JCTC Journal of Chemical Theory and Computation

Improved Boundary Element Methods for Poisson−Boltzmann Electrostatic Potential and Force Calculations

Benzhuo Lu*,†,‡,§ and J. Andrew McCammon†,‡,§,||

Howard Hughes Medical Institute, Center for Theoretical Biological Physics, Department of Chemistry and Biochemistry, and Department of Pharmacology, University of California at San Diego, La Jolla, California 92093-0365

Received January 3, 2007

Abstract: A patch representation differing from the traditional treatments in the boundary element method (BEM) is presented, which we call the constant "node patch" method. Its application to solving the Poisson−Boltzmann equation (PBE) demonstrates considerable improvement in speed compared with the constant element and linear element methods. In addition, for the node-based BEMs, we propose an efficient interpolation method for the calculation of the electrostatic stress tensor and PB force on the solvated molecular surface. This force calculation is simply an $O(N)$ algorithm (N is the number of elements). Moreover, our calculations also show that the geometric factor correction in the boundary integral equations significantly increases the accuracy of the potential solution on the boundary, and thereby the PB force calculation.

1. Introduction

In the 20 years since the first boundary element method (BEM) paper on continuum electrostatics of biological systems,[1] computational scientists have made extensive contributions to the methodological generalization, optimization, and performance improvements in this area. These works include the extension from solving the Poisson equation (PE) and the linear Poisson−Boltzmann equation (PBE),[2,3] to solving the nonlinear PBE,[4−6] from the single molecule case to that of two or more molecules,[7−9] the implementation of accelerating techniques,[9−16] studies on the conditioning of the produced linear system,[9,14,17] the method for rigorous force calculations,[8,9,18] and applications to molecular mechanics or dynamics simulation.[19−22] However, the practical utility of BEM is still very limited in biological electrostatic studies, though it has been widely recognized in engineering applications. Despite its high accuracy of solution and the reduction of system degrees of freedom relative to other numerical approaches such as the finite difference and finite element methods, BEM has its own difficulties: challenges of mesh generation for biomolecules, singular and hypersingular surface integrals, and numerous integral operations. The main hurdle to its practical usage is the speed. We recently made progress on this by using a new version of the fast multipole method (FMM)[9] to achieve a CPU performance comparable or even superior to that of some other numerical methods. This work also focuses on the speed improvement. Two typical low-order BEMs are the constant element method (unknowns are constant in each element) and the linear element method (unknowns are located at the nodes of an element). In this work, we construct a boundary "patch" (or element) differing from the normal facet patch (e.g., triangular element) based on the same mesh and also make a constant unknown approximation in this patch. We call this the constant "node patch" method and demonstrate that this method allows considerable acceleration of the BEM calculation and maintains a similar calculation accuracy. In addition, the details of a fast interpolation method are given for the PB force calculation for the node-based BEMs including the linear element methods and our node patch method. This interpolation method is simply of order N and much faster than our previous $O(N^2)$ algo-

* Corresponding author phone: 858-822-0168; fax: 858-534-4974; e-mail: blu@mccammon.ucsd.edu.
† Howard Hughes Medical Institute.
‡ Center for Theoretical Biological Physics.
§ Department of Chemistry and Biochemistry.
|| Department of Pharmacology.

10.1021/ct700001x CCC: $37.00
Published on Web 04/05/2007

rithms.[8,18] Finally, we also show that the geometric correction to the normal boundary integral equation (nBIE) leads to a much more accurate potential solution, and thereby a much more accurate force calculation.

2. Boundary Integral Equation

2.1. Normal Boundary Integral Equation. In the widely used BEMs to solve the PBE (see refs 2 and 7), the electrostatic potential is expressed as a boundary integral form

$$\phi_p^{\text{int}} = \oint_S \left[G_{pt} \frac{\partial \phi_t^{\text{int}}}{\partial n} - \frac{\partial G_{pt}}{\partial n} \phi_t^{\text{int}} \right] \mathrm{d}S_t + \frac{1}{D_{\text{int}}} \sum_k q_k G_{pk}, \qquad p,k \in \Omega \quad (1)$$

$$\phi_p^{\text{ext}} = \oint_S \left[-u_{pt} \frac{\partial \phi_t^{\text{ext}}}{\partial n} + \frac{\partial u_{pt}}{\partial n} \phi_t^{\text{ext}} \right] dS_t, \qquad p \in \bar{\Omega} \quad (2)$$

where ϕ_p^{int} is the potential at position p inside the molecular domain Ω, q_k is the kth source point charge of the molecule, $S = \partial\Omega$ is its boundary, e.g., solvent-accessible surface, ϕ_p^{ext} is the potential at position p outside domain Ω, D_{int} (D_{ext}) is the interior (exterior) dielectric constant, and n is the outward normal vector at the integral point t. G and u are the fundamental solutions of the PE and the PBE, respectively, where r_{pq} denotes the distance between two points p and q

$$G_{pq} = \frac{1}{4\pi r_{pq}} \quad (3)$$

$$u_{pq} = \exp(-\kappa r_{pq})/4\pi r_{pq} \quad (4)$$

and κ is the reciprocal of the Debye−Hückel screening length determined by the ionic strength of the solution.

When point p approaches surface S, and the boundary conditions $\phi^{\text{int}} = \phi^{\text{ext}}$ and $D_{\text{int}}(\nabla\phi^{\text{int}}\cdot n) = D_{\text{ext}}(\nabla\phi^{\text{ext}}\cdot n)$ are considered, eqs 1 and 2 become a set of self-consistent boundary integral equations (denoted as nBIEs)

$$\frac{1}{2} f_p = \oint_S^{\text{PV}} \left[\epsilon G_{pt} h_t - \frac{\partial G_{pt}}{\partial n} f_t \right] \mathrm{d}S_t + \frac{1}{D_{\text{int}}} \sum_k q_k G_{pk}, \qquad p \in S \quad (5)$$

$$\frac{1}{2} f_p = \oint_S^{\text{PV}} \left[-u_{pt} h_t + \frac{\partial G_{pt}}{\partial n} f_t \right] \mathrm{d}S_t, \qquad p \in S \quad (6)$$

where PV denotes the principal value integral to avoid the singular point when $t \to p$ in the integral equations, $f = \phi^{\text{ext}}$, $h = \nabla\phi^{\text{ext}}\cdot n$, and $\epsilon = D_{\text{ext}}/D_{\text{int}}$.

