Title
TRIFORCE: Tessellated Semianalytical Solvent Exposed Surface Areas and Derivatives.

Permalink
https://escholarship.org/uc/item/078382jc

Journal
Journal of chemical theory and computation, 10(9)

ISSN
1549-9618

Authors
Drechsel, Nils JD
Fennell, Christopher J
Dill, Ken A
et al.

Publication Date
2014-09-01

DOI
10.1021/ct5002818

Peer reviewed
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 approaches18–23 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,
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 $S_i$ 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

$$A = \sum_l A^{(l)}$$  \hspace{1cm} (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 $S_i$
that intersect with $S_0$. For example, circular interface $I_l$ is located
in the plane of intersection between $S_l$ and $S_j$ where the hulls
touch each other. An intersection point $p$ exists between
circular interfaces whenever two interfaces intersect on the
exposed boundary. Two consecutive intersection points $p_k$ and
$p_m$ on an exposed boundary, in conjunction with a tessellation
point $\chi$, form the three points of a triangular patch $A^{(l)}_{jk}$. $\chi$ is the
point on $S_l$ where the tessellation axis $\chi$ intersects. $\chi$ is an
arbitrary vector for each atom fixed for the duration of the
calculations.

Areas $A^{(l)}_{jk}$ are calculated on a unit sphere and therefore have
to be multiplied with the squared radius $r_i$ of sphere $S_i$

$$\hat{A}^{(l)} = \sum_{(jk)\in\text{Seg}_l} A^{(l)}_{jk} r_i^{-2}$$  \hspace{1cm} (2)

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

Patches $A^{(l)}_{jk}$ 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).

In addition, the interface segments of a patch $A^{(l)}_{jk}$ 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 $A^{(l)}_{jk}$ 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 $r_i$ to this negative $\hat{A}^{(l)}$.
Figure 4 shows the principal angles $\Phi$, $\psi$, and $\lambda$ 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 $\chi$ and the intersphere vector. Here, $\Phi$ is the angle between the tessellation plane and a vector from the center of $I_j$ to a $p$. The angle $\psi_j$ is between the tessellation axis and the center of $I_j$. Finally, $\lambda_j$ 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 $T(\Phi,\psi,\lambda)$. $\Phi$ 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 $A_{ijk}(l)$.

From now on, whenever unambiguous, we omit the index ($l$) for simplicity. Throughout the work we will use the symbol $\cdot$ to denote the dot product, symbol $\otimes$ to denote the outer vector product, and symbol $\times$ 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 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:

$\nu_i = x_i - x_l$

$\nu_i = |\nu_i|$

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

where $\nu_i$ is a vector of length $d_i$ between Cartesian coordinates $x_i$ and $x_l$ of spheres $S_i$ and $S_l$. (2) Interface type $f_i$, which is
either positive for convex or negative for concave interfaces, is given by

\[ f_i = \text{sign}(g_i) \]  

(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:

\[ g_i = |g_i| \]  

(8)

Angular calculations are performed on unit vectors:

\[ \mu_i = f_i \frac{\mathbf{u}_i}{d_i} \]  

(9)

which include the radial distances between two interfaces \( I_i \) and \( I_j \):

\[ \rho_{ij} = \arccos(\mu_i \cdot \mu_j) \]  

(10)

as well as the opening angle to the interface,

\[ \lambda_i = \arccos(g_i) \]  

(11)

and the angle to the tessellation axis,

\[ \psi_i = \arccos(\chi_i \cdot \mu_i) \]  

(12)

Multiple circular interfaces will intersect if the sum of their conical opening angles \( \lambda_i \), and \( \lambda_j \), fall below their radial distance \( \rho_{ij} \), and if one does not entirely contain the other. The intersection between these circular interfaces will then result in two intersection points \( p_j \) and \( p_k \). 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. \( \Phi \) angles are calculated with respect to a specific interface, which is illustrated in Figure 6. Here, \( \Phi \) angles for \( p_j \)

Figure 6. Intersections between circular interfaces \( I_p, I_o, \) and \( I_k \) on sphere \( S_i \) resulting in intersection points \( p_o, p_p, p_j, \) and \( p_k \). \( \Phi \) angles are calculated with respect to each interface’s tessellation plane. Here, we show all of the \( \Phi \) angles for interface \( I_p \). Following the path indicated by the arrows, the SASA will be on the right-hand side. This path direction makes \( p_i \) an incoming intersection point of \( I_p \) and \( p_k \) an outgoing intersection point.

differ with respect to interface \( I_p \) or \( I_o \). The curved arrows over the \( \Phi \) indices represent the direction of the intersection. Seen from \( I_p \), \( p_o \) is viewed as an incoming point (connecting \( I_p \) with \( I_o \)) and \( p_k \) as an outgoing point (connecting \( I_p \) with \( I_k \)). The respective angles for interface \( I_k \) are an incoming angle \( \Phi_{ik}^a \) and an outgoing angle \( \Phi_{ik}^o \).

