Adaptive Multilevel Finite Element Solution of the Poisson–Boltzmann Equation II. Refinement at Solvent-Accessible Surfaces in Biomolecular Systems

N. BAKER,[1] M. HOLST,[2] F. WANG[3]

[1]*Department of Chemistry and Department of Mathematics Scientific Computation Group, University of California at San Diego, La Jolla, California 92093*
[2]*Department of Mathematics, University of California at San Diego, La Jolla, California 92093*
[3]*Department of Mathematics, University of California at Irvine, Irvine, California 92697*

Received 25 October 1999; accepted 18 May 2000

ABSTRACT: We apply the adaptive multilevel finite element techniques (Holst, Baker, and Wang [21]) to the nonlinear Poisson–Boltzmann equation (PBE) in the context of biomolecules. Fast and accurate numerical solution of the PBE in this setting is usually difficult to accomplish due to presence of discontinuous coefficients, delta functions, three spatial dimensions, unbounded domains, and rapid (exponential) nonlinearity. However, these adaptive techniques have shown substantial improvement in solution time over conventional uniform-mesh finite difference methods. One important aspect of the adaptive multilevel finite element method is the robust *a posteriori* error estimators necessary to drive the adaptive refinement routines. This article discusses the choice of solvent accessibility for *a posteriori* error estimation of PBE solutions and the implementation of such routines in the "Adaptive Poisson–Boltzmann Solver" (APBS) software package based on the "Manifold Code" (MC) libraries. Results are shown for the application of this method to several biomolecular systems. J Comput Chem 21: 1343–1352, 2000

Keywords: Poisson–Boltzmann equation; adaptive finite element methods; *a posteriori* error estimation; biomolecules; electrostatics

Correspondence to: M Holst; e-mail: mholst@math.ucsd.edu
Contract/grant sponsors: UCSD Hellman Fellowship and NSF CAREER Award (to M.H.); contract/grant number: 9875856

1. Introduction

The Poisson–Boltzmann equation (PBE) is a nonlinear partial differential equation (PDE) fundamental to the Debye–Hückel continuum electrostatic theory.[1] The PBE determines a dimensionless electrostatic potential $u(x) = e_c\Phi(x)/(k_BT)$ around a collection of charges immersed in a salt solution, where $\Phi(x)$ is the electrostatic potential at $x \in \mathbb{R}^3$. For a 1:1 electrolyte, the PBE can be written as

$$-\nabla \cdot \big(\epsilon(x)\nabla u(x)\big) + \bar{\kappa}^2(x)\sinh\big(u(x)\big) = \frac{4\pi e_c^2}{k_BT}\sum_{i=1}^{N_m} z_i\delta(x-\bar{x}_i). \quad (1.1)$$

This PDE is generally solved in some domain $\Omega \subseteq \mathbb{R}^3$ with the condition $u = \bar{u}$ on $\partial\Omega$, the boundary of Ω. The physical system corresponds to $\Omega = \mathbb{R}^3$ and $\bar{u} = 0$. In the PBE, the dielectric value ϵ jumps by one or two orders of magnitude at the interface between the protein and the solvent. Additionally, the modified Debye–Hückel parameter $\bar{\kappa}^2$ (a function of the ionic strength of the solvent) is also discontinuous at an ion-exclusion region surrounding the biological structure. The structure itself (e.g., a biological molecule, or a membrane) is represented implicitly by ϵ and $\bar{\kappa}$, as well as explicitly by the N_m point charges $q_i = z_ie_c$ at the positions $\bar{x}_i$ (with the distributions $\delta(\cdot)$ reflecting the point-charge behavior).

The importance of this equation in biomolecular modeling is well established (cf. refs. 2 and 3 for thorough discussions). A number of articles have appeared in the literature in the last 10 years, presenting various numerical techniques for approximating the solution to (1.1) accurately and efficiently. These methods include uniform-mesh finite difference solutions[3–13] integral equation-based solutions (necessarily restricted to linearizations),[5, 10, 14–17] and nonadaptive finite element methods.[18, 19] Such nonadaptive finite element techniques show a factor of two improvement in performance over uniform-mesh techniques.[19] An adaptive approach for an axi-symmetric case resulting in a two-dimensional problem appears in ref. 20. The work described in ref. 21 for the full three-dimensional nonlinear problem combines adaptive refinement driven by robust *a posteriori* error estimators with multilevel finite element methods to illustrate that the completely general problem can be solved with *orders of magnitude* reduction in solution time compared to uniform-mesh approches.

We will use the adaptive multilevel finite element method of ref. 21 to solve the nonlinear PBE around biomolecules. The *a posteriori* error estimation scheme will be described in the context of the solvent-accessible surface of these protein and nucleic acid systems. Specifically, we will show how the discontinuous functions ϵ and $\bar{\kappa}^2$ can be used in to drive adaptive refinement routines. Additionally, we will outline implementation of routines for the fast evaluation of these functions.

