A NEW LINEAR POISSON-BOLTZMANN EQUATION AND FINITE ELEMENT SOLVER BY SOLUTION DECOMPOSITION APPROACH [*]

JIAO LI [†] AND DEXUAN XIE [‡]

Abstract. The linear Poisson-Boltzmann equation (LPBE) is one well-known implicit solvent continuum model for computing the electrostatic potential of biomolecules in ionic solvent. To overcome its singular difficulty caused by Dirac delta distributions of point charges and to further improve its solution accuracy, we develop a new scheme for solving the current LPBE model, a new LPBE model, and a new LPBE finite element program package based on our previously proposed PBE solution decomposition. Numerical tests on biomolecules and a nonlinear Born ball model with an analytical solution validate the new LPBE solution decomposition schemes, demonstrate the effectiveness and efficiency of the new program package, and confirm that the new LPBE model can significantly improve the solution accuracy of the current LPBE model.

Key words. Poisson-Boltzmann equation, implicit solvent, biomolecular electrostatics, finite element method. **subject classifications.** 92-08,65K10, 65N30

1. Introduction

The linear Poisson-Boltzmann equation (LPBE) is one well-known continuum model used for calculating the electrostatic potential energy of biomolecules (protein or nucleic acids) in ionic solvent [4, 5, 8, 9]. However, LPBE is difficult to be solved numerically due to its solution singularity caused by Dirac delta distributions of point charges. To overcome such a difficulty, three different solution decomposition schemes were developed so far [3, 10, 11, 13]. Recently, the ones from [13] and [3] were well revisited in [2] and [7], respectively, for adaptive finite element solutions of the nonlinear PBE (NPBE) model. In this paper, we only consider the one from [10, 11] since this decomposition naturally splits the NPBE solution u into three parts, G, Ψ, and $\tilde{\Phi}$ (see (3.1)), within both the solute domain D_p and the solvent domain D_s, according to electrostatic contributions from the biomolecular charges, the boundary and interface conditions, and the ionic solvent charges, respectively. Note that Ψ is defined by a linear interface problems, which can be formulated as a weak form without involving any surface integral term on a molecular surface Γ, $\tilde{\Phi}$ is defined by a nonlinear interface problem with continuous interface conditions and a homogenous boundary condition, and both Ψ and $\tilde{\Phi}$ can be calculated numerically on the same finite element function space. Hence, our solution decomposition can lead to efficient numerical algorithms for solving PBE. As an application of this novel solution decomposition, in this paper, we develop a new LPBE model with a higher accuracy of solution and a broader application range than the current LPBE model. Furthermore, we develop new finite element solution splitting schemes for solving the current and new LPBE models, and program them as a new effective LPBE finite element program package.

Our new LPBE model was motivated partially from the following observations:

[*]This work was partially supported by the National Science Foundation, USA, through grant DMS-1226259 and the UWM Research Growth Initiative. Published on *Communications in Mathematical Sciences*, Volume 13, No. 2, pp. 315-325, 2015.

[†]School of Mathematics and Computer Science, Changsha University of Science and Technology, Changsha, Hunan, 410004, P.R. China. This author was partially supported by China Scholarship Council and the UWM Research Growth Initiative during her graduate study at UWM.

[‡]Department of Mathematical Sciences, University of Wisconsin-Milwaukee, Milwaukee, Wisconsin, USA, 53201-0413. Corresponding author, (dxie@uwm.edu, www.uwm.edu/˜dxie).

Traditionally, LPBE is defined under the assumption

$$|u(\mathbf{r})| < 1 \qquad \text{a.e. in } D_s. \tag{1.1}$$

However, we observed that the above assumption often fails to be satisfied in bimolecular calculation. We also observed that $|\tilde{\Phi}|$ from our solution decomposition may be less than $|u|$ in D_s. Linearizing the nonlinear term of NPBE with respect to $\tilde{\Phi}$ may yield a more accurate LPBE model, but this is not feasible in cases where $|\tilde{\Phi}|$ is greater than one. Hence, in general, we need to select a function, w, satisfying the condition

$$|\tilde{\Phi}(\mathbf{r}) - w(\mathbf{r})| < 1 \qquad \text{a.e. in } D_s, \tag{1.2}$$

in order to obtain a LPBE model with a sufficient accuracy. Taking the above condition as an assumption leads to our new LPBE model.