Figure 7 illustrates how these incoming and outgoing angles are calculated utilizing the previously introduced quantities for circular interfaces. First, \( \eta_{ij} \) is derived from the two opening angles \( \lambda_i \) and \( \lambda_j \), and the angle between the two interface vectors \( \rho_{ij} \):

\[ \eta_{ij} = \arccos(\cot(\lambda_i) \cot(\lambda_j) - \cos(\lambda_i) \csc(\lambda_j) \csc(\lambda_j)) \]  

(13)

\( \eta_{ij} \) is the opening angle of the intersection between the interfaces. Next, a normal vector \( \mathbf{n}_o \) to the tessellation plane is established:

\[ \mathbf{v}_i = \chi \times \mathbf{\mu}_i \]  

(14)

\[ \mathbf{n}_o = \frac{\mathbf{v}_i}{|\mathbf{v}_i|} \]  

(15)

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

\[ \mathbf{v}_o = \mathbf{\mu}_i \times \mathbf{\mu}_j \]  

(16)

\[ \mathbf{n}_o = \frac{\mathbf{v}_o}{|\mathbf{v}_o|} \]  

(17)

in which again \( \mathbf{v}_o \) describes the unnormalized vector. The angle between these two normal vectors determines the rotation of \( I_j \) around \( I_i \):

\[ \eta = \arccos(\mathbf{n}_i \cdot \mathbf{n}_o) \]  

(18)

Whether this rotation is to the left or to the right depends on the orientation of \( \chi \) with respect to the interfacial plane of \( I_i \) and \( I_j \):

\[ q_{ij} = -\text{sign}(\mathbf{n}_j \cdot \mathbf{\mu}_i) \]  

(19)

resulting in the signed rotation:

\[ \omega_{ij} = q_{ij} \eta_{ij} \]  

(20)

\( \eta_{ij} \) and \( \omega_{ij} \) need to be combined into both outgoing \( \Phi_{ij}^o \) and incoming \( \Phi_{ij}^a \) angles:

\[ \Phi_{ij}^o = \chi + \omega_{ij} + f_j \eta_{ij} \]  

(21)
\[
\Phi_i^j = \gamma_i^j + \omega_j - f_j \eta_j
\]  
(22)
with \(\gamma_i^j\) being either zero or \(\pi\) and \(\gamma_i^j\) being either zero or \(-\pi\), indicating whether the complement of \(\omega\) needs to be calculated. Both quantities depend on the rotation of \(I_j\) around \(I_i\) and the magnitude of \(\eta_j\):

\[
\gamma_i^j = \pi (\text{sign}(\pi - \omega_j - \eta_j) - 1)
\]  
(23)
\[
\gamma_i^j = -\pi (\text{sign}(-\pi - \omega_j + \eta_j) - 1)
\]  
(24)

Utilizing the preceding equations, we can look-up the areas of two subpatches of a triangular patch \(A_{ik}^{(0)}\) from precomputed tables \(T(\Phi_i^j, \psi_j, \lambda_j)\) and \(T(\Phi_i^j, \psi_j, \lambda_j)\). 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 \(q_{ik}\)), the complementary area needs to be calculated:

\[
q_{ik} = \text{sign}(\Phi_{ik}^j - \Phi_{i}^j)
\]  
(25)
\[
M_j = T(\pi, \psi_j, \lambda_j)
\]  
(26)
\[
A_{ik}^{(1)} = -f_j(M_j(q_{ik}) - 1) - T(f_j(\Phi_{ik}^j, \psi_j, \lambda_j)) + T(f_j(\Phi_{ik}^j, \psi_j, \lambda_j))
\]  
(27)

where \(M_j\) is the maximal area of a triangular patch for a given interface \(I_j\) specified by \(\psi_j\) and \(\lambda_j\).

3. EXPOSED BOUNDARY SEGMENTS

Each sphere can possibly have multiple contributions to \(A_{ik}^{(0)}\) 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 \(p_i\) and \(p_{ik}\). Segments containing \(p_i\) belong to the exposed boundary if, generally speaking, they are not covered by any other interface.

