Research Article **Open Access**

Weihua Geng*

A boundary integral Poisson-Boltzmann solvers package for solvated bimolecular simulations

DOI 10.1515/mlbmb-2015-0004
Received February 20, 2015; accepted May 30, 2015

Abstract: Numerically solving the Poisson-Boltzmann equation is a challenging task due to the existence of the dielectric interface, singular partial charges representing the biomolecule, discontinuity of the electrostatic field, infinite simulation domains, etc. Boundary integral formulation of the Poisson-Boltzmann equation can circumvent these numerical challenges and meanwhile conveniently use the fast numerical algorithms and the latest high performance computers to achieve combined improvement on both efficiency and accuracy. In the past a few years, we developed several boundary integral Poisson-Boltzmann solvers in pursuing accuracy, efficiency, and the combination of both. In this paper, we summarize the features and functions of these solvers, and give instructions and references for potential users. Meanwhile, we quantitatively report the solvation free energy computation of these boundary integral PB solvers benchmarked with Matched Interface Boundary Poisson-Boltzmann solver (MIBPB), a current 2nd order accurate finite difference Poisson-Boltzmann solver.

Keywords: Electrostatics; Solvated biomolecule; Poisson-Boltzmann equation; Boundary integral equation; Treecode

1 Introduction

Electrostatic interactions are vital for the study of biological and chemical systems and processes at the molecular level. In a medium with dielectric constant (or relative permittivity) ε, the Gauss's Law relates the electrostatic potential ϕ in the space to the electrostatic charge density ρ by the classical Poisson's equation $-\nabla\cdot(\varepsilon\nabla\phi) = \rho$. The dielectric coefficient ε in general is a function of the space but can also be simplified as a piecewise constant function with constant value in each particular medium (e.g. for a solvated biomolecule as shown in Figure 1(a), ε=1-10 inside molecule region Ω_1 and ε= 70-80 in the solvent region Ω_2). The charge density ρ usually contains two components. One is the summation of weighted delta functions representing the partial changes assigned to the atomic centers of the biomolecule. The other is the Boltzmann distribution of the mobile ions (dissolved electrolytes), which takes valence number of the ionic species, electrostatic potential, and temperature into account. In addition, we need to consider interface conditions such as the continuity of electrostatic potential ϕ and flux density $\varepsilon\frac{\partial\phi}{\partial\nu}$ normal to dielectric boundary $\Gamma = \partial\Omega_1 = \partial\Omega_2$ as seen in Figure 1(a). By combining all these components and assuming the weak ionic strength we have the linearized Poisson-Boltzmann equation (1)-(2), its interface jump conditions (3), and the boundary condition (4) as the following.

$$-\varepsilon_1\nabla^2\phi(\mathbf{x}) = \sum_{k=1}^{N_c} q_k\delta(\mathbf{x}-\mathbf{y}_k), \quad \mathbf{x}\in\Omega_1, \tag{1}$$

***Corresponding Author: Weihua Geng:** Department of Mathematics, Southern Methodist University, Dallas, TX 75275 USA, E-mail: wgeng@smu.edu

© 2015 Weihua Geng, licensee De Gruyter Open.
This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License.

$$-\varepsilon_2 \nabla^2 \phi(\mathbf{x}) + \bar{\kappa}^2 \phi(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega_2, \tag{2}$$

$$\phi_1(\mathbf{x}) = \phi_2(\mathbf{x}), \quad \varepsilon_1 \frac{\partial \phi_1(\mathbf{x})}{\partial \nu} = \varepsilon_2 \frac{\partial \phi_2(\mathbf{x})}{\partial \nu}, \quad \mathbf{x} \in \Gamma, \tag{3}$$

$$\lim_{|\mathbf{x}| \to \infty} \phi(\mathbf{x}) = 0. \tag{4}$$

where $\varepsilon_1, \varepsilon_2$ are the dielectric constants in Ω_1, Ω_2, respectively, q_k is the partial atomic charge, δ is the Dirac delta function, ν is the outward unit normal vector on Γ, and $\bar{\kappa}$ is the modified Debye-Hückel parameter (modified to dielectric independent) measuring the ionic concentration. We have $\bar{\kappa}^2 = \kappa^2 \varepsilon_2$ where κ^2 is Debye-Hückel parameter. For details about these parameters and units, please refer to the appendix of this article.

In all work mentioned in this article the interface Γ is the solvent excluded surface (also called the molecular surface) obtained by rolling a solvent sphere over the van der Waals surface of the solute [8, 21]. Figure 1(a) gives an illustration in 2-D for the molecular surface. In protein simulation, we use the software MSMS [30] to generate the molecular surface and Figure 1(b) gives the triangulated molecular surface of a sample protein (PDB: 1ajj) with 519 atoms.

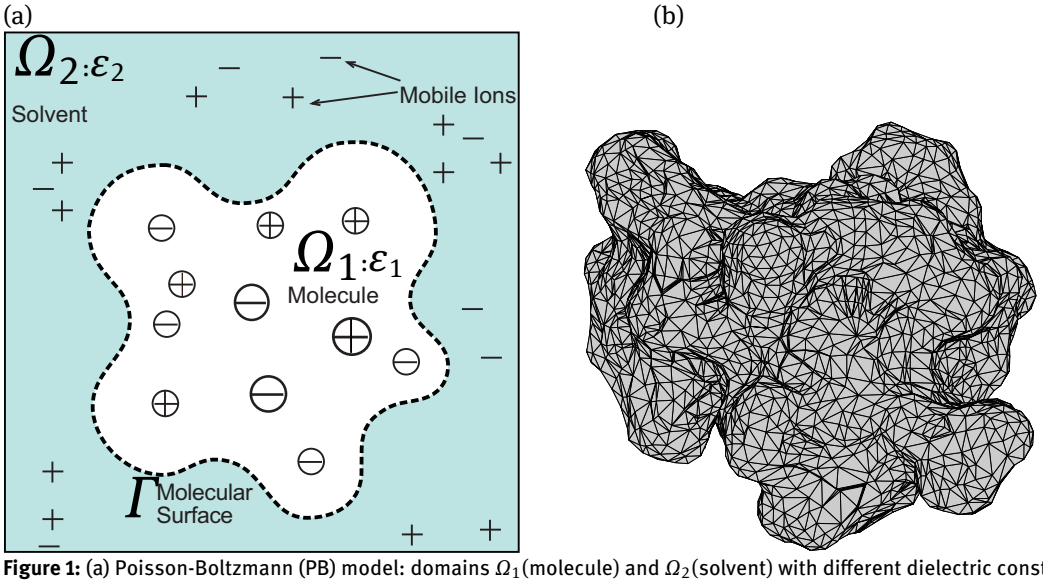

Figure 1: (a) Poisson-Boltzmann (PB) model: domains Ω_1(molecule) and Ω_2(solvent) with different dielectric constants ε_1 and ε_2 respectively; (b) the molecular surface is formed by the trace of solvent probe in contact with the solute (molecule).

Numerical solution of the PB equation (1)-(2) subject to its interface and boundary conditions (3)-(4) is challenging for several reasons:

1) the biomolecule is represented by singular point charges as seen in Equation (1),

2) the molecular surface Γ is geometrically complex,

3) the dielectric constant is discontinuous across the molecular surface, which leads the discontinuity of electric field $-\nabla\phi$ thus the non-smoothness of the solution ϕ across Γ,

4) the domain is unbounded.

Numerical methods for solving the PB equation fall into two categories: 1) grid-based finite-difference methods (FDM) [17, 18, 26–28] and finite-element methods (FEM) [2, 3, 9, 15], and 2) surface-based boundary element methods (BEM) [5–7, 19, 23, 24, 31, 32]. Finite-difference PB solvers are widely used and available in molecular dynamics software packages such as AMBER [26, 27], CHARMM [18], and Delphi [17, 28]. Boundary element PB solvers are less well-developed, but they have advantages:

1) only surface discretizations are needed, rather than volume discretizations,

2) the singular point charges are treated analytically,

3) the molecular surface can be represented accurately using boundary elements,

4) the interface conditions can be explicitly enforced,
5) the far-field boundary condition can be imposed analytically.

