Rapid Grid-Based Construction of the Molecular Surface and the Use of Induced Surface Charge to Calculate Reaction Field Energies: Applications to the Molecular Systems and Geometric Objects

WALTER ROCCHIA,[1] SUNDARAM SRIDHARAN,[2] ANTHONY NICHOLLS,[3] EMIL ALEXOV,[4] ALESSANDRO CHIABRERA,[5,*] BARRY HONIG[4]

[1]*Research Center "E. Piaggio," University of Pisa, Via Diotisalvi 2, 56126 Pisa, Italy*

[2]*Computer Aided Drug Discovery, Pharmacia Corporation, 301 Henrietta St., Kalamazoo, Michigan 49007*

[3]*OpenEye Scientific Software, Inc., 335c, Winische Way, Santa Fe, New Mexico 87501*

[4]*Howard Hughes Medical Institute, Department of Biochemistry and Molecular Biophysics, Columbia University, 630 West 168 Street, New York, New York 10032*

[5]*Biophysical and Electronic Engineering Department, Via All'Opera Pia 11a, 16145, Genova, Italy*

Received 28 February 2001; Accepted 19 June 2001

Abstract: This article describes a number of algorithms that are designed to improve both the efficiency and accuracy of finite difference solutions to the Poisson–Boltzmann equation (the FDPB method) and to extend its range of application. The algorithms are incorporated in the DelPhi program. The first algorithm involves an efficient and accurate semianalytical method to map the molecular surface of a molecule onto a three-dimensional lattice. This method constitutes a significant improvement over existing methods in terms of its combination of speed and accuracy. The DelPhi program has also been expanded to allow the definition of geometrical objects such as spheres, cylinders, cones, and parallelepipeds, which can be used to describe a system that may also include a standard atomic level depiction of molecules. Each object can have a different dielectric constant and a different surface or volume charge distribution. The improved definition of the surface leads to increased precision in the numerical solutions of the PB equation that are obtained. A further improvement in the precision of solvation energy calculations is obtained from a procedure that calculates induced surface charges from the FDPB solutions and then uses these charges in the calculation of reaction field energies. The program allows for finite difference grids of large dimension; currently a maximum of 571^3 can be used on molecules containing several thousand atoms and charges. As described elsewhere, DelPhi can also treat mixed salt systems containing mono- and divalent ions and provide electrostatic free energies as defined by the nonlinear PB equation.

 J Comput Chem 23: 128–137, 2002

Key words: finite-difference method; Poisson–Boltzmann equation; molecular surface; solvent-accessible surface; geometrical objects

Introduction

The Finite Difference Poisson–Boltzmann (FDPB) method is widely used to solve a broad range of electrostatic problems.[1–5] In addition to the electrostatic aspects of the problem, a crucial step in any algorithm involves defining the boundary between the inside and outside of a given molecule because this is the region where the dielectric constant changes from its interior value (usually a low number) to the higher dielectric constant characteristic of many solvents, especially water. Although there is some uncertainty as to where to define this boundary, in most applications it is taken to correspond to the molecular surface (MS) defined as the contact surface between the van der Waals surface of a molecule and the surface of a spherical probe that represents the solvent.[6, 7] The precision and accuracy of the FDPB method depend in part on the ability to obtain an accurate grid representation of the MS. In this work we will present a new algorithm that provides both a fast and accurate procedure to map the MS onto cubic lattices.

*A. Chiabrera died on November 8, 1999.

Correspondence to: B. Honig; e-mail: bh6@columbia.edu

Contract/grant sponsor: NSF; contract/grant number: DBI-9904841

The availability of an accurate description of the molecular surface suggests an alternate approach to the calculation of electrostatic free energies. This is standardly done in FDPB programs by calculating the product of the electrical potential and charge at each grid point where a real charge has been mapped. This, however, implies that the FDPB lattice is used twice; once to solve the PB equation and once to obtain free energies. An alternative approach to the calculation of free energies will be described, which is based on polarization charges mapped onto the MS. This approach, combined with the new method to represent the MS, will be shown to lead to a major improvement in the precision of solvation free energies that can be derived from FDPB calculations.

The MS consists of convex-shaped contact regions of the van der Waals surface, where atoms are accessible to the probe, and concave-shaped reentrant regions, which are defined by the surface of an inward-facing probe when it is simultaneously in contact with the van der Waals surface of more than one atom. Connolly[8] described an analytical method to calculate the MS that is still widely used although new methods to calculate the surface analytically have been described.[9, 10] However, FDPB calculations require a grid representation of the MS. One possible approach is to build the MS analytically and then to map it onto a grid.[11–13] Analytical procedures can be quite time consuming, and do not necessarily offer any advantages for finite difference calculations because the surface must in any case be mapped onto a lattice. The method described in this article is semianalytical in nature, and is extremely fast, with little loss of accuracy.

Perhaps the most common approach to the construction of the MS is to start from a description of the solvent-accessible surface,[13] SAS, which is obtained by increasing the van der Waals radii of all atoms by the probe radius.[6, 7] Thus, the quality of the MS used in FDPB calculations depends on the accuracy of the method used to generate the SAS. Early versions of DelPhi used the *Inkblot* method to generate the MS from the SAS.[14] First, the SAS is mapped onto a grid, and then the MS is generated by assigning to the interior of the molecule those grid points that are farther than a probe radius distance from the SAS. The algorithm used in this work to generate the MS is more accurate than the *Inkblot* procedure at low resolution, and it is not so computationally demanding at high grid resolution. The algorithm first builds the van der Waals volume and then excludes inaccessible regions by a method that iteratively expands the van der Waals surface. A major feature of the method is that only a subset of points on the SAS is generated, and is used to calculate the MS in the reentrant regions. The reduction in the number of surface points both saves computer time and allows for more accurate description of the MS in the reentrant regions, which are the most difficult parts of the surface to describe.

A further development described below is the incorporation of geometric objects into the FDPB method. Although almost all current applications involve molecular systems described in atomic detail, there are many cases where a more schematic representation is desirable. In a biological context, these might correspond to a situation where a structure has not been determined to atomic detail (e.g., many membrane proteins). As another example, simulating a layer of molecules scanned by a tip of an atomic force microscope requires modeling a system that consists of several irregularly shaped objects (molecules) and an object whose structure need not be described in atomic detail but that can be represented by a simple geometrical shape (the tip of the microscope). In such applications as well, a precise description of the molecular surface is critical.