The core numerical procedures are implemented using the Manifold Code (MC)[21–23] libraries, designed by one of the authors (M. Holst) over several years at Caltech and UC San Diego. MC is designed to solve general weak formulations of nonlinear elliptic equations of tensor equations on two- and three-manifolds,[22] a general class of problems that includes the PBE. MC employs *a posteriori* error estimation, adaptive simplex subdivision, unscructured algebraic multilevel methods, global inexact Newton methods, and homotopy methods, for the highly accurate numerical solution of this class of problems. The key features of MC, as well as its performance with respect to alternative methods, are described in ref. 22. These include inexact Newton multilevel solvers, complex mesh data structures, adaptive refinement based on *a posteriori* error estimates, and parallelization of the solution algorithm. The MC software and various supporting tools may be found at the following web site: http://www.scicomp/ucsd.edu/~mholst/codes/mc/.

The specific features required to solve PDEs for biomolecular systems are implemented in the "Adaptive Poisson–Boltzmann Solver" (APBS) software package, designed by one of the authors (N. Baker). This package uses the numerical routines from MC to efficiently solve the PBE for proteins and nucleic acids. The software employs two different solvent accessibility algorithms, implemented in two separate libraries. The first is the SAV algorithm of You and Bashford,[24] described briefly in a later section. The second is the HASH algorithm developed by one of the authors (N. Baker), described in some detail later.

OUTLINE

In Section 2, we review the elements of the adaptive multilevel fiite element method[21] as applied to the PBE. A justification for error estimation and adaptive refinement based on solvent-accessible surfaces is described in Section 3. Fast algorithms for evaluating solvent accessibility are presented in

Section 4. The methodology is tested on several protein systems in Section 5 and the conclusions are presented in Section 6.

2. Review of the Adaptive Multilevel Finite Element Method for the Poisson–Boltzmann Equation

For completeness, we will provide a basic outline of the adaptive multilevel finite element method. Algorithmic details, more comprehensive discussions, and additional references are all presented in the previous article in this series.[21]

WEAK FORM OF THE POISSON–BOLTZMANN EQUATION

The finite element method approximates the exact solution u as a combination of piece-wise polynomial basis fuuctions u_h. For our implementation, we choose the set of piece-wise linear finite element basis functions. This basis spans V_h, a subspace of the Sobolev space $H_0^1(\Omega)$, which includes functions that are differentiable one time under the integral. Clearly, the *strong form* of the PBE, as written in (1.1), requires that the solution be twice differentiable for the equation to hold in the classical sense. Therefore, to accommodate its solution by finite element methods, the equation is represented by the *weak form*,

$$\langle F(u), v\rangle = \int_\Omega (\epsilon\nabla u)\cdot\nabla v + \bar{\kappa}^2 \sinh(u)v - fv\,dx, \quad (2.1)$$

where v is a test function. To employ a Galerkin method, we begin by constructing an approximation $\bar{u}_h$ to the boundary potential function $\bar{u}$, and we then look for a Galerkin approximation that satisfies the weak formulation of the PBE:

$$\text{Find } u_h \in \bar{u}_h + V_h \text{ such that } \langle F(u_h), v_h\rangle = 0, \quad \forall v_h \in V_h. \quad (2.2)$$

METHODS FOR SOLVING THE DISCRETIZED EQUATION

The inexact Newton method allows the nonlinear problem (2.1) to be solved numerically for u_h by iteratively solving linearized versions of (2.1) for corrections to the solution until a desired error tolerance is reached. Not surprisingly, the computational complexity of this method is dominated by the solution of the linear algebraic equations in each iteration. Multilevel methods are the only known provably optimal or nearly optimal methods for solving these types of linear algebraic equations, resulting from discretizations of a large class of general linear elliptic problems.[25–27] These methods apply an interpolation operator to the discretization of the linearized PBE to project it onto a coarser mesh. This coarser discretization provides a fewer number of unknowns, making the problem easier to solve. Several levels of this interpolation scheme leads to a small problem that can be solved exactly. The approximate solution on the finest mesh can then be obtained by using the coarse solutions to accelerate the convergence of iterative methods on the finer levels.

ERROR ESTIMATION AND ADAPTIVE MESH REFINEMENT

Much activity has recently been centered on *a posteriori* error estimation and its use in adaptive mesh refinement algorithms.[28–33] The challenge for a numerical method is to be as efficient as possible, and *a posteriori* estimates are a basic tool in deciding which parts of the solution require additional attention. The solve–estimate–refine structure in simplex-based adaptive finite element codes such as MC exploiting these *a posteriori* estimators, is as follows:

Algorithm 2.1. *(Adaptive multilevel finite element approximation)*

- *While* ($\|u - u_h\|_X > TOL$) *do:*
  1. *Find* $u_h \in \bar{u}_h + V_h$ *such that* $\langle F(u_h), v_h\rangle = 0$, $\forall v_h \in V_h$.
  2. *Estimate* $\|u - u_h\|_X$ *over each element, set* $Q1 = Q2 = \emptyset$.
  3. *Place simplices with large error in "refinement"* Q1.
  4. *Bisect all simplices in* Q1 *(removing them from* Q1*), and place any nonconforming simplices created in* Q2.
  5. Q1 *is now empty; set* Q1 = Q2, Q2 = $\emptyset$.
  6. *If* Q1 *is not empty, goto (4).*
- *End while.*

The refinement scheme is driven by the error estimate in step 2. This adaptive method begins with subdivision of elements with error estimates above a certain tolerance. The details of these error estimates and their relationship to discontinuous coefficients in the PBE will be discussed later. The subsequent simplex subdivision can take a variety of forms, as shown by Figure 1. This figure illustrates a single subdivision of a two-simplex or a

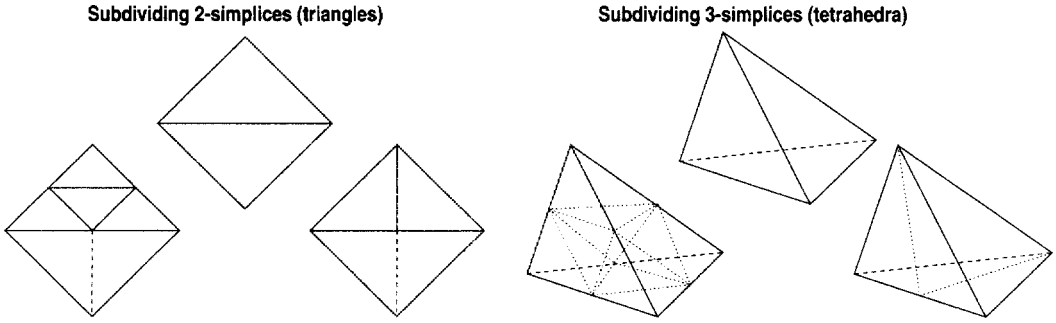

FIGURE 1. Refinement of two- and three-simplices using four-section, eight-section, and bisection.

three-simplex using either four-section (first figure from left), eigth-section (fourth figure from left), or bisection (third and sixth figures from left). All of these methods for mesh refinement are available in MC.[21–23] The conformity loop (4)–(6), required to produce a globally "conforming" mesh (described below) at the end of a refinement step, is guaranteed to terminate in a finite number of steps (cf. refs. 34 and 35), so that the refinements remain local. A globally conforming simplex mesh is defined as a collection of simplices that meet only at vertices and faces; for example, removing the dotted bisection in the third figure from the left in Figure 1 produces a nonconforming mesh.

3. Error Estimation and Adaptive Refinement by Solvent Accessibility

An overview of adaptive error control in finite element methods was given in the companion article.[21] Briefly, the idea is to estimate the behavior of the actual solution to the problem using only a previously computed numerical solution, referred to as *a posteriori* error estimation. One then uses this estimate to build an improved numerical solution by increasing the polynomial order (p-refinement) or refining the mesh (h-refinement) where appropriate. This procedure affects not only approximation quality, but also solution complexity: if a target solution accuracy can be obtained with fewer mesh points by their judicious placement in the domain, the cost of solving the discrete equations ia reduced (sometimes dramatically) because the number of unknowns is reduced (again, sometimes dramatically). Generally speaking, if an elliptic equation has a solution with local singular behavior, such as would result from the presence of abrupt changes in the coefficients of the equation (e.g., the functions ϵ and $\bar{\kappa}$ in the present case), then adaptive methods tend to give dramatic improvements over nonadaptive methods in terms of accuracy achieved for a given complexity price. In this section, we will give a brief overview of the residual-based *a posteriori* error estimator that was derived in the companion article,[21] and will then discuss a new error estimation approach for the Poisson–Boltzmann equation based on geometric solvent accessibility.

NONLINEAR RESIDUAL-BASED *A POSTERIORI* ERROR ESTIMATORS

That one can estimate the error in an approximation when the solution is unknown seems at first surprising, but in fact, such ideas are used even in basic numerical analysis (cf. Richardson extrapolation[36]). In our setting, the main idea can be understood by considering an approximation $\tilde{u}$ to the following linear equation for some nonsingular operator A:

$$Au = f.$$

We can test the quality of our approximation by forming the *residual*:

$$r = f - A\tilde{u}.$$

Note that the error in the approximation $e = u - \tilde{u}$ satisfies the *error equation*:

$$Ae = A(u - \tilde{u}) = Au - A\tilde{u} = f - A\tilde{u} = r.$$

Inverting A and taking norms of both sides givens

$$\|e\| = \left\|A^{-1}r\right\| \leq \left\|A^{-1}\right\| \|r\|.$$

If we have a mechanism for estimating or bounding $\|A^{-1}\|$, then we can detect when $\|e\|$ is large by monitoring $\|r\|$.

