The Double Cubic Lattice Method: Efficient Approaches to Numerical Integration of Surface Area and Volume and to Dot Surface Contouring of Molecular Assemblies

FRANK EISENHABER*

Biochemisches Institut der Charité der Humboldt-Universität zu Berlin, Hessische Str. 3–4, D-10115 Berlin-Mitte, Germany

PHILIP LIJNZAAD, PATRICK ARGOS, CHRIS SANDER, and MICHAEL SCHARF

European Molecular Biology Laboratory, Meyerhofstr. 1, Postfach 10.2209, D-69012 Heidelberg, Germany

Received 24 March 1994; accepted 17 August 1994

ABSTRACT

The double cubic lattice method (DCLM) is an accurate and rapid approach for computing numerically molecular surface areas (such as the solvent accessible or van der Waals surface) and the volume and compactness of molecular assemblies and for generating dot surfaces. The algorithm has no special memory requirements and can be easily implemented. The computation speed is extremely high, making interactive calculation of surfaces, volumes, and dot surfaces for systems of 1000 and more atoms possible on single-processor workstations. The algorithm can be easily parallelized. The DCLM is an algorithmic variant of the approach proposed by Shrake and Rupley (*J. Mol. Biol.*, **79**, 351–371, 1973). However, the application of two cubic lattices—one for grouping neighboring atomic centers and the other for grouping neighboring surface dots of an atom—results in a drastic reduction of central processing unit (CPU) time consumption by avoiding redundant distance checks. This is most

*Author to whom all correspondence should be addressed at European Molecular Biology Laboratory, Postfach 10.2209, Meyerhofstrasse 1, D-69012 Heidelberg, Germany. Frank Eisenhaber is a visiting scientist at the EMBL, Heidelberg.

Journal of Computational Chemistry, Vol. 16, No. 3, 273–284 (1995)

CCC 0192-8651 / 95 / 030273-12

noticeable for compact conformations. For instance, the calculation of the solvent accessible surface area of the crystal conformation of bovine pancreatic trypsin inhibitor (entry 4PTI of the Brookhaven Protein Data Bank, 362-point sphere for all 454 nonhydrogen atoms) takes less than 1 second (on a single R3000 processor of an SGI 4D/480, about 5 MFLOP). The DCLM does not depend on the spherical point distribution applied. The quality of unit sphere tesselations is discussed. We propose new ways of subdivision based on the icosahedron and dodecahedron, which achieve constantly low ratios of longest to shortest arcs over the whole frequency range. The DCLM is the method of choice, especially for large molecular complexes and high point densities. Its speed has been compared to the fastest techniques known to the authors, and it was found to be superior, especially when also taking into account the small memory requirement and the flexibility of the algorithm. The program text may be obtained on request.

Introduction

Surface calculations are widely used in molecular mechanical studies. Surface energy functions are applied to approximate the free energy and enthalpy of solvation.[1-10] A dot representation and a triangulation of a molecular surface is a necessary step in computing the electrostatic reaction field potential via the boundary element method.[11-13] Dot surfaces are used in graphical visualizations of molecules. Folding domains may be located as maxima of the compactness along the protein sequence.[14,15] The contribution of amino acid residues to the solvent accessible surface is used as a parameter in characterization of protein folds and in the generation of three-dimensional (3D) profiles for amino acid sequences.[16-18] Surface shape complementarity is one of the criteria used in predicting docking complexes of ligands with macromolecules.[19]

The surface of a molecule may be defined in different ways. The solvent accessible surface[20] is that part of the surface of a sphere centered at an atom with radius $r_{vdW} + r_{sol}$, where the center of a spherical solvent molecule or probe (r_{sol}) can be placed in contact with the atomic van der Waals sphere (r_{vdW}) without penetrating other atoms. The molecular surface[21] is the envelope of the molecular volume from which solvent is excluded. The molecular surface can often be replaced by the van der Waals surface (accessible surface with zero probe radius), which is easier to calculate.[22]

Accurate methods for calculating molecular surfaces can be grouped into analytical[23-30] and numerical integration approaches. The numerical methods can be characterized by their way of discrete surface approximation: slices of cylindrical surfaces,[20,31] cube compositions,[32-35] and point distributions on atomic spheres[36-41] (the most simple variant). Nearly all algorithms are too slow to be used in a context of multiple invocations, such as in molecular mechanics studies in which the solvation energy of various molecular conformational states is determined.[10]

Recently, two efficient modifications[40,41] of the Shrake and Rupley method[36] have been published. The fastest algorithm, the method of LeGrand and Merz,[40] computes an approximate value of the Shrake and Rupley surface area using a precalculated library of bit strings that code for the burial of atomic surface points by neighboring atoms at some space grid positions. Unfortunately, even at relatively low point density, the library occupies a significant part of the computer memory; for example, 1.4 MBytes for a 256-dot sphere. The method also lacks flexibility, especially if the user alters the distribution or number of surface dots or if a new set of radii for atoms and/or solvent is applied (with larger maximal radii). In such cases, it is necessary to recalculate the library.

The method of Abagyan and Totrov[41] depends on a special dot arrangement, in which the z-coordinate is regularly incremented while the angular position in the z-slice is defined by Fibonacci numbers (Pietr Zielenkiewicz, personal communication). Only dots within a z-axis segment, defined by the projection of the circle of intersection with a neighboring atom onto the z-axis, are checked for burial by the neighbor. However, this point distribution is not very regular, especially at the poles. Abagyan and Totrov rely on a neighbor list based on chemical groups to reduce the number of pairwise atom distance checks.

The double cubic lattice method described in this article overcomes several of the restrictions in the previously described numerical approaches. It is slightly faster than the method of LeGrand and Merz,[40] but the calculated area values are as exact as in the original Shrake and Rupley[36] approach. Additionally, our memory requirement is negligibles in contrast to that of their method. Our technique also does not depend on special properties of the surface dot distribution, as does the method of Abagyan and Totrov.[41] It is simple to implement and to incorporate into existing programs.

The algorithm basically uses an overlayed cubic lattice for grouping spatially close atoms of the molecular assembly. Likewise, another cubic lattice is applied to sort spatially close points of the tessellated atomic sphere. As a result, groups of atoms and points may be immediately recognized as nonoverlapping. Additionally, computer time is saved by the use of the dot product instead of the Euclidean distance for checking overlap.