More details about Poisson-Boltzmann model, formulation, numerical algorithm and applications can be found in some very informative reviews [1, 10, 25]. The contribution of the present boundary integral PB solvers package are threefold. First, the package contains the first boundary integral PB solver using fast treecode and its parallelization to achieve significantly improved efficiency [13]. Second, the package contains the first boundary integral PB solver achieving higher-order accuracy by combining numeral techniques such as curved triangulation, singularity removal, higher-order quadrature, and parallelization [11]. Third, the package contains the first well-posed boundary integral PB solver designed for running on GPUs [12].

The rest of the paper is organized as the following. In section 2, we introduce the related algorithms of the boundary integral PB solvers including formulation, discretization, treecode algorithm, and GPU implementation. In section 3, we provide some brief numerical results for accuracy comparison of different solvers. This paper is ended with a section of concluding remarks.

2 Algorithms

In this section, we introduce numerical algorithms applied in boundary integral PB solvers. The core algorithms are:
1) The well-posed boundary integral formulation of the PB equation,
2) The centroid triangular discretization,
3) The higher-order curved triangulation with singularity removal,
4) MPI-based parallelization,
5) Treecode algorithm,
6) GPU-based acceleration.

There are tradeoffs between accuracy and efficiency when these algorithms are applied or not. With selected combination of these algorithms or not, we have produced three solvers as:
TABI-PB, the treecode-accelerated boundary integral PB solver using algorithms 1),2),4), and 5);
HOBI-PB, the higher-order boundary integral PB solver using algorithm 1), 3), and 4);
GABI-PB, the GPU accelerated boundary integral PB solver using algorithm 1), 2), and 6).

Since all PB solvers use 1) the well-possed boundary integral formulation [19], we started this section with a brief introduction of the formulation followed by other related algorithms.

2.1 Boundary Integral PB Formulation

Starting from Equations (1)-(2) together with their fundamental solutions, the Coulomb potential $G_0(\mathbf{x}, \mathbf{y})$ and screened Coulomb potential $G_\kappa(\mathbf{x}, \mathbf{y})$ as,

$$G_0(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi|\mathbf{x}-\mathbf{y}|}, \quad G_\kappa(\mathbf{x}, \mathbf{y}) = \frac{e^{-\kappa|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|}, \tag{5}$$

utilizing the interface jump conditions in Equation (3), Juffer et al. [19] derived the well-posed integral PB formulation governing the surface potential ϕ_1 and its normal derivative $\frac{\partial\phi_1}{\partial\nu}$ on Γ,

$$\frac{1}{2}(1+\varepsilon)\,\phi_1(\mathbf{x}) = \int_\Gamma \left[K_1(\mathbf{x}, \mathbf{y})\frac{\partial\phi_1(\mathbf{y})}{\partial\nu} + K_2(\mathbf{x}, \mathbf{y})\phi_1(\mathbf{y})\right] dS_\mathbf{y} + S_1(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{6a}$$

$$\frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial\phi_1(\mathbf{x})}{\partial\nu} = \int_\Gamma \left[K_3(\mathbf{x}, \mathbf{y})\frac{\partial\phi_1(\mathbf{y})}{\partial\nu} + K_4(\mathbf{x}, \mathbf{y})\phi_1(\mathbf{y})\right] dS_\mathbf{y} + S_2(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{6b}$$

where $\varepsilon = \varepsilon_2/\varepsilon_1$. The kernels K_{1-4} are defined by

$$K_1(\mathbf{x}, \mathbf{y}) = G_0(\mathbf{x}, \mathbf{y}) - G_\kappa(\mathbf{x}, \mathbf{y}), \quad K_2(\mathbf{x}, \mathbf{y}) = \varepsilon\frac{\partial G_\kappa(\mathbf{x}, \mathbf{y})}{\partial\nu_\mathbf{y}} - \frac{\partial G_0(\mathbf{x}, \mathbf{y})}{\partial\nu_\mathbf{y}}, \tag{7a}$$

$$K_3(\mathbf{x},\mathbf{y}) = \frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}} - \frac{1}{\varepsilon}\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}}, \quad K_4(\mathbf{x},\mathbf{y}) = \frac{\partial^2 G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}\partial \nu_{\mathbf{y}}} - \frac{\partial^2 G_0(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}\partial \nu_{\mathbf{y}}}, \tag{7b}$$

The source terms $S_{1,2}$ are defined by

$$S_1(\mathbf{x}) = \frac{1}{\varepsilon_1}\sum_{k=1}^{N_c} q_k G_0(\mathbf{x},\mathbf{y}_k), \quad S_2(\mathbf{x}) = \frac{1}{\varepsilon_1}\sum_{k=1}^{N_c} q_k \frac{\partial G_0(\mathbf{x},\mathbf{y}_k)}{\partial \nu_{\mathbf{x}}}. \tag{8}$$

Equations (6a)-(6b) comprise a set of coupled second kind integral equations for the surface potential ϕ_1 and its normal derivative $\frac{\partial \phi_1}{\partial \nu}$ on Γ. These well-posed boundary integral equations are applied in PB solvers contained in this package. We next introduced related numerical algorithms.

2.2 Discretization

We discretize the molecular surface Γ by running the software MSMS [30] for triangulation. MSMS takes coordinates and radius of the atoms as input and provides triangles and normal vectors at their vertices as the output. We discretize the integral equations in Equations (6a)-(6b) by using purely the flat triangles [13] with centroid collocation or using curved triangles raised by normal vectors [11] with a higher-order scheme.

2.2.1 centroid collocation

The integrals are discretized by centroid collocation. Let $\mathbf{x}_i, A_i, i = 1,\ldots,N$ denote the centroids and areas of the faces in the triangulation. Then the discretized equations (6a)-(6b) have the following form for $i = 1,\ldots,N$,

$$\frac{1}{2}(1+\varepsilon)\,\phi_1(\mathbf{x}_i) = \sum_{\substack{j=1\\ j\neq i}}^{N}\left[K_1(\mathbf{x}_i,\mathbf{x}_j)\frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_2(\mathbf{x}_i,\mathbf{x}_j)\phi_1(\mathbf{x}_j)\right]A_j + S_1(\mathbf{x}_i), \tag{9a}$$

$$\frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial \phi_1(\mathbf{x}_i)}{\partial \nu} = \sum_{\substack{j=1\\ j\neq i}}^{N}\left[K_3(\mathbf{x}_i,\mathbf{x}_j)\frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_4(\mathbf{x}_i,\mathbf{x}_j)\phi_1(\mathbf{x}_j)\right]A_j + S_2(\mathbf{x}_i). \tag{9b}$$

The term $j = i$ is omitted as a simple but effective way to avoid the kernel singularity.

Equations (9a)-(9b) define a linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ contains the surface potential values $\phi_1(\mathbf{x}_i)$ and normal derivative values $\frac{\partial \phi_1}{\partial \nu}(\mathbf{x}_i)$, and $\mathbf{b}$ contains the source terms $S_1(\mathbf{x}_i), S_2(\mathbf{x}_i)$. The linear system is solved by GMRES iteration which requires a matrix-vector product in each step [29]. Since the matrix is dense, computing the product by direct summation requires $O(N^2)$ operations, which is prohibitively expensive when N is large, and in a later section we describe the treecode algorithm used to accelerate the product. The treecode has been successfully applied on the centroid discretization algorithm, achieving significant acceleration compared with the direct summation [13]. We also pursued a higher-order scheme to improve discretization accuracy. Although including the treecode in this higher-order scheme is rather difficult, accuracy improvement is significant.

2.2.2 higher order schemes

Given the positions and radii of all atoms of a biomolecule, MSMS [30] can generate a discretized surface with a set of $N_f = N$ flat triangles, N_v nodes (vertices) and corresponding normal directions. To achieve higher-order accuracy, we apply the schemes described in [33] to convert these flat triangles to curved triangles for solving the well-posed integral PB equation in Equations (6a) and (6b). We also modify and improve these schemes by using transformation to remove the singularities of kernels K_{1-4} in Equations (7a) and (7b).