In our former work,[8] we extended this form to an interacting system with an arbitrary number of molecules and gave a set of corresponding iterative equations for force calculation.

2.2. Geometric Modification. For a nonsmooth boundary as represented by a practically discretized mesh, the following rigorous BIEs can be obtained when p approaches the boundary using a limiting process from the original eqs 1 and 2:

$$\alpha_p f_p = \oint_S^{\text{PV}} \left[\epsilon G_{pt} h_t - \frac{\partial G_{pt}}{\partial n} f_t \right] \mathrm{d}S_t + \frac{1}{D_{\text{int}}} \sum_k q_k G_{pk}, \qquad p \in S \quad (7)$$

$$(1 - \alpha_p) f_p = \oint_S^{\text{PV}} \left[-u_{pt} h_t + \frac{\partial G_{pt}}{\partial n} f_t \right] \mathrm{d}S_t, \qquad p \in S \quad (8)$$

where the coefficient constant α_p is dependent on the local surface geometry of the node p. For a smooth surface, α_p is $^1/_2$. For a vertex of a polyhedron, which is not a smooth point of the surface, the coefficient α_p is equal to $A_p/4\pi$, where A_p is the interior solid angle at the node. The constant $^1/_2$ is as usually used in the previous BEM PB works, but this work will show that the use of the geometry-dependent coefficient does make a significant improvement in the PB solution, especially the potentials on the surface and the PB force on the molecule. For a mesh with flat elements, the interior solid angle at node p can be calculated by the following formula: where n_p is the total number of neighboring elements of p,

$$A_p = \sum_i^{n_p} \beta_i \text{ - } (n_p - 2) * \pi \quad (9)$$

and the interior dihedral angle β_i at an attached ith edge formed by two neighboring faces with normal vectors, for example, n_1 and n_2, respectively, satisfies $\beta_i = \arccos(-n_1\cdot n_2)$.

There are only weak and strong singular integrals appearing in the above equations, which can be analytically treated by using coordinate transformation and series expansion.[18,23,24]

2.3. Derivative Boundary Integral Formulation. By linearly combining the derivative forms of eqs 5 and 6, the derivative BIEs (dBIEs) can be obtained:[3]

$$\left(\frac{1}{2\epsilon} + \frac{1}{2} \right) f_p = \oint_S^{\text{PV}} \left[(G_{pt} - u_{pt}) h_t - \left(\frac{1}{\epsilon} \frac{\partial G_{pt}}{\partial n} - \frac{\partial u_{pt}}{\partial n} \right) f_t \right] \mathrm{d}S_t + \frac{1}{D_{\text{ext}}} \sum_k q_k G_{pk}, \qquad p \in S \quad (10)$$

$$\left(\frac{1}{2} + \frac{1}{2\epsilon} \right) h_p = \oint_S^{\text{PV}} \left[\left(\frac{\partial G_{pt}}{\partial n_0} - \frac{1}{\epsilon} \frac{\partial u_{pt}}{\partial n_0} \right) h_t - \frac{1}{\epsilon} \left(\frac{\partial^2 G_{pt}}{\partial n_0 \partial n} - \frac{\partial^2 u_{pt}}{\partial n_0 \partial n} \right) f_t \right] \mathrm{d}S_t + \frac{1}{D_{\text{ext}}} \sum_k q_k \frac{\partial G_{pk}}{\partial n_0}, \qquad p \in S \quad (11)$$

where n is the unit normal vector at point t and n_0 is the unit normal vector at point p.

The dBIEs lead to a well-conditioned system of algebraic equations. These dBIEs have been extended to systems with arbitrary numbers of biomolecules, and the solution has been accelerated with an efficient FMM in our former work.[9]

For a discretized mesh that is not smooth in numerical realization, the geometric correction for dBIEs has not been well-implemented in the BEM PB solver, though we have demonstrated the success of applying hypersingular integration techniques to calculate the gradient of potential on the surface as a postprocessing procedure.[18] The direct evaluation

of all the singular integrals and the other terms such as the additional free terms that appear in the geometrically modified BIEs have been theoretically studied.[24,25] The implementation including both the free-term modification and the hypersingular integration techniques for the derivative BEM (dBEM) PB solver has not been demonstrated and is beyond the scope of this work.

3. A Constant "Node Patch" Method

One of the methods often used in BEM is that the unknowns *f* or *h* on an element (face) are treated as constants. Therefore, the number of unknowns is equal to the number of elements. This is the lowest-order BEM method, the so-called "constant element" approach. This treatment is convenient for numerical implementation. Another widely used treatment is the linear element method, in which *f* and *h* at a position in an element are obtained by linear interpolation from the values (unknowns) on the three nodes (for a triangular element). Therefore, the number of unknowns is equal to the number of nodes and is almost half of the number of elements for a triangulated surface (according to Euler's formula for a polyhedron). The disadvantage of a node-based method is the introduction of additional complexities in numerical implementation. The advantage is that it can achieve better accuracy of the solution and seems to gain higher computational speed because the number of unknowns is reduced by about half.

