# THE ADAPTIVE MULTILEVEL FINITE ELEMENT SOLUTION OF THE POISSON-BOLTZMANN EQUATION ON MASSIVELY PARALLEL COMPUTERS

NATHAN A. BAKER, DAVID SEPT, MICHAEL J. HOLST, AND J. ANDREW MCCAMMON

Abstract. Using new methods for the parallel solution of elliptic partial differential equations, the teraflops computing power of massively parallel computers can be leveraged to perform electrostatic calculations on large biological systems. This paper describes the adaptive multilevel finite element solution of the Poisson-Boltzmann equation for a microtubule on the NPACI IBM Blue Horizon supercomputer. The microtubule system is 40 nm in length and 24 nm in diameter, consists of roughly 600,000 atoms, and has a net charge of -1800 e. Poisson-Boltzmann calculations are performed for several processor configurations and the algorithm shows excellent parallel scaling.

## 1. Introduction

Electrostatics play a vital role in determining the specificity, rate, and strength of interactions in a variety of biomolecular processes [1, 2]. The accurate modeling of the contributions of solvent, counterions, and protein charges to the electrostatic field can often be very difficult and typically acts as the rate-limiting step for a variety of numerical simulations. Rather than explicitly treating the solvent and counterion effects in atomic detail, continuum methods such as the Poisson-Boltzmann equation (PBE) are often used to represent the effects of solvation on the electrostatic properties of the biomolecule. Despite this simplification, current methods for the calculation of electrostatic properties from the PBE still require significant computational effort and typically do not scale well with increasing problem size [3].

This paper describes the investigation of a large biomolecular system using the APBS (Adaptive Poisson-Boltzmann Solver) software [4]. APBS is a new Poisson-Boltzmann solver which uses adaptive multilevel finite element techniques [3, 5, 6] to efficiently treat the numerically difficult

aspects of the PBE. APBS is built on the MC (Manifold Code) adaptive multilevel finite element package [3, 5], which is used for most of the numerically intensive aspects of the solution of the PBE.

Section 2 provides some of the background of the Poisson-Boltzmann equation and its relation to biomolecules. A brief overview of adaptive multilevel finite element techniques and their parallelization is presented in Section 3. Discussion of the implementation of these methods in APBS and MC is presented in Section 4 and the results of the solution of the PBE for the electrostatic potential around a microtubule structure are given in Section 5. Finally, conclusions and future work are discussed in Section 6.

## 2. The Poisson-Boltzmann equation

The Poisson-Boltzmann equation is a second-order elliptic partial differential equation which describes the electrostatic potential around a fixed charge distribution in an ionic solution. For more thorough reviews of this equation and its role in biological electrostatics calculations, see Davis and McCammon [1] and Sharp and Honig [7]. There are three components of the solvated biomolecular system that we must consider to accurately model the electrostatic potential: the solute molecule, the solvent, and the solvated ions. The solute molecule is modeled as a dielectric continuum of low polarizability embedded in a dielectric medium of high polarizability which represents the solvent. Reflecting this difference in polarizabilities, the interior of the molecule is typically assigned a relative permittivity (or dielectric constant) between 2 and 20, while the solvent is given a much larger dielectric constant, generally near 80. In most cases, the atomic charge distribution inside the biomolecule is represented by a collection of delta functions. Finally, the solvated ions surrounding the biomolecule are also modeled as a continuum, distributed according to a Boltzmann distribution. Combining Poisson's equation, used to describe the electrostatic behavior of the point charges in the dielectric continuum, with the Boltzmann charge distribution for the solvated ions gives the

nonlinear Poisson-Boltzmann equation (NPBE),

$$-\nabla \cdot (\epsilon(x)\nabla u(x)) + \overline{\kappa}^2(x)\sinh(u(x)) = f(x), \quad u(\infty) = 0, \tag{2.1}$$

or, after linearization of the hyperbolic sine term, the linearized Poisson-Boltzmann equation (LPBE)

$$-\nabla \cdot (\epsilon(x)\nabla u(x)) + \overline{\kappa}^2(x)u(x) = f(x), \quad u(\infty) = 0, \tag{2.2}$$

where the source term is a sum of delta functions,

$$f(x) = \frac{4\pi e_c^2}{kT}\sum_{i=1}^{N_m} z_i\delta(x - x_i). \tag{2.3}$$

In Eqs. 2.1 and 2.2, the variable $u(x) = e_c\phi(x)/kT$ represents a dimensionless electrostatic potential, $\epsilon(x)$ is the dielectric coefficient, $\overline{\kappa}^2$ is the Debye-Hückel screening parameter, which describes ion concentration and accessibility, $kT$ is the thermal energy, $e_c$ is the electron charge, $N_m$ is the number of protein charges, $z_i$ is the partial charge of each protein atom, and $x_i$ is the position of each atom. Figure 1 shows a schematic of a solute (here taken to be a protein), ions, and solvent system modeled by the Poisson-Boltzmann equation. The dielectric coefficient $\epsilon$ changes by nearly two orders of magnitude across the "interior" protein-solvent boundary (solid line in Figure 1) and the screening parameter jumps from zero to a positive value across the "exterior" boundary (dashed line in Figure 1).

The accurate pointwise evaluation of the dielectric and screening parameter coefficients for a typical biomolecule is a nontrivial task. APBS evaluates the dielectric coefficient $\epsilon$ by using the Lee and Richards [8] definition of solvent accessibility. In short, the algorithm considers the volume $\Omega_{SA}$ defined by the union of the (infinite) set of spheres with centers at all $y \in \Omega$ such that $\|y - x_i\| > r_i + \sigma$ for all atoms $i$. Given some point $y$, the coefficient $\epsilon(y)$ is assigned the solvent dielectric constant if it is inside $\Omega_{SA}$, otherwise it is assigned the solute dielectric value. This definition of $\Omega_{SA}$ is shown in more detail in Figure 2 for a simplistic model of a protein molecule

(white circles). The molecular surface (black line) is defined by the accessibility of the solvent probe molecules (hashed circles); points outside this surface are assigned the solvent dielectric value (typically around 80), while points inside are assigned the solute dielectric (generally 2–20). The assignment of the screening parameter values $\overline{\kappa}^2$ is much simpler; points outside a distance $r_i + \sigma_{\text{ion}}$ from all atoms $i$ are assigned the bulk screening parameter value, while points closer than $r_i + \sigma_{\text{ion}}$ to any atom $i$ are assigned a value of 0.

As described here, the PBE equation contains three sources of discontinuities. Both the dielectric coefficient $\epsilon$ and the screening parameter $\overline{\kappa}^2$ have jump discontinuities (analogous to Heaviside step distribution functions) near the protein-solvent interface. Additionally, the source term $f(x)$, which models the point charges at the protein atoms, is represented by a sum of delta functions. While these jump and delta function discontinuities of coefficients in the PBE can pose serious numerical difficulties for traditional uniform or nonadaptive mesh partial differential equation solvers, these features can be efficiently described using the adaptive finite element techniques described in Section 3 [3–5].