Higher-order quadratures (4-point Gauss-Radau quadrature for regular integral and 16-point Gauss-Legendre quadrature for singular integral) are adopted to discretize Equations (6a) and (6b). Here we omit the details which can be found in [11], and only list the discretized formulation.

For $i = 1, 2, \ldots, N_v$, the ith and the $(i+N_v)$th element of the discretized matrix-vector product $\mathbf{Au}$ are given as

$$\{\mathbf{Au}\}_i^r = \frac{1}{2}(1+\varepsilon)\,\phi_1(\mathbf{x}_i) - \sum_{j=1}^{N_i^r}\sum_{m=1}^{N_{GR}}\sum_{n=1}^{3} W_{j,m,n}\left[K_1(\mathbf{x}_i,\mathbf{y}_{j,m})\frac{\partial\phi_1(\mathbf{x}_{j,n})}{\partial\nu_{\mathbf{x}_{j,n}}} + K_2(\mathbf{x}_i,\mathbf{y}_{j,m})\phi_1(\mathbf{x}_{j,n})\right] \tag{10}$$

$$\{\mathbf{Au}\}_{i+N_v}^r = \frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial\phi_1(\mathbf{x}_i)}{\partial\nu_{\mathbf{x}_i}} - \sum_{j=1}^{N_i^r}\sum_{m=1}^{N_{GR}}\sum_{n=1}^{3} W_{j,m,n}\left[K_3(\mathbf{x}_i,\mathbf{y}_{j,m})\frac{\partial\phi_1(\mathbf{x}_{j,n})}{\partial\nu_{\mathbf{x}_{j,n}}} + K_4(\mathbf{x}_i,\mathbf{y}_{j,m})\phi_1(\mathbf{x}_{j,n})\right] \tag{11}$$

In Equations (10) and (11), the superscript r stands for regular triangle, $W_{j,m,n}$ contains the weights and coefficients associated with the quadrature, the transformation Jacobian, and the interpolation coefficients. Note that N_i^r is the number of regular triangles associated with the ith vertex. In these equations, $\mathbf{x}_{j,n}$ are the same set of nodes as $\mathbf{x}_i$, but $\mathbf{y}_{j,m}$ are the quadrature points on the jth curved element, which are mapped from the predetermined points on the unit right triangle. This mismatch brings difficulty to apply Fast Multipole Method or treecode to accelerate the higher-order scheme. Studies about this issue will be proceeded in our future work.

In the treatment of singularities, if the number of quadrature points used in each direction of the Gauass-Lagendre quadrature is N_{GL}, the ith and $(i+N_v)$th element of the discretized matrix-vector product $\mathbf{Au}$ will in stead contain the following singular component

$$\{\mathbf{Au}\}_i^s = \frac{1}{2}(1+\varepsilon)\,\phi_1(\mathbf{x}_i) - \sum_{j=1}^{N_i^s}\sum_{m=1}^{(N_{GL})^2}\sum_{n=1}^{3} W_{j,m,n}\left[K_1(\mathbf{x}_i,\mathbf{y}_{j,m})\frac{\partial\phi_1(\mathbf{x}_{j,n})}{\partial\nu_{\mathbf{x}_{j,n}}} + K_2(\mathbf{x}_i,\mathbf{y}_{j,m})\phi_1(\mathbf{x}_{j,n})\right] \tag{12}$$

$$\{\mathbf{Au}\}_{i+N_v}^s = \frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial\phi_1(\mathbf{x}_i)}{\partial\nu_{\mathbf{x}_i}} - \sum_{j=1}^{N_i^s}\sum_{m=1}^{(N_{GL})^2}\sum_{n=1}^{3} W_{j,m,n}\left[K_3(\mathbf{x}_i,\mathbf{y}_{j,m})\frac{\partial\phi_1(\mathbf{x}_{j,n})}{\partial\nu_{\mathbf{x}_{j,n}}} + K_4(\mathbf{x}_i,\mathbf{y}_{j,m})\phi_1(\mathbf{x}_{j,n})\right] \tag{13}$$

In Equations (12) and (13), the superscript s stands for singular triangle, and $W_{j,m,n}$ contains weights and coefficients associated with quadratures, transformation Jacobian, and interpolation coefficients. Note that N_i^s is the number of (singular) triangles containing the ith vertex.

2.3 Treecode algorithm and its application to accelerate the PB solver

We here summarize the treecode algorithm and refer to previous work for more details [4, 22]. The required sums in Equations (9a)-(9b) have the form of N-body potentials,

$$V_i = \sum_{\substack{j=1\\ j\neq i}}^{N} q_j G(\mathbf{x}_i,\mathbf{x}_j), \quad i = 1,\ldots,N, \tag{14}$$

where G is a kernel, $\mathbf{x}_i$, $\mathbf{x}_j$ are locations of target and source particles, and q_j is a charge associated with $\mathbf{x}_j$. For example, the term involving K_1 on the right side of Equation (9a) has the form given in Equation (14) with $q_j = \frac{\partial\phi_1(\mathbf{x}_j)}{\partial\nu}A_j$. To evaluate the potentials V_i rapidly, the source particles $\mathbf{x}_j$'s are divided into a hierarchy of clusters having a tree structure. The root cluster is a cube containing all the particles and subsequent levels are obtained by dividing a parent cluster into eight children [4]. The process continues until a cluster has fewer than N_0 particles (a user-specified parameter).

Once the clusters are determined, the treecode evaluates the potential in Equation (14) as a sum of particle-cluster interactions,

$$V_i \approx \sum_{c\in N_i}\sum_{\mathbf{x}_j\in c} q_j G(\mathbf{x}_i,\mathbf{x}_j) + \sum_{c\in F_i}\sum_{||\mathbf{k}||=0}^{p} a^{\mathbf{k}}(\mathbf{x}_i,\mathbf{x}_c)\,m_c^{\mathbf{k}}, \tag{15}$$

where c denotes a cluster, and N_i, F_i denote the near-field and far-field of particle $\mathbf{x}_i$. The first term on the right is a direct sum for particles $\mathbf{x}_j$ near $\mathbf{x}_i$, and the second term is a pth order Cartesian Taylor approximation about the cluster center $\mathbf{x}_c$ for clusters that are well-separated from $\mathbf{x}_i$. The Taylor coefficients are given by

$$a^{\mathbf{k}}(\mathbf{x}_i, \mathbf{x}_c) = \frac{1}{\mathbf{k}!} \partial_{\mathbf{y}}^{\mathbf{k}} G(\mathbf{x}_i, \mathbf{x}_c), \tag{16}$$

and the cluster moments are given by

$$m_c^{\mathbf{k}} = \sum_{\mathbf{x}_j \in c} q_j (\mathbf{x}_j - \mathbf{x}_c)^{\mathbf{k}}. \tag{17}$$

Cartesian multi-index notation is used with $\mathbf{k} = (k_1, k_2, k_3)$, $k_i \in \mathbb{N}$, $||\mathbf{k}|| = k_1 + k_2 + k_3$, $\mathbf{k}! = k_1!k_2!k_3!$. A particle $\mathbf{x}_i$ and a cluster c are defined to be well-separated if the following multipole acceptance criterion (MAC) is satisfied,

$$\frac{r_c}{R} \leq \theta, \tag{18}$$

where $r_c = \max_{\mathbf{x}_j \in c} |\mathbf{x}_j - \mathbf{x}_c|$ is the cluster radius, $R = |\mathbf{x}_i - \mathbf{x}_c|$ is the particle-cluster distance, and θ is a user-specified parameter [4]. If the criterion is not satisfied, the code examines the children of the cluster recursively until the leaves of the tree are reached at which point direct summation is used. The Taylor coefficients are computed using recurrence relations [22].