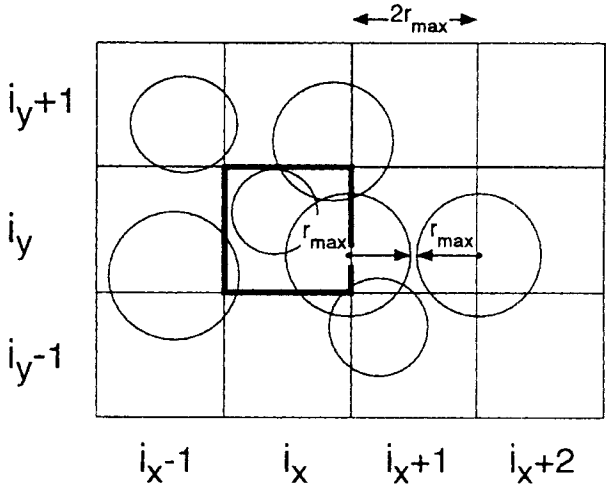

FIGURE 1. The use of a cubic grid for grouping atoms, shown here for the two-dimensional case. Atoms in the central grid cell (i_x, i_y) can only overlap with atoms in the neighboring grid cells $(i_x, i_y \pm 1)$, $(i_x \pm 1, i_y)$, and $(i_x \pm 1, i_y \pm 1)$. Overlap between the large atoms in (i_x, i_y) and $(i_x, i_y \pm 2)$ cannot occur if the grid spacing is chosen to be $2r_{max}$.

Calculation Methods

THE DCLM

The method of artificial grids is a standard approach in computational geometry to group spatially close objects.[41] The cubic lattice is the simplest, allowing the division of the three-dimensional space into disjunct and equal elementary volumes with equal extensions along the three global Cartesian coordinate axes. In the subsequent description of mathematical details, the following notations are used: $\mathbf{a}_i = (x_i, y_i, z_i)$ for the coordinate vector of atoms $i = 1, \ldots, N$ in a Cartesian coordinate space; r_i for the sum of the van der Waals and solvent radii for each respective atom (subsequently referred to as the atomic radius); r_{max}, the maximum of these radii, d_{ij} for the distance between atoms i and j; and $\mathbf{p}_k$ for the coordinate vector of dots $k = 1, \ldots, m$ on a unit sphere centered at the origin of the global coordinate system. The tessellation of the unit sphere is given by an m − tuple $T = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m)$.

The list of neighboring atoms is obtained with the following procedure. The molecular assembly under consideration is placed into a single rectangular box with edge lengths being an integer multiple of $2r_{max}$. This box is subdivided into grid cells by a cubic grid with the spacing $2r_{max}$. As a result, each atom has neighbors only in its own grid cell and in its (maximally 26) neighboring grid cells (Fig. 1). The computational steps involved are as follows:

1. Determination of the minimum and the maximum of the x-, y-, and z-coordinates of all atoms
2. Calculation of the origin of the large rectangular box (x_o, y_o, z_o) and the number of grid cells along each coordinate axis (b_x, b_y, b_z)
3. Assignment of the corresponding subbox number to each atom (i.e., if some atom lies in box (i_x, i_y, i_z) with $0 \le i_x < b_x$, $0 \le i_y < b_y$, and $0 \le i_z < b_z$, then its box number is equal to $i_x + (i_y \cdot b_x) + (i_z \cdot b_x \cdot b_y)$
4. Sorting of the atoms in accordance with box numbers.

The number of boxes occupied by atom centers is small compared with the total number of atoms. Therefore, special techniques of sorting with a small number of integer keys (with order of N operations) can be employed. Step 4 is elegantly executed with an algorithm similar to that published by Sedgewick[42] or, equivalently, using linked lists of atoms rooted in the grid cells to which the atoms belong.[43] The complete neighbor list is computed within a few hundredths of a second even for a protein with more than 3000 atoms.

The idea of neighbor lists based spatial grouping as an alternative to the chemical group method has already been suggested by other authors (see refs. 43 and 44 and references therein). It is generally concluded that the grid cell approach for calculating nonvalent interaction lists is efficient only for systems with about 600 atoms or more. Our neighbor list calculation algorithm is also suitable for much smaller systems. Obviously, the improved performance of our algorithm compared with that of Yip and Elber[44] is explained by two computational details. We use (1) variable box numbers with a fixed grid spacing equal to $2r_{max}$, the cutoff for surface interactions (a relatively small value, about 7 Å); and (2) an effective sorting approach. Cubic lattices have also been applied to approximate electrostatic forces in macromolecules.[45] For long-range interactions, the grid cell approach is less effective, and large memory requirements may cause problems.[43]

Later we will discuss aspects of unit sphere tesselations. The special properties of a point distribution $T = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m)$ on a unit sphere do not affect the present algorithm; however, the distribution should be even as possible to minimize integration error. The points $\mathbf{p}_i$ of the unit sphere sampling the surface of the atom under consideration must be checked for occlusion by neighboring atoms. To speed up this step, we employ a second lattice contained in a cube that is inscribed by the unit sphere (Fig. 2). This box is subdivided into M^3 equal small cubes. A list of surface dots per box is calculated in a way analogous to the previously given recipe (steps 1 through 4).

The information obtained from the second lattice may be used in various ways for surface calculation. The first variant, instigated by the method of Abagyan and Totrov,[41] directly cycles over the elementary cubes containing the dots. In the second approach, the unit sphere lattice is only implicitly used. A third modification investigated by us will be outlined.

Variant I

To computate the surface of atom i, a first loop is performed over all atoms $j \neq i$ from its own grid cell and all atoms j in the neighboring boxes. For a given atom j, the intervals $[X_l', X_u']$, $[Y_l', Y_u']$, and $[Z_l', Z_u']$, which are the projections of the sphere j onto the axes of the local grid (Figs. 2 and 3) are calculated. These intervals define a rectangular

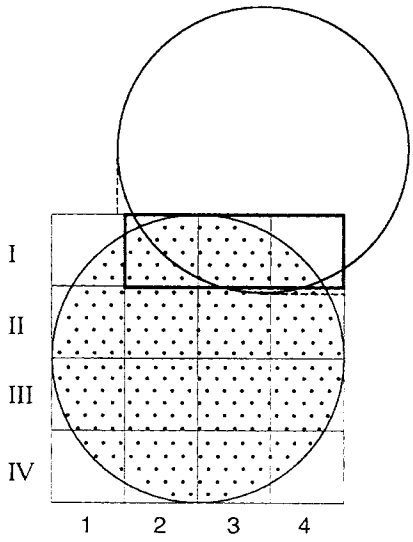

FIGURE 2. The use of a cubic grid for grouping surface dots, shown here for the two-dimensional case. In variant I, the fact is used that only dots lying in the area of overlap can get buried by the neighboring atom. Here, only dots lying in the boxes I2 –4 have to be checked for occlusion (bold box). In a simpler scheme, the full extent of the neighboring atom is taken instead of the extent of the intersection circle. Here, that would result in the need to check the dots in the boxes I1 and II1 –4 (dashed box). In variant II, the grid is employed to sort all dots according to their spatial vicinity (i.e., all dots in one and the same elementary cube are put successively into the array of dot coordinates).