## 3. Parallel multilevel adaptive finite element methods

This section briefly describes the theory behind the parallel multilevel adaptive finite element scheme used to solve the PBE for the electrostatic potential around biomolecules. Section 3.1 describes basic finite element techniques and Section 3.2 discusses the incorporation of adaptivity into these methods. A very short description of multilevel techniques is presented in Section 3.3 and the theory behind parallelization of these methods is described in Section 3.4.

### 3.1. **Finite element discretization.**

In order to solve the PBE on a finite computational platform, we need to truncate and discretize the infinitely large problem domain implicit in Eqs. 2.1 and 2.2. Specifically, we solve the PBE equation inside a polygonal domain $\Omega \subset \mathbb{R}^3$ subject to some

Dirichlet boundary condition

$$u(x) = g \text{ on } \partial\Omega,$$

where $\partial\Omega$ denotes the boundary of $\Omega$. To discretize the problem, we subdivide $\Omega$ by tessellation with tetrahedral simplices. The resulting tetrahedral mesh forms the structure over which we will define $V_h = \text{span}\{v_i\}$, as the space spanned by the piecewise-polynomial basis functions $\{v_i\}$. APBS currently uses the piecewise-linear finite element support provided by MC [5]. A representative basis function is depicted (on a two-dimensional triangular mesh) in Figure 3. The solution to the PBE is approximated by a function $u_h \in \overline{u}_h + V_h$ constructed by a linear combination of the basis functions

$$u_h(x) = \sum_i^N \alpha_i v_i(x). \tag{3.1}$$

The trace function $\overline{u}_h$ is not explicitly constructed, but is assumed to satisfy the Dirichlet boundary conditions.

In order for the construction of $u_h$ from piecewise-linear functions to be successful, we must restate the PBE equations in their "weak" form. Clearly, the second derivative (as required by Eqs. 2.1 and 2.2) of a piecewise-linear function is not well defined. This difficulty can be overcome by integrating the PBE with a test function $\tilde{v}$

$$\int_\Omega \left(-\nabla \cdot \epsilon \nabla u + \overline{\kappa}^2 \sinh(u)\right) \tilde{v} dx = \int_\Omega f \tilde{v} dx, \tag{3.2}$$

and applying integration by parts to the second-order differential term to give

$$\int_\Omega \left(\epsilon \nabla u \cdot \nabla \tilde{v} + \overline{\kappa}^2 \sinh(u) \tilde{v}\right) dx = \int_\Omega f \tilde{v} dx. \tag{3.3}$$

Equation 3.3 can also be written as

$$\langle F(u), \tilde{v} \rangle_{L^2(\Omega)} = \int_\Omega \left(\epsilon \nabla u \cdot \nabla \tilde{v} + \overline{\kappa}^2 \sinh(u) \tilde{v} - f \tilde{v}\right) dx = 0, \tag{3.4}$$

where $\langle \cdot, \cdot \rangle_{L^2(\Omega)}$ denotes the $L^2(\Omega)$ inner product and $F(u)$ is the weak form of the residual. This allows us to restate the PBE in weak form:

$$\text{Find } u_h \in \overline{u} + V_h \text{ such that } \langle F(u_h), v_i \rangle = 0 \text{ for all } v_i \in V_h. \tag{3.5}$$

This form of the PBE requires only one order of differentiation under an integral (with integration in the Lesbegue sense) and is therefore a "weaker" formulation of the PBE than the original second-order differential equations (2.1 and 2.2). Although the above discussion used the NPBE, similar manipulations can be performed for the LPBE to produce an expression for the residual $F(u)$ which is linear in $u$.

Given $u_h$ as a linear combination of the finite element basis functions and the above weak form of the PBE (3.5), we have a discretization of the partial differential equation suitable for numerical solution. In the case in which $F(u)$ is linear (LPBE), Eq. 3.5 explicitly defines a sparse matrix equation that can be solved using standard linear algebra methods or the multilevel methods described in Section 3.3. However, when $F(u)$ is nonlinear (NPBE), we employ a damped inexact Newton iteration as implemented by MC [5] to find $u_h$ [3, 9–13]. In brief, this method iteratively solves the linear matrix equations to determine improvements $w$ to the solution. When these improvements become sufficiently small, the Newton iteration stops and the resulting $u_h$ is used as the solution. The linear systems used to find the solution improvements are defined by the functional (Gateaux) derivatives of the nonlinear residual,

$$\langle DF(u)w, v \rangle = \int_\Omega \left( \epsilon \nabla w \cdot \nabla v + \overline{\kappa}^2 \cosh(u) w v - f v \right) dx, \tag{3.6}$$

and the resulting linear equations for the improvements $w$ are as follows:

$$\text{Find } w \text{ such that } \langle DF(u)w, v_i \rangle = -\langle F(u), v_i \rangle + r \text{ for all } v_i \in V_h, \tag{3.7}$$

where $r$ is a residual that allows for enhanced efficiency by accounting for the possibly inexact solution of Eq. 3.7. The updated solution is then obtained by addition of the improvement times

a damping factor $\lambda$ that stabilizes the algorithm

$$u \leftarrow u + \lambda w. \tag{3.8}$$

For more detailed discussion of the finite element method applied to the PBE, see Holst, Baker, and Wang [3] and Baker, Holst, and Wang [4]. The texts by Axelsson and Barker [14] and Braess [15] are good sources for more general reviews of the finite element method.

3.2. **Error estimation and mesh refinement.** While the methods of the previous section can be used to determine the solution on a given finite element mesh, they do not provide information about the accuracy with which the numerical solution $u_h$ represents the true solution or indicate whether it can be improved. The answers to these two questions lie within the domain of error estimation and adaptive refinement techniques. Again, we present only a cursory overview of this topic, briefly discussing the *a posteriori* error estimation and adaptive refinement techniques that are applied to the PBE in the present work. For more detailed information about the implementation of these methods in the solution of the PBE, see Holst, Baker, and Wang [3]. *A posteriori* error estimation has been the subject of several publications [5, 16–20] which provide much more information about the theory and implementation of these methods.

Adaptive refinement methods typically employ error-estimation techniques to approximate the distance between the numerical and true solution $\|u - u_h\|_X$ (using some norm $\|\cdot\|_X$) and determine the regions of the problem domain where the error is above a certain tolerance. The mesh is then refined in these regions of excess error and the PBE equation is re-solved to provide a more accurate finite element representation of the solution. The error-based refinement of the mesh can also be interpreted as the local enrichment of the finite element basis set in regions where the true solution is not adequately represented. In general, an *a posteriori* error estimator is used to determine the error in each simplex. Simpler *a priori* or geometry-based error estimators can also be used, but the reduction of error in the solution with each level of refinement is typically less efficient.

