_{ab}}{\partial \mathbf{x}_\delta} = q_0 q_1 \frac{\partial \tau_a}{\partial \mathbf{x}_\delta} + q_0 q_2 \frac{\partial \omega^*_{ab}}{\partial \mathbf{x}_\delta} \quad (80)$$

The second addend can be calculated using the same series of eqs 63–74.

$$\frac{\partial \tau_a}{\partial \mathbf{x}_\delta} = \frac{\partial \tau_a}{\partial \mathbf{n}_a}\frac{\partial \mathbf{n}_a}{\partial \mathbf{x}_\delta} + \frac{\partial \tau_a}{\partial \mathbf{n}^*_a}\frac{\partial \mathbf{n}^*_a}{\partial \mathbf{x}_\delta} \quad (81)$$

Both addends can be calculated using eqs 68–74.

AUTHOR INFORMATION

Corresponding Authors

*(N.J.D.D.) E-mail: nils.drechsel@gmail.com.

*(J.V.-F.) E-mail: jordi.villa@uvic.cat.

Funding

We appreciate the financial support provided by NIH Grant GM63592. C.J.F. acknowledges the financial support of startup funds at Oklahoma State University. J.V.-F. acknowledges financial support by AGAUR BE-DGR 2009. This work has also been partially funded by Grant CTQ2008-00755.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We thank Montserrat Corbera for helpful discussions and comments on initial versions of this manuscript. The In addition, we thank the anonymous reviewers of this manuscript for providing helpful comments and suggestions.

REFERENCES

(1) Uhlig, H. H. *J. Phys. Chem.* **1937**, *41*, 1215–1226.
(2) Hermann, R. B. *J. Phys. Chem.* **1972**, *76*, 2754–2759.
(3) Nicholls, A.; Sharp, K. A.; Honig, B. *Proteins: Struct., Funct., Genet.* **1991**, *11*, 281–296.
(4) Gallicchio, E.; Kubo, M. M.; Levy, R. M. *J. Phys. Chem. B* **2000**, *104*, 6271–6285.
(5) Eisenberg, D.; McLachlan, A. D. *Nature* **1986**, *319*, 199–203.
(6) Hawkins, G. D.; Cramer, C. J.; Truhlar, D. G. *J. Phys. Chem.* **1996**, *100*, 19824–19839.
(7) Hawkins, G. D.; Cramer, C. J.; Truhlar, D. G. *J. Phys. Chem. B* **1997**, *101*, 7147–7157.
(8) Wang, J.; Wang, W.; Huo, S.; Lee, M.; Kollman, P. A. *J. Phys. Chem. B* **2001**, *105*, 5055–5067.
(9) Gallicchio, E.; Zhang, L. Y.; Levy, R. M. *J. Comput. Chem.* **2002**, *23*, 517–529.
(10) Wagoner, J. A.; Baker, N. A. *Proc. Natl. Acad. Sci. U. S. A.* **2006**, *103*, 8331–8336.
(11) Tan, C.; Tan, Y.-H.; Luo, R. *J. Phys. Chem. B* **2007**, *111*, 12263–12274.
(12) Gallicchio, E.; Paris, K.; Levy, R. M. *J. Chem. Theory Comput.* **2009**, *5*, 2544–2564.
(13) Fennell, C. J.; Dill, K. A. *J. Am. Chem. Soc.* **2010**, *132*, 234–240.
(14) Fennell, C. J.; Kehoe, C. W.; Dill, K. A. *Proc. Natl. Acad. Sci. U. S. A.* **2011**, *108*, 3234–3239.
(15) Hummer, G.; Garde, S.; Garcia, A. E.; Pohorille, A.; Pratt, L. R. *Proc. Natl. Acad. Sci. U. S. A.* **1996**, *93*, 8951–8955.
(16) Hummer, G. *J. Am. Chem. Soc.* **1999**, *121*, 6299–6305.
(17) Lazaridis, T.; Karplus, M. *Proteins: Struct., Funct., Genet.* **1999**, *35*, 133–152.
(18) Connolly, M. L. *J. Appl. Crystallogr.* **1983**, *16*, 548–558.
(19) Richmond, T. J. *J. Mol. Biol.* **1984**, *178*, 63–89.
(20) Perrot, G.; Cheng, B.; Gibson, K. D.; Vila, J.; Palmer, K. A.; Nayeem, A.; Maigret, B.; Scheraga, H. A. *J. Comput. Chem.* **1992**, *13*, 1–11.

(21) Fraczkiewicz, R.; Braun, W. *J. Comput. Chem.* **1998**, *19*, 319−333.
(22) Hayryan, S.; Hu, C.-K.; Skřivánek, J.; Hayryan, E.; Pokorný, I. *J. Comput. Chem.* **2005**, *26*, 334−343.
(23) Klenin, K. V.; Tristram, F.; Strunk, T.; Wenzel, W. *J. Comput. Chem.* **2011**, *32*, 2647−2653.
(24) Wodak, S. J.; Janin, J. *Proc. Natl. Acad. Sci. U. S. A.* **1980**, *77*, 1736−1740.
(25) Sridharan, S.; Nicholls, A.; Sharp, K. A. *J. Comput. Chem.* **1995**, *16*, 1038−1044.
(26) Weiser, J.; Shenkin, P. S.; Still, W. C. *J. Comput. Chem.* **1999**, *20*, 217−230.
(27) Rychkov, G.; Petukhov, M. *J. Comput. Chem.* **2007**, *28*, 1974−1989.
(28) Lee, B.; Richards, R. M. *J. Mol. Biol.* **1971**, *55*, 379−400.
(29) Steinbach, P. J.; Brooks, B. R. *J. Comput. Chem.* **1994**, *15*, 667−683.
(30) Samudrala, R.; Levitt, M. *Proteins* **2000**, *9*, 1399−1401.
(31) Rueda, M.; Ferrer-Costa, C.; Meyer, T.; Perez, A.; Camps, J.; Hospital, A.; Gelpi, J. L.; Orozco, M. *Proc. Natl. Acad. Sci. U. S. A.* **2007**, *104*, 796−801.