Methods

Van Der Waals Surface (VDWS) Generation

The VDWS of a molecule or of an object is determined from the VDW surfaces of individual atoms and/or surfaces of objects. The surfaces are mapped onto the grid (Fig. 1). Any midpoint (between two consecutive lattice points) is classified as either internal or external according to its position. If it falls within an object or an atom, it is considered to be "in" midpoint (filled circles in Fig. 1). If it lies outside objects and/or atoms, it is marked as an "out" midpoint (empty circles in Fig. 1). After the mapping, each grid point is classified as follows based on its six nearest midpoint positions: *External* grid point—all surrounding midpoints are in solution; *Internal* grid point—all surrounding midpoints are in a homogeneous medium different from the solvent; *Boundary Grid Point* (BGP) ("▲" in Fig. 1), is subclassified as an: *Internal boundary* grid point—none of its midpoints is in the solvent but some have been assigned to regions having different dielectric constants. This corresponds to a point that lies at a boundary between atoms (or ob-

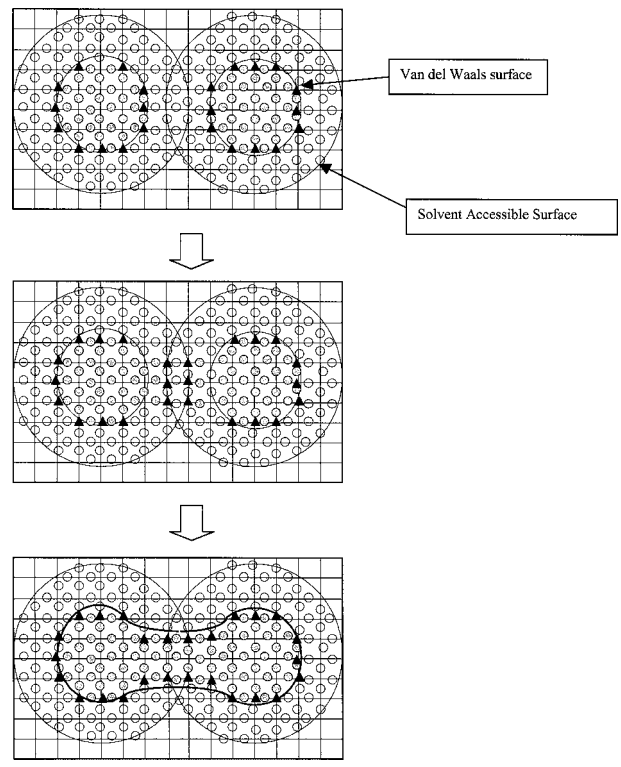

Figure 1. Illustration of molecular surface generation (smoothed numerical surface—SNS) by iterative growth from the Van der Waals surface. "●" are "in" midpoints, "○" are "out" midpoints and "▲" are boundary grid points.

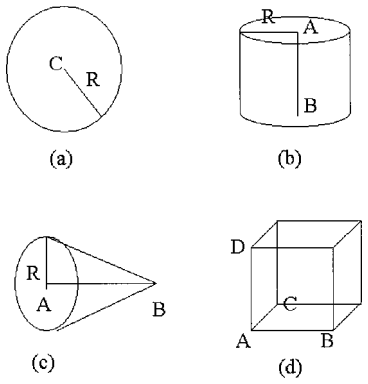

Figure 2. Objects included in DelPhi.

jects) having different dielectric properties; *External boundary* grid point—some midpoints are in solution and some are not.

Solvent-Accessible Surface (SAS)

We refer to the extended volume, EV, of an atom as the volume occupied by a sphere having the same center as the atom and a radius increased by the probe radius (the same rational is used for objects). The SAS is the surface that delimits the extended volume of the system, which contains molecules and objects together. After the VDWS has been mapped onto a lattice, defining the epsmap, the SAS is determined as the surface obtained expanding the VDWS by a probe radius (Fig. 1), and in turn, is used to map the MS onto the grid. A detailed description of the algorithm used to calculate the SAS is given in Appendix A.

Growing the Molecular Surface—the Smoothed Numerical Surface (SNS) Method

To calculate the MS, the distance to the SAS for each "out" midpoint that surrounds an external boundary grid point needs to be calculated. If the given "out" midpoint is closer to the SAS than a probe radius, this means that it is in a contact region and it will remain "out," i.e., it will retain its status as point that is accessible from the solution. Conversely, if it is further from the SAS than a probe radius, this implies that it is in an invagination where the probe cannot enter, and its status will be changed to "in" midpoint ("●" in the middle panel of Fig. 1). Such a midpoint is inaccessible from the solvent, and formally should be assigned to the interior. The process of calculating the relative position of a midpoint with respect to a surface is accomplished by simple surface distance calculations which, at least in the case of atomic spheres, is trivial and not very time consuming; this method is applied iteratively and takes a limited number of steps to converge.

Next, each neighbor grid point is checked to see if its status has changed (middle panel in Fig. 1). If it was a boundary grid point it may become an interior point, in which case it gets deleted from the list of boundary points, or it may retain its boundary status. If the neighbor was an external grid point then it may have been converted to a boundary grid point, BGP, and added to the list of boundary points to be checked. After reclassifying all its six neighbor midpoints, the BGP is checked again, and if it became an internal point, then it is deleted from the list of boundary points. The process then moves onto the next BGP in the list. After the first cycle of iteration, some of the boundary points will have lost their boundary status, and many new boundary points will have been added to the list. In the next iteration the boundary points to be checked are the ones that were added to the list in the first cycle. The iteration stops when no new boundary points were added in the previous cycle (the bottom panel in Fig. 1).

To summarize, the method is defined as semianalytical in the sense that the contact part is derived analytically, whereas the reentrant part is calculated in a numerical, albeit grid scale independent, way. The accuracy should be quite similar to that of a fully analytical method because the number of dots on the circles of intersection (COI—see Appendix A) is quite high (each circle of intersection is described by 384 points, which results in an almost continuous line). Such accuracy is much higher than the typical grid resolution used in FDPB calculations (grid size ranging from 1 to 0.2 Å). Therefore, the accuracy of the surface used in the FDPB calculations is determined almost entirely by the grid resolution.

The iteration converges very rapidly; for grid spacings greater than 0.5 Å convergence requires only two to three iterations. The epsmap modified in this way will form the input for a Poisson–Boltzmann solver (DelPhi in our case). DelPhi has a dynamic memory allocation feature that automatically calculates the grid-size and assigns memory to the various grid-based arrays depending on molecular size and grid resolution.[15, 16]

Solvation Free Energy Calculation

If one wishes to calculate the reaction field of a given molecule in a high dielectric solvent using the "traditional" method, two PB runs are required. First, half the sum of the product of the charge at each grid point by the corresponding grid potential is calculated for the molecule placed in the solvent. This term includes coulombic contribution, reaction field contributions, ionic contributions from the solvent, if any, and a self-energy term needed to assemble the charges. The last term, which is infinite for a classical electrostatic system, is finite in the finite difference representation. In a second step, the same calculation is carried out in a medium of uniform dielectric constant corresponding to that of the molecule. The two numbers are then subtracted so as to eliminate the self-energy terms, which have no physical meaning.