APBS employs the residual-based *a posteriori* error estimation framework provided by MC which generates a per-simplex error estimate $\eta_s$ in simplex $s$ by using the residual defined by the strong form (Eqs. 2.1 and 2.2) of the PBE [3, 19],

$$\eta_s^2 = h_s^2 \left\| -\nabla \cdot \epsilon \nabla u_h + \overline{\kappa}^2 \sinh(u_h) - f \right\|_{L^2(s)}^2 + \frac{1}{2} \sum_{f \in s} h_f \left\| \lfloor n_f \cdot \epsilon \nabla u_h \rfloor_f \right\|_{L^2(f)}^2 , \tag{3.9}$$

where $h_s$ denotes the size of the simplex, $f \in s$ denotes a face of simplex $s$, $h_f$ is the size of the face $f$, $\lfloor \tilde{v} \rfloor_f$ denotes the jump across the face $f$ of some function $\tilde{v}$, $n_f \cdot \epsilon \nabla u_h$ is the component of $\epsilon \nabla u_h$ normal to simplex face $f$, and the $L^2$ norm over a simplex or face is given by

$$\|v\|_{L^2(s \text{ or } f)}^2 = \int_{s \text{ or } f} |v|^2 \, dx. \tag{3.10}$$

Since each error estimate is defined over a simplex or simplex face, the solution is linear over the entire domain of the norm (3.9) and the contribution from the second-order term $-\nabla \cdot \epsilon \nabla u_h$ is zero. In general, the second (jump) term of the error estimator typically dominates $\eta_s$; therefore, the first term is not implemented in APBS. An estimate of the global error over the problem domain is obtained as the root mean square of the per-simplex estimates

$$\eta_{\text{global}} = \frac{1}{N} \left( \sum_i \eta_s^2 \right)^{1/2} . \tag{3.11}$$

Although this $\eta_{\text{global}}$ provides only an upper bound (within a constant) of the true error in the solution, it offers a practical measure for the reduction of error during solution of the PBE.

Given a per-simplex error estimate, those simplices with errors above a certain tolerance $\eta_{\text{tol}}$ are marked for subdivision. APBS employs the longest edge simplex subdivision algorithm in MC [5] for adaptivity. This subdivision method, along with other examples, is shown in Figure 4. Subdivision of *only* the marked simplices typically results in a nonconforming mesh, *i.e.*, a mesh in which the faces of some simplices intersect the vertices of other simplices. This situation is depicted in Figure 4 where, without the subdivision depicted by the dotted line in the left-hand figure, the triangular mesh would be non-conforming. Since non-conforming finite element meshes pose a variety of

numerical difficulties, adaptive mesh refinement is carried out in an iterative fashion, via a "queue swapping" algorithm [3, 5]. This algorithm, as implemented by the MC libraries [5], creates two empty queues ($Q1$, $Q2$) and fills one ($Q1$) with the list of simplices marked for refinement by the error estimator. The simplices in $Q1$ are refined and the resulting non-conforming simplices (if any) are placed in $Q2$. After all the simplices in $Q1$ have been refined, the roles of the queues are swapped ($Q1 = Q2$, $Q2 = \emptyset$) and the algorithm is repeated. This loop continues until the entire mesh is conforming, whereupon both queues are empty.

3.3. **Multilevel solution.** The time required to solve the linear algebra equations, either within the Newton steps for NPBE or explicitly defined by the LPBE, generally dominates the solution of the PBE. Therefore, it is important to make these steps as efficient as possible. Multilevel methods are well-established techniques for efficiently solving such equations through algebraic hierarchies [9, 21–26]. Such methods have been shown to give optimal (for uniformly refined meshes [27]) or nearly optimal (for adaptively refined meshes [6]) time and memory complexity for the solution of the linear matrix equations.

APBS employs the multilevel finite element solver technology in MC [5] to form an algebraic hierarchy of problems based on the refinement of the mesh [3, 5, 28]. Specifically, a prolongation operator $P_k$ is constructed which relates basis functions on refinement levels $k$ and $k+1$ of the finite element mesh. Given operator $A_k$ on level $k$ of the mesh, the prolongation operator $P_k$ can be used to reconstruct the problems $A_{k+1}$ from coarser levels of the mesh by applying $P_k$ to the current problem $A_k$ via $A_{k+1} = P_k^T A_k P_k$. Using this prolongation-based reconstruction, the problem can then be solved in a multilevel fashion, employing a direct solver for the problem on the coarsest level.

3.4. **Parallel finite element methods.** Using the parallel refinement techniques of Bank and Holst [29], the methods described in the previous sections can be performed in a parallel fashion. In the parallel implementation, each of the $P = 2^p$ processors is given the same initial mesh. Using

the techniques described in Section 3.2, each processor solves the problem on this coarse mesh and generates an *a posteriori* error estimate for every simplex in the mesh. This error estimate is then used to weight a spectral bisection method which partitions the mesh into $P$ pieces of approximately equal error. Finally, each of these mesh partitions $M_i$ is assigned to a different processor $i$. After partitioning, the usual solution and adaptive refinement methods of the previous sections are performed with only a small modification: When the per-simplex error estimates are calculated on processor $i$, only simplices within the local partition $M_i$ and a boundary region of size $\sigma$ surrounding $M_i$ are given nonzero error estimates. The simplices with errors greater than a specified tolerance are marked for refinement, and these marked simplices (which are located only in $M_i$ and the overlap) are subdivided. The queue-swapping algorithm then proceeds as usual with simplices from any partition being refined to ensure fully conforming mesh. The initial error-based partitioning step acts as the load-balancing mechanism for this algorithm; the number of simplices in an error-based adaptive refinement is related to the total error in in the mesh region it is refining. By partitioning the mesh such that all processors have roughly the same amount of error, this algorithm provides a reasonable amount of *a priori* load balancing.

The overlap region surrounding each mesh partition is implemented by APBS in a simple fashion. Let $x_i$ be the center of geometry of partition $M_i$, and let $R_i$ be the radius of the sphere circumscribing $M_i$. The parameter $\sigma \geq 1$ is the desired relative size of the overlap region with respect to $R_i$. APBS then enforces parallel refinement with partition overlap by only allowing error-based simplex marking (on processor $i$) of simplices within a distance $\sigma R_i$ of the center $x_i$ of partition $M_i$. A two-dimensional example of this method applied to a four processor system is shown in Figure 5. In this example, all simplices within $\sigma = 1.2$ times the radius of the green partition were given the same error (which was chosen to be greater than the error-based marking tolerance). The resulting refinement over the green processor's partition and the overlap region is evident, as is the additional refinement outside the radius $\sigma R_i$ required for conformity.

