Derivatives of Molecular Surface Area and Volume: Simple and Exact Analytical Formulas

KONSTANTIN V. KLENIN,[1,2] FRANK TRISTRAM,[3] TIMO STRUNK,[3] WOLFGANG WENZEL[2,3]

[1]*Steinbuch Center for Computing, Karlsruhe Institute of Technology, P.O. Box 3640, D-76021 Karlsruhe, Germany*

[2]*Center for Functional Nanostructures, Karlsruhe Institute of Technology, P.O. Box 6980, D-76049 Karlsruhe, Germany*

[3]*Institute of Nanotechnology, Karlsruhe Institute of Technology, P.O. Box 3640, D-76021 Karlsruhe, Germany*

Received 10 January 2011; Revised 26 April 2011; Accepted 26 April 2011
DOI 10.1002/jcc.21844
Published online 8 June 2011 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: The computational effort of biomolecular simulations can be significantly reduced by means of implicit solvent models in which the energy generally contains a correction depending on the surface area and/or the volume of the molecule. In this article, we present simple derivation of exact, easy-to-use analytical formulas for these quantities and their derivatives with respect to atomic coordinates. In addition, we provide an efficient, linear-scaling algorithm for the construction of the power diagram required for practical implementation of these formulas. Our approach is implemented in a C++ header-only template library.

© 2011 Wiley Periodicals, Inc. J Comput Chem 32: 2647–2653, 2011

Key words: molecular modeling; implicit solvent models; solvent accessible surface area; molecular volume; power diagram

Introduction

In the last decade, great progress has been made in all-atom computer simulations of biological macromolecules, such as proteins and nucleic acids. This progress was possible not only because of the enormous increase of computer power available but also because of improvement of the models used in the simulations. In particular, force fields were optimized to provide better agreement with experiment, even though difficulties still remain to describe large scale conformational changes. One attractive choice to reduce the computational effort has been the use of an implicit solvent model, which permits elimination of many degrees of freedom associated with the surrounding water molecules and ions. In this model, the behavior of the solute macromolecule is governed by a potential of mean force that can be formally achieved by averaging over all possible conformations of the molecules of solvent. Practically, the solute macromolecule is considered to be immersed into an infinite homogeneous medium characterized by a certain dielectric constant and ionic strength. An additional advantage is that the problems related to periodic boundary conditions can be avoided in such simulations. These problems include not only physical limitations (self-interactions, external potentials, etc.) but also increasingly computational difficulties, because efficient evaluation of electrostatic interactions with periodic boundary conditions has been challenging on emerging hybrid computational architectures, such as graphics accelerators or the Cell processor, which are expected to play an ever increasing role in novel emerging computational architectures.

In the implicit solvent models, there is a well-defined boundary between the simulated solute macromolecule and the surrounding medium. Each atom is treated as a ball of a certain radius and the whole macromolecule as a union of such balls. The solvation energy is then usually decomposed into three contributions. (1) The first term is the energy of creation of a cavity in the solvent. This term is closely related to the excluded volume of the solute macromolecule, i.e., the volume that becomes inaccessible to the centers of mass of the solvent molecules. For evaluation of this volume, the radii of the balls should be put to the van der Waals radii of the corresponding atoms augmented by the effective radius of the water molecule ($\sim$1.4 Å). (2) The second term accounts for the energy of the van der Waals interactions and the hydrophobic/hydrophilic effects. The contribution to this term from each atom is proportional to the area of its surface that is exposed to the solvent (also called the solvent

Correspondence to: W. Wenzel; e-mail: wolfgang.wenzel@kit.edu.

Contract/grant sponsor: Deutsche Forschungsgemeinschaft, Center for Functional Nanostructures; contract/grant number: C5.1

accessible surface area, SASA).[1] More precisely, this area can be defined as the uncovered surface area of the corresponding ball, with the ball radii being specified above. (3) The third term is a correction to the electrostatic energy. It can be estimated either by solution of the Poisson-Boltzmann equation or by the generalized Born approximation, which is less accurate but also less time-consuming.[2]

Therefore, in many currently used force fields, the energy explicitly depends on the surface area of individual atoms or groups and/or the volume of a solute macromolecule.[3–10] For some simulation methods, such as Monte Carlo, it is enough to be able to evaluate the energy as a function of the atomic coordinates. For others, like molecular dynamics, evaluation of the forces acting on the atoms is required. Hence, it is important to have efficient algorithms for computation not only of the surface area and volume but also of the derivatives of these quantities with respect to atomic coordinates as well.

In the last years, several exact[11–26] and approximate[27–42] solutions of this problem have been published (see Ref. 43 for review). Numerical integrations of the surface or volume are straightforward, but involve an enormous number of integration points. Accurate numerical estimates of derivatives often require computational costs that are so large as to make the use of implicit solvent models inefficient. More practical approaches mostly rely on relatively simple approximate analytical expressions for these quantities, which can be easily differentiated with respect to the atomic coordinates. However, in this case, the errors arising in implicit solvent simulations stem not only from the physical approximations underlying the model but also from the approximations intrinsic to the evaluation procedures. Regarding the exact methods, most of the existing works followed a strategy where an analytical expression for the surface or volume was obtained first. The derivatives are then obtained by application of the standard differentiation rules. In the following, we show that the analytical expressions for the derivatives can be found much easier than for the surface and volume themselves.