An alternative way to calculate the solvation free energy is to use the concept of induced charges. Classical electrostatics specifies that reaction field effects due to a dielectric boundary can be exactly reproduced by an appropriate distribution of induced polarization charge placed at the dielectric boundary.[17] In the Finite-Difference formalism, the induced charge at boundary point (i, j, k) is given by the numerical implementation of Gauss's law:

$$q^{\mathrm{ind}}(i, j, k) = -q^{\mathrm{real}}(i, j, k) + \frac{3h}{2\pi}\left(\phi(0) - \frac{1}{6}\sum_{l=1}^{6}\phi(l)\right) \quad (1)$$

where $q^{\mathrm{ind}}(i, j, k)$ is the induced charge in the (i, j, k) grid cube, $q^{\mathrm{real}}(i, j, k)$ is the fixed charge in the same cube, h is the grid spacing, $\phi(0)$ is the potential at the (i, j, k) grid point, and $\phi(l)$ is the potential at neighboring grid points. The reaction field energy is obtained with a simple Coulomb's law calculation between these

induced charges and the real charges as if they where in vacuum. There are several differences between the "traditional" and the induced charge methods. The most obvious is that the "traditional" method uses two runs and uses the grid positions for both real and induced charges, while the induced charge method needs only one run and uses the actual position for real charges and the optimized position for induced charges, i.e., grid polarization charges projected onto the MS. Moreover, the first method uses the potential at charged points, where it is inaccurate (the analytical potential of a point charge is infinitely large at the position of the charge). In contrast, the potential used to calculate the induced charges is at the surface of the molecule, where no fixed charges are present. Thus, the solvation energy calculated by the method of induced charges is more accurate.

The improved estimate of the reaction field energy, obtained by repositioning the induced grid charges on the exact molecular surface, is referred to as the *Scaled Solvation Energy*. The induced grid charges are projected on the Molecular Surface according to the following procedure: for points in the contact region, a parametric expression for the line joining the grid point and the closest exposed atom center is derived, and then the point is projected on the surface by means of a simple distance calculation. For points in reentrant regions, the grid-induced charge is first projected on the closest point of the SAS and then projected back on the MS along the line joining the SAS point and the boundary grid point in a similar way.

Geometrical Objects

Four types of geometrical objects can be inserted in the present version of DelPhi; spheres, cones, cylinders, and boxes (Fig. 2). Each of these objects is defined as a geometrical figure with a particular dimension, position, and dielectric constant. The objects are mapped into the grid by modifying the epsmap, i.e., assigning an appropriate value for dielectric constant at each midpoint. The total number of midpoints is, neglecting the borders of the space, three times the number of grid points, which usually ranges from 65^3 to 571^3. Thus, scanning any midpoint to check if it belongs to a particular object could make the calculation extremely slow.

Thus, one needs to know a restricted subset of midpoints that might belong to an object—this is equivalent to the problem of finding the smallest box, with its edges parallel to the coordinate axes, containing a given object. Secondly, one needs an algorithm that is able to decide whether a midpoint of that subset is internal to the current object or not. Formally, finding such a box is equivalent to the problem of finding the integer extremes of functions x, y, and z on a given geometrical shape. This, in general, could be accomplished by using techniques like Lagrange multipliers, but for the simple shapes of interest here, geometric analysis is sufficient. In Appendix B, we describe both the procedures to find the smallest box that can accommodate an object and to determine whether a given point belongs to an object.

Geometrical-Shaped Charge Distributions

The new version of DelPhi can handle various types of charge distributions that could be associated with different objects. The charge distributions that have been allowed may have the same overall shape of the objects previously described, and may also have a linear, rectangular, circular, or point form. A data structure has been built to contain: an ordinal number that labels the distribution; a field that defines whether the distribution is in the volume or on the surface; a field that defines whether the distribution is "free" or belongs to one of the previously inserted objects; a number called "distrtype" that determines the shape of the distribution, and finally the parameters related to the dimension and position of the charge distribution.

For both surface and volumetric charge distributions, the charge density is a constant, and the major problem is how to distribute this value over a surface or in a volume composed of discrete points, i.e., on the grid points used in the Finite Difference algorithm. The mathematical description of the methodology used to distribute the charge requires several transformations between different coordinate systems, and so involves a significant number of operations. This description, together with the criterion that decides the local charge density, is provided in Appendix C. The underlying logic of the procedure is as follows.

Consider that quantity of charge Q has to be distributed over an object. One has to find the corresponding surface or the volume charge density, depending on the task. The density must be such that the integral (over the surface or the volume of the object) is equal to Q. The best way to calculate the integral is to use an appropriate coordinate system (spherical for a sphere, cylindrical for a cylinder, etc.). Then the same integral must be transformed into grid "coordinates," and continuum density must be converted to partial charges assigned at appropriate grid points.

Results

Timings and Accuracy of the Surface Generation Algorithm

The SNS and *Inkblot* methods were compared for the protein, BPTI.[18] The quality of the epsmap surface was tested with a probe sphere with radius $R = 1.4$ Å, corresponding to that of a water molecule. The center of the sphere was positioned at each "in" grid midpoint and the sphere surface was modeled as equally distributed surface points. If any of the surface points of the probe sphere was found to be accessible from solution, then this "in" midpoint was counted as an error point. The ratio of the number of error "in" midpoints to the total number of "in" midpoints is the estimate of the error.

Results are shown in Figure 3. It can be seen that the *Inkblot* method generates a significant number of errors, especially at low grid resolution. At high grid resolution the number of points inside the template cube is also high; thus, the error decreases by approximately twofold, but still remains much higher than the error of the SNS method. Figure 3 demonstrates that the SNS method is insensitive to the grid resolution, and that it is extremely accurate, even at low grid resolution. Although SNS performance degrades slightly worst as the resolution increases, due to of the use of the same density of surface arc points for all resolutions, the error is negligible compared to the error of *Inkblot* procedure at the same resolution. This makes the SNS very effective for modeling large objects, where the grid resolution is limited by the size of the object.

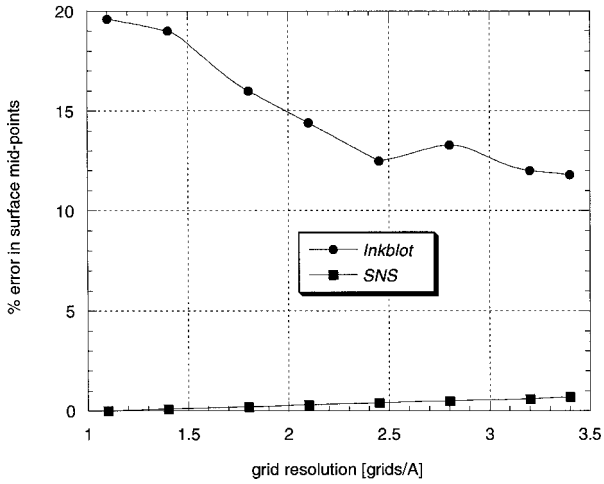

Figure 3. Error in surface midpoints for the *Inkblot* and SNS methods.