As noted by Bank and Holst [29], this error-decoupling parallel algorithm essentially trades computation for communication. While the algorithm requires little or no communication between processors, it compensates by duplicating the computational effort spent in some portions of the solution algorithm. Specifically, partitioning steps of the mesh and computations on the the solution in overlap regions are duplicated across processors. Although the overlap region can be neglected for some problems [29], a nonzero overlap region proportional to the size of $M_i$ must generally be implemented to satisfy the requirements underlying the decoupled error estimates [29, 30].

## 4. Implementation

The APBS program provides parallel and sequential implementations of the multilevel adaptive finite element techniques described in Section 3 to solve the linear and nonlinear versions of the PBE around biomolecules in ionic solutions. APBS makes extensive use of MC [5] for a variety of tasks, including the implementation of finite element mesh structures, refinement algorithms and data structures, assembly and solution of the linear and nonlinear equations, spectral bisection mesh partitioning, and residual-based error estimation. Because of the underlying hardware abstraction design of MC, the APBS code is designed for portability and can be used, with no modifications, on both single-processor workstations and massively parallel supercomputers. Parallel communication is currently provided via vendor implementations of the MPI 1.1 standard, however, future plans include support for OpenMP protocols to better leverage the capabilities of shared memory platforms. APBS is currently in beta testing phase, but is scheduled for release in open-source form pending the addition of OpenMP support and other features. Information about obtaining MC [5] and related tools is available at http://www.scicomp.ucsd.edu/∼mholst.

The sequence of operations followed by APBS during a typical solution of the PBE is outlined in Algorithm 4.1.

**Algorithm 4.1.** *APBS program execution*

1. *Initialize $P$ processors; all subsequent steps are carried out by each processor.*
2. *Read in the very coarse initial mesh (pre-mesh) and molecule systems.*
3. *For each molecule (or molecular system):*
   (a) *Assign atomic, dielectric, and ionic strength data.*
   (b) *Map atomic charges to mesh simplices.*
   (c) *Construct nonlinear and/or linear algebra structures for each molecular system.*

   *End for.*
4. *Uniformly refine the mesh to contain no less than $cP$ simplices (where $c$ is usually 10-100).*
5. *Solve the PBE and estimate the error for a reference molecular system.*
6. *Partition the mesh into $P$ pieces and assign each piece to a processor.*
7. *While the size of simplices on the molecule-solvent boundaries is too big:*
   (a) *Mark those simplices in the local partition and overlap region (see Section 3.4) which contain a point charge or lie on the boundary between the solvent and the interior of any molecule.*
   (b) *Refine marked simplices; refine the mesh to conformity.*

   *End while.*
8. *While the global error estimate is too big:*
   (a) *Solve the weak form of the PBE for each molecular system.*
   (b) *Estimate the per-simplex error using the residual-based error estimator.*
   (c) *Mark simplices on the local partition or overlap region with errors greater than some tolerance.*
   (d) *Refine marked simplices; refine the mesh to conformity.*

   *End while.*

The details of implementation for the particular steps of this algorithm have been mostly covered in previous sections. In summary, after initialization of the problem, partitioning of the mesh and

initial *a priori* refinement, APBS carries out the adaptive solve-estimate-refine procedure outlined in Section 3.2 until a target accuracy has been reached.

Appropriate *a priori* refinement of the initial mesh can provide acceleration in convergence of the expensive solve-estimate-refine steps (Step 8 in Algorithm 4.1). The first step in the procedure is the reading of a very coarse "pre-mesh" (Step 2 in Algorithm 4.1) which completely encapsulates the desired problem domain. In general, this coarse mesh can be of any polyhedral shape; in practice it is typically a simple rectangular prism comprised of six tetrahedra. To allow for the accurate representation of boundary conditions by an analytical Green's function model, the outer boundary of the pre-mesh should be at least twice the radius of the sphere which circumscribes the molecular complex. After the pre-mesh is read, it must be uniformly refined to a desired number ($cP$) number of simplices for partitioning (Step 4, Algorithm 4.1). In general, a suitable number of simplices for partitioning is roughly $10-100$ times the target number of partitions. This number of initial simplices not only provides a reasonable error estimate for error-weighted partitioning of the mesh, but also provides adequate flexibility for the spectral bisection partitioning algorithm. While uniform refinement is not a requirement, the errors on the very coarse pre-mesh are typically so large that error-based refinement schemes lead to uniform marking and subdivision. Following error-weighted partitioning of the mesh, geometry-based refinement near point charges and molecular surfaces is carried out on the local partition and overlap region until the specified mesh resolution is reached (Step 7 in Algorithm 4.1). This process is essentially an *a priori* estimate of the problematic features of the system that will be refined by subsequent *a posteriori* estimate-refine steps. By subdividing simplices (on the local partition and overlap region of each processor) which lie across the dielectric or ionic strength boundaries or contain charges, the *a priori* refinement scheme is attempting to resolve some of the overall structure of the discontinuous problem coefficients prior to the more costly solve-estimate-refine loop.

## 5. Application to biomolecular systems

One of the advantages of using adaptive methods to solve the PBE is the ability to study large biomolecular systems that are untenable with uniform mesh techniques. One such system of interest is the cytoskeleton, the complex array of filaments and proteins within every eukaryotic cell. The largest cytoskeletal component, the microtubule, is a hollow cylindrical filament (see Figure6) assembled from the long protofilaments composed of tubulin subunits [31, 32]. The microtubule cylinders are 25 nm in diameter and, depending on function, can have lengths from nanometers to several millimeters. While microtubules are the most rigid structures in the cell and play an important structural role, they are also involved in variety of other functions, including cellular transport, motility, and division. Many of these more dynamic functions involve interactions with other proteins or filaments in the cell, often through electrostatic interactions. For this reason, the ability to calculate the electrostatic properties of a microtubule can provide important insight into many cellular processes. It is the large size of microtubules that poses tremendous computational challenges; for example, the atomically detailed solution of the PBE for a 1-$\mu$m-long microtubule requires more than 21 million delta functions in the source term of the PBE to model the charge distribution to full atomic detail.

APBS was used to solve the LPBE for a 40-nm-long microtubule consisting of 605,205 atoms with a net charge of -1800 e. The microtubule structure was assembled by D. Sept using microtubule structures derived from the work of Nogales, Whittaker, Milligan, and Downing [33]. The biomolecule was assigned an internal dielectric constant of 2 and surrounded by a solvent of dielectric 78.54 and ionic strength of 150 mM. The molecular volume was defined with 0.14-nm-radius solvent probes, and the ion accessibility was calculated using 0.20-nm probes. The pre-mesh was a 6-tetrahedron cubic box with 90-nm sides. For each $P$-processor calculation, the pre-mesh was uniformly refined to over $100P$ simplices and partitioned by error-weighted spectral bisection. In order to ensure the best possible load balancing, no *a priori* adaptive refinement was performed.