The matrix-vector product amounts to evaluating the sums on the right side of Equations (9a)-(9b). The kernels K_{1-4} appearing there, defined in Equations (7a)-(7b), are linear combinations of the Coulomb potential G_0, the screened Coulomb potential G_κ, and their first and second normal derivatives. Terms involving the potentials can be evaluated as explained previously, but terms involving the normal derivatives require a slight modification, which resembles the simple idea that for a function expressed with Taylor expansion, its derivative can be conveniently obtained by taking the derivative of each terms of the expansion. The details are described in [13].

2.4 Parallelization

In order to solve the linear algebraic system from Equations (9a) and (9b) or from Equations (10)-(13), the matrix-vector product $\mathbf{Ax}$ needs to be computed at each iteration step. This is by nature a straight-forward parallelization. The majority of the computing time is taken by the following subroutines.
(1) Compute the source term in Equation (8).
(2) Perform matrix-vector product as in Equations. (9a) and (9b) on each GMRES iteration.
(3) Compute electrostatic solvation energy.
Among all of these subroutines, subroutine (2) is the most expensive one and is repeatedly computed on each GMRES iteration. We compute all of these routines in parallel.

Furthermore, the treecode algorithm as described in the previous section, loops around all particles. Each target particle interacts with all source particles in terms of clusters (far field) or particles (low field) in a pre-constructed hierarchical tree. Since there is no interaction between target particles, the parallelization is again straightforward. We implemented the MPI-based parallelization in HOBI-PB and TABI-PB. Detailed algorithms can be found in [11, 13]. In these distributed-memory implementation, the tree are built within memories associated with individual CPUs, which is convenient but subject to the memory capacity. It is expected the OpenMP based shared-memory architecture can bypass this limitation at the cost of relatively more complicated implementation. We will further investigate this application in the future.

2.5 GPU acceleration

GPU can be treated as a collection of a lot of mini-CPUs to perform simple but repeated work. GPU has the advantage of possessing many parallel treads and the disadvantage of limited memory. For example, a latest GPU card Tesla K40 has 2880 CUDA cores but only 12GB memory. Two facts make the boundary integral

PB solvers good candidates for GPU implementation. First, the resulting linear algebraic matrix is not stored thus the memory usage is small. For example, in our TABI-PB solver with treecode acceleration, solving the PB equation for a protein whose surface is discretized into 536,886 elements only uses 600 MB memory. In addition, in our GABI-PB solver, we only need to allocate $6N_c + 11N$ size-of-double device memory for a protein with N triangular surface elements and N_c partial charges. Second, as described in the previous section about parallelization, the target-sources interactions are repeatedly computed for N times and there is no interaction between targets. Thus the targets-sources interaction computation is straightforward in parallelization. We implemented the direct-sum based computation for the linear algebraic system from Equations (9a)-(9b) with CUDA on a GPU card (Nvidia Tesla M2070) and achieved 150X speedup compared with the implementation on a single CPU (Intel L5640 2.27 GHz). The implementation is rather straightforward as far as the copy of data between memories in CPU and GPU are correctly handled and the workload are properly distributed among CPU and GPU. Details of implementation can be found in our recent publication [12]. We are currently implementing treecode algorithm into GPU computing, which is a more challenging and rewarding work.

3 The Boundary Integral PB Solvers Package

In this section, we introduce the boundary integral PB solvers package published at the author's academic website (http://faculty.smu.edu/wgeng/research/bipb.html). All code are deposited at sourceforge as linked from the author's website. This package contains three boundary integral PB solvers. Using boundary integral PB solvers is a smart choice if the accuracy of electrostatic potentials and fields near or on the molecular surface are of the major concerns of one's research. Among these solvers, TABI-PB generally solves the PB equation with the best combination of efficiency and accuracy. HOBI-PB provides the most accurate electrostatic potential on the triangular molecular surface at the cost of increased CPU time and Memory use. GABI-PB performs extremely well for solving PB equation with less than 300,000 boundary elements as it utilizes the GPU-accelerations under the direct summation scheme (no treecode). Interested users can download these solvers and use them for academic purpose by following the New BSD License and citing the related publication.

Treecode-Accelerated Boundary Integral Poisson-Boltzmann (TABI-PB) Solver [13]

TABI-PB employs a well-conditioned boundary integral formulation for computing the electrostatic potential and its normal derivative on the molecular surface. The surface is triangulated and the integral equations are discretized by centroid collocation. The linear system is solved by GMRES iteration and the matrix-vector product is carried out by a Cartesian terraced which reduces the cost from $O(N^2)$ to $O(N \log N)$, where N is the number of faces in the triangulation. The TABI-PB solver can be applied to compute the electrostatic potential on molecular surface and solvation energy. The solver can be obtained with two different approaches.

One way is to download the binary version package (Windows, Linux, Mac). Note: For this option, testing proteins in forms of APBS's pqr file (sample) should be stored in the sub-directory with the name of "test_proteins", while MSMS of the corresponding operation system (Windows, Linux, Mac), and the input file (sample) should be in the same directory with the binary files. Users also need to change the property of the binary files (tabipb.exe and msms) to executable. More details can be found from the User's Guide. The other way is to download the sourcecode (written in Fortran 77/90/95) from sourceforge and more details can be found from the User's Guide.

Higher-Order Boundary Integral Poisson-Boltzmann (HOBI-PB) Solver [11]

HOBI-PB contains the parallel higher-order boundary integral method to solve the linear Poisson-Boltzmann (PB) equation. In our method, a well-posed boundary integral formulation is used to ensure the fast convergence of Krylov subspace linear solver such as GMRES. The molecular surfaces are first discretized with

flat triangles and then converted to curved triangles with the assistance of normal information at vertices. To maintain the desired accuracy, four-point Gauss-Radau quadratures are used on regular triangles and sixteen-point Gauss-Legendre quadratures together with regularization transformations are applied on singular triangles. We take advantage of the straightforward parallel feature of boundary integral formulation, and parallelize the schemes with the message passing interface (MPI) implementation.

Users can download the source code (written in Fortran 77/90/95 with Open MPI) from sourceforge. Note Users need to have Open MPI installed on ones' system.

GPU-Accelerated Boundary Integral Poisson-Boltzmann (GABI-PB) Solver [12]
GABI-PB presents a GPU-accelerated direct-sum boundary integral method to solve the linear Poisson-Boltzmann (PB) equation. In our method, a well-posed boundary integral formulation is used to ensure the fast convergence of Krylov subspace based linear algebraic solver such as the GMRES. The molecular surfaces are discretized with flat triangles and centroid collocation. To speed up our method, we take advantage of the parallel nature of the boundary integral formulation and parallelize the schemes within CUDA shared memory architecture on GPU. The schemes use only $11N + 6N_c$ size-of-double device memory for a biomolecule with N triangular surface elements and N_c partial charges.

Users can download the source code (written in CUDA C) from sourceforge. Note the Users need to have GPUs access and nvcc complier enabled.

4 Numerical Results

4.1 On a spherical cavity with analytical solutions

We first solve the linear Poisson-Boltzmann equation on a spherical cavity with radius 2Å and a centered change $1e_c$ submerged in water with zero ionic strength. The result here has been reported in [11] and we restated since this article describes a software package containing all related PB solvers. We report the electrostatic solvation energy E_{sol} and surface potential errors e_ϕ computed with all above-mentioned methods in Table 1. The results of APBS [2] and MIBPB are from reference [14].

We defined the relative $L-\infty$ errors of the surface potential as

$$e_\phi = \frac{\max_{i=1,\ldots,N} |\phi^{num}(x_i) - \phi^{exa}(x_i)|}{\max_{i=1,\ldots,N} |\phi^{exa}(x_i)|} \tag{19}$$

where N is the number of unknowns. Note N is the number of vertices for HOBI-PB, the number of triangular elements for TABI-PB, and the number of close-to-surface mesh points (irregular points) for APBS and MIBPB. The notation ϕ^{num} represents numerically solved surface potentials and ϕ^{exa} denotes the analytical solutions obtained by Kirkwood's spherical harmonic expansion [20]. The discretization of APBS and MIBPB are on the Cartesian grid with mesh size h. The discretization of HOBI-PB and TABI-PB are on the molecular surface with density d, number of vertices per Å^2.