This idea lies at the core of the derivation of the *a posteriori* error estimator in the companion articles,[21] although the derivation is somewhat more complex due to the fact that the equation we work with is a nonlinear integral weak form rather

than a simple linear operator equation. The somewhat technical derivation of the complete nonlinear residual-based error estimator employed by MC can be found in Appendix A in ref. 21, and the realization of the estimator in the specific case of the Poisson–Boltzmann equation can be found in Appendix B in ref. 21. In the remainder of this section, we will discuss the use of a second error estimator based on solvent accessibility.

GEOMETRIC *A POSTERIORI* ERROR ESTIMATORS BASED ON SOLVENT ACCESSIBILITY

The presence of local singularities in the PBE coefficients leads to rapid changes in solution the neighborhood of these discontinuities. When nonlinear residual-based error estimators are employed, these rapid changes are detected in the *jump* term in the estimator (cf. the expressions in Appendices A and B in ref. 21), and as a result, almost all refinement occurs at the discontinuity interface. This is, in fact, the desired refinement behavior; it is in these regions of change, generally associated with the protein–solvent interface, that we wish to refine simplices to obtain higher levels of accuracy.

Rather than use the somewhat expensive residual-based error estimator in ref. 21 to drive the refinement at the dielectric boundary, we describe now a geometric error indicator that will mark dielectric boundary elements for refinement more directly, resulting in a less expensive error indicator. Note that in general, the functions $\epsilon(x)$ and $\bar{\kappa}^2(x)$ are defined as

$$\epsilon(x) = \begin{cases} \epsilon_w & x \in \Gamma \\ \epsilon_p & \text{otherwise} \end{cases}$$
$$\bar{\kappa}^2(x) = \begin{cases} \kappa & x \in \Gamma' \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\Gamma' \subseteq \Gamma \subset \Omega$ are (often complicated) volumes in $\mathbb{R}^3$ and ϵ_s, ϵ_w, and κ are positive parameters. Although these discontinuous representations of ϵ and $\bar{\kappa}^2$ are widely used, there are multiple definitions for Γ and Γ' employed in PBE solvers.

The various choices for Γ and Γ' can be distinguished with the aid of definitions from ref. 24. Let P be the set of ordered pairs $(\bar{x}_i, r_i)$ representing the centers and radii of all atoms, and let $\rho \geq 0$ be a probe molecule's radius. The free volume $F(\rho, P)$ is simply the collection of acceptable locations for the center of a probe sphere where there is no overlap between the probe and protein atoms,

$$F(\rho, P) = \left\{ y \in \Omega \colon \operatorname{dist}(\bar{x}_i, y) \geq r_i + \rho,\ \forall (\bar{x}_i, r_i) \in P \right\}, \tag{3.2}$$

where $\operatorname{dist}(a, b)$ is the distance between two points a and b. The solvent accessible volume $S(\rho, P)$ enlarges the free volume by the radii of the probe spheres in $F(\rho, P)$,

$$S(\rho, P) = \left\{ y \in \Omega \colon \operatorname{dist}(x, y) \leq \rho,\ x \in F(\rho, P) \right\}. \tag{3.3}$$

Both of these volume definitions are commonly used for Γ and Γ', with ρ often taken as zero or the radius of a solvent or ion probe. The APBS code allows the use of either of two "solvent accessibility" algorithms (cf. Section 4), which efficiently determine whether a point lies in Γ or Γ'. Through the judicious use of these algorithms, APBS can efficiently handle any of the volume definitions.

Adaptive refinement is implemented by bisection of those simplices that cross discontinuities in ϵ or $\bar{\kappa}^2$. During the initial mesh generation and subsequent refinement, vertices are assigned values for ϵ and $\bar{\kappa}^2$ using one of the methods in Section 4. Refinement is performed on any simplex with differing values of these variables at its vertices. For efficiency, new vertices are labeled with ϵ and $\bar{\kappa}^2$ only if they are associated with simplices falling across the discontinuous boundaries. A similar treatment is used for the evaluation of quadrature points. The solvent-accessibility algorithms are called only if the quadrature point lies in a simplex with differing values of ϵ or $\bar{\kappa}^2$ at its vertices. This method reduces the overhead associated with checking solvent accessibility.

GENERATING THE INITIAL MESHES TO START MESH REFINEMENT

Two methods have been implemented in the APBS software to create initial meshes for the start of mesh refinement.

The first method constructs initial finite element triangulations by using the charge centers within the molecule as a subset of the initial vertex locations. Additional nodal points are added to three spheres which enclose the biomolecule. The largest of these spheres is used to denote the Dirichlet boundary of the computational domain. This outer sphere should made sufficiently large so that a simple boundary condition can be assumed. The nodal points on the two interior spheres are used to prevent excessively thin or otherwise badly shaped simplices. Application of a Delaunay mesh generating routine to this set of charge and sphere vertices

completes the initial triangulation. Although this method provides very accurate representation of the delta functions, the nonoptimal Delaunay algorithm can cause long initial setup times for large biomolecules and generate meshes that are too large to be handled efficiently by the multilevel algorithm. Because the molecules described in this article are fairly small, this method will be used for the numerical examples presented in this article.