Instead, each partition in the mesh was subject to the solve-estimate-refine adaptive refinement loop using the residual-based *a posteriori* error estimator until each processor had the target number of vertices (40,000). These calculations were performed on 1, 2, 4, 8, 16, and 32 processors of the NPACI Blue Horizon supercomputer. Blue Horizon is a massively parallel computational platform with eight-way SMP IBM 222 MHz Power3 nodes and 4 GB RAM per node. Because of the low communication costs, each job was submitted to the IP space queues. Jobs requiring less than 32 processors were run with all 8 processors per node, giving each processor roughly 400 MB of heap memory. However, to provide adequate memory for the initial mesh refinement and partitioning steps, the 32-processor job was run with 4 processors per node, allowing approximately 800 MB of memory per processor.

The parallel scaling results for these calculations are shown in Figure 7. Although it was anticipated that each calculation would take roughly the same amount of execution time with the various processor configurations, the actual runs showed a decrease in the execution time with increasing number of processors. The execution times fit the polynomial $t_1 P + t_0$ with a correlation coefficient $r^2 = 0.91$, a slope of $t_1 = (-140 \pm 20)$ s/processor, and an intercept of $t_0 = (14800 \pm 300)$ s. The solid line in Figure 7 shows the *global* number of simplices in the mesh $L(P)$ (the sum of the number of simplices from each partition) as a function of the number of processors. This function was fit to a straight line $L(P) = l_1 P + l_0$ with correlation coefficient $r^2 = 0.999$, slope $l_1 = (2.05 \pm 0.03) \times 10^5$ simplices/processor, and intercept $l_0 = (1.7 \pm 0.6) \times 10^5$ simplices. Finally, the parallel efficiency was defined as

$$E(P) = \frac{L(P)}{PL(1)} \tag{5.1}$$

and plotted as the dashed line on Figure 7. The efficiency was also fit to a linear polynomial $E(P) = e_1 P + e_0$ with correlation coefficient $r^2 = 0.61$, slope $e_1 = (7 \pm 3) \times 10^{-3}$ per processor, and intercept $e_0 = (1.08 \pm 0.05)$. The mean efficiency of the six runs was $\overline{E} = 1.0 \pm 0.1$.

As shown in Figure 7, APBS exhibits excellent scaling behavior for up to 32 processors. The parallel efficiency is very high for all processor configurations and shows only a slight decrease for larger (16 and 32) calculations. The "superlinear" scaling ($E(2) = 1.9$ and $E(4) = 1.08$) of the two and four processor configurations can be considered an artifact of the parallel efficiency definition. Since it is very difficult to refine mesh partitions to an exact number of simplices, the solve-estimate-refine loop can only be constrained to refine the partition to contain *more* than a specified number of simplices, therefore not providing an exact cutoff for each partition and processor configuration. The resulting scatter in $L(1)$ and $L(P)$ leads to parallel efficiencies that can deviate, both positively and negatively, from their "ideal" values.

Due to the large size of the resulting electrostatic potential data sets, it was not possible to visualize the results of the parallel calculations shown in Figure 7. However, a much lower resolution calculation was performed on a slightly larger (60 nm long, 901,083 atoms, -3000 e charge) microtubule to generate the electrostatic potential contours shown in Figure 8. As expected, the highly charged microtubule shows mostly negative electrostatic potential near the molecular surface (Figure 8, red contour). However, several regions of positive potential are visible, especially near the ends of the microtubule. Such localized variations in electrostatic potential often play important roles in molecular recognition and binding and suggest interesting modes of microtubule assembly and stability.

## 6. Conclusions

Using new methods for the parallel solution of elliptic partial differential equations, it is possible to leverage the teraflops computing power of massively parallel computers to perform electrostatic calculations on biological systems at scales approaching the cellular level. Using the APBS and MC software on the NPACI IBM Blue Horizon supercomputer, it was possible to solve the Poisson-Boltzmann equation for the electrostatic potential around a microtubule containing more than

600,000 atoms. The code showed excellent parallel scaling, providing incentive to attempt further calculations to probe the structure and function of even larger macromolecules.

Future work is aimed at the inclusion of OpenMP extensions into APBS to take better advantage of shared-memory machines and reduce the memory overhead associated with the duplicated storage of biomolecular atomic information. Work is also in progress to utilize more of the Blue Horizon's parallel capabilities and investigate much larger biomolecular systems involving calculations on millions of atoms. By enabling such research through the use of parallel computing technology, theoreticians should be able to move from computational chemistry at molecular scales to computational biology at the cellular level.

## Acknowledgements

N. Baker's work was supported by predoctoral fellowships from the Howard Hughes Medical Institute and the Burroughs Wellcome La Jolla Interfaces in Science program. M. Holst's work was supported in part by a UCSD Hellman Fellowship, and in part by NSF CAREER Award 9875856. J.A. McCammon's work was supported in part by grants from NIH and NSF. N. Baker is indebted to Amit Majumdar, Giridhar Chukkapalli, and Greg Johnson at the San Diego Supercomputer Center for help with technical issues on the Blue Horizon and visualization of the large data sets. N. Baker would also like to thank Dr. Adrian Elcock (University of California San Diego, Dept. of Chemistry and Biochemistry) for helpful explanations of the various solvent- and ion-accessible volume definitions used in PBE solvers.

## References

[1] M. E. Davis and J. A. McCammon. Electrostatics in biomolecular structure and dynamics. *Chem. Rev.*, 94:7684–7692, 1990.

[2] B. Honig and A. Nicholls. Classical electrostatics in biology and chemistry. *Science*, 268:1144–1149, 1995.

[3] Michael J. Holst, Nathan A. Baker, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I: algorithms and examples. *J. Comp. Chem*, in press.

[4] Nathan A. Baker, Michael J. Holst, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II: refinement at solvent accessible surfaces in biomolecular systems. *J. Comp. Chem*, in press.

[5] Michael J. Holst. *Adaptive multilevel finite element methods on manifolds and their implementation in MC.* (In preparation; currently available as a UCSD Dept. of Mathematics technical report and user's guide to the MC software).

[6] R. E. Bank, T. F. Dupont, and H. Yserentant. The hierarchical basis multigrid method. *Numer. Math.*, 52:427–458, 1988.

[7] K. A. Sharp and B. Honig. Calculating total electrostatic energies with the nonlinear Poisson-Boltzmann equation. *J. Phys. Chem.*, 94:7684–7692, 1990.

[8] B. Lee and F. M. Richards. The interpretation of protein structures: estimation of static accessibility. *J. Mol. Biol.*, 55:379–400, 1971.

[9] M. Holst and F. Saied. Numerical solution of the nonlinear Possion-Boltzmann equation: developing more robust and efficient methods. *J. Comput. Chem.*, 16:337–364, 1995.