In either kind of the BEMs, most operations are the far-field integrations. When an element patch ΔS_i (also denoted as its area) is far from the evaluation point *p*, in the constant element treatment, the boundary integrals on this patch are approximated as

$$\int_{\Delta S_i} G_{pt} h_t \, dS \simeq h_i G_{pi} \Delta S_i \tag{12}$$

$$\int_{\Delta S_i} \frac{\partial G_{pt}}{\partial n} f_t \, dS \simeq f_i \nabla G_{pi} \cdot n_i \Delta S_i \tag{13}$$

where n_i is the unit normal vector of the *i*th element and the *i*th position is taken as the element center. In the above formulas, constant approximations of *f* and *h* are used, which is the meaning of "constant element" treatment, and the values of function *G* and its derivative are also approximated as constants on the integral patch because of the far-field approximation. For near-patch integration, a normal quadrature method is used. Similar treatments apply to the integrations for the kernel *u* and its derivative, as well as for the other second-derivative terms if the dBIEs are used.

Here, we construct a patch around each node, instead of directly using the facet patch (element), and suppose that *f* and *h* are constants on this new "node patch". We call this the constant "node patch" treatment. A simple way to construct these new patches is illustrated in Figure 1 in which an example "node patch" at the *i*th node that has five neighboring elements is constructed. All the centroids $\{O_l, l = 1, ..., 5\}$ of the five adjacent triangles and the midpoints $\{C_l, l = 1, ..., 5\}$ of the attached five edges are listed, and then the area formed by $\{O_1,C_1,O_2,C_2,...O_5,C_5,O_1\}$ is the new patch that we want. Therefore, there is one-third of the area

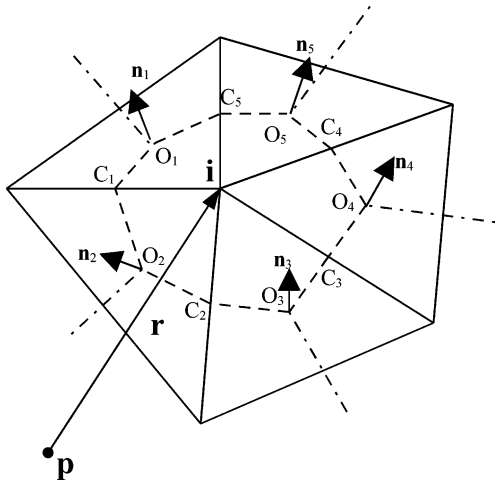

Figure 1. "Node" patch constructed on a triangular mesh. *O* and *n* are the centroid and normal vector of an element, repsectively, and *C* is the middle point of an edge.

of each neighboring triangle occupied by the "node patch" ΔS_i. The other patches are similarly constructed. All the node patches connect and cover the whole surface. Now, the far-field integrals on the new patch ΔS_i become

$$\int_{\Delta S_i} G_{pt} h_t \, dS \simeq h_i G_{pi} \Delta S_i^a \tag{14}$$

$$\int_{\Delta S_i} \frac{\partial G_{pt}}{\partial n} f_t \, dS \simeq f_i \nabla G_{pi} \cdot \Delta S_i^b \tag{15}$$

where, supposing all neighboring elements of *i*th node form a set {L},

$$\Delta S_i^a = \frac{1}{3} \sum_{l \in \{L\}} \Delta S_l \tag{16}$$

$$\Delta S_i^b = \frac{1}{3} \sum_{l \in \{L\}} \Delta S_l n_l \tag{17}$$

where n_l is the unit normal vector of the *l*th neighboring element, ΔS_l is the area of the *l*th adjacent triangular element, and ΔS_i^b here should also be considered as a vector. As in the constant element treatment, for near-patch integration, a normal quadrature method is used. Similar treatments apply to the integrations for the kernel *u* and its derivative, as well as for the other second-derivative terms if the dBIEs are used.

There are three main advantages of this "node patch" treatment in the BEM. First, as aforementioned, the unknowns are reduced by almost half relative to the constant element method, and the computational time on solving the resulting linear system in each iteration step or in direct matrix inversion is also reduced by the same factor. The only additional computation is a preprocessing of some geometric coefficients ΔS_i^a and ΔS_i^b as in eqs 16 and 17, which can be saved for repeated usage in the iterative solving procedure. The CPU cost of this preprocessing represents a negligible portion of the whole PBE solution time. However, this time reduction (by about half) due to reduced unknowns in our constant node patch method does not hold in the normal linear element approach, though the number of unknowns

is the same as the number of nodes. The reason is that there are more additional numerical operations (interpolations and quadratures) for the integration on every element. Therefore, in the iterative solving procedure, though the unknowns in the linear element approach are reduced by about half, the total computational time is not reduced much. If direct matrix inversion is used, time saving is expected due to the fact that the linear system size is reduced and all the matrix coefficients are just calculated once; however, the iterative method has to be used for any sizable biological system.

Thus, compared with the constant element method, the constant "node patch" method can save considerable computational time. This will also be demonstrated in numerical tests. And because both methods use similar assumptions (constant f and h in a patch), and are based on the same mesh resolution, there is no loss in the accuracy of the solution. It is worth noting that the linear element treatment that uses linear interpolation to get f and h in a patch instead of using a constant approximation should achieve relatively higher computational accuracy.

The second advantage, which is not so explicit, lies in the case when it is necessary to store the matrix coefficients from the "near points" (local list) integration. The local list is typically given by cutoff criteria or the FMM local neighbor-box list output if FMM is implemented. It is found that, relative to the linear element method, our constant "node patch" method tremendously saves time searching and locating the local list in order to store the calculated coefficients in a practical matrix storage format, for example, the Harwell−Boeing sparse matrix format (HB),[26] or modified sparse column (row) format. In the linear element treatment, the CPU cost for this part is large enough to be comparable with the whole PBE solution time. The BEM mesh is a kind of unstructured mesh. In the linear element method, each matrix coefficient relates to a node pair (the unknowns are located at nodes), while the local list is normally given as an element list because the integrations are performed on each element and need the nodal value interpolations. Therefore, the node pair information needs to be searched from the element list. Another fact to be considered is that each node is normally shared by several elements; therefore, each matrix coefficient corresponding to a node pair has contributions from the integrations on all the connected elements. Due to these reasons, storing the calculated coefficients (from geometric integrals) requires complex embedded loops to search in both element and node indices to locate the position in the storage frame. For example, four embedded loops in our original implementation are required for the whole matrix storage, which makes the coefficient saving not obviously more efficient than the direct matrix−vector multiplication calculation in every iteration step. However, for our node patch method, or similarly the constant element approach, the local list is also a node list; therefore, the location in the sparse matrix of the coefficient corresponding to each local point is straightforward by counting the local nodes and looking at their indices. This can finally save CPU time in the whole iterative solution procedure.