The second method starts with a very coarse triangulation of the problem domain and refines those simplices containing more than a target number of delta functions (i.e., charges). This algorithm is very fast, and can be stopped at any desired target number of coarse meah simplices. However, this method requires the storage and look-up of the delta function locations that can create additional time and memory overhead during the solve–estimate–refine stages of the PBE solution.

4. Solvent Accessibility Algorithms

These algorithms attempt to rapidly evaluate the solvent accessibility of points in Ω for use in quadrature and refinement routines. Both algorithms give exact results for their respective models of solvent accessibility.

SOLVENT ACCESSIBLE VOLUME (SAV) ALGORITHM

The SAV algorithm, created by You and Bashford,[24] uses atomic radii and position to calculate a set of geometry objects that provide an analytical representation of the solvent-accessible volume $S(\rho, P)$, as defined in (3.3). Reference 24 shows how particular combinations of conical, toroidal, tubular, and spherical shell regions can exactly describe $S(\rho, P)$. Furthermore, they show that the complexity of describing the structure and evaluating the solvent accessibility of points is linear in the number of atoms.

This algorithm begins by constructing a set of data structures for the analytical description of the solvent-accessible volume. These structures include objects describing the geometry associated with the shells, cones, and tubes described above. Although this initial construction of the structures is time consuming, it only occurs in the initialization phase of SAV; subsequent evaluations of solvent accessibility are much faster. By determining the location of a point with respect to a few of these geometry objects, SAV can quickly assess whether a point lies within $S(\rho, P)$.

The SAV algorithm was provided by Dr. D. Bashford at the Scripps Research Institute as C++ code. This was modified from lattice to single-point evaluation and wrapped in C by one of the authors (F.W.) to define the PBE coefficients to MC. This wrapper around SAV was later integrated into APBS along with VHASH by one of the authors (N.B.).

ATOMIC HASH LIST (HASH) ALGORITHM

In many cases, the simpler $F(\rho, P)$ free volume suffices for PBE calculations. Although SAV can be used to calculate this volume (in the limit $\rho \to 0$), less complicated methods are also available. One such method is based on the Verlet cell index metbod common to large molecular dynamics simulations.[37–39] As used in molecular dynamics, the cell method divides the computational domain into a collection of cells to aid in the evaluation of neighbor lists, or atoms within a certain distance from each other. A similar approach is used here in the HASH algorithm. The set of atoms in the molecule is divided into lists associated with particular regions of space. These can then be used to quickly and accurately evaluate the accessibility of any point in space.

The HASH algorithm begins with the creation of a hash table; our data structure associated with the Verlet cell index lattice. Given a desired number of simplices, a uniform hexagonal mesh is created to surround the biomolecule, leaving at least ρ space from the surface of all atoms. Next, a list of atoms is associated with each simplex in the mesh. Let

$$E_i(\rho, P) = \left\{ y \in \Omega \colon \operatorname{dist}(y, \bar{x}_i) < r_i + \rho \right\}$$

be the solvent excluded volume due to atom i, such that $F(\rho, P) = \Omega - \bigcup_{i=1}^{N} E_i(\rho, P)$. Any simplex that contains a portion of $E_i(\rho, P)$ will have atom i assigned to at least one of its vertex lists. Like SAV, this initial assignment only needs to be performed once. Given a point y, accessibility is determined by finding the simplex which contains y and eveluating $\operatorname{dist}(y, \bar{x}_i) < r_i + \rho$ for each atom i in the simplex list. If this inequality is false for all atoms in the simplex list, then y is in $F(\rho, P)$.

This method has several parameters that can be adjusted to suit the particular system of interest. Obviously, the number of simplices can be tuned to obtain a desired average number of atoms per simplex. However, the hash table lattice can also be altered to fit the particular system. Although APBS currently uses a uniform hexagonal lattice in the HASH implementation, any mesh in which a point can be associated with a simplex in $\mathcal{O}(1)$ time is suitable for this algorithm.

5. Numerical Test Results

The use of adaptive multilevel finite element method to obtain solutions to the PBE was tested by numerical experiments on various biomolecules. APBS and MC were compiled with gcc (flags '-g -O') and run under Linux on a 500 MHz Intel Pentium III (256 MB RAM) computer. The initial meshes were generated using the atomic coordinates and an outer sphere of three times the biomolecular radius a. The value of a was taken to be the largest distance of any atom from the protein center of geometry. The Dirichlet boundary condition $\bar{u}$ on this outer sphere was calculated from the free space solution of the linearized PBE for a spherical, nonpolarizable ion located at $\bar{x}$

$$\bar{u}(x) = \left(\frac{e_c^2}{k_B T}\right) \frac{z e^{-\kappa(a-r)}}{\epsilon_s \epsilon_0 (1+\kappa a)|x-\bar{x}|} \qquad (5.1)$$

where z is the total protein charge, ϵ_s is the solvent dielectric, and κ is related to the ionic strength of the salt solution.[13] Solutions were generated for both the linear ($\bar{\kappa}^2 = 0$ everywhere) and nonlinear ($\bar{\kappa}^2$ chosen to represent 150 mM ionic strength) forms of the PBE at each level. The preconditioned conjugate gradient method was applied using the algebraic multigrid method as the preconditioner until an error of 0.01 per simplex was reached.