From Table 1, we can see that APBS provides acceptable value in electrostatic solvation energy compared with the true value 81.98 kcal/mol since only the potential at the center of the spherical cavity is required to compute the electrostatic solvation energy. However, the surface potentials computed by APBS method show large errors. This is due to the approximation on interface conditions and singular charges of standard finite difference methods.

MIBPB uses a more sophisticated finite difference scheme. The rigorous treatment on interface conditions and singular charges significantly improves accuracy. The electrostatic solvation energy is nearly perfect even at coarse grid and the surface potential is very accurate with solid 2nd order convergence pattern relative to mesh-size h.

Table 1: Accuracy comparison of different PB solvers on a spherical cavity (radius=2Å , $q(0,0,0) = 1e_c$, $\varepsilon = 80$, $\kappa = 0$); h the mesh size; d the number of vertices per Å^2.

	APBS			MIBPB				HOBI-PB			TABI-PB		
h	E_{sol}	e_ϕ	ord.	E_{sol}	e_ϕ	ord.	d	E_{sol}	e_ϕ	ord.	E_{sol}	e_ϕ	ord.
1	−83.44	1.94e+0		−81.95	1.24e-2		5	−81.98	1.45e-4		−83.41	4.07e-4	
0.5	−85.85	1.31e+0	0.6	−81.98	1.91e-3	2.7	10	−81.98	5.26e-5	1.5	−83.13	2.55e-4	0.7
0.2	−82.58	5.76e-1	0.9	−81.98	3.87e-4	1.7	20	−81.98	1.52e-5	1.8	−82.82	1.82e-4	0.5
0.1	−82.27	2.94e-1	1.0	−81.98	1.07e-4	1.9	40	−81.98	6.13e-6	1.3	−82.60	1.27e-4	0.5
0.05	−82.03	1.49e-1	1.0	−81.98	2.31e-5	2.2	80	−81.98	1.85e-6	1.7	−82.43	8.58e-5	0.6

Table 2: Computing solvation energy with PB solvers: Columns 5-9 are electrostatic solvation energies in kcal/mol computed with different solvers and parameters; "d10" indicates 10 vertices per Å^2 as the density for the triangulation;"h.5" indicates mesh size of finite difference is 0.5Å.

ID	PDB	# of el.	# of at.	TABI (d10)	TABI (d20)	HOBI (d10)	MIB (h.5)	MIB (h.25)
1	1ajj	40496	519	−1140.6	−1137.3	−1130.9	−1137.2	−1139.9
2	2erl	43214	573	−959.6	−954.1	−963.8	−948.8	−948.2
3	1cbn	44367	648	−306.5	−304.9	−304.1	−303.8	−303.5
4	1vii	47070	596	−913.8	−907.2	−917.8	−901.2	−900.4
5	1fca	47461	729	−1215.4	−1207.0	−1224.6	−1200.1	−1199.0
6	1bbl	49071	576	−1000.2	−993.4	−1004.7	−986.8	−986.5
7	2pde	50518	667	−822.8	−820.5	−809.7	−820.9	−819.1
8	1sh1	51186	702	−758.8	−755.7	−755.1	−753.3	−752.2
9	1vjw	52536	828	−1255.6	−1246.8	−1268.8	−1237.9	−1237.0
10	1uxc	53602	809	−1152.9	−1145.2	−1158.6	−1138.7	−1137.2
11	1ptq	54260	795	−882.1	−877.7	−882.8	−873.1	−872.0
12	1bor	54629	832	−863.8	−858.1	−866.8	−853.7	−858.7
13	1fxd	54692	824	−3341.6	−3323.2	−3391.8	−3300.0	−3299.2
14	1r69	57646	997	−1098.1	−1092.1	−1097.5	−1089.5	−1086.2
15	1mbg	58473	903	−1366.4	−1357.9	−1378.6	−1346.1	−1347.5
16	1bpi	60600	898	−1319.5	−1311.0	−1326.7	−1301.9	−1298.1
17	1hpt	61164	858	−824.6	−818.9	−826.9	−814.3	−810.7
18	451c	79202	1216	−1034.0	−1028.2	−1030.7	−1024.6	−1023.1
19	1svr	81198	1435	−1727.3	−1717.2	−1723.3	−1711.2	−1709.0
20	1frd	81972	1478	−2882.9	−2866.5	−2901.7	−2851.9	−2848.5
21	1a2s	84527	1272	−1938.8	−1927.2	−1958.9	−1913.5	−1911.8
22	1neq	89457	1187	−1753.9	−1741.9	−1762.6	−1730.1	−1727.7
23	1a63	132134	2065	−2399.5	−2385.0	−2401.9	−2373.5	−2370.6
24	1a7m	147121	2809	−2178.6	−2167.8	−2178.0	−2155.5	−2153.3

TABI-PB gives the electrostatic solvation energy in the same accurate level as APBS but produces much more accurate surface potentials than APBS does. These surface potentials converge at the 0.5th order relative to density d.

HOBI-PB is obviously a more accurate method. It shows 1.5th order of convergence relative to density d as reflected from surface potential errors and its accuracy is even better than results from MIBPB.

These results demonstrate HOBI-PB is the most accurate PB solver among its peers. According to our knowledge, for this classic benchmark test, there is no other PB solvers can achieve the same level of accuracy as HOBI-PB does.

4.2 On 24 proteins

In this section, we provide the benchmark test of computing electrostatic solvation energy with different PB solvers. The electrostatic solvation energy is calculated as:

$$E_{\text{sol}} := \frac{1}{2}\sum_{k=1}^{N_c} q_k \phi_{\text{reac}}(\mathbf{y}_k) = \frac{1}{2}\sum_{k=1}^{N_c} q_k \int_\Gamma \left[K_1(\mathbf{y}_k, \mathbf{y}) \frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{y}_k, \mathbf{y}) \phi_1(\mathbf{y}) \right] dS_{\mathbf{y}}, \tag{20}$$

where $\phi_{\text{reac}}(\mathbf{x}) = \phi(\mathbf{x}) - S_1(\mathbf{x})$ is the reaction potential. Note the second equality in Equation (20) is for boundary integral PB solvers. For finite difference PB solver, the reaction potential $\phi_{\text{reac}}(\mathbf{x})$ need to be interpolated from potentials solved at the grid points.

We focus on accuracy rather than efficiency therefore only the HOBI-PB and TABI-PB results are provided. Note the GABI-PB result is very close to that of TABI-PB as we have shown in [13] that treecode truncation error is negligible compared with the discretization error. For comparison purpose, we also provided the results from MIBPB, a second order finite difference PB solver with rigorous treatment of jump conditions, complex geometry, and charge singularities [14]. The results are shown in Table 2. In this table, the first column is the index of 24 selected proteins in the ascending order of surface areas. The four digits protein data bank (PDB) ID of these proteins are listed in the second column, followed by the number of boundary elements (triangles) in the third column, and the number of atoms in the fourth column. Columns 5-9 are data of electrostatic solvation energies. These data are consistent in general with some differences, which can be better illustrated in Figure 2.