The implementation of our approach is based on the power diagram of the solute macromolecule. In this respect, our approach is similar to that of Edelsbrunner et al.[25,44–46] who developed the software package PROGEOM[47] for calculation of molecular surface, volume, and their derivatives. However, our derivations and final formulas are much simpler, and our software is essentially more efficient in terms of central processing unit (CPU) time, accuracy, and stability.

Surface Area and Its Derivatives

Consider a molecule as a union of overlapping balls (atoms) of different radii. Let us select a single arbitrary atom. We denote its radius by ρ and the position of its center by $\boldsymbol{r}$. The buried area of its surface has a boundary that consists of the arcs resulting from the intersection with the neighboring atoms. Consider a single contour formed by n such arcs as illustrated in Figure 1. We denote the arcs by $a_1, a_2, \ldots, a_n$, with the indices increasing as one moves round the buried area in an counterclockwise manner. For simplicity, we ascribe the same indices to the corresponding neighboring atoms (note that, in general, one atom

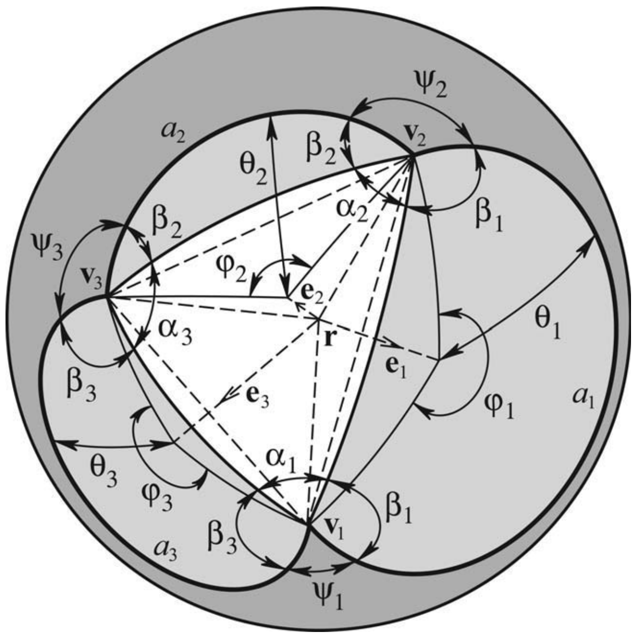

Figure 1. The surface of the selected atom (example). The buried area is shown in white and light-gray. The arcs forming the boundary are represented by the thick solid lines. The thinner solid lines (without arrows) are geodesics. The dashed lines are straight segments that do not belong to the surface.

may obtain several different indices). The problem is to find the derivatives $dS/d\boldsymbol{r}_i$, where S is the accessible area of the selected atom and $\boldsymbol{r}_i$ is the center of the ith neighbor ($i = 1, 2, \ldots, n$).

Let $\boldsymbol{e}_i$ be the unit vector in the direction to the ith neighbor (i.e., from the point $\boldsymbol{r}$ to the point $\boldsymbol{r}_i$). We will first calculate the component of the gradient $dS/d\boldsymbol{r}_i$ that is parallel to this vector. The total area of the cap covered by the ith atom is

$$S_i^{(\mathrm{cap})} = 2\pi(1-\cos\theta_i)\cdot\rho^2, \tag{1}$$

where θ_i is the angular radius of the cap. From a simple geometrical consideration, it can be easily obtained that

$$\cos\theta_i = \frac{1}{2\rho}\left(b_i + \frac{\rho^2-\rho_i^2}{b_i}\right), \tag{2}$$

where ρ_i is the radius of the ith neighbor and b_i is the distance to it (i.e., the distance between the points $\boldsymbol{r}$ and $\boldsymbol{r}_i$). Let φ_i be the central angle of the arc a_i. By an infinitesimal increment of b_i, the fraction of the cap boundary contributing to the change of the buried area is equal to $\varphi_i/2\pi$. Thus, using eqs. (1) and (2), we get

$$\frac{dS}{d\boldsymbol{r}_i}\boldsymbol{e}_i = -\frac{\varphi_i}{2\pi}\frac{dS_i^{(\mathrm{cap})}}{db_i} = \frac{\varphi_i\rho}{2}\left(1+\frac{\rho^2-\rho_i^2}{b_i^2}\right). \tag{3}$$