Each initial mesh underwent seven levels of adaptive refinement according to a posterior error estimates of the PBE solution. The Delaunay generator naturally generates simplex elements near and inside the biomolecule in the coarse meshes, which are on the order of a few angstroms due to the charge point separations in the data, which form the vertices of the simplex mesh. Because the simplex elements touching the dielectric boundaries are always marked for refinement by the solvent-accessibility error indicator, the simplex diameters on the dielectric boundaries approach 2^{-7} Å. We are not aware of any results in the literature that come anywhere close to this type of resolution. For example, typical AChE finite difference methods evaluate the solution on a 110-Å^3 uniform cubic mesh with a 1-Å resolution, requiring the evaluation of roughly 10^6 unknowns.

However, to obtain the roughly the same accuracy available at the 2^{-7} Å resolution of adaptive methods, nearly 10^{26} unknowns would need to be determined. Clearly, this is a number too large for even the most powerful of current supercomputing resources.

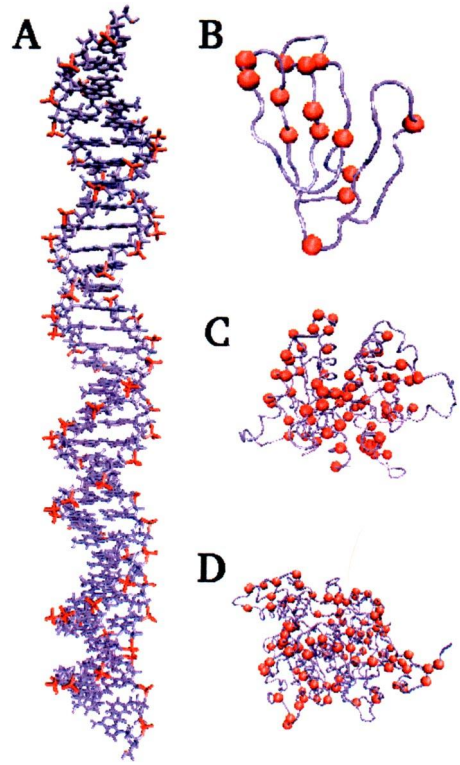

FIGURE 2. Structures of the test biomolecules (not to scale) as generated by VMD.[40] Charged groups are displayed as spheres; other bonds are shown as lines or by a protein backbone tube. The DNA 36-mer is shown in (A), fasciculin-2 in (B), HIV integrase in (C), and mouse AChE in (D).

Four biomolecules were chosen for their different electrostatic properties. First, the neutral HIV integrase dimer (HIV IN, Fig. 2C) offers the chance to observe biomolecular electrostatic fields in the absence of a net charge. Mouse acetylcholinesterase (mAChE, Fig. 2D) was chosen for its negative charge distribution, which has been implicated in the fast catalysis of positively charged substrates;[41]

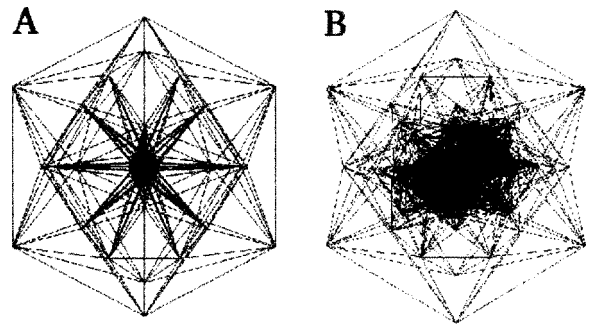

FIGURE 3. Initial meshes from Delaunay triangulation of charge points for the DNA 36-mer [(A), viewed along DNA axis] and FAS2 (B).

TABLE I.
Numerical Test Results for Linear Problems.

Molecule	Number of Atoms	Total Charge (e)	Number of Nodes		Time (s)	
			HASH	SAV	HASH	SAV
FAS2	906	+4	213601	182317	540	645
HIV IN	3008	0	408079	338140	1095	1829
mAChE	8340	−8	450759	397314	1393	—
DNA	2302	−70	510272	440072	1640	2188

This lists the CPU time required to refine a precalculated mesh seven times and obtain the multilevel solution of the PBE on each refinement level. Please refer to the text for additional details.
[a] Due to discrepancies in the SAV C++ and MC code, we were unable to obtain conforming results for this test case.

while fasciculin-2 (FAS2, Fig. 2B) posses a net positive charge, which contributes to its inhibition mAChE.[42] Finally, a DNA 36-mer (DNA, Fig. 2A) was chosen as a example of the family of highly charged nucleic acids.