In order to better show the differences of results provided by different solvers/parameters, we subtract the computed solvation energies from their average among the five columns for the same protein. The average is not necessarily a better result but a reasonable way to shift the solvation energy to better show the differences. Note the biggest difference is about 90 kcal/mol for protein 13 and the solvation energy is about 3300 kcal/mol, which result in less than 3% error. Thus we conclude results computed by all these solvers are consistent. We will focus on differences as shown on the figure 2 with the following interesting patterns.
(1) The results from MIBPB at different mesh sizes (h=0.5 and h=0.25) only show slight difference. Thus this method has the best convergence attributing its rigorous treatment of complex geometry, interface conditions, and charge singularities. Another reason is that for MIBPB implementation the molecular surface is discretized at (d=10) and then converted to different finite difference mesh h. The triangular discretization errors are therefore not reflected in results at different h. We also observed the pattern that the result of MIBPB at h=0.25 (light blue with stars) is closer to the result from TABI-PB (green with squares and blue with empty circle) than result of MIBPB at h=0.5 (purple with solid circle) is, showing a convergence pattern from finite difference method toward boundary elements method.
(2) The results from TABI-PB at different density (d=10 vs d=20) show some differences but are sufficently close. The differences are major from triangular discretization errors, which are the dominant errors. We also observed the pattern that the result of TABI-PB at d=20 (green with square) is closer to the results from MIBPB (purple with solid circle and light blue with star) than result of TABI-PB at d=10 (blue with empty circle) is, showing a convergence pattern from boundary elements method toward finite difference method.
(3) HOBI-PB result at d=10 is out of expectation. We expect the curve (red with triangle) lie in between results from TABI-PB and HOBI-PB but it does not. This leaves a puzzle to be solved in the future and here is the explanation for now. In [11], for an example with analytical solution, we rigorously showed that HOBI-PB achieves the best accuracy among all PB solvers. However, when proteins are used, the unavailability of accurate normal condition and radial projection ruined the higher-order scheme more than the centroid collocation. Meanwhile, the normal vectors computed at the quadrature position introduces further approximation, which might leads the differences observed here as well.

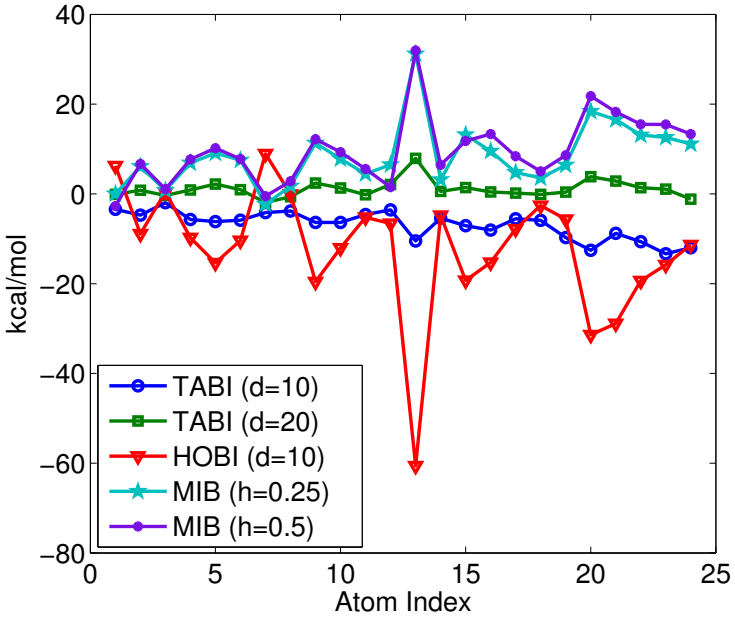

Figure 2: Solvation energy differences as subtracted from the average results of all solvers with different parameters

5 Discussion and Future Work

We introduced three boundary integral PB solvers in this paper. The differences among these solvers are at using or without using particular numerical schemes for the tradeoff between accuracy and efficiency. Among these solvers, TABI-PB generally solves the PB equation with the best combination of efficiency and accuracy. HOBI-PB provides the most accurate electrostatic potential on the triangular molecular surface at the cost of increased CPU time and Memory use. GABI-PB performs extremely well for solving PB equation with less than 300,000 boundary elements. The package published with this paper is only the first stage of work of our plan. It contains important components of our numerical algorithms such as boundary integral formulations, treecode, higher-order quadrature, parallelization, GPU etc. We will continue to improve the package in the following directions: 1) A Python wrapper for computing binding energy, pKa, and electrostatic surface potential at atomic and residual level, 2) The combination of Treecode and GPU in accelerating the PB Solver, 3) Nonlinear PB solver, 4) Boundary integral PB based electrostatic solvation forces for molecular dynamics simulations.

Acknowledgement: This work is partially supported by NSF grants DMS-0915057, DMS-1318898, and DMS-1418957.

Conflict of interest: Author state no conflict of interest.

Appendix 1: Units of PB equation

This section describes the units of PB equation under its different numerical implementation. We believe the introduction here will make the comparison between different PB solvers easier. For convenience, we restate

the PB equation as

$$-\varepsilon_1 \nabla^2 \phi(\mathbf{x}) = \sum_{k=1}^{N_c} q_k \delta(\mathbf{x} - \mathbf{y}_k), \quad \mathbf{x} \in \Omega_1, \tag{21}$$

$$-\varepsilon_2 \nabla^2 \phi(\mathbf{x}) + \bar{\kappa}^2 \phi(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega_2 \tag{22}$$

where ε_1 and ε_2 are relative permittivities inside and outside the molecule, and we also define $\varepsilon = \frac{\varepsilon_2}{\varepsilon_1}$ in deriving the integral formulation. Here the Debye-Hückel parameter κ is defined $\kappa^2 = \frac{8\pi N_A e_c^2}{1000\epsilon_2 k_B T} I_s$ where $N_A = 6.022045 \times 10^{23}$ is the Avagadro's number, $e_c = 4.8032424 \times 10^{-10}$ esu= $1.60217646 \times 10^{-19}$ C is the Fundamental charge, $k_B = 1.380662 \times 10^{-16}$ erg/K= 1.380662×10^{-23} J/K is Boltzmann's constant, T is the absolution temperature in K, $I_s = \frac{1}{2}\sum_{i=1}^{N_i} c_i z_i^2$ is the ionic strength with N_i the number of different types of ions, and c_i the molar concentration of ion type i with charge $q_i = z_i e_c$. In Equation (22), we use $\bar{\kappa}^2 = \kappa^2/\varepsilon_2$ to keep the permittivity constant ε_2 with the diffusion term to be consistent with Equation (21). In simulation, we have $\kappa^2 = 8.430325455 * I_s/\varepsilon_2$ for $T = 300$ K. r.f. Holst's thesis [16] for details.

The solution ϕ to Equations (21) and (22) is the electrostatic potential, whose units could be various depending on which units system is applied. Here we use an intuitive way to define its units. We assume the model is under molecular level, therefore the application of Åas distance units will be an convenient choice. Meanwhile, the charges carried by atoms are most conveniently measured by e_c, the charge carried by an electron. These two assumptions basically defines the units of the values in our calculation. If we observe the right-hand side of Equation (21), the units indicated will be $e_c/Å^3$ (The units of the δ function is $1/Å^3$, since its integral over the entire space gives constant 1). Consequently, we know that the units of the potential will be given by $e_c/Å$ if we use e_c and Å as our basic units. However, the most popular units of potential as the output of many current bimolecular software is given by kcal/mol/e_c. From the calculation shown in Holst's thesis, we need multiply $332.06364(4\pi)$ (adjusted to $332.0716(4\pi)$ to conform the calculation of CHARMM and APBS) with our output ϕ to give values of electrostatic potential with units kcal/mol·e_c. All our reports in the following will use this units for electrostatic potential and use kcal/mol for solvation energy.

We also mentioned briefly the units applied in APBS since many our results are compared with that from APBS. For the solvation energy, the output of APBS is in the units of kJ/mol, therefore it can be converted to $kcal/mol$ by dividing 4.184. APBS can also output potentials, whose units are of $k_B T/e_c$. With this information, we can derive the exchange rate between $k_B T/e_c$ and kcal/mol/e_c as $1 k_B T/e_c = \frac{300\ \text{K} \times 1.3806504 \times 10^{-26}\ \text{kJ}}{K e_c} = \frac{300\ \text{K} \times 1.3806504 \times 10^{-26}\ \text{kcal} \times 6.02214078 \times 10^{23}}{4.184\ \text{K} \times \text{mol} \times e_c}$=0.5961634386 kcal/mol/$e_c$.

Appendix 2: Analytical solution to PB equation for a spherical cavity

Kirkwood's dielectric sphere provides an excellent benchmark for testing Poisson Boltzmann (PB) solvers in terms of accuracy, speed of convergence and efficiency. In his paper [20], Kirkwood provided very detailed physical analysis and solutions to the PB equation of a spherical molecule in solvent. Here is a brief summary of Kirkwood's results used in the present work.