box that contains the only points on atom i that can be occluded by atom j. For the x-axis, we find

$$X_l' = \max\left(\frac{x_j - x_i - (r_i - r_j)}{2r_i}, 0\right) \quad \text{and} \quad \text{(1a)}$$

$$X_u' = \min\left(\frac{x_j - x_i + (r_i + r_j)}{2r_i}, 1 - \frac{1}{M}\right) \quad \text{(1b)}$$

The expressions for the other directions are similar. The function

$$g(X) = \text{int}(M \cdot X) \quad \text{(1c)}$$

translates the interval boundaries into grid position where int represents the nearest smaller integer. For point densities higher than about 400 points per sphere (M larger than 4), it is advantageous to take only the projection of the intersection circle onto the three coordinate axes as limits for the respective intervals $[X_l, X_u]$, $[Y_l, Y_u]$, and $[Z_l, Z_u]$ in the same way, as Abagyan and Totrov

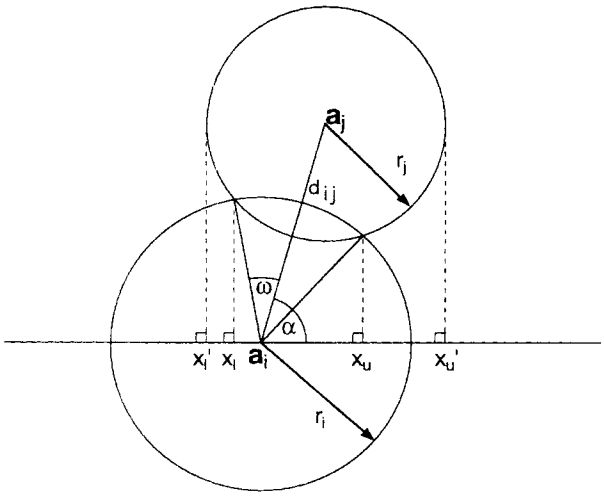

FIGURE 3. Construction of the projection of an atom's extent, or the extent of the intersection circle, onto an axis of the grid. Shown here for the x-axis. $\mathbf{a}_{i,j}$, $r_{i,j}$, and d_{ij} as usual; $X'_{l,u}$ the lower and upper limit of the projection of atom j's extent onto the x-axis; $X_{l,u}$ the lower and upper limit of the projection of the intersection circle onto the x-axis; a the angle of the vector $\mathbf{a}_j - \mathbf{a}_i$ with the positive x-axis; u the radius of the intersection circle, expressed as an arc over the surface of the unit sphere. In the text, $X'_{l,u}$ and $X_{l,u}$ are expressed as fractions of r_i.

did for one axis[40] (Figs. 2 and 3). The interval boundaries $[X_l, X_u]$ then become

$$X_l = \begin{cases} 0 & \text{if } \cos(\alpha) \le -\cos(\omega) \\ \dfrac{\cos(\alpha + \omega) + 1}{2} & \text{otherwise} \end{cases} \tag{1a'}$$

and

$$X_u = \begin{cases} 1 - \dfrac{1}{M} & \text{if } \cos(\alpha) \ge \cos(\omega) \\ \dfrac{\cos(\alpha - \omega) + 1}{2} & \text{otherwise} \end{cases} \tag{1b'}$$

Although calculation of one square root per coordinate axis cannot be avoided, the computational costs are mitigated by an additional reduction of surface dot checks. The cosines of $\alpha + \omega$ and $\alpha - \omega$ are calculated via

$$\cos(\alpha)\cos(\omega) = \frac{x_j - x_i}{r_i} C \quad \text{and} \tag{2a}$$

$$\sin(\alpha)\sin(\omega) = \frac{1}{r_i}\sqrt{\left(\frac{r_i^2}{d_{ij}^2} - C^2\right)\left((y_j - y_i)^2 + (z_j - z_i)^2\right)} \tag{2b}$$

where

$$C = \frac{d_{ij}^2 + r_i^2 - r_j^2}{d_{ij}^2} \tag{2c}$$

For each nonempty grid cell, we loop over all dots that are not yet buried. The occlusion of the surface dot $\mathbf{p}_k$ by the neighboring atom j is not tested by the squared distance as usually done, but via the dot product, which is slightly faster to compute. Dot $\mathbf{p}_k$ is occluded by atom j if

$$(\mathbf{a}_j - \mathbf{a}_i) \cdot \mathbf{p}_k > \frac{d_{ij}^2 + r_i^2 - r_j^2}{2r_i} \tag{3}$$

where the value on the right side has to be calculated only once for each neighboring atom j. It was found empirically that M is a good choice if

$$M^3 \le m/2 < (M + 1)^3 \tag{4}$$

where m is the number of points per sphere. In this case, the computer time consumption is low for a wide range of m, but it may not be the optimal selection of M for a special value of m.

Variant II

Here the loop over the dots k on the sphere of atom i precedes the loop over the neighboring atoms j. The rationale is that most of the occluded surface dots are buried by a few close atoms. If dot $\mathbf{p}_k$ is occluded by neighboring atom j, the number j is stored and it is checked whether the next dot $\mathbf{p}_{k+1}$ is buried by the same atom j. In this variant, it is necessary to precalculate a temporary list of neighbors j overlapping atom i on the basis of the neighbor list obtained with the first cubic grid. Also, arrays with the difference vectors $\mathbf{a}_j - \mathbf{a}_i$ and the corresponding dot products [see eq. (3)] must be prepared.

This algorithm is sensitive to the order in the m-tuple $T = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m)$ of point vectors. Given that atom j is occluding dot $\mathbf{p}_k$, it would be desirable to have all other dots covered by atom j following $\mathbf{p}_k$ in the m-tuple T. This can be partly fulfilled by reordering the point vectors in such a manner that $\mathbf{p}_k$ is followed by a spatially close dot $\mathbf{p}_{k+1}$ in T. If the algorithm generating the tessela-

tion T does not place the dots in such an order automatically (as in DSSP[46]), the cubic lattice on the unit sphere can be applied and all dots in one elementary cube are placed successively into the corresponding arrays. Additionally, dots from neighboring boxes would similarly follow each other.