Selecting a function for w is a research issue onto itself. In this paper, we consider two possibilities. One is to set $w = 0$, which works when $|\tilde{\Phi}|$ is less than 1 in D_s at most mesh points. The other one is to generate w from an available LPBE model (e.g., the current LPBE model or our new LPBE model using $w = 0$).

Our LPBE finite element program package was written in Python, C++, and Fortran based on the FEniCS project [1]. To generate a finite element mesh for a given protein molecule, we adapted a molecular surface and volumetric mesh generation program package, GAMer [12], as one part of our program package. In order to speed up the calculation, we wrote Fortran subroutines and C++ functions for inputting and outputting data and for computing G (see (3.4)), its gradient ∇G (see (3.5)), and a multiple Debye-Hückel boundary value function g (see (2.5)). We converted them into Python modules as a part of our program package. Because of the FEniCS project [1], the new and the current LPBE models can be approximated by linear, quadratic, and cubic finite element methods and then solved by a linear iterative solver from the PETSc library (*http://www.mcs.anl.gov/petsc*). The details on the formulation of finite element equations and the iterative solution processes are omitted due to page limitation.

We performed numerical experiments on a nonlinear Born ball model with analytical solution (see (5.1)) and six biomolecules including four proteins, one protein-DNA complex, and one DNA. Two criteria were proposed (See Criterion 1 and 2) and used in accuracy comparisons for finite element solutions of two different LPBE models. Numerical results (see Figures 5.1 and 5.2 and Tables 5.1 to 5.3) validated the new LPBE solution decomposition schemes and our program package, demonstrated the effectiveness and efficiency of our program package, and confirmed that the new LPBE model can significantly improve the solution accuracy of the current LPBE model.

The remaining sections of this paper are organized as follows: In Section 2, we briefly review the NPBE and LPBE models. In Section 3, we describe the new scheme for solving the LPBE model. In Section 4, we present the new LPBE model. The program package and numerical results are reported in Section 5.

2. The NPBE and LPBE models

Let Ω be a sufficiently large bounded domain of $\mathbb{R}^3$ satisfying $\Omega = D_p \cup D_s \cup \Gamma$ with Γ denoting the interface between the solute domain D_p and solvent domain D_s. As an implicit solvent approach, both D_p and D_s are treated as continuum media with different dielectric constants ϵ_p and ϵ_s, respectively. D_p contains a three dimensional molecular structure of a biomolecule with n_p atoms while D_s contains sodium (Na^+) and chloride (Cl^-) ions. The position $\mathbf{r}_j$ and charge number z_j of the j-th atom are known. In this paper, we set $\epsilon_p = 2.0$ and $\epsilon_s = 80.0$.

In the above notation, the dimensionless NPBE model is defined by

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha \sum_{j=1}^{n_p} z_j \delta_{\mathbf{r}_j}, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta u(\mathbf{r}) + \bar{\kappa}^2 \sinh(u(\mathbf{r})) = 0, & \mathbf{r} \in D_s, \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), \quad \epsilon_s \frac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ u(\mathbf{s}) = g, & \mathbf{s} \in \partial\Omega, \end{cases} \tag{2.1}$$

where u is the dimensionless electrostatic potential, g is a given function, $\partial\Omega$ denotes the boundary of Ω, $\delta_{\mathbf{r}_j}$ is a Dirac delta distribution at point $\mathbf{r}_j$, $\mathbf{n}(\mathbf{s})$ is the unit outward normal vector of D_p, and α and $\bar{\kappa}$ are defined by

$$\alpha = 4\pi e_c^2/(k_B T) \quad \text{and} \quad \bar{\kappa} = \sqrt{8\pi e_c^2 N_A I_s/(1000 k_B T)}. \tag{2.2}$$

Here e_c is the elementary charge, k_B is the Boltzmann constant, T is the absolute temperature, I_s is the ionic strength, and N_A is the Avogadro constant, which estimates the number of ions per mole as $6.02214129 \times 10^{23}$. In electrostatic units (esu), e_c and k_B are estimated by

$$e_c = 4.80320425 \times 10^{-10} esu, \qquad k_B = 1.3806488 \times 10^{-16} erg\ /\ K,$$

where K denotes Kelvin, and $1 erg = 1 esu^2$ per centimeter (cm).