The third advantage, as with other node-based methods such as the linear element method, is that it is convenient to compute the potential and its gradient (not only the normal derivative) at any position near or on the molecular surface for stress and force calculation through an interpolation method. As shown in the following section, a simple linear interpolation can be performed in a constructed prism, where only the potential f and its normal derivative h on the nodes are required. It might be more complicated to calculate the gradient of potential at an arbitrary surface point in the constant element method.

4. Interpolation Method for Calculation of the Gradient of the Potential

Our previous work[8,18] introduced two rigorous methods, a variational approach and a hypersingular integral method, for the PB force calculation. Both are order N^2 algorithms. Here, we'll introduce an interpolation method that is simply of order N. In an ionic solution, the full stress tensor on the boundary should include an additional term accounting for the ionic pressure besides the conventional Maxwell stress tensor.[27] It is

$$T_{ij} = D_{\text{ext}} E_i E_j - \frac{1}{2} D_{\text{ext}} E^2 \delta_{ij} - \frac{1}{2} D_{\text{ext}} \kappa^2 \phi^2 \delta_{ij} \qquad (18)$$

where E is the electrostatic field and δ_{ij} is the Kronecker δ function. Therefore, to obtain the boundary stress tensor, the derivative of the potential, that is, the negative of E on the boundary, should be known.

To get the gradient of the potential on the surface, a potential function in the vicinity of the molecular surface can be constructed using an interpolation method. From the potential data on the nodes of a triangulated surface, we can construct a C^1 or C^2 modeled potential field on the surface, for example, by using piecewise trivariate polynomials that are defined on a three-dimensional triangulation called the simplicial hull and defined over the domain surface.[28] Because, in our case, we not only know the potential values but also know their normal derivatives on the nodes, a more efficient and simple way can be taken to construct the approximated potential functions on the surface by using both the potential and its normal derivative values. The idea is to construct a small three-sided prism attached on each triangular element on the surface in which a piecewise interpolated function is defined. The prism is defined by the triangular element and the three normal vectors on the nodes as shown in Figure 2.

A concern that has to be tackled is that the normal vector $\mathbf{V}$ at a node (vertex) on a surface mesh is not defined due to the discontinuity. Hence, the calculation or assignment of this normal vector is not trivial. The normal vector at the node is also required in the node-based dBEMs including the node patch dBEM presented in this work, because the vector n_0 at the point p appears in the dBIEs 10 and 11. Several averaging methods may be used, such as direct averaging from the neighbor facet normals, neighbor angle weighting, neighbor facet area weighting, or using weights from facet normals and supposing adjacent vertices are inscribed in a sphere.[29] If the mesh is generated from software

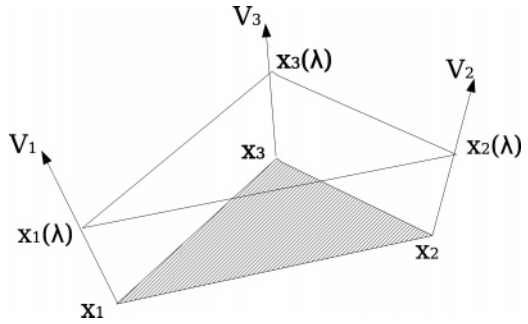

Figure 2. Prism constructed on a triangular element. The shadowed triangle is one of the boundary elements; V_1, V_2, and V_3 are three unit normal vectors at the three nodes; and λ is a parameter to describe the third dimension of the prism.

such as MSMS[30] that can output the normal vectors, the data can be directly used and need not be calculated again. We have tested these methods for vertex normal calculations for both spherical cavity and protein cases. Preliminary results seem to show that no one method is superiorly accurate and preferred. The overall PB solutions obtained with different normal vector definitions do not differ much except for some points where the local geometry is very uneven, and the final energy calculations are close to each other. Here, the simplest way is adopted to calculate the normal vector:

$$cV_p = \sum_{\text{adjacent faces}} n_i \tag{19}$$

where n_i is the normal vector of the adjacent ith element of node p and c is a factor used to normalize the unit vector V_p.

Any point in a triangle can be described using two parameters (parametric coordinates), for example, (ξ, η). On the normal direction, we use another parameter λ to locate the position. For example, a point on the normal direction of node i has coordinates $X_i(\lambda) = X_i + \lambda V_i$, $i = 1-3$, where X_i is the position of ith node and V_i is its unit normal vector. It is easy to see that $X_i(0) = X_i$. And the potential value at $X_i(\lambda)$ can be approximated as $f_i(\lambda) = f_i + \lambda h_i$ when λ is small enough. Here, it is supposed that the potential derivative in the normal direction V_i at the ith node is equal to the PB solution h_i. We'll discuss this later. Then, for the same λ at three nodes, the three points $\{X_1(\lambda), X_2(\lambda), X_3(\lambda)\}$ form a new parametric triangle layer. Therefore, the position and function values at any point in this layer can be interpolated using the parametric coordinates (ξ, η, λ) as

$$X(\xi,\eta,\lambda) = (1 - \xi - \eta)(X_1 + \lambda V_1) + \xi(X_2 + \lambda V_2) + \eta(X_3 + \lambda V_3) \tag{20}$$

$$f(\xi,\eta,\lambda) = (1 - \xi - \eta)(f_1 + \lambda h_1) + \xi(f_2 + \lambda h_2) + \eta(f_3 + \lambda h_3) \tag{21}$$

where f_i denotes the potential at the ith node, and h_i denotes its derivative in the normal direction V_i. The trivariate function $f(\xi,\eta,\lambda)$ is then constructed to model the potential in the vicinity of the surface element. When $\lambda = 0$, the triangle layer is reduced to the surface element. All the prisms connect and cover the whole molecular surface without overlap. The interpolated function is piecewise C^1 continuous on the whole molecular surface; therefore, the gradient can be calculated everywhere. Because we only want to calculate the potential gradient on the molecular surface, $\lambda = 0$, this makes the approximation of the interpolation (small λ) on the normal direction acceptable. The gradient of the potential at any point $(\xi, \eta, \lambda = 0)$ on the surface can be obtained from the following relationship:

$$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = J^{-1} \begin{pmatrix} f_\xi \\ f_\eta \\ f_{\lambda=0} \end{pmatrix} \tag{22}$$

where J is the coordinate transformation matrix from (ξ, η, λ) to (x, y, z) at $\lambda = 0$. These function derivatives are the required fields in eq 18.