If now a dot $\mathbf{p}_k$ is occluded by one neighboring atom j, the likelihood that the same atom will overlap the following dot $\mathbf{p}_{k+1}$ is high. Therefore, it is worthwhile to store the atom j and to check it with the next dot. If atom j does not occlude dot $\mathbf{p}_{k+1}$, only then is the iteration over the temporary neighbor list restarted. As a result, the central loop of the corresponding program is remarkably short. Our statistics for the solvent accessible surface of compact protein structures show that about 80% of all dots are recognized as buried after checking ony one neighboring atom j. The worst case is an accessible dot for which a complete loop over all neighbors j has to be made.

In this variant, the grid spacing of the unit sphere depends mainly on the depth of overlap of a good neighbor and not on the dot density. Good performance can be achieved for $M = 3$, 4, 5, and 6; $M = 4$ is a generally acceptable value for all dot densities and protein structures studied.

Variant III

It is attractive to loop, for a given atom i, over all elementary cubes occupied with dots in the unit sphere grid and then cycle over all neighboring atoms j. The corner of the elementary cube with the largest distance from the center of atom i can be rapidly determined by considering each coordinate axis individually. For the x-axis, the coordinate difference d_x between the center of atom j and the farthest corner is

$$d_x = \max\left(\frac{x_j - x_i}{r_i} - X_l, \frac{x_j - x_i}{r_i} - X_l - \Delta\right) \quad (5)$$

where X_l and $X_l + \Delta$ are the two possible x-coordinates of the eight vertices of cube l and Δ is the grid spacing. If this corner is occluded by atom j, all dots in cube l are also not accessible. Similarly, an elementary cube is not occluded by any atom j if its circumscribed sphere has no overlap with any neighbor, and consequently all dots in the corresponding cube are accessible.

The status of most of the dots is determined by one of the two conditions just described.

Relative Performance of the Variants

We also investigated several other combinations of the previously described algorithmic elements. All variants proved faster than previously published algorithms. If implemented on a UNIX workstation (SGI or DEC), variant II is faster than any other of our variants. Algorithm II outperformed variant I by a factor of almost 2 and variant III by a factor of about 1.5. All results presented in this article have been obtained with variant II.

We think that the preferences may be different for other machine architectures and/or compilers. The burden of conditional jumps, the ability to execute integer and float operations in parallel, and the size of fast cache memory will be decisive for this choice.

NOTES ON PARALLELIZATION

The time-consuming part of the calculation is the check of the dots on a given atomic sphere for burial by neighboring atoms. The calculations for atoms i are independent from each other and can be placed on independent processors. The use of the *m_fork* utilities (Sequent Computer Systems parallel programming primitives) as available on SGI multiprocessor machines is a cheap possibility. The reduction in computation time is expected to be almost linear with the number of processors.

COMPUTATION OF SURFACE, VOLUME, AND COMPACTNESS

The surface area A is obtained by

$$A = 4\pi \sum_i r_i^2 \frac{m_{\text{acc}}(i)}{m} \quad (6)$$

where $m_{\text{acc}}(i)$ is the number of dots on atom i not occluded by neighboring atoms and the summation is over all atoms in the molecule. The accuracy may be improved if the surface dots are weighted by area in accordance with the Voronoi subdivision of the unit sphere for a given tessellation.

The molecular volume V is calculated through the Gauss-Ostrogradskii theorem,

$$\int_V \nabla \cdot \mathbf{F}(\mathbf{r})\, dV = \oint_S \mathbf{F}(\mathbf{r}) \cdot dS$$

with $\mathbf{F}(\mathbf{r}) = \mathbf{r}$, the coordinate vector. As a result, the volume is equal to

$$V = \frac{4\pi}{3m} \sum_i \left[r_i^2 \mathbf{a}_i \cdot \left(\sum_{k(i)} \mathbf{p}_k \right) + r_i^3 m_{\text{acc}}(i) \right] \quad (7)$$

where all dots are assigned equal weights. The second summation is carried out only over all accessible dots $\mathbf{p}_k$ on atom i. The numerical error in the volume calculation is also affected by the average length of the vectors $\mathbf{a}_i$. This influence can be suppressed by placing the global origin at the geometrical center of the molecular assembly under study. The volume in eq. (7) is the CPK (Corey-Pauling-Koltun) or van der Waals volume if the atom radii are not incremeted by the probe radius. This volume is a good approximation to the solvent-excluded volume defined by Connolly[47] for low molecular compounds. With atom radii including the probe radius, eq. (7) yields the volume inside the solvent accessible surface used in compactness calculations.

The compactness z of a molecule is defined as the solvent accessible surface area divided by the minimum possible surface area, which is that of a sphere of equal volume.[15] Thus,

$$Z = A \cdot (36\pi \cdot V^2)^{-1/3} \quad (8)$$

TESSELATION OF THE UNIT SPHERE

The spherical point distribution to be used for a Shrake and Rupley type of surface calculation has to be as even as possible. Subdivisions based on inscribed regular polyhedra are natural candidates. Tesselations have been frequently studied, with applications ranging from architecture (geodesic domes) to biology (structure of viral coats).[48]

An icosahedral tesselation produces $10\tau + 2$ vertices (20τ faces, 30τ edges), where τ is of the form

$$\tau = b^2 + bc + c^2$$

Three classes of tesselations can be distinguished: $c = 0$ (class I); $b = c$ (class II); and $b \neq c$ (class III, skewed tesselations).

The quality of a tesselation, here defined as the ratio of the maximum and minimum edge lengths of the Delaunay triangulations it generates, depends on the sophistication of the algorithm. A yardstick for the quality is the case of the tesselation frequency tending to infinity. The five triangles surrounding the icosahedral poles then form regular pentagons, resulting in an edge length ratio of $2\sin(36°) \approx 1.1756$.

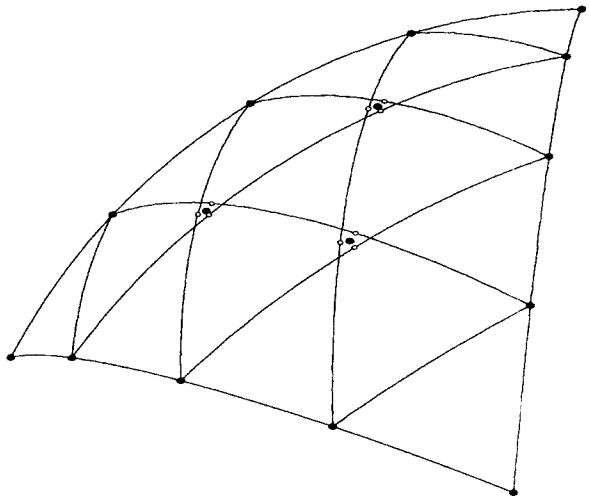

FIGURE 4. Tessellation of an icosahedral face with frequency 4. Equiangular subdivision of the icosahedral edges does not yield coincident equivalent grid points inside the face. The average point (filled circles) of three equivalent grid points (open circles) is taken instead.