An infinitesimal displacement $d\boldsymbol{r}_i$ of the ith neighbor in the direction perpendicular to $\boldsymbol{e}_i$ can be treated as a rotation by the angle

$$d\gamma_i = (\boldsymbol{e}_i \times d\boldsymbol{r}_i)/b_i \tag{4}$$

about the central point $\boldsymbol{r}$. If the surface tension of the selected atom is σ, then

$$\boldsymbol{T}_i d\gamma_i = -\sigma dS, \tag{5}$$

where $\boldsymbol{T}_i$ is the corresponding torque. Substituting Eq. (4) into Eq. (5) and changing the order of multipliers in the triple product, we get

$$d\boldsymbol{r}_i(\boldsymbol{T}_i \times \boldsymbol{e}_i) = -\sigma b_i dS. \tag{6}$$

Now, we will make use of the fact that the torque acting on any line on the ball surface depends only on the end points of the line but not on its particular shape. We will find $\boldsymbol{T}_i$ as the torque acting on the geodesic that connects the end points $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$ of the arc a_i. For symmetry reasons, the vector $\boldsymbol{T}_i$ points to the same direction as the vector $(\boldsymbol{v}_{i+1} - \boldsymbol{v}_i)$ (if $i = n$, the index $i + 1$ should be substituted by 1). If we rotate the geodesic by the full angle 2π about the axis that passes through the ball center and is parallel to $\boldsymbol{T}_i$, then the swept area is $4\pi\rho^2 \cdot |\boldsymbol{v}_{i+1} - \boldsymbol{v}_i|/2\rho = 2\pi\rho|\boldsymbol{v}_{i+1} - \boldsymbol{v}_i|$. If the angle of rotation is $d\gamma_i$, then the swept area, corrected by the factor of $d\gamma_i/2\pi$, becomes $dS = \rho|\boldsymbol{v}_{i+1} - \boldsymbol{v}_i|d\gamma_i$. Hence, we have:

$$\boldsymbol{T}_i = -\sigma dS/d\gamma_i = \sigma\rho(\boldsymbol{v}_{i+1} - \boldsymbol{v}_i). \tag{7}$$

After the substitution into Eq. (6) we get

$$dS = -(\rho/b_i)[(\boldsymbol{v}_{i+1} - \boldsymbol{v}_i)\times\boldsymbol{e}_i]d\boldsymbol{r}_i, \quad d\boldsymbol{r}_i \perp \boldsymbol{e}_i, \tag{8}$$

Combining eqs. (3) and (8), we obtain finally

$$\frac{dS}{d\boldsymbol{r}_i} = \frac{\varphi_i\rho}{2}\left(1 + \frac{\rho^2 - \rho_i^2}{b_i^2}\right)\boldsymbol{e}_i - \frac{\rho}{b_i}[(\boldsymbol{v}_{i+1} - \boldsymbol{v}_i)\times\boldsymbol{e}_i]. \tag{9}$$

Note that if two different indices i and j correspond to the same neighboring atom then this formula just provides the additive contributions from the arcs a_i and a_j. The generalization to the case when the boundary of the buried area consists of several contours is straightforward. If a single neighboring atom gives rise to k arcs possibly belonging to different contours, the total derivative is the sum of k terms, each of which is given by Eq. (9). It is also obvious that the contribution from one contour to the derivative $dS/d\boldsymbol{r}$ (with respect to the position of the selected atom) is given by

$$\frac{dS}{d\boldsymbol{r}} = -\sum_{i=1}^{n}\frac{dS}{d\boldsymbol{r}_i}. \tag{10}$$

Now, we will provide an elementary derivation of the Connolly formula[11] for the accessible area S. At first, we assume that the boundary consists of a single contour as illustrated in Figure 1. The buried surface can be represented as the union of the central polygon with the vertices $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ (white area in Fig. 1) and the caps truncated by the sides of this polygon (light-gray areas in Fig. 1). The polygon area is

$$S^{(\text{polygon})} = \left[\sum_{i=1}^{n}\alpha_i - \pi(n-2)\right]\cdot\rho^2. \tag{11}$$

Here and further the notations are as shown in Figure 1. The area of the ith truncated cap can be found as the area of the sector with the central angle φ_i [compare Eq. (1)]

$$S_i^{(\text{sector})} = (\varphi_i/2\pi)S_i^{(\text{cap})} = \varphi_i(1 - \cos\theta_i)\cdot\rho^2 \tag{12}$$

plus (if $\varphi_i > \pi$) or minus (if $\varphi_i < \pi$) the area of the triangle with the vertices $\boldsymbol{v}_i$, $\boldsymbol{v}_{i+1}$, and the center of the ith cap. For example, the area of the first triangle with $\varphi_1 > \pi$ (Fig. 1) is

$$\begin{aligned} S_1^{(\text{triangle})} &= [(2\pi - \varphi_1) + 2(\beta_1 - \pi/2) - \pi]\cdot\rho^2 \\ &= (2\beta_1 - \varphi_1)\cdot\rho^2. \end{aligned} \tag{13}$$

The area of the second triangle with $\varphi_2 < \pi$ is

$$S_2^{(\text{triangle})} = [\varphi_2 + 2(\pi/2 - \beta_2) - \pi]\cdot\rho^2 = (\varphi_2 - 2\beta_2)\cdot\rho^2. \tag{14}$$