In biomolecular calculation, the domain Ω is measured in angstroms ($\AA$). Thus, the length unit should be converted from centimeter (cm) to angstrom ($1\ cm = 10^8$Å). For $T = 298.15K$ and $I_s = 100\ mM$, which are often used in numerical tests, α and $\bar{\kappa}^2$ can be found to have the following values

$$\alpha = 7042.939216246565, \qquad \bar{\kappa}^2 = 0.8482715011423737. \tag{2.3}$$

Under condition (1.1), (2.1) can be approximated as the LPBE model

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha \sum_{j=1}^{n_p} z_j \delta_{\mathbf{r}_j}, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta u(\mathbf{r}) + \bar{\kappa}^2 u(\mathbf{r}) = 0, & \mathbf{r} \in D_s, \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), \quad \epsilon_s \frac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ u(\mathbf{s}) = g, & \mathbf{s} \in \partial\Omega. \end{cases} \tag{2.4}$$

In calculation, g is often set as the multiple Debye-Hückel boundary value function

$$g = \frac{e_c^2}{k_B T \epsilon_s} \sum_{j=1}^{n_p} \frac{e^{-\kappa|\mathbf{r}-\mathbf{r}_j|}}{|\mathbf{r}-\mathbf{r}_j|} z_j, \tag{2.5}$$

where $\kappa = \bar{\kappa}/\sqrt{\epsilon_s}$, which is called the Debye-Hückel parameter [6].

3. A new LPBE solution decomposition scheme

From [10, 11] it is known that a solution u of (2.1) can be split in the form

$$u = \tilde{\Phi} + \Psi + G, \tag{3.1}$$

where $\tilde{\Phi}$ is a solution of the nonlinear boundary value problem

$$\begin{cases} \Delta\tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta\tilde{\Phi}(\mathbf{r}) + \bar{\kappa}^2 \sinh(\tilde{\Phi} + \Psi + G) = 0, & \mathbf{r} \in D_s, \\ \tilde{\Phi}(\mathbf{s}^+) = \tilde{\Phi}(\mathbf{s}^-), \quad \epsilon_s \frac{\partial\tilde{\Phi}(\mathbf{s}^+)}{\partial\mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial\tilde{\Phi}(\mathbf{s}^-)}{\partial\mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \tilde{\Phi}(\mathbf{s}) = 0, & \mathbf{s} \in \partial\Omega, \end{cases} \tag{3.2}$$

Ψ is a solution of the linear boundary value problem

$$\begin{cases} \Delta\Psi(\mathbf{r}) = 0, & \mathbf{r} \in D_p \cup D_s, \\ \Psi(\mathbf{s}^+) = \Psi(\mathbf{s}^-), \quad \epsilon_s \frac{\partial\Psi(\mathbf{s}^+)}{\partial\mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial\Psi(\mathbf{s}^-)}{\partial\mathbf{n}(\mathbf{s})} + (\epsilon_p - \epsilon_s)\frac{\partial G(\mathbf{s})}{\partial\mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \Psi(\mathbf{s}) = g - G(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{3.3}$$

and G is given by

$$G(\mathbf{r}) = \frac{e_c^2}{k_B T \epsilon_p} \sum_{j=1}^{n_p} \frac{z_j}{|\mathbf{r} - \mathbf{r}_j|}. \tag{3.4}$$