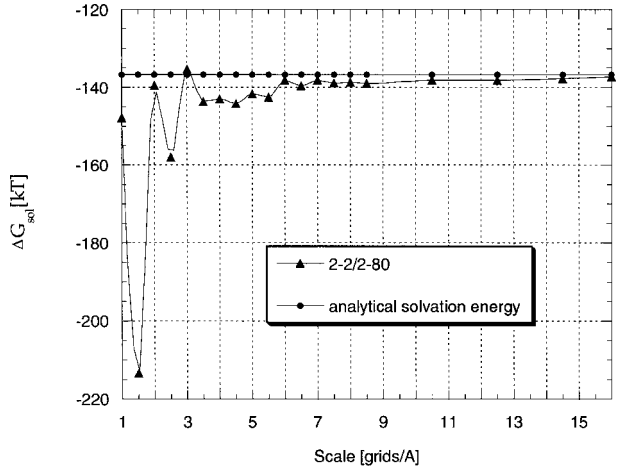

Figure 5. Solvation free energy (ΔG_{sol}) for model sphere as a function of grid resolution.

The efficiency of this method is illustrated in Figure 4, where the time required to construct the surface is plotted as a function of grid resolution. At higher grid resolutions the *Inkblot* method becomes highly inefficient. This is because a cube filled has to be generated for every BGP, so that the method scales as the fifth order of grid size (two orders for the BGPs and three from the filled cubes). In contrast, the SNS method is much less dependent on grid size, and its CPU time is almost a linear function of the grid resolution (Fig. 4).

Solvation Free Energies

The effect of grid resolution on the solvation free energy was determined for a model sphere and compared to the Born energy.[19] The radius of the sphere was taken as 1 Å, and the charge was 1 electron unit. The corresponding Born energy in water ($\varepsilon = 80$) is -136.74 kT at 25°C. The scaled solvation energy (not shown in Fig. 5) is essentially independent of grid resolution and is within 0.005 kT of the Born energy. In contrast, the solvation energy calculated by the traditional method, marked as 2-2/2-80 in Fig. 5, strongly depends on grid resolution. At low grid resolution it overestimates the solvation energy, and as the grid resolution increases the traditional method approaches the analytical solution.

Figure 6 tests the two methods on the protein BPTI. Radii and charges were taken from Parse set.[20] At low grid resolution the magnitude of the solvation free energies obtained from the two methods are quite different, while at higher resolution they appear to approach a similar value. Clearly, the scaled solvation free energies fluctuate far less, and are much more stable. However, in all applications it is important to ensure that the resolution of the lattice is sufficiently high.

The numerical description of the MS can be transformed into a triangulated surface mesh by using the Marching Cubes algorithm. We have developed a highly efficient surface triangulation program

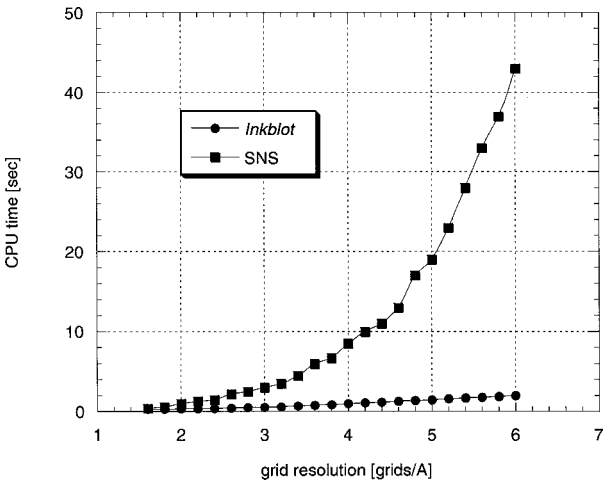

Figure 4. Timings for surface generation on a SGI INDY for BPTI, file 6PTI.[18]

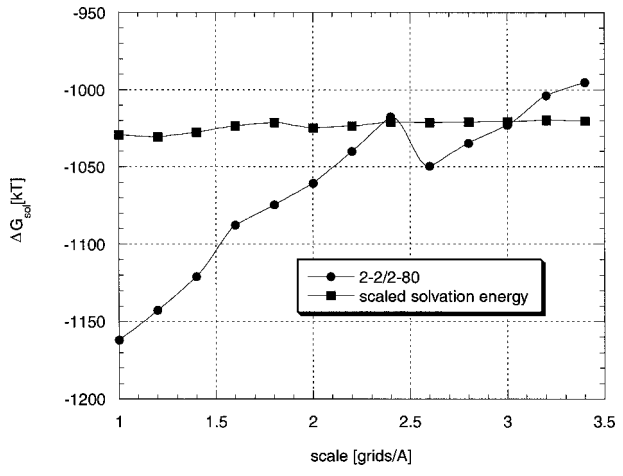

Figure 6. Solvation free energy (ΔG_{sol}) for BPTI as a function of grid resolution.

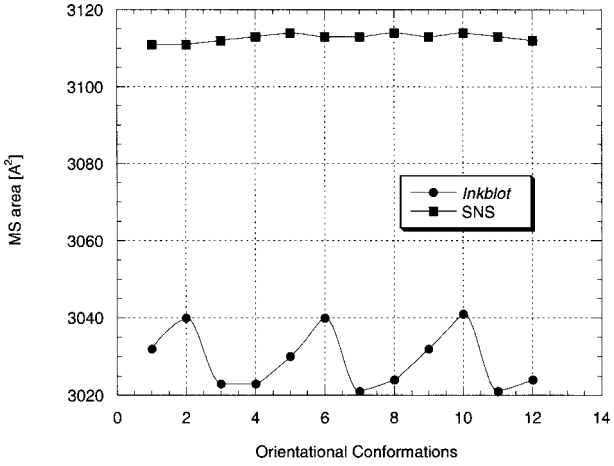

Figure 7. The molecular surface area of BPTI as a function of its positioning with respect to the grid. The grid resolution is 1.3 grids/Å.

for molecules in boundary element electrostatic calculations.[21] We tested the quality of the triangulated MS by computing the surface area of 6PTI and plotting it for several orientations of the same molecule within the grid (Fig. 7) and as a function of grid resolution (Fig. 8). Again, the surface produced by the *Inkblot* method shows considerable variability with respect to grid positioning, whereas the surface area calculated with the SNS method is essentially independent of the position of the molecule within the grid.

The SNS algorithm also converges much faster than the *Inkblot* method (Fig. 8). It essentially reaches its asymptotic value at grid resolutions higher than two grids/A, while the *Inkblot* method does not converge in the calculated grid interval (although the magnitude of the MS area asymptotically approaches the value calculated by SNS as resolution increases). This again demonstrates the advantage of the SNS method, especially in modeling large molecules or ensemble of molecules.