In both cases, the area of a truncated cap is given by

$$S_i^{(\text{tr.cap})} = (2\beta_i - \varphi_i\cos\theta_i)\cdot\rho^2. \tag{15}$$

Using eqs. (11) and (15), we obtain the Connolly formula[11] for the accessible surface area:

$$S = \left[2\pi + \sum_{i=1}^{n}(\varphi_i\cos\theta_i + \psi_i - \pi)\right]\cdot\rho^2, \tag{16}$$

where$\psi_i = 2\pi - \alpha_i - \beta_i - \beta_{i-1}$, as illustrated in Figure 1. The generalization for an arbitrary number of contours is obvious. One should sum up the contributions from the single contours [given by Eq. (16)] and then put the resulting value to the interval [0, $4\pi\rho^2$] by subtracting $4\pi\rho^2$, the necessary number of times.

The angle $(\pi - \psi_i)$ can be found as the angle between the vectors $[\boldsymbol{e}_{i-1}\times(\boldsymbol{v}_i - \boldsymbol{r})]$ and $[\boldsymbol{e}_i\times(\boldsymbol{v}_i - \boldsymbol{r})]$:

$$\cos(\pi - \psi_i) = \frac{\boldsymbol{e}_{i-1}\boldsymbol{e}_i - \cos\theta_{i-1}\cos\theta_i}{\sin\theta_{i-1}\sin\theta_i} \tag{17}$$

(if $i = 1$, then $i - 1$ should be substituted by n). The angle φ_i can be found as the angle between the vectors $[\boldsymbol{v}_i - (\boldsymbol{r} + \boldsymbol{e}_i\rho\cos\theta_i)]$ and $[\boldsymbol{v}_{i+1} - (\boldsymbol{r} + \boldsymbol{e}_i\rho\cos\theta_i)]$:

$$\cos\varphi_i = \frac{(\boldsymbol{v}_i - \boldsymbol{r})(\boldsymbol{v}_{i+1} - \boldsymbol{r}) - \rho^2\cos^2\theta_i}{\rho^2\sin^2\theta_i}, \tag{18a}$$

$$\sin\varphi_i = \frac{\boldsymbol{e}_i[(\boldsymbol{v}_i - \boldsymbol{r})\times(\boldsymbol{v}_{i+1} - \boldsymbol{r})]}{\rho^2\sin^2\theta_i}. \tag{18b}$$

The most difficult problem is to construct the contour of arcs and to identify the vertices $\boldsymbol{v}_i$ among all the other triple intersections of the atomic spheres. It can be done by means of the power diagram, which is the Voronoi diagram modified for the set of the balls with different radii.[18,44,45] The vertices $\boldsymbol{v}_i$ are the intersections of the selected atom surface with the edges of the corresponding power cell. An algorithm for the construction of the power diagram is given in the Appendix.

Volume and Its Derivatives

The problem now is to obtain the derivative of the total volume V of the molecule with respect to the position $\boldsymbol{r}$ of the selected atom. This derivative can be written as

$$dV/d\boldsymbol{r} = -\boldsymbol{F}/P, \tag{19}$$

where P is the external pressure and $\boldsymbol{F}$ is the corresponding force acting on the accessible surface. This force does not depend on the particular shape of the surface, but is entirely determined by its boundary. At first, we assume that the boundary is a single contour consisting of n arcs (Fig. 1). Then $\boldsymbol{F}$ can be easily found as the force acting on the auxiliary surface that is composed of the following $2n$ flat pieces: (i) the n triangles with the vertices $\boldsymbol{r}$, $\boldsymbol{v}_i$, and $\boldsymbol{v}_{i+1}$ and (ii) the n circular segments bounded by the arcs a_i and the chords connecting their end points (i.e., $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$). The force acting on the ith triangle is

$$\boldsymbol{F}_i^{(\text{triangle})} = \frac{1}{2}[(\boldsymbol{v}_i - \boldsymbol{r})\times(\boldsymbol{v}_{i+1} - \boldsymbol{r})] \cdot P. \tag{20}$$

The force acting on the ith circular segment is

$$\boldsymbol{F}_i^{(\text{circ.segm.})} = \frac{1}{2}\rho^2 \sin^2\theta_i(\varphi_i - \sin\varphi_i) \cdot P\boldsymbol{e}_i. \tag{21}$$

Thus, combining eqs. (19)–(21), we get

$$\frac{dV}{d\boldsymbol{r}} = -\frac{1}{2}\sum_{i=1}^{n}[\boldsymbol{v}_i\times\boldsymbol{v}_{i+1} + \rho^2\sin^2\theta_i(\varphi_i - \sin\varphi_i)\boldsymbol{e}_i]. \tag{22}$$

If the boundary consists of several contours, Eq. (22) provides the additive contribution from each of them.