Here $\frac{\partial G(\mathbf{s})}{\partial\mathbf{n}(\mathbf{s})} = \nabla G \cdot \mathbf{n}$ with ∇G being given by

$$\nabla G(\mathbf{r}) = -\frac{e_c^2}{k_B T \epsilon_p} \sum_{j=1}^{n_p} z_j \frac{\mathbf{r} - \mathbf{r}_j}{|\mathbf{r} - \mathbf{r}_j|^3}. \tag{3.5}$$

Clearly, under the condition (1.1), (3.2) can be linearilized as

$$\begin{cases} \Delta\tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta\tilde{\Phi}(\mathbf{r}) + \bar{\kappa}^2 \tilde{\Phi} = -\bar{\kappa}^2(\Psi + G), & \mathbf{r} \in D_s, \\ \tilde{\Phi}(\mathbf{s}^+) = \tilde{\Phi}(\mathbf{s}^-), \quad \epsilon_s \frac{\partial\tilde{\Phi}(\mathbf{s}^+)}{\partial\mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial\tilde{\Phi}(\mathbf{s}^-)}{\partial\mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \tilde{\Phi}(\mathbf{s}) = 0, & \mathbf{s} \in \partial\Omega. \end{cases} \tag{3.6}$$

With a solution $\tilde{\Phi}_{l0}$ of (3.6), we construct a function, $u_{l0} = \tilde{\Phi}_{l0} + \Psi + G$, and verify that u_{l0} is a solution of the LPBE model (2.4). In this way, a new LPBE solution decomposition scheme is obtained in Algorithm 1.

Algorithm 1 (A new LPBE solution decomposition scheme) *Let u_{l0} be a solution of the LPBE model* (2.4). *It can be found in four steps:*

1. *Calculate G and its gradient ∇G by* (3.4).
2. *Find a solution Ψ of* (3.3).
3. *Find a solution $\tilde{\Phi}_{l0}$ of* (3.6).
4. *Construct u_{l0} by the splitting formula $u_{l0} = \tilde{\Phi}_{l0} + \Psi + G$.*

In this paper, we only consider a finite element method for solving (3.3) and (3.6) numerically since the interface conditions of (3.3) and (3.6) can be much more easily treated by the finite element method than by a finite difference method.

4. A new LPBE model

When a solution $\tilde{\Phi}$ of (3.2) satisfies

$$|\tilde{\Phi}(\mathbf{r})| < 1 \qquad \text{a.e. in } D_s, \tag{4.1}$$

we have $\sinh(\tilde{\Phi}) \approx \tilde{\Phi}$, and $\cosh(\tilde{\Phi}) \approx 1$. Thus, by (3.1), $\sinh(u)$ is linearized as

$$\begin{aligned} \sinh(u) = \sinh(\tilde{\Phi} + \Psi + G) &= \sinh(\Psi + G)\cosh(\tilde{\Phi}) + \cosh(\Psi + G)\sinh(\tilde{\Phi}) \\ &\approx \sinh(\Psi + G) + \tilde{\Phi}\cosh(\Psi + G), \end{aligned}$$

resulting in a new linearization of (3.2) as follows:

$$\begin{cases} \Delta\tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r}\in D_p,\\ -\epsilon_s\Delta\tilde{\Phi}(\mathbf{r}) + \bar{\kappa}^2\cosh(\Psi+G)\tilde{\Phi}(\mathbf{r}) = -\bar{\kappa}^2\sinh(\Psi+G), & \mathbf{r}\in D_s,\\ \tilde{\Phi}(\mathbf{s}^+) = \tilde{\Phi}(\mathbf{s}^-), \quad \epsilon_s\frac{\partial\tilde{\Phi}(\mathbf{s}^+)}{\partial\mathbf{n}(\mathbf{s})} = \epsilon_p\frac{\partial\tilde{\Phi}(\mathbf{s}^-)}{\partial\mathbf{n}(\mathbf{s})}, & \mathbf{s}\in\Gamma,\\ \tilde{\Phi}(\mathbf{s}) = 0, & \mathbf{s}\in\partial\Omega. \end{cases} \tag{4.2}$$