[10] R. E. Bank and R. K. Smith. Parameter selection for Newton-like methods applicable to nonlinear partial differential equations. *SIAM J. Numer. Anal.*, 17:806–822, 1980.

[11] R. E. Bank and R. K. Smith. Global approximate Newton methods. *Numer. Math.*, 37:279–295, 1981.

[12] R. E. Bank and R. K. Smith. Analysis of a multilevel iterative method for nonlinear finite element equations. *Math. Comp.*, 39:453–465, 1982.

[13] R. S Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[14] O. Axelsson and V. A. Barker. *Finite element solution of boundary value problems. Theory and computation.* Academic Press, San Diego, CA, 1984.

[15] D. Braess. *Finite elements. Theory, fast solvers, and applications in solid mechanics.* Cambridge Univ. Press, New York, 1997.

[16] M. Holst and D. Bernstein. Adaptive finite element solution of the initial-value problem in general relativity: Theory and algorithms. *Comm. Math. Phys.*, submitted.

[17] I. Babuška and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15:736–754, 1978.

[18] I. Babuška and W. C. Rheinboldt. A posteriori error estimates for the finite element method. *Int. J. Numer. Meth. Engrg.*, 12:1597–1615, 1978.

[19] R. Verfürth. A posteriori error estimates for nonlinear problems. Finite element discretization of elliptic equations. *Math. Comp.*, 62:445–475, 1994.

[20] R. Verfürth. *A review of a posteriori error estimation and adaptive mesh-refinement techniques*. John Wiley, New York, 1996.

[21] R. E. Bank and J. Xu. The hierarchical basis multigrid method and incomplete LU decomposition. In D. Keyes and J. Xu, editors, *Seventh international symposium on domain decomposition methods for partial differential equations*, pages 163–173. AMS, 1994.

[22] A. Brandt. Algebraic multigrid theory: the symmetric case. *Appl. Math. Comp.*, 19:23–56, 1986.

[23] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and its applications*. Cambridge Univ. Press, 1984.

[24] M. Holst and F. Saied. Multigrid solution of the Poisson-Boltzmann equation. *J. Comput. Chem.*, 14:105–113, 1993.

[25] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid methods*, volume 3 of *Frontiers in applied mathematics*, pages 73–130. SIAM, Philadelphia, 1987.

[26] W. Hackbusch. *Multi-grid methods and applications*. Springer-Verlag, Berlin, 1985.

[27] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34:581–613, 1992.

[28] M. Holst and S. Vandewalle. Schwarz methods: to symmetrize or not to symmetrize. *SIAM J. Numer. Anal*, 34:699–722, 1997.

[29] R. Bank and M. Holst. A new paradigm for parallel adaptive meshing algorithms. *SIAM J. Sci. Comput.*, in press.

[30] Jinchao Xu and Aihui Zhou. Local and parallel finite element algorithms based on two-grid discretizations. *Math. Comp.*, 69:881–909, 2000.

[31] P. Dustin. *Microtubules*. Springer-Verlag, Berlin, 1984.

[32] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson. *Molecular biology of the cell*. Garland Publishing, New York, 1994.

[33] E. Nogales, M. Whittaker, R. A. Milligan, and K. H. Downing. High-resolution model of the microtubule. *Cell*, 96:79–88, 1999.

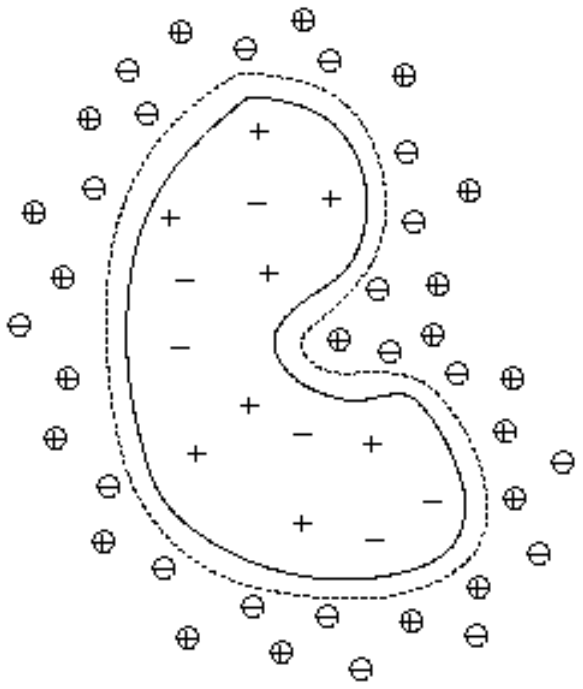

FIGURE 1. Schematic of a model protein-solvent system. Charges within the protein are depicted as plus and minus symbols. The first protein-solvent boundary (solid line) represents discontinuities in the dielectric coefficient $\epsilon$, while the second boundary (dashed line) represents discontinuities in the screening parameter $\overline{\kappa}^2$. Finally, the solvated ions surrounding the protein are depicted by the circled plus and minus symbols.

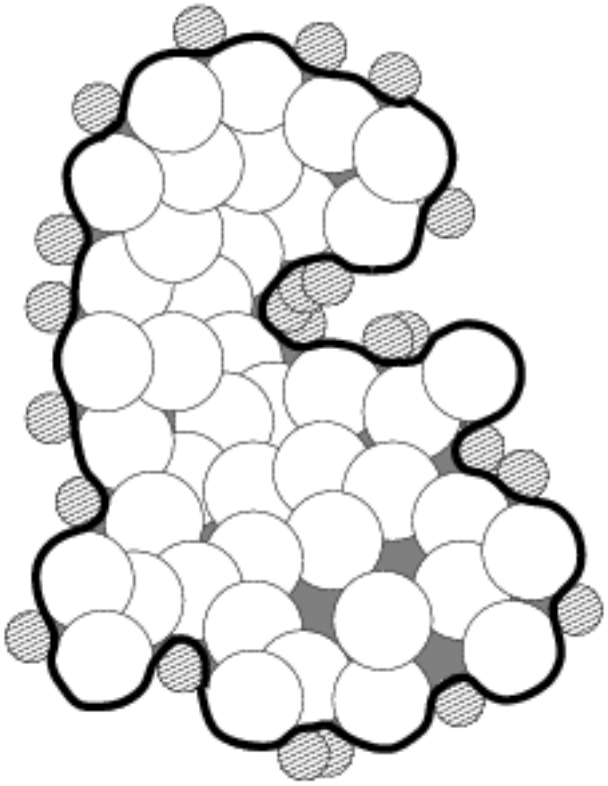

FIGURE 2. Representation of dielectric assignment based on solvent accessibility. The molecular surface (heavy black curve) is defined by the solvent probes (hashed circles) and the protein atoms (white circles). The gray areas represent regions outside the atomic radii that are not accessible to the solvent molecules and therefore inside the molecular surface.