The same auxiliary surface is very useful in the calculation of the volume itself. Outside this surface, the ball can be decomposed into the following fractions of simple geometry (we consider, again, a single contour of n arcs). (i) The fraction corresponding to the accessible surface. One can imagine it as being cut off by the straight segment with one end fixed at the ball center $\boldsymbol{r}$ and the second end running along the contour of arcs. Its volume is

$$V^{(1)} = \frac{1}{3}S\rho, \tag{23}$$

where S is given by Eq. (16). (ii) The n cones with the bases being the circular segments embraced by the arcs a_i and with the apex at the ball center $\boldsymbol{r}$. The height of the ith cone is $\rho \cos\theta_i$. The total volume of n cones can be written as an algebraic sum [compare Eq. (21)]

$$V^{(2)} = \frac{1}{6}\rho^3\sum_{i=1}^{n}\cos\theta_i\sin^2\theta_i(\varphi_i - \sin\varphi_i) \tag{24}$$

with some of the terms possibly being negative (for $\theta_i > \pi/2$). With this choice of signs, the sum $V^{(1)} + V^{(2)}$ provides the correct volume of the ball fraction outside the auxiliary surface.

If one builds the auxiliary surface for each contour on each atom and then cuts off the external fraction of the molecule, the remaining (internal) part will represent one or several polyhedra, with all the faces being the triangles. Let us connect all the vertices of the polyhedra with the space origin by straight segments, so that each face becomes the base of a tetrahedron with the apex at the space origin. Then the total volume of the internal part can be found as the algebraic sum of the volumes of these tetrahedra. The volume of a tetrahedron is positive, if the corresponding face is looking from the origin and negative otherwise. The contribution from the n faces associated with a single contour of arcs is

$$V^{(3a)} = -\frac{1}{6}\sum_{i=1}^{n}\boldsymbol{r}(\boldsymbol{v}_i\times\boldsymbol{v}_{i+1}). \tag{25}$$

To obtain the total volume of the molecule, one has to sum up eqs. (23)–(25) over all contours and over all atoms and add the results together (by summation, if the accessible surface of one atom becomes larger than $4\pi\rho^2$, it should be reduced by $4\pi\rho^2$). Note that the final result does not depend on the position of the space origin. However, this dependence is present in Eq. (25) and, for this reason, the three terms $V^{(1)} + V^{(2)} + V^{(3a)}$, summed over all contours belonging to one atom, cannot be interpreted as the volume of this atom.

For computation of the volume of an individual atom, the third term should be modified. It is natural to define the atom volume as the volume of intersection of the corresponding ball with its cell in the power diagram (described in Appendix). In this case, the term $V^{(3a)}$, summed over all contours, should be substituted by the volume of the internal part of the cell truncated by the auxiliary surfaces:

$$V^{(3b)} = \frac{1}{3}\sum_{j=1}^{m}\sigma_j(\rho\cos\theta_j), \tag{26}$$

where σ_j is the area of the jth truncated face of the cell, i.e., the area of the face fraction that lies strictly inside the auxiliary surfaces. The expression $\rho\cos\theta_j$ is equal to the distance from the face plane to the center of the atom. The summation is performed over all faces of the cell for which the absolute value of this distance is smaller than the atom radius ρ (or, in other words, for which $\cos\theta_j$ is defined). Each individual term of the sum represents the volume of a pyramid, the base of which coincides with a truncated face and the apex with the center of the atom. If the center of the atom lies outside the corresponding cell, the contribution from some terms in Eq. (26) becomes negative, which is automatically accounted for by the sign of cos θ_j. For a totally covered atom, $V^{(3b)}$ is equal to the volume of the entire cell.

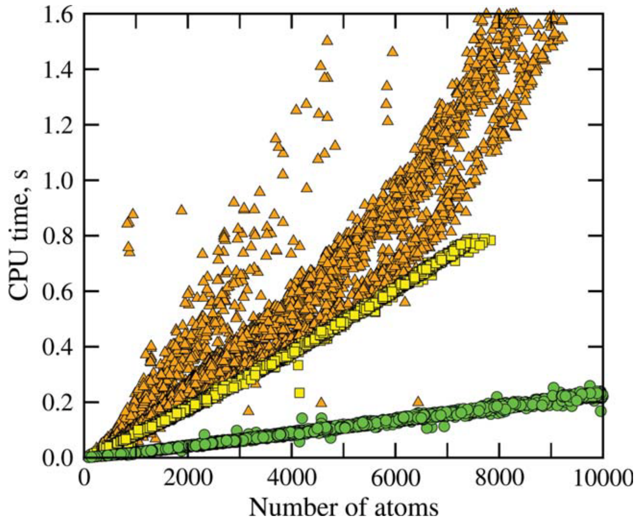

Figure 2. Benchmark for performance of POWERSASA (green circles), ASC (orange triangles), and ALPHASURF (yellow squares). The CPU time required for calculation of the SASA is plotted against the number of atoms in a protein structure. The computations were performed on a 64-bit PC with an AMD Opteron 246 processor (2.0 GHz) and 8 GB RAM running under Linux 2.6.32. The set of structures comprised the proteins from PDB released in 2010.

Thus, to obtain the volume of an individual atom, one has to sum up the terms $V^{(1)}$ and $V^{(2)}$ [eqs. (23) and (24)] over all its contours and add the term $V^{(3b)}$ [eq. (26)].