Now, the Maxwell tensor $\mathbf{T}$ can be calculated using eq 18, and the PB force F and torque M acting on a molecule are calculated by integrations

$$F = \int_S \mathbf{T}(x)\, \mathrm{d}S(x) \tag{23}$$

$$M = \int_S r_c(x) \times [\mathbf{T}(x)\, \mathrm{d}S(x)] \tag{24}$$

where $r_c(x)$ is a vector from the center of mass of the target molecule to the surface point x and the dot and cross vector multiplication are applied to the vector and tensor quantities.

It is worth it to note that this force calculation procedure can be further improved. As stated above, the PB solution h_i is also taken as the potential derivative in the node normal direction V_i, which is a rough approximation. In the BEM formula, h should be considered as the projection of the potential gradient in the surface normal direction at each quadrature (integration) point. Therefore, a better approximation is that the value h_i at node i is the projection of the potential gradient (not known) at the node in the adjacent element normal direction (all the projections on the adjacent elements equal the same h_i due to the linear element treatment). Normally, the projection of the potential gradient on the node normal vector V_i may deviate from the value h_i to some extent, which is dependent on the mesh geometry. Therefore, the potential gradient at each node, denoted as $\vec{H}_i$, can be fitted from the above consideration. We calculate the value and direction of each $\vec{H}_i$ through minimizing the quantity

$$\sum_{l \in \{L\}} (\vec{H}_i \cdot n_l - h_i)^2 \tag{25}$$

where $\{L\}$ is the collection of the adjacent elements of the ith node. And it is found from the above interpolation procedure that the prism construction actually only needs an (arbitrary) vector, not necessarily the normal vector, and the directional derivative on this vector at each node. Therefore, the interpolation procedure for force calculation based on the set $\{\vec{H}_i\}$ is similar to that on the set $\{V_i, h_i\}$ and should lead to better results, especially for very irregular mesh geometry.

5. Results and Discussion

5.1. Performance Comparison of Different Patch Treatments. In the iterative PBE solver, the surface integration

Table 1. Performance Comparison of the Constant Node Patch (First Line) and Linear Element (Second Line) Approaches[a]

mesh size (vertices and faces)	CPU time dissection			$E_{solvation}$ (kcal/mol)	error in f_i (%)	error in h_i (%)
	coeff saving	GMRES	total			
162, 340	0.01	0.07	0.10	−82.3	9.1	2.8
	0.11	0.08	0.20	−84.7	8.0∼9.3	3.6∼6.5
642, 1280	0.08	0.37	0.51	−81.3	4.5	1.3
	0.42	0.40	0.83	−82.7	3.6∼5.0	1.4∼3.3
4841, 9678	0.88	9.39	11.49	−544.3		
	13.17	10.07	26.23	−554.8		
7525, 15046	6.38	11.90	20.05	−531.5		
	33.53	12.33	50.49	−537.8		

[a] Both calculations use the derivative BIE forms, and seven quadrature points are taken for each local element integration in the linear element method. The first two meshes are for spherical cavity calculations, the last two meshes for a protein (fasciculinII). Protein surface mesh is generated using the program MSMS.[30] As a reference, the exact Born solvation energy $E_{solvation}$ of a unit spherical cavity is −80.9 kcal/mol, and the protein solvation energy is decreased to −522.0 kcal/mol when higher resolution mesh is used.

is performed during each iterative step (implicit matrix−vector multiplication). To accelerate the speed and also consider the memory requirement, some matrix coefficients related to the local integration (corresponding to a sparse matrix) are saved, while the far-field integrations are directly calculated. The task for the coefficient saving includes two main parts: coefficient computation and its position placement in a practical matrix-saving format. A compressed sparse column format is used in the current code (similar to HB format). When the local matrix coefficients are saved, the main work in each GMRES iteration step is the far-field integration. This part can be accelerated by the already implemented FMM.[9]

Table 1 provides a comparison of the calculation speed and accuracy between the constant node patch and the linear element BEMs. Calculations are made for both a spherical cavity case and a protein case. As an example, Figure 3 shows the surface mesh and potential of the protein molecule (fasciculinII) calculated using the current node patch method. It is found that in all the cases the CPU time costs in the GMRES iterations of both BEMs are very close. Because we just use one quadrature point in the linear element method, that is, equivalent to the constant element treatment, for far-element integration, this only makes a small difference in the calculation speed (when FMM is used) compared with the constant node patch treatment. However, there is much difference in the sparse matrix-saving step between the two approaches, and it is also found that this is basically the origin of the difference of the total CPU times for the PBE solution by the two BEMs. It is worth it to point out that in the linear element method the most time-consuming step lies in the position placement of the matrix coefficients to be stored, which is much more expensive than the coefficient calculation operations. The CPU time cost of this step is comparable with the total time cost of the PBE solution. As mentioned in the method section, the reason is that in the linear element method both the (unstructured) node and element indices are required to search for the coefficient storage positions, and we have to use several embedded search loops in the code to reach these. In the constant node patch method, on the other hand, each calculated coefficient to be saved only corresponds to one node in the local node list; therefore, the position location is straightforwardly indicated by the node index itself. This significantly saves CPU time and becomes practical if some matrix coefficients need to be stored.