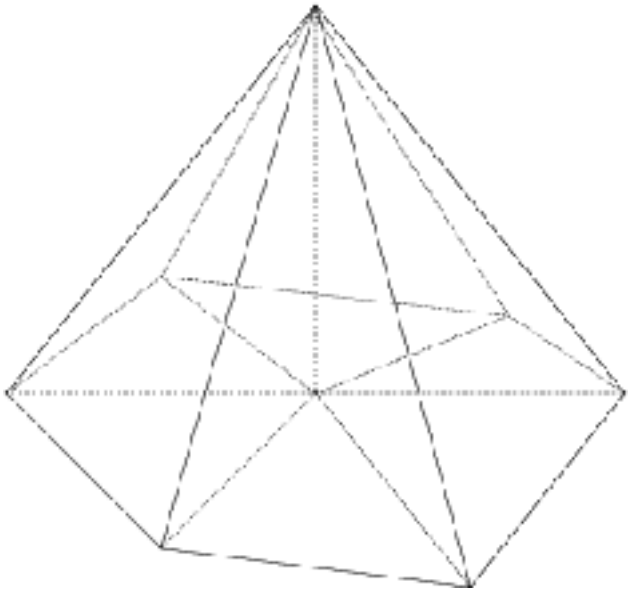

FIGURE 3. Piecewise-linear finite element basis function on a two-dimensional (triangular) mesh.

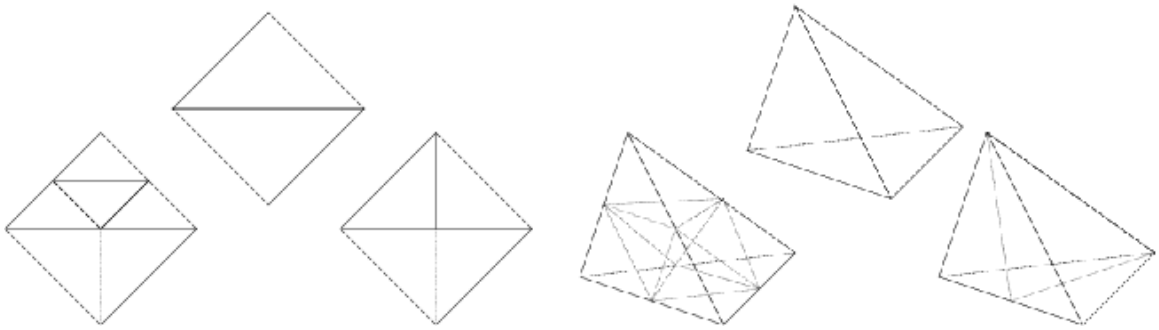

FIGURE 4. Simplex subdivision schemes. The figures on the left show simplex subdivision of a mesh in two dimensions (upper) via quadrasection (lower left) and longest-edge bisection (lower right). The dotted lines on the lower figures show the additional subdivisions that must be performed to ensure a conforming mesh. The figures on the right show subdivision of a single simplex in three dimensions (upper) via octasection (lower left) and longest-edge bisection (lower right).

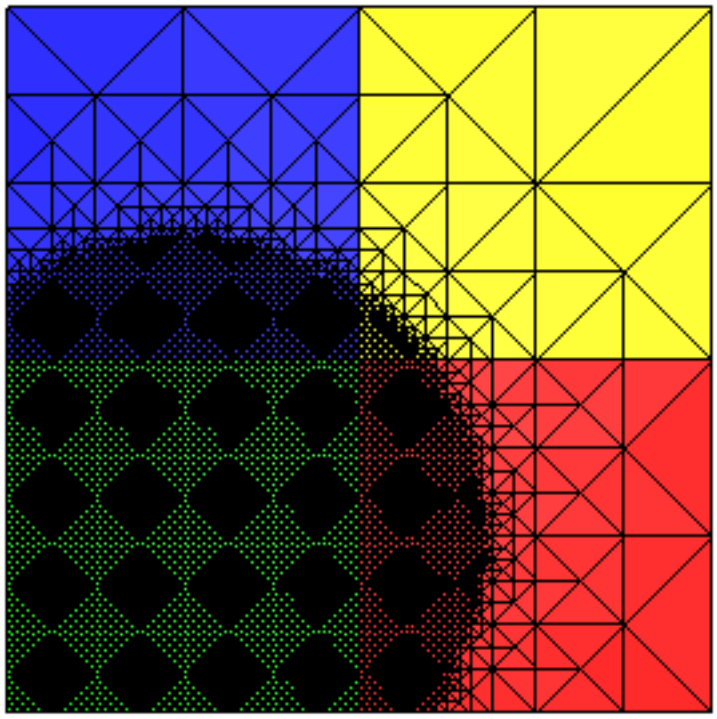

FIGURE 5. Overlapping refinement on a partitioned mesh. All simplices within $\sigma = 1.2$ times the radius of the green partition were given the same error (greater than the simplex marking tolerance). Longest-edge bisection was performed via the queue-swapping algorithm (see Section 3.2) until the mesh was conformal. The checkerboard pattern within the refined region is an artifact of the image, a Moiré pattern due to the high density of simplex edges.

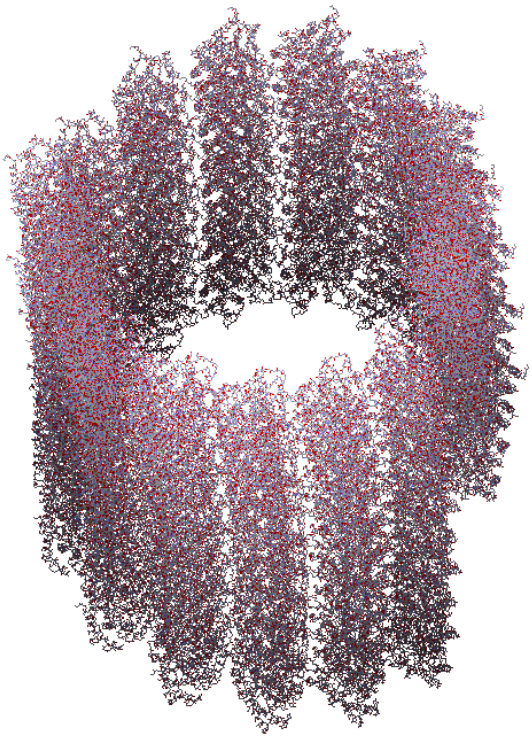

FIGURE 6. The amino acid backbone atoms of a microtubule. This structure is 25 nm in diameter, 40 nm in length and has 901,083 atoms and a -3000 e charge.