To calculate the three principal angles \(\Phi, \psi, \text{and} \lambda\) 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, \(\Phi\) 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 \(\Phi\) 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.

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 interactions.

4.1. Contact Point between Two Spheres. Imagine the following scenario for which key quantities are shown in Figure 9: two nonintersecting atoms \(S_i\) and \(S_j\) 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. To compensate, we smooth the force and area around the contact point with a logistic function $\Lambda(t)$. Figure 9 illustrates the effect of $\Lambda(t)$. 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 $\Lambda(t)$ using a term $t_i$ which depends on the distance between the spheres:

$$\overline{A}_{ij}(t) = \Lambda(t_i) A_{ij}(t)$$

(28)

$$t_i = p_0 \left( 1 - \frac{d_i}{r_i + \eta} \right)$$

(29)

$$\Lambda(x) = \frac{1}{1 + \exp(-2x)}$$

(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 $\psi$, $\lambda$, and $\Phi$ (which involves $\eta$ and $\omega$) lead to the calculation of fractional terms in which the denominator vanishes once the angles approach either 0 or $\pi$. This gives rise to numerically anomalous very large (or infinite) forces. $\lambda$ and $\eta$ are two angles that can describe extreme geometrical constellations: a very small $\lambda$ refers to two spheres that just barely touch, a very small $\eta$ refers to two interfaces that just barely touch, and a very large $\eta$ 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.

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

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

$$\tau_i = \arccos(n_i^* \cdot n_i)$$

(31)

The original $\omega_i$ can be reconstructed:

$$\omega_i = \frac{1}{2}(\theta_i - \theta_i q_i + q_i \theta_i q_i - q_i \theta_i) + q_i \theta_i q_i$$

(32)

where the relationship between the new and old tessellation planes depend on the geometrical constellation of $\chi^*$ and $n_i$. $q_0$ indicates if the tessellation plane and the interfacial plane of $I_i$ and $I_j$ are similarly orientated with respect to the tessellation axis:

$$q_0 = \text{sign}(\chi \cdot n_i) \cdot \text{sign}(\chi' \cdot n_i)$$

(33)

$q_1$ and $q_2$ determine the angular relationship between $\tau_i$ and $\omega_i^*$, and whether the complement of $\omega_i^*$ needs to be computed:

$$q_1 = \begin{cases} -1 & (q_0 \tau_i > \omega_i) \land (q_0 \tau_i > q_0 \omega_i) \\ 1 & \text{else} \end{cases}$$

(34)

$$q_2 = \begin{cases} -1 & (q_0 \tau_i > \omega_i) \land (q_0 \tau_i < q_0 \omega_i) \\ 1 & \text{else} \end{cases}$$

(35)

The advantage of eq 32 is that it does not depend on the possibly singular angle between $n_i$ and $n_i^*$ instead, it depends on the guaranteed nonsingular angles $\omega_i^*$ and $\tau_i$ 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 $\lambda$ and $\psi$ is equal to $\pi$ or if it exceeds $\pi$. During the calculation of the integration tables, we map $\Phi$ angles to $\phi$ angles, which represent longitudinal lines that pass through the same intersection point (see the Appendix for a more detailed explanation of $\phi$ angles). The former case would map one $\Phi$ angle to an infinite number of $\phi$ angles. The latter would map $\Phi$ angles to $\phi$ angles in such a way that close to the singularity the look-up tables would be too sparse. This gives rise to impricse areas and derivatives. To handle this, we split each sphere into two independently tessellated hemispheres. As a result, no $\Phi$ 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. 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 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 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

| PDB code | heavy atoms | LCPO | TRIFORCE |
|----------|-------------|------|----------|
|          |             | 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   |
| 1IF6     | 436         | 3.8105 | 0.0486   | 0.0003   |
| 4PT1     | 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   |
| 1FQV     | 545         | 5.7127 | 0.0146   | 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   |
| 1JS1     | 721         | 4.9532 | 0.0614   | 0.0031   |
| 2CDV     | 801         | 5.4789 | 0.0316   | 0.0020   |
| 1OFC     | 805         | 5.0891 | 0.0361   | 0.0039   |
| 1KTE     | 818         | 5.4651 | 0.0403   | 0.0021   |
| 1NSO     | 858         | 4.2843 | 0.0275   | 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        | 0.1128 | 0.0012   |
| 1LIT     | 1045        | 3.4352 | 0.0219   | 0.0012   |
| 1B7      | 1208        | 3.1121 | 0.0254   | 0.0015   |
| 2IB      | 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        | 5.4146 | 0.0344   | 0.0012   |
| 2PTN     | 1629        | 4.6474 | 0.0286   | 0.0013   |
| 1SPAD    | 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        | 7.9677 | 0.0455   | 0.0034   |
| 2TMN     | 2432        | 7.6176 | 0.0414   | 0.0031   |
| 2TSS     | 2457        | 3.8305 | 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

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 tessellation axis, as the measured fluctuations when rotating the molecule and/or the tessellation 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 (GETAREA) values. We show the TRIFORCE results as a function of interpolation grid detail. With an extremely coarse grid of $3 \times 3 \times 3$, 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% integration time was 60 ms. Grid details of 64 and 80 have average integration times of 70 ms, a greater than 15% performance penalty.