The accuracies of both potential solution and energy calculation are also well-maintained in the node patch method. In the sphere case, the solution error with the constant node patch approximation is stable (nearly stays at a same value), which is reasonable, while the error varies over a range with the linear element method due to different shapes of the triangular elements on the sphere. In addition, the constant element method is also tested and shows lower accuracy than the linear element method as discussed above. For example, for the above sphere case with a 642 node mesh, the constant element method results in an energy of −84.9 kcal/mol. And the speed is also much slower than our constant node patch method (data not shown here).

5.2. Effects of the Geometric Factor Modification. We check the effects on the PBE solution of using the geometric correction coefficient α_p instead of $^1/_2$ in the left-hand side of the BIEs 7 and 8. The test is performed on a single sphere model, in which a unit positive charge is positioned at the center of a unit sphere (radius of 1 Å). The relative interior and exterior dielectric constants are set as 2 and 80, respectively. The ionic concentration is 0. Table 2 shows the BEM solution values obtained with the normal BIEs with coefficient $^1/_2$, and with the geometry-dependent coefficients on the first five nodes of the mesh. The analytical values are also shown as references.

It is found that the normal BEM (nBEM, based on the nBIEs) gives a solution on the nodes with around ∼8% relative errors, while the geometrically corrected BEM makes surprising improvements in the potential solution *f*, with less than 0.05% relative error. However, the normal derivative of the potential, *h*, is not improved. This may be due to the fact that in the left-hand sides of both eqs 7 and 8 the geometric correction only explicitly couples with *f*. This indicates that the geometric correction on the dBIE may improve the accuracy of both *f* and *h*, which will be studied in the future work related to the aforementioned free terms and hypersingular integrals that appear in the dBIE.

5.3. Force Calculations. For a test model, we calculate the electrostatic interaction forces between two spherical molecules and compare with those computed by our previous methods.[8,18] We choose two point charges in a vacuum, in which each charge is surrounded by a unit sphere discretized by 320 flat triangular elements (162 nodes). Both charges are put on the *x* axis, so that the nonzero force component is along the *x* direction; that is, F_x, and the other two components F_y and F_z are zero in theory. Figure 4 shows the forces along the *x* direction F_x as a function of the distance between the two point charges calculated using the present interpolation method, a hypersingular integral method, the variational approach, and the analytical formula. It is found that the interpolation method based on the PB solution with geometric correction in the nBIEs is the most accurate

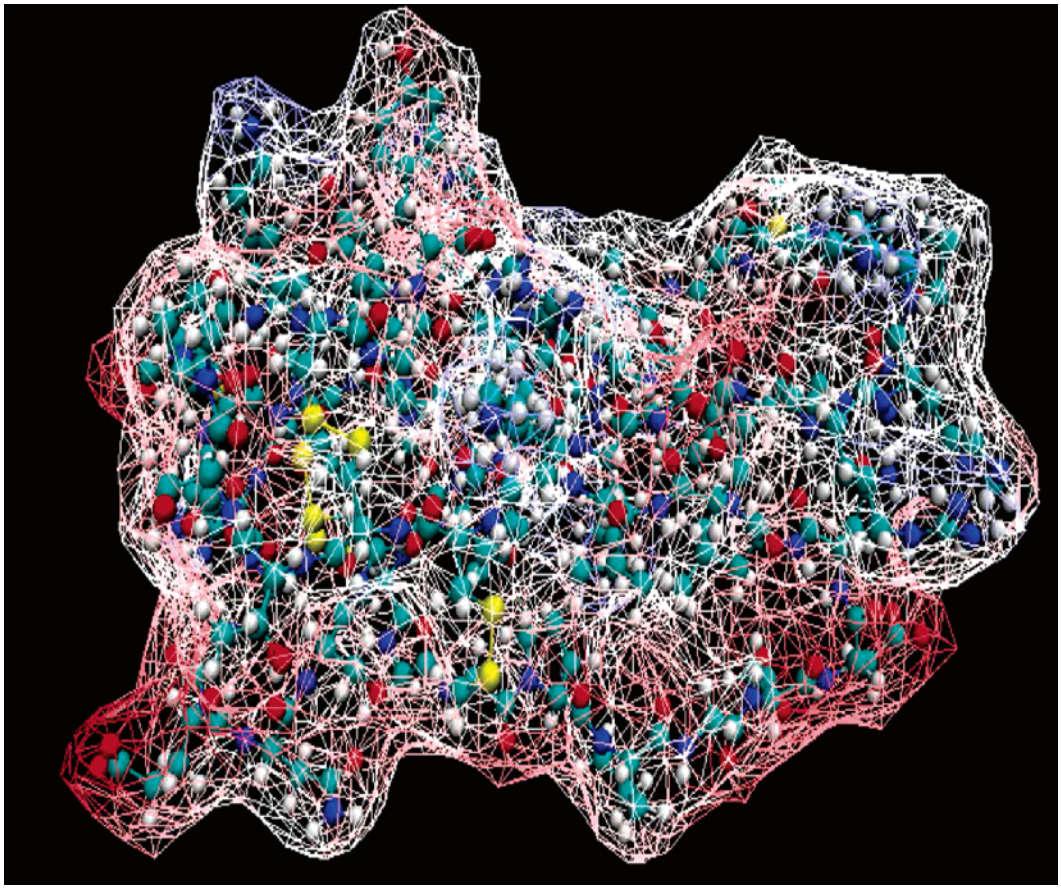

Figure 3. Surface mesh and potential map of fasciculinII. The mesh wireframe is colored from red to blue to represent the potential increasing from negative to positive values. The figure is generated using VMD[31] by running a tcl script