Implementation and Benchmark

We implemented the above formulas for computation of the molecular surface area, volume, and their derivatives in a C++ header-only template library POWERSASA that comprises two files: power_diagram.h and power_sasa.h. Both float and double precision are provided. At present, we work on the next versions adapted for parallel and graphics processing unit (GPU) computing. More information about this library is available from the authors on request.

As a benchmark for our software, we performed the computation of SASA for the set of all proteins in the Protein Data Bank (PDB) released in 2010 (total number 7375). The PBD files containing macromolecules other than proteins were not included. If multiple models were available, we took only the first one. When an "alternate location indicator" was present, we considered only the structure marked with "A." In the case of multiple ligand positions, the ligand was removed. The water molecules were deleted. For the atom radii, the following values were used (in angstroms): 1.8 (C), 1.2 (H), 1.5 (O), 1.6 (N), 1.75 (S), and 3.14 (others). These values were augmented by the effective radius of the water molecule 1.4 Å. The benchmark was carried out with a float precision on a 64-bit personal computer (PC) with an AMD Opteron 246 processor (2,0 GHz) and 8 GB RAM running under Linux 2.6.32. The CPU time required for one evaluation of the SASA as a function of the number of atoms in a structure, N, is shown in Figure 2.

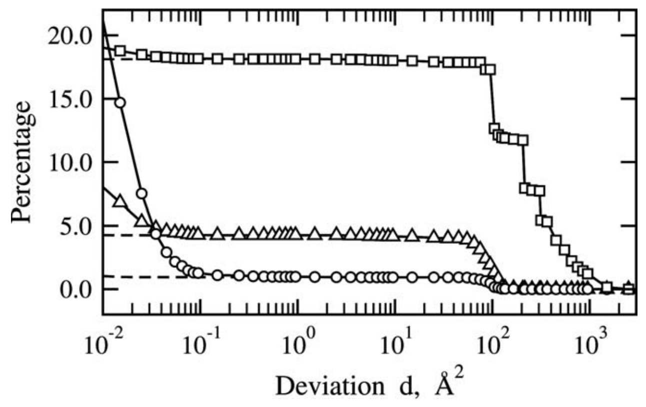

Figure 3. Percentage of calculated SASA values for which the tentative deviation exceeds the given threshold d: POWERSASA, float precision (circles), ASC (triangles), and ALPHASURF (squares). Tentative deviation was measured not with respect to the actual SASA value (which is mostly unknown), but with respect to the nearest result of the other two programs. The dashed lines correspond to the case when the POWERSASA computations were performed with the double precision.

For comparison, the results for two other programs, ASC[15,48] and ALPHASURF from the PROGEOM package,[25,47] are presented as well. For ALPHASURF, no data with $N > 7834$ are available, as the program always crashed with segmentation fault. For the lower N values, ALPHASURF crashed or entered an infinite loop in 3.9% of all the cases. ASC proved to be more stable and failed to produce a result for only 0.27% of the whole set of structures. Our software was always stable. As one can see in Figure 2, its performance is essentially better than that of the other programs. The computational time scales are practically linear with N. This linearity persists also for the larger N values not shown in Figure 2 (the largest structure in the set, of 101,798 atoms, required 2.6 s).

As all the three programs are based on exact analytical formulas, they should, in principle, provide very similar numbers for the SASA values. Indeed, in the most of the cases, the five to six leading digits of the results were identical. As a typical SASA value for a structure with ~8000 atoms is of the order of ~40,000 Å^2, the results usually differed by less than 0.1 Å^2. However, larger deviations took place sometimes. These deviations can be attributed to the wrong identified vertices $\boldsymbol{v}_i$ for some atoms. As a tentative measure d_a for accuracy of a SASA value S_a calculated by a certain program a, we took its minimal deviation from the values S_b and S_c given by the other two programs: $d_a = \min(|S_a - S_b|, |S_a - S_c|)$. Thus, if $S_a = S_b = S_c$ (all the programs give the same result), then $d_a = d_b = d_c = 0$ (it is assumed that this result is precise). If $S_a = S_b \neq S_c$ (the first two results are the same but differ from the third one), then $d_a = d_b = 0$ and $d_c > 0$ (the coinciding results are assumed to be precise, and the third one has the deviation $d_c = |S_c - S_a|$). For all the three programs, the percentage of the cases when the tentative deviation exceeds a given threshold value d is plotted versus d in the Figure 3. Note that, in most of the cases, the deviations are either less than 0.1 Å^2 or greater than 50 Å^2. For our software, the tentative deviation exceeded 1 Å^2 for 57 structures out of the total number of 5800 (when the results from all the three programs were available). It does not mean that the calculated SASA values are not correct: they just differ from the results of the other both programs. For these 57 questionable structures, we performed additional calculations of the SASA by a reliable (though very slow) Monte Carlo procedure and found out that the actual deviation of our results lies within 1 Å^2. Thereupon, we concluded that our results were, in fact, always correct. The tentative deviation registered for 57 structures was due to the simultaneous errors in the other programs.