The simplest way to subdivide a spherical triangle is through projection onto the unit sphere of a triangular grid lying on a triangular face (gnomonic projection.[49] This, however, results in poor quality, with a ratio of longest to shortest arc of about 1.5 because the subdivision is equal with respect to chords but not spherical arcs.

We have cast this way of subdivision in spherical terms, by covering the spherical triangle with segments of great circles that connect corresponding subdivisions of the edges of the spherical triangle (Fig. 4). These segments form a grid similar to the triangular grid previously mentioned. However, these segments, and their subsequent subdivisions into the spherical triangular grid points, do not coincide generally (Fig. 4). This problem was solved by always taking the average of the three resulting equivalent grid points. When such a method of tesselating a face is applied to the icosahedron or pentakisdodecahedron (32 vertices, 90 edges, 60 faces), one obtains class I or II icosahedral tesselations, respectively.

We found the tesselation quality of this procedure for any frequency always to be better than the limiting ratio of 1.1756. This is better than any of the methods currently in use.[39,46,50,51] Another

advantage is the good control over the number of points produced, as one can choose $10b^2 + 2$ points (class I tesselations) or $30b^2 + 2$ points (class II tesselations), with b being the tesselation frequency (which can be any positive integer).

The procedure used to triangulate the point distributions is based on a so-called greedy planar triangulation[52] but extended to deal with points on the unit sphere. It generates a Delaunay triangulation from which edge lengths and areas of faces and Voronoi regions are calculated. Areas are those of spherical polygons.

TABLE I.
Comparison of Numerical and Analytical Surface Area and Volume Calculations and the Influence of Point Density.

	Analyt.	122	362	642	1002	1472
		Ia. 4PTI (454 heavy atoms, 58 residues)				
t	2.97	0.50	0.91	1.36	1.86	2.54
A	3973.8	3961.4	3971.8	3967.9	3974.1	3975.7
ΔA	0.0	13.4	2.0	5.9	0.3	1.9
ΔA_{atom}	0.0	4.1	1.5	1.0	0.6	0.6
V	11915.3	11871.3	11923.7	11885.7	11911.1	11911.9
ΔV	0.0	44.0	8.4	29.6	4.2	4.6
Z	1.575	1.574	1.574	1.575	1.576	1.576
		Ib. 3FXN (1073 heavy atoms, 138 residues)				
	Analyt.	122	362	642	1002	1472
t	8.15	1.29	2.21	3.15	4.31	5.63
A	6943.8	6968.3	6933.4	6944.4	6939.1	6943.3
ΔA	0.0	24.5	10.4	0.6	4.7	0.5
ΔA_{atom}	0.0	4.5	1.5	1.3	0.7	0.6
V	26350.0	26373.6	26281.9	26370.6	26338.8	26352.3
ΔV	0.0	23.6	69.1	20.6	11.2	2.3
Z	1.621	1.626	1.622	1.621	1.621	1.621
		Ic. 1TIM (3740 heavy atoms, 492 residues, 2 chains)				
	Analyt.	122	362	642	1002	1472
t	32.1	5.23	8.18	11.55	15.25	20.51
A	20002.8	19970.9	19997.1	19998.7	20012.2	19997.0
ΔA	0.0	31.9	5.7	4.1	9.4	5.8
ΔA_{atom}	0.0	4.2	2.1	1.3	1.0	0.7
V	89100.7	89153.7	88972.7	89029.1	89121.0	89067.3
ΔV	0.0	53.0	128.0	71.6	21.0	33.4
Z	2.073	2.069	2.075	2.074	2.074	2.073

Results of the program NSC for selected protein structures (Tables Ia–c) are presented. Coordinates of all ATOMS records were taken from the corresponding entries of the Brookhaven Protein Data Bank.[53,54] The atom radii were taken from Eisenberg and McLachlan.[3] The solvent radius was set equal to 1.4 Å.

The time t (in seconds) is the CPU time for calculating the surface (on a single R3000 processor of a SGI / 4D 480 as in ref. 30, about 5 MFLOP). The time values here are directly comparable with $t_{ana,3}$ in Tables 1 and 2 in ref. 30. Both the numerical method and the analytical routine (ASC[30]) show a significantly improved performance. For the numerical surface computation, we applied variant II (see the Calculation Methods section).

A (in Å^2) is the solvent accessible surface area for the whole structure. ΔA denotes the absolute value of the difference between the value of A and the analytically calculated surface area using the program ASC. ΔA_{atom} is the maximum of the absolute deviation of the surface of a single atom and its analytically computed area. The volume V (in Å^3) inside the solvent accessible surface, the absolute deviation from the analytically calculated volume, and the compactness Z [eq. (4)] are also presented.

To assess numerical error, the analytical results for surface area (from ASC[30]) and volume are listed in the second column. The analytical volume was computed with the program PQMS[46] using atom radii incremented by the solvent radius and a zero probe radius. In this case, cusps do not occur and the volume calculation is completely analytical.

The following columns present the numerical results for several dot densities of the atomic spheres; tessellations 122 and 1472 are based on the dodecahedron (class II), whereas the remaining are icosahedral tessellations (class I): (a) 4PTI (454 heavy atoms, 58 residues); (b) 3FXN (1073 heavy atoms, 138 residues); (c) 1TIM (3740 heavy atoms, 492 residues, 2 chains).

Results

A computer program NSC (numerical surface calculation) was written in standard C and tested on several types of UNIX workstations (SGI, DEC, SUN). For file input and command interpretation, it was attached to the shell of ASC.[30] Best optimization results were obtained with the compiler *gcc* distributed by the Free Software Foundation (version 2.5.7, options *-O2 -ffast-math -finline-functions -funroll-all-loops*). NSC selects among the available unit sphere tesselations the one with the smallest number of dots above the density required by the user. The calculation results for selected protein structures are presented in Tables Ia–c and II.

The accuracy of our numerical surface integration technique was checked against the analytical computation of the surface area for the entire molecule and also on a per-atom basis. For this purpose, the program ASC[30] was used. The accuracy is improved with increased dot density (Tables Ia–c). More than 600 dots per unit sphere ensure an accuracy better than 1.5 Å^2 for the surface area for every atom which is about 1% of the solvent accessible surface area of an isolated atom.