2-layer model

The model is a spherical molecule with dielectric boundary (radius a) and ionic boundary (radius b) in solvent. The dielectric constant is ϵ^- inside dielectric boundary and ϵ^+ in the exclusion layer and the solvent domain. The Laplace equation in sphere coordinates is given by:

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial \phi}{\partial r}\right) + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial \phi}{\partial \theta}\right) + \frac{1}{r^2 \sin^2\theta}\frac{\partial^2 \phi}{\partial \varphi^2} = 0, \tag{23}$$

where r is the radial coordinate, $\theta \in [0, \pi]$ is the polar angle from the z-axis and $\varphi \in [0, 2\pi)$ is the azimuthal angle in the $x - y$ plane with respect to the x-axis.

The solution of Eq. (23) has the following expansions

$$\phi = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (K_{mn} r^{-n-1} + J_{mn} r^n) P_n^m(\cos\theta) e^{im\varphi}, \tag{24}$$

where P_n^m are the associated Legendre function of the first kind, and K_{mn} and J_{mn} are constants to be determined by boundary or interface conditions.

– Inside the spherical molecule, where singular charges exist, we have

$$-\nabla^2 \phi^- = 4\pi \sum_{i=1}^{N_m} q_i \delta(\mathbf{r} - \mathbf{r}_i). \tag{25}$$

The potential ϕ^-, as the solution to Eq. (25), can be written as

$$\phi^- = \phi^\star + \hat{\phi}, \tag{26}$$

where $\phi^\star$ is the Green's function

$$\phi^\star = \sum_{i=1}^{N_m} \frac{q_i}{\epsilon^- |\mathbf{r} - \mathbf{r}_i|} \tag{27}$$

where N_m is number of charges in the spherical molecule, q_i and $\mathbf{r}_i$ are the charge and position of the ith atom, respectively. The Green's function can be further expanded as

$$\phi^\star = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{E_{mn}}{\epsilon^- r^{n+1}} P_n^m(\cos\theta) e^{im\varphi}, \tag{28}$$

where

$$E_{mn} = \frac{(n - |m|)!}{(n + |m|)!} \sum_{i=1}^{N_m} q_i r_i^n P_n^m(\cos\theta_i) e^{im\varphi_i}, \tag{29}$$

with $(r_i, \theta_i, \varphi_i)$ being the polar coordinates of the position of the ith charge. However, in practical calculations, this expansion converges slowly for small r. Therefore, it is more convenient to directly evaluate Eq. (27).

The non-singular term, $\hat{\phi}$, is given by

$$\hat{\phi} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (B_{mn} r^n) P_n^m(\cos\theta) e^{im\varphi}. \tag{30}$$

– In the spherical shell, i.e. the ion exclusion layer bounded by the spherical surfaces of radii a and b, the potential ϕ^e governed by the equation $\nabla^2 \phi^e = 0$ is given by:

$$\phi^e = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (C_{mn} r^{-n-1} + G_{mn} r^n) P_n^m(\cos\theta) e^{im\varphi}. \tag{31}$$

However, this part was not used in our model.

– Outside of the sphere of radius b, the potential ϕ^+ is governed by

$$\nabla^2 \phi^+ - \kappa^2 \phi^+ = 0, \tag{32}$$

where κ is the ionic strength defined previously, ϕ^+ is given as

$$\phi^+ = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (A_{mn} r^{-n-1}) e^{-\kappa r} K_n(\kappa r) P_n^m(\cos\theta) e^{im\varphi}, \tag{33}$$

where

$$K_n(x) = \sum_{s=0}^{n} \frac{2^s n!(2n - s)!}{s!(2n)!(n - s)!} x^s. \tag{34}$$

– All the coefficients A_{mn}, C_{mn}, G_{mn} and B_{nm} are to be determined by boundary conditions and interface condition: $\phi^- = \phi^e$, $\epsilon^- \frac{\partial \phi^-}{\partial r} = \epsilon^+ \frac{\partial \phi^e}{\partial r}$, $\phi^e = \phi^+$, $\frac{\partial \phi^e}{\partial r} = \frac{\partial \phi^+}{\partial r}$ and $\phi^+(\infty) = 0$.

one-layer model

In fact, we used a two-domain model in this work — the ion exclusion layer was not considered. Assuming weak ionic strength, the potential inside the molecule, ϕ^-, with E_{mn} defined as in Eq. (29), is given by:

$$\phi^- = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(\frac{E_{mn}}{\epsilon^- r^{n+1}} + B_{mn}r^n)P_n^m(\cos\theta)e^{im\varphi}, \tag{35}$$

and the potential outside the molecule, ϕ^+, is given by

$$\phi^+ = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{C_{mn}}{r^{n+1}}e^{-\kappa r}K_n(\kappa r)P_n^m(\cos\theta)e^{im\varphi}. \tag{36}$$

Coefficients B_{mn} and C_{mn} can be obtained via boundary and interface conditions: $\phi^- = \phi^+$ and $\epsilon^-\frac{\partial\phi^-}{\partial r} = \epsilon^+\frac{\partial\phi^+}{\partial r}$ at $r = a$.

To give analytical expression, we plug the above series into the two interface conditions to obtain

$$C_{nm}e^{-\kappa a}K_n(\kappa a) = \frac{E_{nm}}{\epsilon^-} + a^{2n+1}B_{nm} \tag{37}$$

$$\epsilon^+ C_{nm}e^{-\kappa a}[(n+1+\kappa a)K_n(\kappa a) - \kappa a K_n'(\kappa a)] = (n+1)E_{nm} - B_{nm}na^{2n+1}\epsilon^- \tag{38}$$

To cancel $C_{nm}e^{-\kappa a}$, we have

$$\epsilon^+(\frac{E_{nm}}{\epsilon^-} + a^{2n+1}B_{nm})\mathbb{X} = (n+1)E_{nm} - B_{nm}na^{2n+1}\epsilon^- \tag{39}$$

where

$$\mathbb{X} = \frac{[(n+1+\kappa a)K_n(\kappa a) - \kappa a K_n'(\kappa a)]}{K_n(\kappa a)} = (n+1) + \frac{(\kappa a)^2 K_{n-1}(\kappa a)}{(2n-1)K_n(\kappa a)}.$$

Finally, we have

$$B_{nm} = \frac{((n+1) - \frac{\epsilon^+}{\epsilon^-}\mathbb{X})}{a^{2n+1}(n\epsilon^- + \epsilon^+\mathbb{X})}E_{nm}$$

except when $n = 0$, $B_{00} = \frac{E_{00}}{a}[\frac{1}{(1+\kappa a)\epsilon^+} - \frac{1}{\epsilon^-}]$. And

$$C_{nm} = \frac{e^{\kappa a}[(1/\epsilon^-) + a^{2n+1}\mathbb{C}]}{K_n(\kappa a)}E_{nm}$$

where $\mathbb{C} = \frac{B_{nm}}{E_{nm}}$

Analytic solution in closed form

When the unit charge is located at the center of the sphere with radius R. The solution of PB equation can be analytically given in closed form without resorting to the multiple expansions. This solution, due to its simplicity in form, was used as the fundamental test case of any PB solver. In the one-layer model, the general solutions in each region for the PB equation are given as

$$\phi_1(r) = c_1 + \frac{1}{4\pi r} \quad \texttt{in } \Omega_1, \tag{40}$$

$$\phi_2(r) = c_2\frac{e^{-\kappa r}}{4\pi r} \quad \texttt{in } \Omega_2, \tag{41}$$

$$\tag{42}$$

The constants c_1 and c_2, which can be derived by applying the above equations in the the continuity conditions Eq. (3), have the form

$$c_1 = \frac{1}{4\pi\epsilon(1+\kappa R)R} - \frac{1}{4\pi R}, \quad c_2 = \frac{e^{\kappa R}}{\epsilon(1+\kappa R)}. \tag{43}$$