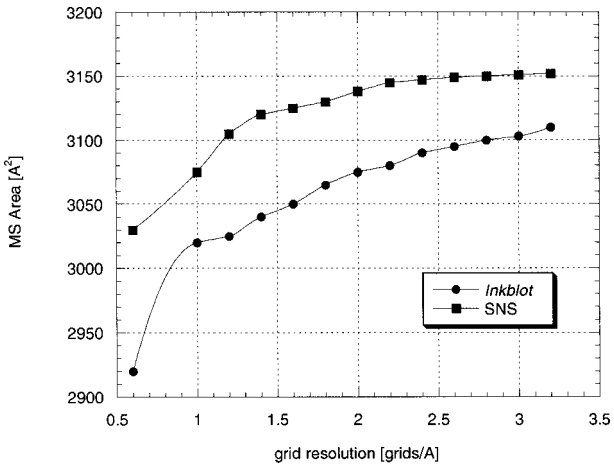

Figure 8. Molecular surface area of BPTI as a function of grid resolution.

Discussion

The numerical description of various surfaces associated with molecular systems has many applications in the physical and biological sciences. In particular, the molecular surface is widely used to define dielectric boundaries in electrostatic calculations based on the PB equation. In this article we have reported a new algorithm to build the molecular surface and to map it onto a three-dimensional lattice used to obtain finite difference solutions to the PB equation. The surface algorithm is extremely fast and precise, and results in improved performance of the PB solver. Further improvement is obtained by a new procedure to obtain electrostatic free energies from surface charges that are positioned on the molecular surface rather than on lattice points. It is worth noting that various improvements in numerical methods to solve the PB equation may be limited in their effectiveness if they use an imprecise mapping of the molecular surface.

A new version of DelPhi has been developed that incorporates these features. The program also allows the user to define simple geometrical objects (spheres, cylinders, cones, and parallelepipeds) as well as continuous charge distributions of different geometric shapes. This latter facility should greatly expand the applicability of the Finite Difference Poisson–Boltzmann method by allowing applications to systems that cannot be described in atomic detail or for which such a description may not be relevant. Another new feature is the ability to assign different dielectric constants to different regions of space, for example, every object can have a different dielectric constant.[16]

As described in a previous article, the new version of DelPhi can also treat mixed salt solutions consisting of monovalent and divalent ions and provide electrostatic free energies for such systems within the framework of the nonlinear PB equation.[16] Dynamical memory allocation allows for calculations on extremely large grids—a 571^3 grid was recently treated on an SGI Origin 2000. Academic users can obtain the program can obtain the program from trantor.bioc.columbia.edu; a commercial version will be available from Accelrys.

Appendix A

To derive the MS from the SAS, we need to be able to calculate the distance from a midpoint to the SAS. The essential information needed to determine the MS is contained in a small subset of the SAS, one reason is that the concave part of the MS coincides with the VDWS and has already been mapped on the grid. To save time and memory, we exploited this fact and found an algorithm that avoids the generation of a complete set of dots on the SAS. We generate only a set of dots lying on the *arcs* that arise from the intersection of the extended surfaces of the atoms/objects because it is these arcs that are used to generate the surface in the reentrant regions. The algorithm has to handle objects and atoms differently, and therefore, it makes a number of approximations based on the assumption that there will be many more atoms than objects and that the atoms will be smaller in size. If we are dealing with intersections between pairs of atoms as in Figure 1, or with the intersection of an atom and an object, a table of all intersecting pairs is constructed, and for each pair the center, radius, and normal

vector of its circle of intersection (COI) are found. A fixed template of evenly spaced dots, called vertices, is generated on the circumference of the COI. For each dot one determines if it is "buried" in the EV of any other atom or if it faces the solvent.

The following possibilities can arise for a consecutive set of two vertices: (1) they are both buried by the same atom, (2) they are buried by two different atoms, (3) one is buried and one is exposed, and (4) both are exposed. For case 1 neither dot belongs to the arc, and we assume that this is the case for any point on the circumference that lies between them. For all other cases the coordinates of the exposed vertices are saved in a data structure, and the midarc point is checked for occlusion by nearby atoms. The same bisection procedure is repeated until a certain resolution threshold is reached.

For object–object intersections a totally different algorithm has to be used because one can make no assumption about the shape of the intersection. The recursive cubing algorithm that we developed first partitions the space into a very coarse grid. It then calculates the number of objects in each grid cell and recursively refines the grid only in cells where the intersection might possibly occur. When the refinement has reached the resolution required, the vertices are calculated analytically with an appropriate procedure.

Application of these procedures for atoms and/or objects results in a set of vertices that belong to the SAS arcs, and that are more closely spaced in the vicinity of arc-to-arc intersections. The procedure that builds the MS, needs to know which atoms/objects or which vertices are in proximity of a particular point, because it is too time consuming to consider all existing pairs of atoms/objects and vertices. This is why a cubing algorithm that partitions the space into cells has been used. This procedure returns a structured look-up table for each cell in which the space has been subdivided, which specifies the elements, for example, atoms, objects, or vertices it contains, or which are located in one of the nearest neighbor cells. In practice, a cubing procedure that partitions the molecule into cells of dimension equal to the largest expanded atom diameter is actually used to find the closest atoms to a point, whereas the vertices are grouped into cells with the cell dimension equal to the probe radius.

Appendix B

Here we shall describe in some detail the criteria that have been used to decide if a point is inside or outside an object, and the determination of the smallest box with edges parallel to the coordinate axes containing the object.

Sphere (Fig. 2a)

For a sphere of radius R and centered in $C(x_c, y_c, z_c)$, the extreme values are: $x = x_c \pm R$, $y = y_c \pm R$, and $z = z_c \pm R$, and therefore, a point $P(x, y, z)$ is internal if:

$$(x - x_c)^2 + (y - y_c)^2 \leq (R + z - z_c)(R - z + z_c) \qquad \text{(B.1)}$$

Cylinder (Fig. 2b)

For a cylinder of radius R and axis $(A–B)$, a parametric representation of a point $P(x, y, z)$ belonging to its surface in a cylindrical coordinate system is given by:

$$P–B = R\cos\hat{\xi} + R\sin\theta\hat{\eta} + t(A–B), \qquad \text{(B.2)}$$

where $\{\hat{\xi}, \hat{\eta}\}$ is an appropriate orthonormal basis for the two-dimensional linear space orthogonal to the vector $(A–B)$, and θ and t are two parameters ranging, respectively, from 0 to 2π and from 0.0 to 1.0. In this way, the problem is reduced to finding the extremes of a function of two variables on a rectangular domain. We now describe the procedure for finding the extremes with respect to the x coordinate (the procedure for the y and z coordinates is identical).