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.

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 ($C_{15}H_{10}$) 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
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 \( \Xi \) 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 \( \Phi, \psi, \lambda \) and \( \lambda \) values. \( \psi \) and \( \lambda \) 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 \( \Omega \) and integrating over \( \theta \) and \( \varphi \):

\[
A = \int_{ \varphi_{0} }^{ \varphi_{f} } \int_{ 0 }^{ \Phi(\varphi, \psi, \lambda) } \psi \sin(\theta) \, d\theta \, d\psi \sin(\theta) \left| \frac{\partial \Omega}{\partial \theta} \times \frac{\partial \Omega}{\partial \psi} \right| \quad (36)
\]

\( \theta \) is the radial length of an arc drawn between the tessellation point and the boundary of a virtual interface. \( \varphi \) 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 \( \theta \) depend on the parametrization of the interface, as well as the rotation \( \varphi \), and is given by

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

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 \( \mathbf{x}_i \) and the change of coordinates of intersecting atoms \( \mathbf{x}_i^{(0)} \). For the sake of readability, we omit
functions of

\[ \frac{\partial A}{\partial x_5} = \sum_i \frac{\partial A^{(i)}}{\partial x_5} \]  

(38)

If \( \delta \notin \text{Int} \) or \( \delta \neq I \), the derivative will be zero. The derivatives of each atom are comprised by derivatives from triangular patches:

\[ \frac{\partial A^{(i)}}{\partial x_5} = r_i^2 \sum_{(i\mu) \in \text{Seg}_5} \frac{\partial A^{(i)}}{\partial x_5} \]  

(39)

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

\[ \frac{\partial A^{(i)}}{\partial x_5} = - (q_{\delta i} - 1) \frac{\partial M_i}{\partial x_5} + \left( \frac{\partial T(P_k)}{\partial x_5} - \frac{\partial T(P_j)}{\partial x_5} \right) \]  

(40)

With \( P_k = (\Phi_{jk}^i, \psi_{jk}, \lambda_j) \) and \( P_j = (\Phi_{ij}^k, \psi_{ij}, \lambda_i) \).

8.2.2. Integration Tables. The integration tables are functions of \( \psi, \lambda, \Phi \), and their derivatives are split accordingly:

\[ \frac{\partial T(P_i)}{\partial \Phi_{ij}} = G^{(\text{ab})}(P_i) \]  

(42)

\[ \frac{\partial T(P_j)}{\partial \psi_{ij}} = G^{(\text{ab})}(P_j) \]  

(43)

\[ \frac{\partial T(P_k)}{\partial \lambda_j} = G^{(\text{ab})}(P_k) \]  

(44)

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

\[ \frac{\partial T(P_i)}{\partial \Phi_{ij}} = G^{(\text{ab})}(P_i) \]  

(45)

\[ \frac{\partial T(P_j)}{\partial \psi_{ij}} = G^{(\text{ab})}(P_j) \]  

(46)

\[ \frac{\partial T(P_k)}{\partial \lambda_j} = G^{(\text{ab})}(P_k) \]  

(47)

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

8.2.3. Angle \( \psi \). Derivatives for angle \( \psi_i \), illustrated in Figure 4 and eq 12, are

\[ \frac{\partial \psi_i}{\partial x_5} = \frac{\partial \psi_i}{\partial x_5} \]  

(49)

\[ \frac{\partial \psi_i}{\partial x_5} = \frac{\partial \psi_i}{\partial x_5} \]  

(50)

\[ \frac{\partial \psi_i}{\partial x_5} = \frac{\partial \psi_i}{\partial x_5} \]  

(51)

\[ \frac{\partial \psi_i}{\partial x_5} = \frac{\partial \psi_i}{\partial x_5} \]  

(52)

\[ \frac{\partial \psi_i}{\partial x_5} = \frac{\partial \psi_i}{\partial x_5} \]  