However, condition (4.1) may still fail in biomolecular calculation. In such a case, we select a function, w, satisfying (1.2), and linearize $\sinh(u)$ as shown below:

$$\begin{aligned} \sinh(u) &= \sinh(\tilde{\Phi} - w + w + \Psi + G)\\ &= \sinh(\tilde{\Phi} - w)\cosh(w+\Psi+G) + \cosh(\tilde{\Phi} - w)\sinh(w+\Psi+G)\\ &\approx \tilde{\Phi}\cosh(w+\Psi+G) - w\cosh(w+\Psi+G) + \sinh(w+\Psi+G). \end{aligned}$$

where $\sinh(\tilde{\Phi} - w) \approx \tilde{\Phi} - w$ and $\cosh(\tilde{\Phi} - w) \approx 1$ have been used. Consequently, we obtain another new linearization of (3.2) in the form

$$\begin{cases} \Delta\tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r}\in D_p,\\ -\epsilon_s\Delta\tilde{\Phi}(\mathbf{r}) + \bar{\kappa}^2\cosh(w+\Psi+G)\tilde{\Phi}(\mathbf{r}) = f, & \mathbf{r}\in D_s,\\ \tilde{\Phi}(\mathbf{s}^+) = \tilde{\Phi}(\mathbf{s}^-), \quad \epsilon_s\frac{\partial\tilde{\Phi}(\mathbf{s}^+)}{\partial\mathbf{n}(\mathbf{s})} = \epsilon_p\frac{\partial\tilde{\Phi}(\mathbf{s}^-)}{\partial\mathbf{n}(\mathbf{s})}, & \mathbf{s}\in\Gamma,\\ \tilde{\Phi}(\mathbf{s}) = 0, & \mathbf{s}\in\partial\Omega, \end{cases} \tag{4.3}$$

where f is defined by $f = \bar{\kappa}^2[w\cosh(w+\Psi+G) - \sinh(w+\Psi+G)]$. From (4.3) it leads to the new LPBE model, which we define in Algorithm 2 for clarity.

Algorithm 2. (Definition of the new LPBE model) *For a given function w satisfying* (1.2)*, the new LPBE model is defined in four steps.*

1. *Calculate G and its gradient ∇G by* (3.4)*.*
2. *Find a solution Ψ of* (3.3)*.*
3. *Find a solution $\tilde{\Phi}_l$ of* (4.3)*.*
4. *Define a solution u_l of the new LPBE model as $u_l = \tilde{\Phi}_l + \Psi + G$.*

In this paper, two choices of w are considered. One is to simply set $w = 0$, which gives the case of (4.2). The other one is to set $w = \tilde{\Phi}_{l0}$ to gain a more accurate LPBE solution than the current LPBE model.

Clearly, different selections of w may lead to different LPBE solutions. To judge one solution to be "better" than another one, we introduce Criterion 1.

Criterion 1 (Solution accuracy judgment) *Let u_1 and u_2 be two LPBE solutions. Then u_1 is said to have a higher accuracy (or is better) than u_2 provided that*

$$\|u_1 - u\|_{L_2(\Omega)} < \|u_2 - u\|_{L_2(\Omega)}, \tag{4.4}$$

where u denotes a corresponding solution u of NPBE, and $\|\cdot\|_{L_2(\Omega)}$ denotes the L_2 norm, which is defined by $\|v\|_{L_2(\Omega)} = (\int_\Omega |v|^2 d\mathbf{r})^{\frac{1}{2}}$ for all $v\in L_2(\Omega)$.