Equation (B.2) can be solved with respect to x:

$$x(\theta, t) = x_B + R\sqrt{\hat{\xi}_x^2 + \hat{\eta}_x^2}\cos\left[\theta - \arctan\left(\frac{\hat{\eta}_x}{\hat{\xi}_x}\right)\right] + t(A–B)_x. \qquad \text{(B.3)}$$

Using this expression, one can calculate separately the extremes with respect to the θ and t variables, and find:

$$\begin{aligned} \max_t(\max_\theta x) &= x_B + R\sqrt{\hat{\xi}_x^2 + \hat{\eta}_x^2} + \max\left[0; (A–B)_x\right], \\ \min_t(\min_\theta x) &= x_B + R\sqrt{\hat{\xi}_x^2 + \hat{\eta}_x^2} + \min\left[0; (A–B)_x\right]. \end{aligned} \qquad \text{(B.4)}$$

In this way one can define the smallest dimensions of a box that will completely accommodate the cylinder.

Determining whether a given point is internal to the cylinder involves the following procedure. Let H be the projection of the point P on the cylinder axis, and let t be a real number such that $(H–B) = t(A–B)$. The first requirement is that t must belong to the [0; 1] interval. It can be shown that:

$$t = \frac{(P–B)\cdot(A–B)}{(A–B)^2}. \qquad \text{(B.5)}$$

The second requirement, which needs to be verified only if t lies in the right interval, is that the distance between P and H must be less than the radius of the cylinder. This can be determined with the following condition:

$$(P–B)^2 - t^2(A–B)^2 < R^2 \qquad \text{(B.6)}$$

The orthonormal basis $(\{\hat{\eta}, \hat{\xi}\})$ for the two-dimensional linear space orthogonal to the vector $(A–B) = (a, b, c)$, can be expressed with respect to the coordinates of $(A–B)$ (a, b, and c), as follows:

$$\begin{aligned} \hat{\xi} &= \frac{1}{\sqrt{a^2 + b^2}}(b, -a, 0), \qquad \text{and} \\ \hat{\eta} &= \frac{1}{\sqrt{a^2 + b^2}}\frac{1}{\sqrt{a^2 + b^2 + c^2}}\left(ac, bc, -a^2 - b^2\right), \end{aligned} \qquad \text{(B.7)}$$

for the case $a = b = 0$, the basis will be given by: $\hat{\xi} = (1, 0, 0)$, and $\hat{\eta} = (0, 1, 0)$.

Cone (Fig. 2c)

A cone can be defined by its spread angle 2α $(0 \leq \alpha < \pi/2)$ and its axis $(A–B)$. A parametric representation of a point $P(x, y, z)$

belonging to its surface is:

$$(P-B) = (1-t)|A-B|\tan\alpha[\cos\theta\hat{\xi} + \sin\theta\hat{\eta}] + t(A-B), \quad \text{(B.8)}$$

where θ and t range between 0 to 2π and between 0 to 1, respectively.

The expression for the x coordinate is:

$$x(\theta,t) = x_B + (1-t)|A-B|\tan\alpha\sqrt{\hat{\xi}_x^2+\hat{\eta}_x^2} \times \cos\left[\theta - \arctan\left(\frac{\hat{\eta}_x}{\hat{\xi}_x}\right)\right] + t(A-B)_x. \quad \text{(B.9)}$$

The extremes of x can be found as:

$$\max_t(\max_\theta x) = x_B + \max\left\{(A-B)_x; |A-B|\tan\alpha\sqrt{\hat{\xi}_x^2+\hat{\eta}_x^2}\right\},$$
$$\min_t(\min_\theta x) = x_B + \min\left\{(A-B)_x; -|A-B|\tan\alpha\sqrt{\hat{\xi}_x^2+\hat{\eta}_x^2}\right\}. \quad \text{(B.10)}$$

The same equations can be used for the other two coordinates.

A given point will be inside the cone if its position satisfies two conditions: (1) following the same procedure described for cylinder, one can define a variable t, which belongs to the interval [0; 1]:

$$t = \frac{(P-B)\cdot(A-B)}{(A-B)^2}. \quad \text{(B.11)}$$

(2) the distance between P and H must be less than the radius of the local section of the cone:

$$(P-B)^2 - (A-B)^2 < \left[t^2 + (1-t)^2\tan^2\alpha\right]. \quad \text{(B.12)}$$

The orthonormal basis ($\{\hat{\xi},\hat{\eta}\}$) is the same as in case of the cylinder [see (B.7)].

Parallelepiped (Fig. 2d)

The extremes in the case of a parallelepiped correspond to the intersection between planes, and, therefore, they are located on the vertices. In fact, from Calculus one knows that, given a convex function F defined on a convex set $D \subset \Re^n$, if $F \in C^1$, then extremes of F are located on the borders of D. In particular, if D is determined by the intersection of planes, the extremes lie on the vertices. In our case, each position coordinate is a weak convex function; thus, one knows where the extremes are to be found.

Given a box like the one in Figure 2d, the points A, B, C, and D allow for complete characterization of its dimension and position with respect to a coordinate system with origin O. The procedure for finding extreme values is based on simple geometrical rules, as is as the procedure of finding the status of each point.[22]

Appendix C

Surface Charge Distribution

A charge Q can be distributed over a given surface of area A by finding a surface charge density function $\sigma(x, y, z)$ such that:

$$\iint_{\text{surface}} \sigma(x,y,z)\,dx\,dy\,dz = Q. \quad \text{(C.1)}$$

For applications involving the Finite Difference algorithm, the charge Q has to be distributed over a set of grid points. Let $P(u, v) = (x, y, z)$ be the parameterization of the surface, and T the transformed domain such that $T = \{(u, v)/[x(u, v), y(u, v), z(u, v)] \in \text{surface}\}$. Then, surface charge density can be given as a function of new variables "u" and "v":

$$Q = \iint_T \sigma\big(P(u,v)\big)\big|\vec{N}(u,v)\big|\,du\,dv, \quad \text{(C.2)}$$

where $\vec{N}$ is the vector obtained by the cross product of

$$\left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right) \quad \text{and} \quad \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right),$$

which is normal to the surface in each point, and surface integral is taken over the transformation domain T. For sake of simplicity, we will define $\tilde{\sigma}(u, v) = \sigma(P(u, v))|\vec{N}(u, v)|$. Then (C.2) can be rewritten as:

$$Q = \iint_T \tilde{\sigma}(u,v)\,du\,dv = \sum_i \Delta q_i, \quad \text{(C.3)}$$

where "i" labels grid points of finite difference cubes that belongs to the surface of the object. Thus,

$$\tilde{\sigma}(u,v) = \frac{Q}{A}\big|\vec{N}(u,v)\big| \quad \text{(C.4)}$$

and

$$dS = \big|\vec{N}(u,v)\big|\,du\,dv \approx h^2, \quad \text{(C.5)}$$

and, finally, the partial charge on the grid points is:

$$\Delta q_i = \int_{u_i-\Delta u/2}^{u_i+\Delta u/2}\int_{v_k-\Delta v/2}^{v_k+\Delta v/2} \tilde{\sigma}(u,v)\,du\,dv \approx \frac{Qh^2}{A}. \quad \text{(C.6)}$$