(53)

\[ \frac{\partial \psi_i}{\partial x_5} = \frac{\partial \psi_i}{\partial x_5} \]  

(54)

8.2.5. Angle \( \Phi \). Derivatives for angle \( \Phi \), as illustrated in Figure 4 and eqs 21–22, are

\[ \frac{\partial \Phi_{ij}}{\partial x_5} = \frac{\partial \Phi_{ij}}{\partial x_5} + \frac{\partial \phi_{ij}}{\partial x_5} \]  

(55)

\[ \frac{\partial \Phi_{ij}}{\partial x_5} = \frac{\partial \Phi_{ij}}{\partial x_5} + \frac{\partial \phi_{ij}}{\partial x_5} \]  

(56)

8.2.6. Angle \( \eta \). Derivatives for angle \( \eta \), as illustrated in Figure 7 and eq 13, are

\[ \frac{\partial \eta_{ab}}{\partial x_5} = \frac{\partial \eta_{ab}}{\partial x_5} + \frac{\partial \phi_{ab}}{\partial x_5} \]  

(57)

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

\[ \frac{\partial \eta_{ab}}{\partial x_5} = \frac{\partial \eta_{ab}}{\partial x_5} \]  

(58)

\[ \frac{\partial \eta_{ab}}{\partial x_5} = \frac{\partial \eta_{ab}}{\partial x_5} \]  

(59)

\[ \frac{\partial \eta_{ab}}{\partial x_5} = \frac{\partial \eta_{ab}}{\partial x_5} \]  

(60)

8.2.7. Angle \( \rho \). Derivatives for angle \( \rho \), as illustrated in Figure 5 and eq 10, are

\[ \frac{\partial \rho_{ab}}{\partial x_5} = \frac{\partial \rho_{ab}}{\partial x_5} \]  

(61)

\[ \frac{\partial \rho_{ab}}{\partial x_5} = \frac{\partial \rho_{ab}}{\partial x_5} \]  

(62)

The following identity applies: \( \rho_{ab} = \rho_{ab} \).

8.2.8. Angle \( \omega \). Derivatives for angle \( \omega \), as illustrated in Figure 7 and eq 20, are

\[ \frac{\partial \omega_{ab}}{\partial x_5} = \frac{\partial \omega_{ab}}{\partial x_5} \]  

(63)
interface

The logistic smoother is applied to the area term by multiplication which creates the following additional terms:

\[
\frac{\partial \sigma_{ab}}{\partial x_i} = \frac{\partial \sigma_{ab}}{\partial n_i} + \frac{\partial \sigma_{ab}}{\partial n_{ab}}
\]

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

\[
\frac{\partial \sigma_{ab}}{\partial n_i} = \frac{n_i}{1 - (n_i \otimes n_{ab})^2}
\]

Both addends can be calculated using eqs 68–74.

\[
\frac{\partial \sigma_{ab}}{\partial n_{ab}} = \frac{1}{d_{ab}} (I - n_i \otimes n_{ab})^T
\]

\[
\frac{\partial \sigma_{ab}}{\partial \mu_i} = I \times \mu_i
\]

\[
\frac{\partial \sigma_{ab}}{\partial \mu_{ab}} = -I \times \mu_i
\]

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:

\[
\frac{\partial \Lambda^{(t)}}{\partial x_i} = \Lambda(t) \frac{\partial \Lambda^{(t)}}{\partial x_i} + \frac{\partial \Lambda(t)}{\partial x_i} \Lambda^{(t)}
\]

\[
\frac{\partial \Lambda^{(t)}}{\partial \mu_i} = \frac{\partial \Lambda(t)}{\partial \mu_i} \frac{\partial \Lambda^{(t)}}{\partial \mu_i}
\]

\[
\frac{\partial \Lambda(t)}{\partial \mu_i} = \frac{1}{1 + \cosh(p_1 - 2xp_1)}
\]

\[
\frac{\partial \mu_i}{\partial x_i} = \frac{\mu_i p_i}{r_i + r_i}
\]

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:

\[
\frac{\partial \sigma_{ab}}{\partial x_i} = q_{ab} \frac{\partial \tau_i}{\partial x_i} + q_{ab} \frac{\partial \sigma_{ab}}{\partial x_i}
\]

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 additional authors 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.; 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.; Keohoe, 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. Protams: Struct., Funct., Genet. 1999, 35, 153–152.
18. Connolly, M. L. J. Appl. Crystallogr. 1983, 16, 548–558.
19. Richmond, T. 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.; Srivánek, J.; Hayryan, E.; Pokorny, 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.