Some of the initial meshes generated using the Delaunay triangulation scheme described above are shown in Figure 3. The solutions to the linear PBE were calculated using the APBS and MC software packages and the previously described refinement schemes. Although results for the nonlinear PBE are presented, the solutions can be easily found with the APBS/MC software package. Table I shows the CPU time for refinement of the initial mesh and obtaining the solution to the linear PBE with the HASH and SAV algorithms. This table also lists the number of atoms in each molecule, the total charge, and the numbers of nodal points (vertices) on the finest level of refinement. Although the HASH algorithm was able to compute solvent accessible volumes for the mAChE molecule, discrepancies in the SAV code caused run-time errors that prevented a solution with the SAV algorithm. The results of the PBE calculations are shown in Figures 4 and 5. In Figure 4,

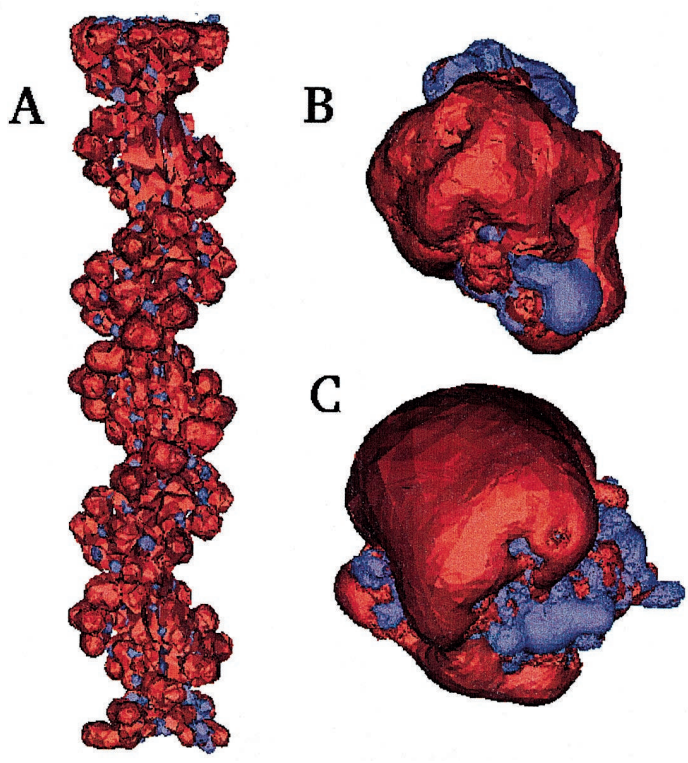

FIGURE 4. Potential contours from solution to linear PBE. (A) Contours for DNA 36-mer (red = −5, blue = +0.1); (B) HIV IN contours (red = −0.15, blue = +0.15); and (C) the contours for the mAChE potential (red = −0.15, blue = +0.15).

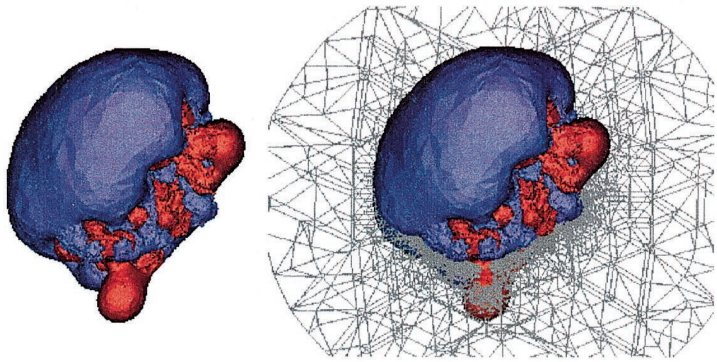

FIGURE 5. Potential contours from solution to linear PBE for FAS2 shown with a slice through the finite element mesh.

contours of the potentials generated for DNA, HIV IN, and mAChE are shown. Figure 5 shows potential contours for the FAS2 system as well as a slice through the finite element mesh.

6. Conclusions

We applied the adaptive multilevel finite element techniques described in ref. 21 to the nonlinear Poisson–Boltzmann equation in the context of biomolecules. Although numerical solution of the PBE in this setting is usually difficult to accomplish due to presence of discontinuous coefficients, delta functions, three spatial dimensions, and rapid nonlinearity, we have shown here and in the companion article[21] that that combining error estimation and adaptivity with the global inexact Newton–multilevel methods presented in ref. 11 yields a rapid and extraordinarily accurate approximation technique. We also demonstrated here the effectiveness of solvent accessibility as an alternative tool for *a posteriori* error estimation in the particular case of PBE solutions, where the dielectric boundary drives most of the refinement. We then gave a series of numerical experiments with the implementation of such techniques in the "Adaptive Poisson–Boltzmann Solver" (APBS) software package based on the "Manifold Code" (MC) libraries, together with the use of two solvent accessibiity libraries, VSAV and HASH. Retsults were shown for the application of this method to several biomolecular systems.