Here A is the surface area of the object and h is the grid spacing. Applying the above formulas to the specific objects that were incorporated in DelPhi, one has:

Sphere

For a sphere:

$$P(\theta,\varphi) = \begin{cases} x = R\cos\varphi\sin\theta \\ y = R\sin\varphi\sin\theta \\ z = R\cos\theta \end{cases} \qquad (0 \le \varphi \le 2\pi,\ 0 \le \theta \le \pi); \quad \text{(C.7)}$$

the normal vector is $\vec{N}(\theta, \varphi) = R \sin\theta(\cos\varphi \sin\theta, \sin\varphi \sin\theta, \cos\theta)$, and

$$\begin{cases} \theta_j = j\Delta\theta \\ \varphi_k = k\Delta\varphi \end{cases}, \qquad \begin{cases} \Delta\theta = \dfrac{h}{R} \\ \Delta\varphi = \dfrac{h}{R\sin\theta_j} \end{cases};$$

$$\text{where } j = 1, \ldots, \text{int}\left[\frac{R\pi}{h}\right] \quad \text{and}$$

$$k = 1, \ldots, \text{int}\left[\frac{2R\pi \sin\theta_j}{h}\right], \tag{C.8}$$

$$\Delta q_i = \frac{Qh \sin\frac{\Delta\theta}{2}}{2\pi R} \approx \frac{Qh^2}{4\pi R^2}, \tag{C.9}$$

j and k are linked to i in a nested loop logic, where i spans over all the loops. Thus, Δq_i can be determined knowing three parameters—total charge Q, grid size h and the radius R of the sphere.

Cylinder

The parameterized surface of a cylinder is:

$$P(\theta, t) = B + R\cos\theta\hat{\xi} + R\sin\theta\hat{\eta} + t(A\text{–}B); \qquad (0 \le \theta \le 2\pi,\ 0 \le t \le 1). \tag{C.10}$$

In discrete notation

$$\begin{cases} \theta_j = j\Delta\theta \\ t_k = k\Delta t \end{cases}, \qquad \begin{cases} \Delta\theta = \dfrac{h}{R} \\ \Delta t = \dfrac{h}{|A\text{–}B|} \end{cases};$$

$$\text{where } j = 1, \ldots, \text{int}\left[\frac{2R\pi}{h}\right] \quad \text{and}$$

$$k = 1, \ldots, \text{int}\left[\frac{|A\text{–}B|}{h}\right], \tag{C.11}$$

and the partial charges assigned at grid points is:

$$\Delta q_i = \frac{Qh^2}{2\pi R|A\text{–}B|}. \tag{C.12}$$

Cone

For the lateral surface of a cone, one defines $R = |A\text{–}B|\tan\alpha$ and $r = (1-t)R$, and obtains:

$$P(r, \theta) = B + r\left[\cos\theta\hat{\xi} + \sin\theta\hat{\eta}\right] + (1 - r/R)(A\text{–}B); \qquad (0 \le \theta \le 2\pi,\ 0 \le r \le R); \tag{C.13}$$

$$\begin{cases} r_j = j\Delta r \\ \theta_k = k\Delta\theta \end{cases}, \qquad \begin{cases} \Delta r = h\sin\alpha \\ \Delta\theta = \dfrac{h}{r_j} \end{cases};$$

$$\text{where } j = 1, \ldots, \text{int}\left[\frac{R}{h\sin\alpha}\right] \quad \text{and}$$

$$k = 1, \ldots, \text{int}\left[\frac{2\pi r_j}{h}\right], \tag{C.14}$$

$$\Delta q_i = \frac{Qh^2\cos\alpha}{\pi|A\text{–}B|^2\tan\alpha}. \tag{C.15}$$

Parallelepiped

For the lateral surface of a parallelepiped, instead, each face has been treated as a subset of a geometric plane, being its normal vector one of the following: $\hat{\xi}$, $\hat{\eta}$ or $(D\text{–}B)/|D\text{–}B|$. Here,

$$\begin{cases} \xi_j = j\Delta\xi \\ \eta_k = k\Delta\eta \\ w_l = l\Delta w \end{cases}, \qquad \begin{cases} \Delta\xi = h \\ \Delta\eta = h \\ \Delta w = h \end{cases};$$

$$\text{where } j = 1, \ldots, \text{int}\left[\frac{|A\text{–}B|}{h}\right]$$

$$k = 1, \ldots, \text{int}\left[\frac{|C\text{–}B|}{h}\right], \quad \text{and}$$

$$l = 1, \ldots, \text{int}\left[\frac{|D\text{–}B|}{h}\right], \tag{C.16}$$

ξ_j, η_k, and w_l are the components along the vectors $(A\text{–}B)$, $(C\text{–}B)$, and $(D\text{–}B)$, respectively, while the incremental charge has the following value:

$$\Delta q_i = \frac{Qh^2}{\text{Area}_{\text{lateral}}}. \tag{C.17}$$

Volumetric Charge Distribution

The Finite Difference algorithm interpolates real charges on the nearest eight grid points. Thus, any charged point is split onto eight neighboring grid points that are separated by a grid spacing distance from one another. Thus, charging the volume of a given object requires an algorithm that generates points internal to the object. This can be accomplished by considering the parametric equation of the desired shape and using a procedure that derives $\{\hat{\xi}, \hat{\eta}\}$ as described previously. Spatial discretization of the charge is a critical factor. For example, a charge Q could be split into a set of partial charges positioned at distance h from one another, giving rise to a density of $1/h^3$. A suitable algorithm will readily interpolate these charges onto the closest grid points. Therefore, a density greater than this would be useless, because it would not affect the real resolution of the charge distribution on the grid. On the other hand, a too low a density would not meet the requirement of approximating a continuous distribution. But even a density close to $1/h^3$ might have the drawback of generating many charged points, which would dramatically slow down the finite difference algorithm. To deal with this problem, DelPhi implements, where possible, a procedure that assigns charges with a higher density in the proximity of the surface and a somewhat lower density in the core of objects. Examples are provided below for sphere and parallelepiped (the methodology for a cylinder and cone is given elsewhere in meta-language[22]).