The quality of the unit sphere tesselation is decisive for the spatial invariance[38] (independence of the surface area on the orientation of the molecule with respect to the reference frame of the point distribution). Our polyhedral point distributions yield spatially invariant surface area values because the accuracy of surface area per atom depends only on point density for various protein structures (differently oriented in their crystals; see Tables Ia–c).

Interestingly, especially for higher point densities, the surface area results are practically the same, whether the polyhedral or the Zielenkiewicz point distribution[41] is applied. This is surprising because the tesselation quality of the latter is only between 2.5 and 3. The edge length distribution of the corresponding Delaunay triangulation is shown in Figure 5a and can be seen to be rather poor. Nonetheless, the face areas and the Voronoi areas (for our purposes more relevant) are well distributed (see Figs. 5b and 5c). Consequently, the Zielenkiewicz distribution can safely be applied for the calculation of surface areas of entire molecules, but for most other purposes (e.g., for generating a dot surface to be used in trianglation), a polyhedral distribution is preferred. It can also be seen from Figure 5c that the introduction of point weights for the polyhedral tesselation would not significantly improve the area accuracy if the dot density is low (less than 1000).

The computation speed is influenced by the number of points per atomic sphere, by the number of atoms, and by the conformation of the molecule. In more compact molecules, each atom has a larger number of neighboring atoms (Table II). Our numerical routine is normally faster than ASC[30] for point densities below 1000. Because of the rapid neighbor list calculation and a reduced number of distance checks between atoms and surface dots, our method is faster than the approach of Abagyan and Totrov; for example, NSC needs only 34% of their CPU time for flavodoxin (atom coordinates of entry 3FXN in the Protein Data Bank). The calculation of the solvent accessible surface area of the crystal conformation of bovine pancreatic trypsin inhibitor (entry 4PTI of the Brookhaven Protein Data Bank, 362 point sphere for all 454 nonhydrogen atoms) takes less than 1 second (Table Ib). The same CPU time was achieved by the algorithm of Merz and LeGrand[40] on a comparable machine, but with a lower point density. It should also be noted that NSC was used inside the command shell of ASC and, as a result, the access to coordinates and radii was not optimized by the compiler together with the numerical surface calculation procedure. This is a realistic situation because surface calculations are often only one of the computational steps in a complex

TABLE II.
Influence of Conformation.

	1CRN		$CRN_{extended}$	
	Analyt.	642	Analyt.	642
t	1.96	0.96	0.91	0.80
A	2988.3	2995.8	5813.5	5815.8
ΔA	0.0	7.5	0.0	2.3
ΔA_{atom}	0.0	1.1	0.0	1.1
V	8687.3	8703.5	10829.8	10819.2
ΔV	0.0	16.2	0.0	10.6
Z	1.462	1.464	2.456	2.458

The data show the dependency of computational efficiency on structural compactness. Crambin is a small protein with 327 heavy atoms and 46 residues. The data are presented for the PDB version (1CRN[53,54]) and for a fully extended conformation ($CRN_{extended}$). The symbols are assigned as in Table I. The atom coordinates for the extended conformation of crambin were obtained using the program package ICM.[41]

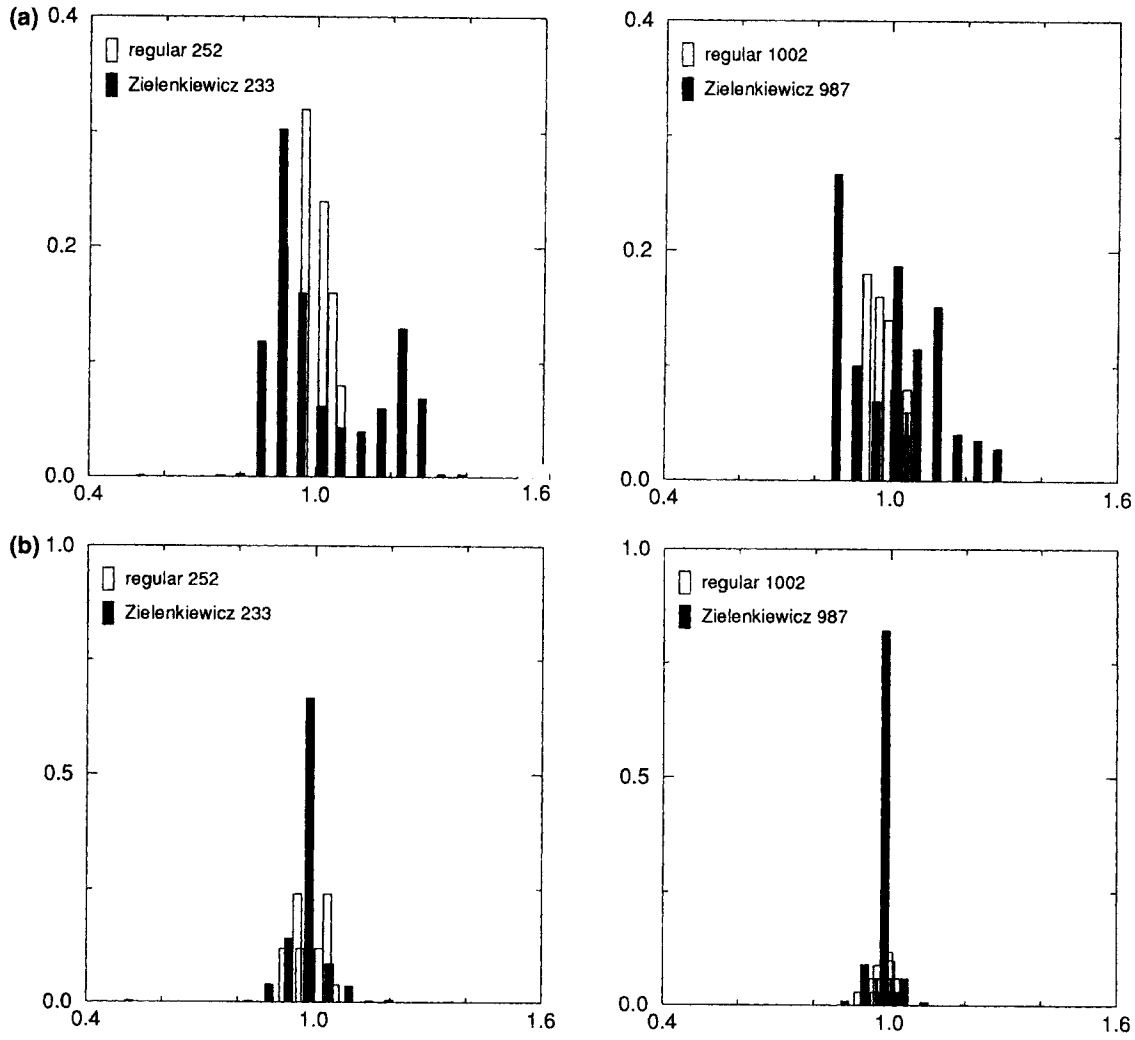

FIGURE 5. Comparison of the quality of point distributions at different numbers of points per sphere. The facial and Voronoi areas and edge lengths refer to those in a Delaunay triangulation[52] of the distributions. The plots represent histograms of fractional deviations from the ideal value. The data were grouped in 20 bins of equal width; plotted vertically is the occurrence as fraction of the total. (a) Edge lengths. The ideal edge length L' is calculated under the assumption that the unit sphere is uniformly covered with equilateral spherical triangles of arc length L. For the area ε of one such triangle, we can write