Appendix: Algorithm for Construction of a Power Diagram

The power distance from an arbitrary point $\boldsymbol{x}$ in the three-dimensional space to atom i is defined as $(\boldsymbol{x} - \boldsymbol{r}_i)^2 - \rho_i^2$, where, as before, $\boldsymbol{r}_i$ is the center of atom i and ρ_i is its radius. The power cell of atom i is the set of all those points, for which the power distance to atom i is not larger than to any other atom. The cell faces are shared between two cells, the cell edges are shared between at least three cells, and the cell vertices are shared between at least four cells. For simplicity, we will further assume that each edge is shared by exactly three cells and each vertex is shared by exactly four cells, as this can always be achieved by slight modification of the atom radii.

The collection of all the cells defines the power diagram. In our algorithm, implemented in a C++ program, the power diagram is represented by the following objects. (1) The atoms with indication of their position, radius, and with the pointers to the vertices of their cell. (2) The vertices specified by their coordinates, the pointers to the four partner vertices that terminate the outgoing edges at the opposite ends, and the pointers to the four nearest atoms (in terms of the power distance), which we will further call generators. In addition, a vertex is characterized by the power value that is equal to the power distance to any of its generators. (3) There is also a special sort of vertices, called infinite vertices, that can be thought as the "farthest" points of those edges that stretch to infinity. Instead of position, an infinite vertex is characterized by direction. It has only three generators and no power value. However, it still has the pointers to four partner vertices: one of these vertices is finite and terminates the corresponding edge, the rest three are infinite and will be defined later.

The initial step of our algorithm is the construction of the power diagram for the first four atoms (that should not lie in the same plane). We will not describe this step in detail, as it is quite trivial. This initial power diagram contains five vertices of which only one is finite. Each vertex has pointers to the other four. Further, we add the rest of the atoms one by one, modifying the power diagram accordingly each time.

Suppose we have already constructed the power diagram for the first j atoms. After adding the next one, the modifications are carried out by the procedure that is outlined as follows.

1. First, we find and delete the vertices with the power value larger than the power distance to the new atom. An infinite vertex is deleted if the new atom lies "nearer" to it than its generators. (More precisely, this has the following meaning. The three generators of the infinite vertex define a plane. The new atom may lie either exactly on this plane or to one of the two sides from it. The infinite vertex is deleted if the new atom lies to the side specified by the vertex direction. In the case then the new atom lies exactly on the plane, the infinite vertex is deleted only if its finite partner is deleted.) The deleted vertices are put temporarily into a "trash," so that the information stored in them remains available during the next step, after which the trash is emptied.
2. Second, we construct the new finite vertices, which obviously lie on the "pending" edges, i.e., the edges with one terminal vertex survived and the other deleted. The three atoms whose cells share the pending edge and the new atom constitute the four generators for the new vertex. At this stage, for the new vertices only one pointer to a partner vertex is specified.
3. Next, we connect the new vertices by appropriate new edges. Namely, we find all those pairs of the new vertices that have three generators in common and provide the new partners with the pointers to each other.
4. After that, some pointers of the new vertices still remain vacant. They should obviously point to the new infinite vertices, which we now construct. If, among the generators of a new finite vertex, there are three that are not shared by any of its existing finite partners, then this triplet of generators defines the new infinite vertex.
5. Finally, each pair of the infinite vertices that have two generators in common is provided with the pointers to each other.

It should be noted that the search for the vertices to be deleted (step 1) is not required to be exhaustive. We start with an arbitrary finite vertex and, if it survives, jump to its finite partner that has the smallest difference between the power distance to the new atom and the power value. We iterate these jumps until we come to a vertex for which this difference is negative and find, in this way, the first vertex to be deleted. (In the case then all possible jumps would lead to an increase of the specified difference, we check if the infinite partners are to be deleted. If not, the new atom does not contribute to the power diagram.) As soon as the first vertex is found, the further search becomes trivial, because all the vertices to be deleted can be reached by jumps from one such vertex to another. The search efficiency improves when the starting vertex belongs to a cell of the atom that is covalently bound to the new one. In this case, the time required for adding a new atom does not depend explicitly on j, but only on the local surrounding. Thus, for usual molecular conformations, the entire algorithm scales almost linearly with the total number of atoms.

The power diagram constructed by the above algorithm occupies the whole space, which is somewhat redundant, as we are interested only in the region near the simulated molecule. The efficiency and numerical stability of the algorithm can be further improved if we restrict the construction of the power diagram to some fixed parallelepiped comprising the molecule.