Sphere

Spherical symmetry has the very useful property that the distance between each point and the surface can be derived from the distance to the center. Thus, the points that carry the partial charges are distributed on spherical shells starting at the distance $(R - h/2)$ from the center, i.e., the first layer with thickness $h/2$ from the surface. The rationale for this choice is that the interpolation routine for the charge position should not assign any point farther than

$h/2$ from the surface. The approximating value for the volume of each cell within the sphere is $r^2 \sin\theta \Delta r \Delta\theta \Delta\varphi$. The angles can be expressed as a function of distance r and grid spacing h as $\Delta\theta = h/r$ and $\Delta\varphi = h/(r \sin\theta)$, while Δr will take values of $h, 2h, 3h, \ldots$ as long as the point goes toward the center. Thus, assuming that Δr_{cell} is the "radial width" of each cell, we cannot say that it is equal to the difference $\Delta r_{\text{lin}} = (r_{i+1} - r_i)$ from two successive different cell centers. It is easy to show that:

$$(\Delta r_{\text{lin}})_i = \frac{(\Delta r_{\text{cell}})_{i+1} + (\Delta r_{\text{cell}})_i}{2}. \tag{C.18}$$

For instance, if $(\Delta r_{\text{cell}})_i = ih$, we obtain $(r_{i+1} - r_i) = -(i + 1/2)h$, and, consequently: $r_i = R - (i^2/2)h, i = 1, 2, \ldots$.

Thus, the charge that must be assigned to each cell is:

$$\begin{aligned}\Delta q_i &= \frac{Q}{\text{Vol}} \iiint_{\text{cell}} r^2 \sin\theta \, dr \, d\theta \, d\varphi \\ &= \frac{Q}{\text{Vol}} \int_{r_i-(1/2)\Delta r}^{r_i+(1/2)\Delta r} \int_{\theta-(1/2)\Delta\theta}^{\theta+(1/2)\Delta\theta} \int_{\varphi-(1/2)\Delta\varphi}^{\varphi+(1/2)\Delta\varphi} r^2 \sin\theta \, dr \, d\theta \, d\varphi \\ &= \frac{Q}{\frac{3}{4}\pi R^3} ih^2 \left(2 \sin\frac{h}{2r_i}\right)\left(r_i + \frac{i^2h^2}{12r_i}\right) \approx \frac{Q}{\frac{3}{4}\pi R^3} ih^3. \end{aligned} \tag{C.19}$$

The rationale for starting from the boundary of the solid is that one would not be able to determine the distance to the surface of the last assigned points, if one started from the center. Moreover, the number of cells into which the solid will be decomposed is not easily predictable, thus allowing a certain amount of charge to be unassigned. Specific routines are designed to account for this problem.

Parallelepiped

Let L_u, L_v, L_w, be the lengths of the three dimensions of the parallelepiped, such that $L_u \leq L_v \leq L_w$. The algorithm subdivides the shortest dimension into Δl_u, so that $\Delta l_u = ih$. The cell dimensions along the other directions, namely Δl_v and Δl_w, are spanned in two nested loops. For each of them, a Δl is chosen, which depends on both the distance from the sides along the current direction and the Δl corresponding to the immediately external loop, in such a way that $\Delta l_w \leq \Delta l_v \leq \Delta l_u \; \forall u, v, w$.

As in the previous case, the starting points are of the form $(L - (1/2)h)$, but an explicit expression like

$$l_u = \left[L_u - \frac{u^2}{2}h\right]$$

holds only for the u direction, whereas the others are defined at run time.

The charge to be assigned in each cell is:

$$\Delta q_i = \frac{Q \Delta l_u \Delta l_v \Delta l_w}{L_u L_v L_w}. \tag{C.20}$$

This procedure is far more complex to describe than to implement in any programming language, and that it does not require an excessive computational effort to be accomplished.

Glossary

Solvent Accessible Surface (SAS): surface obtained by increasing the van der Waals radii of all atoms by the probe radius. Molecular Surface (MS): The contact surface between the van der Waals surface and that defined by a probe sphere rolling on the surface of a molecule. Van der Waals surface (VDWS): VDWS of a molecule or of an object is determined from the VDW surfaces of individual atoms and/or surfaces of object. Inkblot Method: Procedure to construct the MS on a cubic grid which starts with the SAS on the grid and excludes all grid points that are within a probe radius from each of the boundary grid points. Smoothed Numerical Surface (SNS) method: The method reported in this article that effectively smoothes the MS defined on a cubic grid. Midpoint: In the finite difference algorithm, a "midpoint" is the middle point of each edge of grid cube. Epsmap: A map that specifies the value of the dielectric constant at each midpoint. Boundary Grid Point (BGP): Grid points for which some of the nearest neighbor midpoints lie in the solvent and the rest in the solute. The BGP can be subdivided into external and internal BGP, based on the nature of the neighbor's midpoints.

Acknowledgments

We thank to Kim Sharp for useful discussions. It is with profound sadness that the coauthors of Alessandro Chiabrera acknowledge his passing on November 8, 1999.

References

1. Warwicker, J.; Watson, H. C. J Mol Biol 1982, 157, 671.
2. Klapper, I.; Hagstrom, R.; Fine, R.; Sharp, K.; Honig, B. Proteins Struct Funct Genet 1986, 1, 47.
3. Davis, M. E.; McCammon, J. A. J Comput Chem 1989, 10, 386.
4. Nicholls, A.; Honig, B. J Comput Chem 1991, 12, 435.
5. Holst, M.; Baker, N.; Wang, M. J Commput Chem 2000, 21, 1319.
6. Lee, B.; Richards, F. M. J Mol Biol 1971, 55, 379.
7. Richards, F. M. Annu Rev Biophys Bioeng 1977, 6, 151.
8. Connolly, M. L. Science 1983, 221, 709.
9. Sanner, M.; Olson, A.; Spehner, J. Biopolymers 1996, 38, 305.
10. Liang, J.; Edelsbrunner, H.; Fu, P.; Sudhakar, P.; Subramanian, S. Proteins 1998, 33, 18.
11. Zauhar, R.; Morgan, R. J Comput Chem 1990, 11, 603.
12. Eisenhaber, F.; Argos, P. J Comput Chem 1993, 14, 1272.
13. You, T.; Bashford, D. J Comput Chem 1995, 16, 743.
14. Gilson, M. K.; Sharp, K. A.; Honig, B. H. J Comput Chem 1987, 9, 327.
15. Nicholls, A.; Honig, B. J Comput Chem 1991, 12, 435.
16. Rocchia, W.; Alexov, E.; Honig, B. J Phys Chem 2001, 105, 6507.
17. Jackson, J. D. Classical Electrodynamics; John Wiley & Sons: New York, 1962.
18. Fehlhammer, H.; Bode, W.; Huber, R. J Mol Biol 1977, 111, 415.
19. Born, M. Zeitschrift Phys 1920, 1, 45.
20. Sitkoff, D.; Sharp, K. A.; Honig, B. J Phys Chem 1994, 98, 1978.
21. Bharadwaj, R.; Windemuth, A.; Sridharan, S.; Honig, B.; Nicholls, A. J Comput Chem 1995, 16, 898.
22. Rocchia, W. New Devices for Information Processing: Quantum Mechanical Approach; Ph.D. Thesis, University of Genoa, 1999.