Table 2. The BEM Solution at the First Five Nodes on a Unit Spherical Surface[a,b]

node index	x	y	z	f_{analy}	h_{analy}	f_N (err%)	h_N (err%)	f_M (err%)	h_M (err%)
1	0.000	0.000	1.000	4.150	−4.150	4.485	−4.202	4.150	−4.202
						(8.077)	(1.272)	(0.002)	(1.258)
2	0.273	0.000	0.962	4.150	−4.150	4.499	−4.200	4.150	−4.203
						(8.406)	(1.208)	(0.004)	(1.272)
3	0.084	0.260	0.962	4.150	−4.150	4.500	−4.192	4.150	−4.196
						(8.444)	(1.022)	(−0.001)	(1.131)
4	0.526	0.000	0.851	4.148	−4.146	4.518	−4.174	4.147	−4.186
						(8.924)	(0.668)	(−0.027)	(0.950)
5	0.362	0.263	0.894	4.151	−4.152	4.527	−4.219	4.152	−4.233
						(9.054)	(1.616)	(0.023)	(1.945)

[a] f_N and h_N are the potential and its normal derivative, respectively, obtained with the normal BIE, and f_M and h_M are from the geometrically modified BIE. f_{analy} and h_{analy} are the corresponding analytical values. The errors (shown in the parentheses) are relative to the analytical results and in percentages. The corresponding units are in angstroms, moles, and kilocalories. [b] Calculations use the normal BIE form on a surface mesh with 162 nodes and 320 elements.

one among three approaches, and the calculated forces F_x nearly overlap with the analytical results over the whole distance range. However, if the force interpolation is based on the normal BIE solution without geometric correction, then the results are not so accurate as those of the other two methods (see the blue dot line in Figure 4). This means that the accuracy of the interpolation method heavily depends on the solution accuracy of the PBE.

The superior advantage of the interpolation method for force calculation is its calculation efficiency. In the above test case, in which 30 calculation points (distances) were selected, the CPU time on an Intel Pentium IV (2 GH) for the whole nBEM calculation is 28.3 s for the variational approach, 23.0 s for the hypersingular integral method, and 12.0 s for the present interpolation method. Moreover, as shown in the method description, the CPU time spent on the force calculation is simply proportional to the number of boundary elements of the target molecule.

6. Conclusions

New boundary patches around each node in the BEM are simply constructed on the basis of the usual triangulated mesh. This kind of BEM can be considered as a compromise of the traditional constant element method (constant node patch now) and the linear element method (unknowns located at nodes) and draws upon the advantages of both methods:

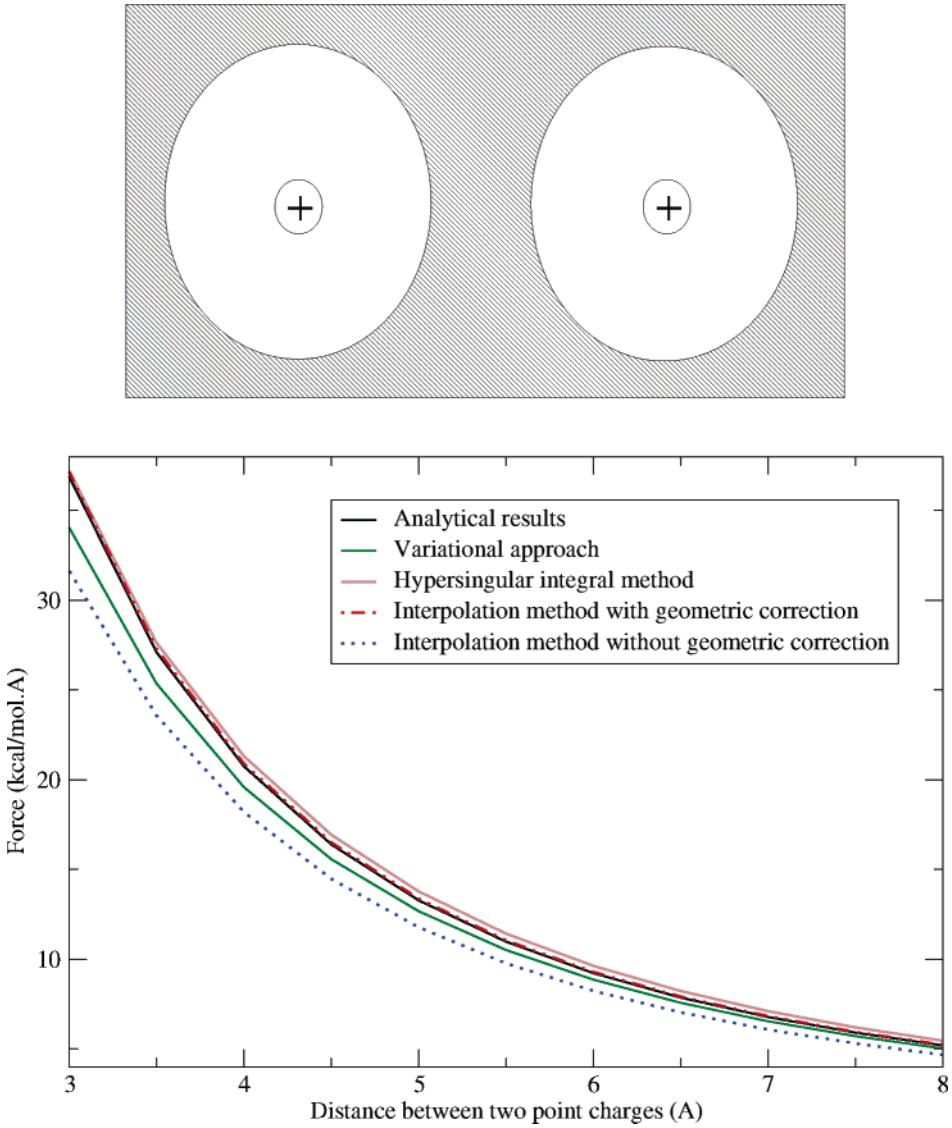

Figure 4. Calculations of electrostatic forces between two spheres using different approaches with normal BIE form. The data from the hypersingular integral method and variational approach are taken from refs 8 and 18.

reduction of the unknowns, simplified numerical complexity, and convenience for coefficient storage. Because the same mesh resolution is still kept, the calculation accuracy is not lost, and our numerical results on both sphere and protein cases also demonstrate this.