By plugging Eq. (43) into Eqs. (40) and Eqs. (41), we have the analytical solution

$$\phi_1(r) = \frac{1}{4\pi\epsilon(1+\kappa R)R} - \frac{1}{4\pi R} + \frac{1}{4\pi r} \text{ in } \Omega_1, \tag{44}$$

$$\phi_2(r) = \frac{e^{\kappa R}}{4\pi\epsilon(1+\kappa R)}\frac{e^{-\kappa r}}{r} \text{ in } \Omega_2. \tag{45}$$

And correspondingly the normal derivatives at interface are:

$$\frac{\partial\phi_1}{\partial r} = -\frac{1}{4\pi r^2} \text{ in } \Omega_1, \tag{46}$$

$$\frac{\partial\phi_1}{\partial r} = -\frac{e^{\kappa R}}{4\pi\epsilon(1+\kappa R)}\frac{e^{-\kappa r}(\kappa r+1)}{r^2} \text{ in } \Omega_2. \tag{47}$$

When no mobile ion are present in the solvent, i.e. $\kappa = 0$, the above forms are reduced to the solution of the Poisson equation.

$$\phi_1(r) = (\frac{1}{\epsilon} - 1)\frac{1}{4\pi R} + \frac{1}{4\pi r} \text{ in } \Omega_1, \tag{48}$$

$$\phi_2(r) = \frac{1}{4\pi r\epsilon} \text{ in } \Omega_2. \tag{49}$$

And correspondingly the normal derivatives at interface are:

$$\frac{\partial\phi_1}{\partial r} = -\frac{1}{4\pi r^2} \text{ in } \Omega_1, \tag{50}$$

$$\frac{\partial\phi_2}{\partial r} = -\frac{1}{4\pi\epsilon r^2} \text{ in } \Omega_2. \tag{51}$$

References

[1] Baker, N. A. (2005). Improving implicit solvent simulations: a Poisson-centric view. *Current Opinion in Structural Biology*, 15(2):137–43.

[2] Baker, N. A., Sept, D., Holst, M. J., and Mccammon, J. A. (2001a). The adaptive multilevel finite element solution of the Poisson-Boltzmann equation on massively parallel computers. *IBM Journal of Research and Development*, 45(3-4):427–438.

[3] Baker, N. A., Sept, D., Joseph, S., Holst, M. J., and McCammon, J. A. (2001b). Electrostatics of nanosystems: Application to microtubules and the ribosome. *Proceedings of the National Academy of Sciences of the United States of America*, 98(18):10037–10041.

[4] Barnes, J. and Hut, P. (1986). A hierarchical o(n log n) force-calculation algorithm. *Nature*, 324(6096):446–449.

[5] Bordner, A. J. and Huber, G. A. (2003). Boundary element solution of the linear Poisson-Boltzmann equation and a multipole method for the rapid calculation of forces on macromolecules in solution. *Journal of Computational Chemistry*, 24(3):353–367.

[6] Boschitsch, A. H. and Fenley, M. O. (2004). Hybrid boundary element and finite difference method for solving the nonlinear Poisson-Boltzmann equation. *Journal of Computational Chemistry*, 25(7):935–955.

[7] Boschitsch, A. H., Fenley, M. O., and Zhou, H.-X. (2002). Fast boundary element method for the linear Poisson-Boltzmann equation. *The Journal of Physical Chemistry B*, 106(10):2741–2754.

[8] Connolly, M. L. (1985). Depth buffer algorithms for molecular modeling. *J. Mol. Graphics*, 3:19–24.

[9] Cortis, C. M. and Friesner, R. A. (1997). Numerical solution of the Poisson-Boltzmann equation using tetrahedral finite-element meshes. *Journal of Computational Chemistry*, 18:1591–1608.

[10] Fogolari, F., Brigo, A., and Molinari, H. (2002). The Poisson-Boltzmann equation for biomolecular electrostatics: a tool for structural biology. *Journal of Molecular Recognition*, 15(6):377–92.

[11] Geng, W. (2013). Parallel higher-order boundary integral electrostatics computation on molecular surfaces with curved triangulation. *Journal of Computational Physics*, 241(0):253 – 265.

[12] Geng, W. and Jacob, F. (2013). A GPU-accelerated direct-sum boundary integral Poisson-Boltzmann solver . *Computer Physics Communications*, 184(6):1490 – 1496.

[13] Geng, W. and Krasny, R. (2013). A treecode-accelerated boundary integral Poisson-Boltzmann solver for electrostatics of solvated biomolecules. *Journal of Computational Physics*, 247(0):62 – 78.

[14] Geng, W., Yu, S., and Wei, G. W. (2007). Treatment of charge singularities in implicit solvent models. *Journal of Chemical Physics*, 127:114106.

[15] Holst, M. and Saied, F. (1993). Multigrid solution of the Poisson-Boltzmann equation. *Journal of Computational Chemistry*, 14(1):105–13.

[16] Holst, M. J. (1994). *The Poisson-Boltzmann Equation: Analysis and Multilevel Numerical Solution*. PhD thesis, UIUC.

[17] Honig, B. and Nicholls, A. (1995). Classical electrostatics in biology and chemistry. *Science*, 268(5214):1144–9.

[18] Im, W., Beglov, D., and Roux, B. (1998). Continuum solvation model: electrostatic forces from numerical solutions to the Poisson-Boltzmann equation. *Computer Physics Communications*, 111(1-3):59–75.

[19] Juffer, A., E., B., van Keulen, B., van der Ploeg, A., and Berendsen, H. (1991). The electric potential of a macromolecule in a solvent: a fundamental approach. *J. Comput. Phys.*, 97:144–171.

[20] Kirkwood, J. G. (1934). Theory of solution of molecules containing widely separated charges with special application to zwitterions. *J. Comput. Phys.*, 7:351 – 361.

[21] Lee, B. and Richards, F. M. (1971). The interpretation of protein structures: estimation of static accessibility. *J Mol Biol*, 55(3):379–400.

[22] Li, P., Johnston, H., and Krasny, R. (2009). A Cartesian treecode for screened Coulomb interactions. *J. Comput. Phys.*, 228(10):3858–3868.

[23] Liang, J. and Subranmaniam, S. (1997). Computation of molecular electrostatics with boundary element methods. *Biophys. J.*, 73:1830–1841.

[24] Lu, B., Cheng, X., Huang, J., and McCammon, J. A. (2006). Order N algorithm for computation of electrostatic interactions in biomolecular systems. *Proceedings of the National Academy of Sciences*, 103(51):19314–19319.

[25] Lu, B. Z., Zhou, Y. C., Holst, M. J., and McCammon, J. A. (2008). Recent progress in numerical methods for the Poisson-Boltzmann equation in biophysical applications. *Communications in Computational Physics*, 3(5):973–1009.

[26] Lu, Q. and Luo, R. (2003). A Poisson-Boltzmann dynamics method with nonperiodic boundary condition. *Journal of Chemical Physics*, 119(21):11035–11047.

[27] Luo, R., David, L., and Gilson, M. K. (2002). Accelerated Poisson-Boltzmann calculations for static and dynamic systems. *Journal of Computational Chemistry*, 23(13):1244–53.

[28] Rocchia, W., Alexov, E., and Honig, B. (2001). Extending the applicability of the nonlinear Poisson-Boltzmann equation: Multiple dielectric constants and multivalent ions. *J. Phys. Chem.*, 105:6507–6514.

[29] Saad, Y. and Schultz, M. (1986). GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869.

[30] Sanner, M. F., Olson, A. J., and Spehner, J. C. (1996). Reduced surface: An efficient way to compute molecular surfaces. *Biopolymers*, 38:305–320.

[31] Vorobjev, Y. N. and Scheraga, H. A. (1997). A fast adaptive multigrid boundary element method for macromolecular electrostatic computations in a solvent. *Journal of Computational Chemistry*, 18(4):569–583.

[32] Zauhar, R. J. and Morgan, R. S. (1985). A new method for computing the macromolecular electric potential. *Journal of Molecular Biology*, 186(4):815–20.

[33] Zauhar, R. J. and Morgan, R. S. (1988). The rigorous computation of the molecular electric potential. *Journal of Computational Chemistry*, 9(2):171–187.