Acknowledgments

The authors thank R. Bank and J. A. McCammon for many enlightening discussions, D. Bashford for use of his solvent accessibility C++ library, and J. A. McCammon, T. Mordasi, and A. Elcock for access to the biomolecule data. N. Baker is a predoctoral fellow of the Howard Hughes Medical Institute and the Burroughs Wellcome Fund La Jolla Interfaces in Science training program.

References

1. Debye, P.; Hückel, E. Physik Z 1923, 24, 185.
2. Briggs, J. M.; McCammon, J. A. Comput Phys 1990, 6, 238.
3. Sharp, K. A.; Honig, B. Annu Rev Biophys Chem 1990, 19, 301.
4. Allison, S. A.; Sines, J. J.; Wierzbicki, A. J Phys Chem 1989, 93, 5819.
5. Gilson, M. K.; Sharp, K. A.; Honig, B. H. J Comput Chem 1988, 9, 327.
6. Jayaram, B.; Sharp, K. A.; Honig, B. Biopolymers 1989, 28, 975.
7. Luty, B. A.; Davis, M. E.; McCammon, J. A. J Comput Chem 1992, 13, 1114.
8. Nicholls, A.; Honig, B. J Comput Chem 1991, 12, 435.
9. Rashin, A. A.; Malinsky, J. J Comput Chem 1991, 12, 981.
10. Sharp, K. A.; Honig, B. J Phys Chem 1990, 94, 7684.
11. Holst, M.; Saied, F. J Comput Chem 1995, 16, 337.
12. Holst, M. Ph.D. thesis, Numerical Computing Group, University of Illinois at Urbana-Champaign, 1993; Also published as Technical Report UIUCDCS-R-03-1821.
13. Holst, M. Tech. report, Applied Mathematics and CRPC, California Institute of Technology, 1994.
14. Davis, M. E.; McCammon, J. A. J Comput Chem 1989, 10, 386.
15. Juffer, A. H.; Botta, E. F. F.; van Keulen, B. A. M.; van der Ploeg, A.; Berendsen, H. J. C. J Comput Phys 1991, 97, 144.
16. Niedermeier, C.; Schulten, K. Tech. report, Department of Physics and Beckman Institute, University of Illinois at Urbana-Champaign, 1990.
17. Yoon, B. J.; Lenhoff, A. M. J Comput Chem 1990, 11, 1080.
18. Cortis, C. M.; Friesner, R. A. J Comput Chem 1997, 18, 1570.
19. Cortis, C. M.; Friesner, R. A. J Comput Chem 1997, 18, 1591.

20. Bowen, W. R.; Sharif, A. O. J Colloid Interface Sci 1997, 187, 363.
21. Holst, M.; Baker, N.; Wang, F. J Comput Chem 2000, 21, 1319.
22. Holst, M. In preparation; currently available as a technical report and User's Guide to the MC software.
23. Holst, M.; Bernstein, D. Commun Math Phys, submitted.
24. You, T.; Bashford, D. J Comput Chem 1995, 16, 743.
25. Bank, R. E.; Dupont, T. F. Math Comp 1981, 36, 35.
26. Hackbusch, W. Multi-Grid Methods and Applications; Springer-Verlag: Berlin, Germany, 1985.
27. Xu, J. SIAM Rev 1992, 34, 581.
28. Babuška, I.; Rheinboldt, W. C. Int J Numer Meth Eng 1978, 12, 1597.
29. Babuška, I.; Rheinboldt, W. C. SIAM J Numer Anal 1978, 15, 736.
30. Bank, R. E. PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 8.0; Software, Environments and Tools; SIAM: Philadelphia, PA, 1998, vol. 5.
31. Verfürth, R. Math Comp 1994, 62, 445.
32. Verfürth, R. A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques; John Wiley & Sons Ltd: New York, 1996.
33. Xu, J.; Zhou, A. Math Comp, to appear.
34. Rivara, M. C. Int J Num Meth Eng 1984, 20, 745.
35. Rivara, M. C. J Comput Appl Math 1991, 36, 79.
36. Isaacson, E.; Keller, H. B. Analysis of Numerical Methods; John Wiley & Sons, Inc.: New York, 1966.
37. Allen, M. P.; Tildesley, D. J. Computer Simulation of Liquids; Clarendon Press: Oxford, 1987.
38. Hockney, R. W.; Eastwood, J. W. Computer Simulation Using Particles; McGraw-Hill: New York, 1981.
39. Quentrec, B.; Brot, C. J Comput Phys 1975, 13, 430.
40. Humphrey, W.; Dalke, A.; Schulten, K. J Mol Graph 1996, 14, 33.
41. Bourne, Y.; Taylor, P.; Bougis, P.; Marchot, P. J Biol Chem 1999, 274, 2963.
42. Marchot, P.; Bourne, Y.; Prowse, C.; Bougis, P.; Taylor, P. Toxicon 1998, 36, 1613.