In addition, we describe an efficient interpolation approach for force calculation that is proper for any kind of node-based BEM. This is simply of order N with a small prefactor. The accuracy of force calculation by this approach is determined by the accuracy of the PBE solution, f and h. Our calculations also show that the geometry factor correction in BIE can significantly improve the accuracy of potential solution on the surface, thereby improving the accuracy of force calculation using the interpolation approach.

The code will be made available by the authors.

Acknowledgment. We would like to thank Prof. Chandrajit Bajaj for discussion of the interpolation method, Dr. Xiaolin Cheng for his help, and Dr. Justin Gullingsrud for visualization aid. This work was supported in part by the NIH, NSF, the Howard Hughes Medical Institute, National Biomedical Computing Resource, the NSF Center for Theoretical Biological Physics, SDSC, the W. M. Keck Foundation, and Accelrys, Inc.

References

(1) Zauhar, R. J.; Morgan, R. S. *J. Mol. Biol.* **1985**, *186*, 815−820.

(2) Yoon, B. J.; Lenhoff, A. M. *J. Comput. Chem.* **1990**, *11*, 1080−1086.

(3) Juffer, A. H.; Botta, E. F. F.; Vankeulen, B. A. M.; Vanderploeg, A.; Berendsen, H. J. C. *J. Comput. Phys.* **1991**, *97*, 144−171.

(4) Vorobjev, Y. N.; Grant, J. A.; Scheraga, H. A. *J. Am. Chem. Soc.* **1992**, *114*, 3189−3196.

(5) Zhou, H. X. *J. Chem. Phys.* **1994**, *100*, 3152−3162.

(6) Boschitsch, A. H.; Fenley, M. O. *J. Comput. Chem.* **2004**, *25*, 935−955.

(7) Zhou, H. X. *Biophys. J.* **1993**, *65*, 955−963.

(8) Lu, B. Z.; Zhang, D. Q.; McCammon, J. A. *J. Chem. Phys.* **2005**, *122*, 214102.

(9) Lu, B. Z.; Cheng, X. L.; Huang, J. F.; McCammon, J. A. *Proc. Natl. Acad. Sci. U.S.A.* **2006**, *103*, 19314−19319.

(10) Bharadwaj, R.; Windemuth, A.; Sridharan, S.; Honig, B.; Nicholls, A. *J. Comput. Chem.* **1995**, *16*, 898−913.

(11) Zauhar, R. J.; Varnek, A. *J. Comput. Chem.* **1996**, *17*, 864−877.

(12) Vorobjev, Y. N.; Scheraga, H. A. *J. Comput. Chem.* **1997**, *18*, 569−583.

(13) Totrov, M.; Abagyan, R. *Biopolymers* **2001**, *60*, 124−133.

(14) Boschitsch, A. H.; Fenley, M. O.; Zhou, H. X. *J. Phys. Chem. B* **2002**, *106*, 2741−2754.

(15) Bordner, A. J.; Huber, G. A. *J. Comput. Chem.* **2003**, *24*, 353−367.

(16) Kuo, S. S.; Altman, M. D.; Bardhan, J. P.; Tidor, B.; White, J. K. Fast Methods for Simulation of Biomolecule Electrostatics. In *ICCAD '02: Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design*; ACM Press: New York, 2002.

(17) Liang, J.; Subramaniam, S. *Biophys. J.* **1997**, *73*, 1830−1841.

(18) Lu, B. Z.; Cheng, X. L.; Hou, T. J.; McCammon, J. A. *J. Chem. Phys.* **2005**, *123*, 084904.

(19) Zauhar, R. J. *J. Comput. Chem.* **1991**, *12*, 575−583.

(20) Wan, S. Z.; Wang, C. X.; Xiang, Z. X.; Shi, Y. Y. *J. Comput. Chem.* **1997**, *18*, 1440−1449.

(21) Wang, C. X.; Wan, S. Z.; Xiang, Z. X.; Shi, Y. Y. *J. Phys. Chem. B* **1997**, *101*, 230−235.

(22) Lu, B. Z.; Wang, C. X.; Chen, W. Z.; Wan, S. Z.; Shi, Y. Y. *J. Phys. Chem. B* **2000**, *104*, 6877−6883.

(23) Pozrikidis, C. *A Practical Guide to Boundary-Element Methods with the Software Library BEMLIB*; Chapman and Hall: London, 2002.

(24) Guiggiani, M. Formulation and Numerical Treatment of Boundary Integral Equations with Hypersingular Kernels. In *Singular Inetgrals in Boundary Element Methods*; Sladek, V., Sladek, J., Eds.; Computational Mechanics Publications: Southampton, U. K., 1998.

(25) Frangi, A.; Guiggiani, M. *Z. Angew. Math. Mech.* **2001**, *81*, 651−664.

(26) Duff, I. S.; Grimes, R. G.; Lewis, J. G. *ACM Trans. Math. Software* **1989**, *15*, 1−14.

(27) Gilson, M. K.; Davis, M. E.; Luty, B. A.; McCammon, J. A. *J. Phys. Chem.* **1993**, *97*, 3591−3600.

(28) Bajaj, C. L.; Xu, G. Modeling Scattered Function Data on Curved Surfaces. In *Pacific Graphics '94: Proceeding of the Second Pacific Conference on Fundamentals of Computer Graphics*; World Scientific Publishing Co., Inc.: River Edge, NJ, 1994.

(29) Max, N. *J. Graph. Tools* **1999**, *4*, 1−6.

(30) Sanner, M. F.; Olson, A. J.; Spehner, J. C. *Biopolymers* **1996**, *38*, 305−320.

(31) Humphrey, W.; Dalke, A.; Schulten, K. *J. Mol. Graphics* **1996**, *14*, 33−38.

CT700001X