$$\varepsilon = \frac{4\pi}{F} = 4 \arctan\left(\sqrt{\tan\left(\tfrac{3}{4}L\right)\tan^3\left(\tfrac{1}{4}L\right)}\right)$$

(formula of L'Huilier from spherical trigonometry), where F is the number of triangular faces. After solving for L, the ideal edge length is given by $L' = 2\sin(L/2)$. (b) Face areas. The ideal value is $4\pi/F$, where F is the number of faces per unit sphere. (c) Areas of Voronoi regions. The ideal value is $4\pi/m$, where m is the number of dots per unit sphere.

context. With a one-file program, even better performance can be expected. For a molecule of about 1000 atoms and a medium dot density per atomic sphere, the calculations are in fact done interactively even on a normal workstation.

The volume values V calculated with our routine differ less than 3% from the analytically calculated volumes (program PQMS),[46] beginning with a point density of 600 dots per atomic sphere (Tables Ia–c). Similarly, the compactness values computed with our routine are identical to those calculated by Zehfus[15] with an accuracy of 2%.

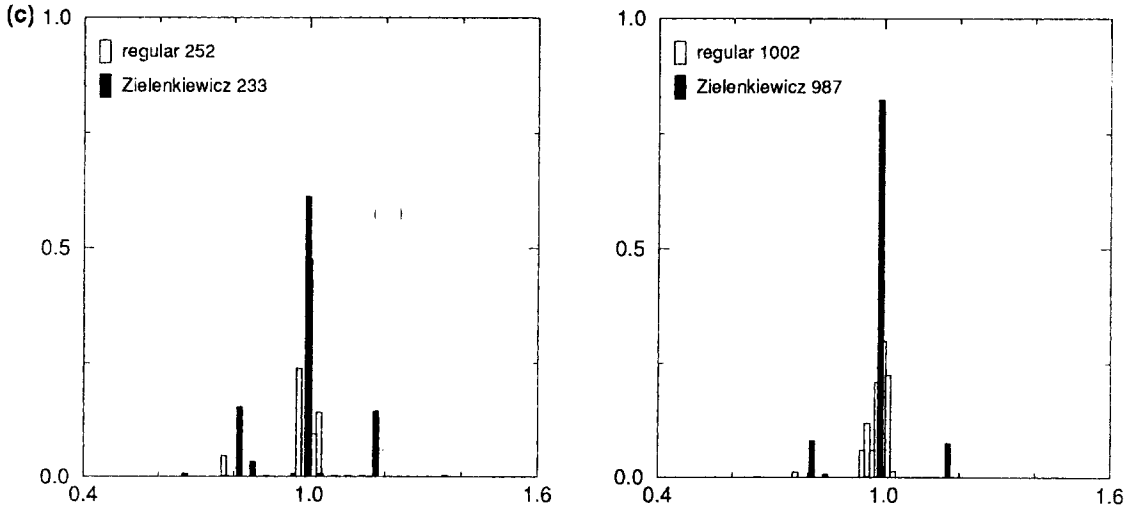

FIGURE 5. (Continued)

Discussion

All surface calculation procedures have advantages and deficiencies. The choice of the best method for a certain purpose depends on many aspects; for example, the desired accuracy of the surface area (overall value or area per atom), computational speed, memory requirements, algorithmic complexity, available time for programming efforts, and the context of the calculation.

An analytical algorithm[30] is to be preferred if the accuracy of the surface area per atom is important, such as in optimization protocols. The noise of numerical surface evaluation may render the total energy function misbehaved to numerical optimization routines. Analytic surface computation is also advantageous if it is desirable to obtain at low additional computational costs accurate analytical derivatives of the surface area (with respect to atom coordinates) or the connectivity of surface pieces.[47,55] The computation speed is not a reason to reject the analytical algorithm. The program ASC[30] (version 2.0, February 1994) calculates the surface area more than two times faster than previously described (especially for large and compact molecules) as a result of several algorithmic innovations (compare t in the columns for analytical results of Tables Ia–c and II with $t_{ana,3}$ in Tables 1 and 2 of ref. 30). The need for extensive and sophisticated programming, however, is a drawback of the analytical method.

A numerical algorithm as presented here is advantageous (1) if the accuracy of surface area values in the range of ± 0.5 to 3.0 Å^2 per atom (depending on point density) is sufficient and (2) if it is desired to determine at almost no additional computational cost the values of the volume (via the Gauss-Ostrogradskii theorem; see the Calculation Methods section) and of the compactness or the coordinates of dots representing the surface of the molecule. Our polyhedral dot distributions are well suited for generating dot surfaces that may be further used for graphics, triangulation, boundary element calculations (dot selection via the spoke method[13]), and geometrical docking. The algorithms described in this article will be applied in new releases of the programs DSSP[46] and Melc.[13]

The code of the subroutine NSC in standard C is available for distribution; the request should be directed to the authors by regular post or electronic mail: Eisenhaber@EMBL-Heidelberg.DE on Internet. The routine NSC is also incorporated into the program ASC[30] (beginning with version 2.0) and can be invoked with the commands "nsc", "nsc_vol", and "nsc_dots". The package ASC is available via FTP (send a request for details).

Acknowledgments

The authors are grateful to C. J. Kitrick for suggestions regarding tessellation of the unit sphere and for providing his sphere tessellation programs to us. F. E. is grateful to R. Abagyan for generously giving insight into details of his program package ICM[40] and thanks A. Juffer for stimulating discussions and C. Frömmel for encourage-

ment. The authors acknowledge financial support from the German Bundesministerium für Forschung und Technologie (grant FG5-1075 to P.A.) and from the Fund Wissenschaftler-Integrationsprogramm (grant 020386/B to F.E.) jointly administered by the East German Länder and the government of the Federal Republic of Germany.