References

1. Lee, B.; Richards, F. M. J Mol Biol 1971, 55, 379.
2. Still, W. C.; Tempczyk, A.; Hawley, R. C.; Hendrickson, T. J Am Chem Soc 1990, 112, 6127.
3. Eisenberg, D.; McLachlan, A. Nature 1986, 319, 199.
4. Ooi, T.; Oobatake, M.; Némethy, G.; Scheraga, H. A. Proc Natl Acad Sci USA 1987, 84, 3086.
5. Wesson, L; Eisenberg, D. Protein Sci 1992, 1, 227.
6. Fraternali, F.; Van Gunsteren, W. F. J Mol Biol 1996, 256, 939.
7. Gibson, K. D.; Scheraga, H. A. Proc Natl Acad Sci USA 1967, 58, 420.
8. Kang, Y. K.; Némethy, G.; Scheraga, H. A. J Phys Chem 1987, 91, 4105.
9. Kang, Y. K.; Gibson, K. D.; Némethy, G.; Scheraga, H. A. J Phys Chem 1988, 92, 4739.
10. Gogonea, V.; Ōsawa, E. Supramol Chem 1994, 3, 303.
11. Connolly M. L. J Appl Cryst 1983, 16, 548.
12. Richmond, T. J. J Mol Biol 1984, 178, 63.
13. Gibson, K. D.; Scheraga, H. A. Mol Phys 1987, 62, 1247.
14. 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.
15. Eisenhaber, F.; Argos, P. J Comput Chem 1993, 14, 1272.
16. Sridharan, S.; Nicholls, A.; Sharp, K. A. J Comput Chem 1995, 16, 1038.
17. Sanner, M. F.; Olson, A. J.; Spehner, J.-C. Biopolymers 1996, 38, 305.
18. Fraczkiewicz, R.; Braun, W. J Comput Chem 1998, 19, 319.
19. Bryant, R.; Edelsbrunner, H.; Koehl, P.; Levitt, M. Discrete Comput Geom 2004, 32, 293.
20. Hayryan, S.; Hu, C.-K.; Skřivánek, J.; Hayryan, E.; Pokorný, I. J Comput Chem 2005, 26, 334.
21. Rowlinson, J. S. Mol Phys 1963, 6, 517.
22. Connolly M. L. J Am Chem Soc 1985, 107, 1118.
23. Avis, D.; Bhattacharya, B. K.; Imai, H. Vis Comput 1988, 3, 323.
24. Gogonea, V.; Ōsawa, E. J Comput Chem 1995, 16, 817.
25. Edelsbrunner, H; Koehl, P. Proc Natl Acad Sci USA 2003, 100, 2203.
26. Buša, J.; Džurina, J.; Hayryan, E.; Hayryan, S.; Hu, C.-K.; Plavka, J.; Pokorný, I.; Skřivánek, J.; Wu, M.-C. Comput Phys Commun 2005, 165, 59.
27. Shrake, A.; Rupley, J. A. J Mol Biol 1973, 79, 351.
28. Wodak, S. J.; Janin, J. Proc Natl Acad Sci USA 1980, 77, 1736.
29. Hasel, W.; Hendrikson, T. F.; Still, W. C. Tetrahed Comput Method 1988, 1, 103.
30. LeGrand, S. M.; Merz, K. M. J Comput Chem 1993, 14, 349.
31. Stouten, P. F. W.; Frömmel, C.; Nakamura, H.; Sander, C. Mol Simul 1993, 10, 97.
32. Street, A. G.; Mayo, S. L. Folding Design 1998, 3, 253.
33. Weiser, J.; Shenkin, P. S.; Still, W. C. Biopolymers 1999, 50, 373.
34. Lee, S. M.; Feig, M.; Salsbury, F. R., Jr.; Brooks C. L., III. J Comput Chem 2003, 24, 1348.
35. Gallicchio, E.; Levy, R. M. J Comput Chem 2004, 25, 479.
36. Zhang, N. G.; Zeng, C.; Wingreen, N. S. Proteins 2004, 57, 565.
37. Rychkov, G.; Petukhov, M. J Comput Chem 2007, 28, 1974.
38. Durham, E.; Dorr, B.; Woetzel, N; Staritzbichler, R.; Meiler, J. J Mol Model 2009, 15, 1093.
39. Pavanï, R.; Ranghino, G. Comput Chem 1982, 6, 133–135.
40. Gavezzotti, A. J Am Chem Soc 1983, 105, 5220.
41. Grant, J. A.; Pickup, B. T. J Phys Chem 1995, 99, 3503.
42. Augspurger, J. D.; Scheraga, H. A. J Comput Chem 1996, 17, 1549.
43. Connolly, M. L. Molecular Surfaces: A Review. Network Science Corporation. Available at http://www.netsci.org/Science/Compchem/feature14.html, accessed 26 April 2011.
44. Edelsbrunner, H. Discrete Comput Geom 1995, 13, 415.
45. Edelsbrunner, H.; Shah, N. R. Algorithmica 1996, 15, 223.
46. Edelsbrunner, H.; Koehl, P. Discrete and Computational Geometry. MSRI Publications, 2005, 52, 243.
47. PROGEOM package. Available at http://nook.cs.ucdavis.edu/∼koehl/ProShape/download.html, accessed 26 April 2011.
48. ASC package. Available at http://mendel.imp.ac.at/studies/asc_download.jsp, accessed 26 April 2011.