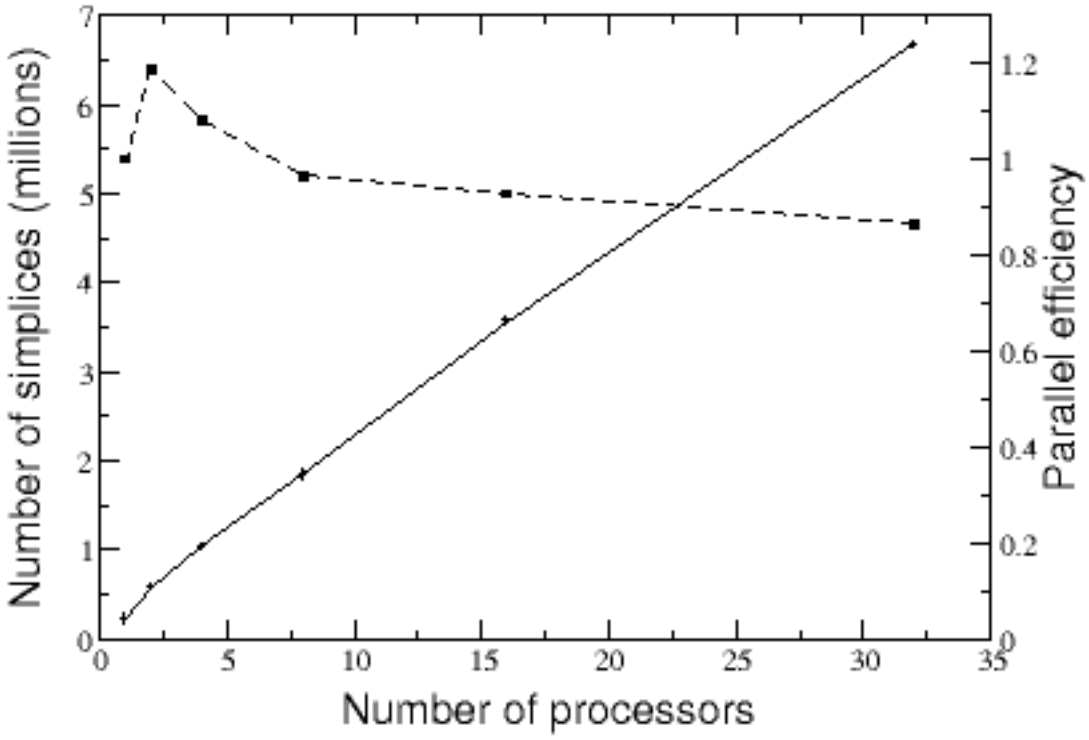


FIGURE 7. Parallel scaling of electrostatics calculations for a microtubule using APBS on the NPACI Blue Horizon supercomputer. The solid line shows the number of simplices in the global mesh and the dotted line depicts the parallel efficiency.

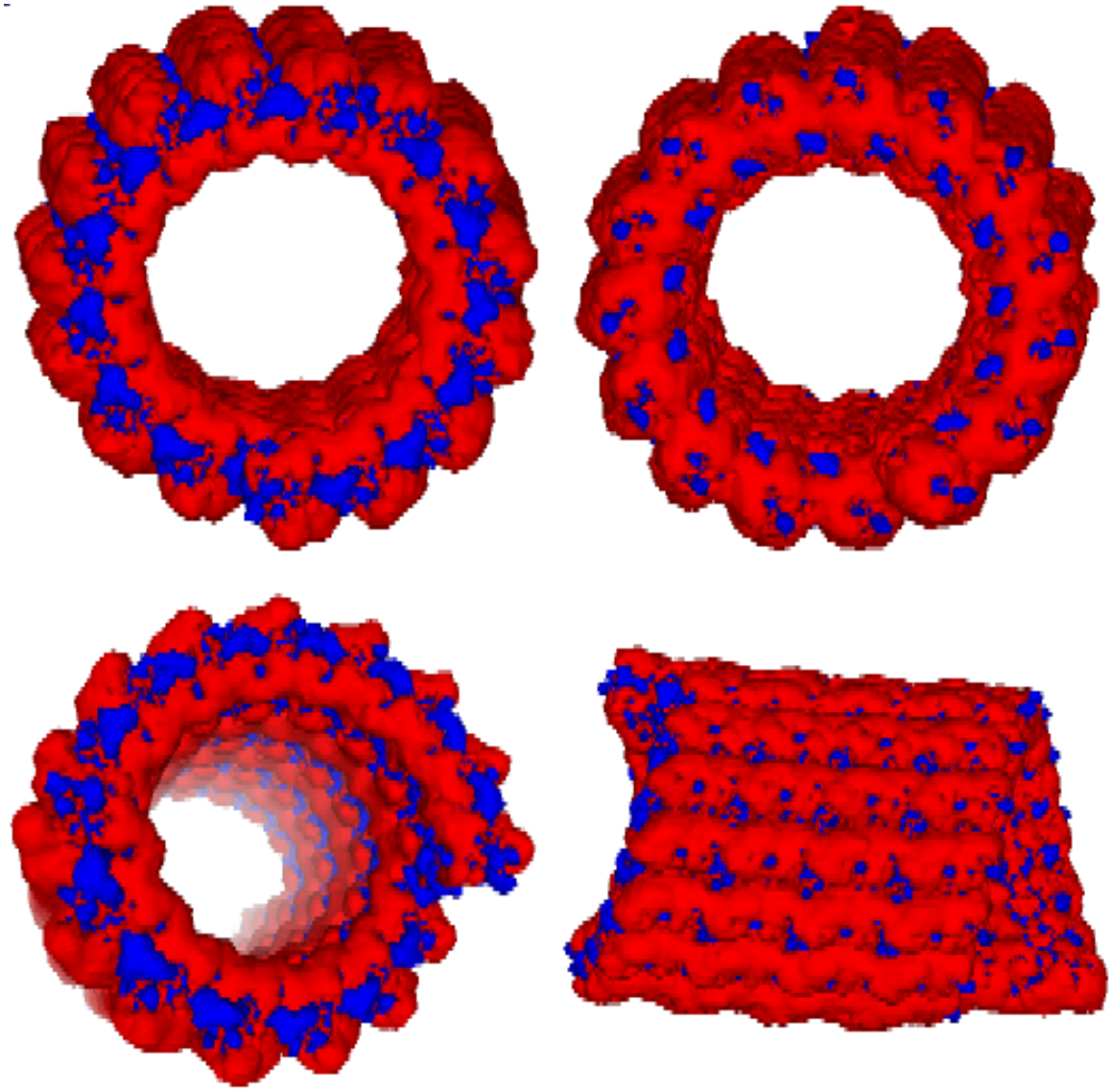

FIGURE 8. Electrostatic potential of a 605,205-atom microtubule fragment at 150 mM ionic strength. Potential contours are shown at +0.5 $kT/e$ (blue) and -0.5 $kT/e$ (red). Each image represents a different view of the macromolecule. The upper images show the electrostatic complementarity at ends of the microtubule, while the lower images show the varying positive and negative regions of the potential on the exterior and interior of the protein.