References

1. W. Hasel, T. Hendrickson, and W. C. Still, *Tetr. Comp. Method.*, **1**, 103 (1988).
2. W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson, *J. Amer. Chem. Soc.*, **112**, 6127 (1990).
3. D. Eisenberg and A. D. McLachlan, *Nature*, **319**, 199 (1986).
4. L. Wesson and D. Eisenberg, *Protein Science*, **1**, 227 (1992).
5. C. A. Schiffer, J. W. Caldwell, R. M. Stroud, and P. A. Kollman, *Protein Science*, **1**, 396 (1992).
6. C. A. Schiffer, J. W. Caldwell, P. A. Kollman, and R. M. Stroud, *Molecular Simulation*, **10**, 121 (1993).
7. B. v. Freyberg and W. Braun, *J. Comp. Chem.*, **14**, 121 (1993).
8. B. v. Freyberg, T. J. Richmond, and W. Braun, *J. Mol. Biol.*, **233**, 275 (1993).
9. T. Ooi, M. Ootobake, G. Nemethy, and H. A. Scheraga, *Proc. Natl. Acad. Sci.*, **84**, 3086 (1993).
10. K. C. Smith and B. Honig, *Proteins*, **18**, 119 (1994).
11. R. J. Zauhar and R. S. Morgan, *J. Comp. Chem.*, **11**, 603 (1990).
12. B. J. Yoon and A. M. Lenhoff, *J. Comp. Chem.*, **11**, 1080 (1990).
13. A. H. Juffer, E. F. F. Botta, B. A. M. van Keulen, A. van der Ploeg, and H. J. C. Berendsen, *J. Comp. Phys.*, **97**, 144 (1991).
14. M. Zehfus and G. D. Rose, *Biochemistry*, **25**, 5759 (1986).
15. M Zehfus, *Proteins*, **16**, 293 (1993).
16. J. Bowie, R. Luthy, and D. Eisenberg, *Science*, **253**, 164 (1991).
17. R. Luthy, J. Bowie, and D. Eisenberg, *Nature*, **356**, 83 (1992).
18. R. Abagyan, D. Frishman, and P. Argos, *Proteins*, **19**, 132 (1994).
19. J. Cherfils and J. Janin, *Curr. Op. Struct. Biol.*, **3**, 265 (1993).
20. B. Lee and F. M. Richards, *J. Mol. Biol.*, **55**, 379 (1971).
21. F. M. Richards, *Ann. Rev. Biophys. Bioeng.*, **6**, 151 (1977).
22. D. C. Rees and G. M. Wolfe, *Protein Science*, **2**, 1882 (1993).
23. M. L. Connolly, *J. Appl. Cryst.*, **16**, 548 (1983).
24. M. L. Connolly, *J. Appl. Cryst.*, **18**, 499 (1985).
25. T. J. Richmond, *J. Mol. Biol.*, **178**, 63 (1984).
26. K. D. Gibson and H. A. Scheraga, *Mol. Physics*, **62**, 1247 (1987).
27. K. D. Gibson and H. A. Scheraga, *Mol. Physics*, **64**, 641 (1988).
28. L. R. Dodd and D. N. Theodorou, *Mol. Physics*, **72**, 1313 (1991).
29. G. Perrot, B. Cheng, K. D. Gibson, J. Vila, K. A. Palmer, A. Nayeem, B. Maigret, and H. A. Scheraga, *J. Comp. Chem.*, **13**, 1 (1992).
30. F. Eisenhaber and P. Argos, *J. Comp. Chem.*, **14**, 1272 (1993).
31. T. J. Richmond and F. M. Richards, *J. Mol. Biol.*, **119**, 537 (1978).
32. J. J. Müller, *J. Appl. Cryst.*, **16**, (1983).
33. M. Y. Pavlov and B. A. Fedorov, *Biopolymers*, **22**, 1507 (1983).
34. A. Y. Meyer, *J. Comp. Chem.*, **9**, 18 (1988).
35. H. R. Karfunkel and V. Eyraud, *J. Comp. Chem.*, **10**, 628 (1989).
36. A. Shrake and J. A. Rupley, *J. Mol. Biol.*, **79**, 351 (1973).
37. H. Wang and C. Levinthal, *J. Comp. Chem.*, **12**, 868 (1991).
38. J. L. Pascual-Ahuir and E. Silla, *J. Comp. Chem.*, **12**, 1047 (1991).
39. E. Silla, I. Tunon, and J. L. Pascual-Ahuir, *J. Comp. Chem.*, **12**, 1077 (1991).
40. S. M. LeGrand and K. M. M. Merz, Jr., *J. Comp. Chem.*, **14**, 349 (1993).
41. R. A. Abagyan, M. M. Totrov, and D. Kuznetsov, *J. Comp. Chem.*, **15**, 488 (1994).
42. R. Sedgewick, *Algorithms in C++* (2nd ed.), Addison-Wesley, Reading, MA, 1992, pp. 112–114 and pp. 373–386.
43. W. F. van Gunsteren, H. J. C. Berendsen, F. Colonna, D. Perahia, J. P. Hollenberg, and D. Lelouch, *J. Comp. Chem.*, **5**, 272 (1984).
44. V. Yip and R. Elber, *J. Comp. Chem.*, **10**, 921 (1989).
45. D. B. Beglov and A. A. Lipanov, *J. Biomol. Struct. Dyn.*, **9**, 205 (1991).
46. W. Kabsch and C. Sander, *Biopolymers*, **22**, 2577 (1983).
47. M. L. Connolly, *J. Am. Chem. Soc.*, **107**, 1118 (1985).
48. H. S. M. Coxeter, In *A Spectrum of Mathematics*, J. C. Butcher, Ed., Auckland University Press and Oxford University Press, 1972, pp. 98–107.
49. H. Wenninger, *Spherical Models*, Cambridge University Press, Cambridge, UK, 1979.
50. C. J. Kitrick, *Structural Topology*, **11**, 15 (1985).
51. C. J. Kitrick, *Int. J. Space Struct.*, Special Issue on Geodesic Forms, **5**, 223 (1990).
52. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
53. F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer, Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi, *J. Mol. Biol.*, **112**, 535 (1977).
54. E. E. Abola, F. C. Bernstein, S. H. Bryant, T. F. Koetzle, and J. Weng, In *Crystallographic Databases—Information Content, Software Systems and Scientific Applications*, F. H. Allen, G. Berghoff, and R. Sivers, Eds., International Union of Crystallography, Bonn/Cambridge/Chester, 1987, p. 107.
55. P. Alard and S. J. Wodak, *J. Comp. Chem.*, **12**, 918 (1991).