Another way to judge a LPBE solution accuracy is to compare its solvation free energy with that of a corresponding NPBE solution. For a solution v defined in the form $v = \tilde{\Phi} + \Psi + G$, the solvation free energy can be simply calculated by

$$F_{sol}(v) = \frac{1}{2}\sum_{j=1}^{n_p} z_j(\tilde{\Phi}(\mathbf{r}_j) + \Psi(\mathbf{r}_j)). \tag{4.5}$$

Here we ignore any physical unit for simplicity. This consideration leads to Criterion 2:

Criterion 2 (Solution accuracy judgment) *Let u_1 and u_2 denote two LPBE solutions. Then u_1 is said to have a higher accuracy than u_2 provided that*

$$|F_{sol}(u_1) - F_{sol}(u)| < |F_{sol}(u_2) - F_{sol}(u)|, \tag{4.6}$$

where u denotes a corresponding solution of NPBE.

When exact solutions of LPBE and NPBE are unavailable, the functions u_1, u_2 and u in Criteria 1 and 2 can be simply set as the finite element solutions defined on the same finite element function space, like what we did in Section 5. Such a selection is reasonable if we are interested in an accuracy of a numerical LPBE solution as an approximation of the corresponding NPBE numerical solution.

5. Program package and numerical results

We programmed Algorithms 1 and 2 as a finite element program package in Python based on the library DOLFIN from the FEniCS project [1]. We adapted the mesh generation program package GAMer [12] and then converted it as a Python module by the software development tool SWIG (*http://www.swig.org*). By calling this Python module, a finite element mesh can be generated directly within our program package for each input biomolecule represented in a PQR file. To improve the computer performance of our program package, we wrote Fortran subroutines for computing the values of G, ∇G, and g at mesh nodes, and then converted them into Python modules by the Fortran to Python interface generator F2PY (*http://cens.ioc.ee/projects/f2py2e/*). All the involved boundary value problems were approximated by a finite element method, and then solved by the conjugate gradient method (CG) with the ILU preconditioner from the PETSc library. Another linear iterative solver GMRES was used when CG failed, which occurred occasionally in solving for Ψ due to too large jump discontinuities on some interface mesh points.

In all the numerical tests, we set the relative and absolute tolerances of CG and GMRES as 10^{-10}, and used the values of α and $\bar{\kappa}$ given in (2.3). All the tests were made on one 2.8 GHz Intel Core i7 processor of a MacBook Pro with 4GB memory.

5.1. Test cases of a nonlinear Born ball model

To verify our solution decomposition schemes defined in Algorithms 1 and 2 and our program package, we constructed the nonlinear Born ball model

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha z \delta, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta u(\mathbf{r}) + \bar{\kappa}^2 \sinh(u(\mathbf{r})) = \rho, & \mathbf{r} \in D_s, \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), \quad \epsilon_s \frac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}} = \epsilon_p \frac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}}, & \mathbf{s} \in \Gamma, \\ u(\mathbf{s}) = \frac{\alpha z}{4\pi \epsilon_s |\mathbf{s}|}, & \mathbf{s} \in \partial\Omega, \end{cases} \tag{5.1}$$

where $\Omega = \{\mathbf{r} : |\mathbf{r}| < R_s\}$, $D_p = \{\mathbf{r} : |\mathbf{r}| < R_p\}$, $D_s = \{\mathbf{r} : R_p < |\mathbf{r}| < R_s\}$, $\Gamma = \{\mathbf{r} : |\mathbf{r}| = R_p\}$, z is a charge number, $\rho = \bar{\kappa}^2 \sinh(\frac{\alpha z}{4\pi\epsilon_s |\mathbf{r}|})$, and δ is the Dirac delta distribution at the origin. The analytical solution of this Born ball model is given by

$$u(\mathbf{r}) = \begin{cases} \frac{\alpha z}{4\pi R_p}\left(\frac{1}{\epsilon_s} - \frac{1}{\epsilon_p}\right) + \frac{\alpha z}{4\pi \epsilon_p |\mathbf{r}|}, & \mathbf{r} \in D_p, \\ \frac{\alpha z}{4\pi\epsilon_s |\mathbf{r}|}, & \mathbf{r} \in D_s. \end{cases} \tag{5.2}$$

The only change we made to our program package to make it work in solving the Born ball model was to add the function ρ of (5.1) to the second equation of (3.2).

We solved the new and current LPBE models for the above Born ball model with $w = 0$, $R_p = 1$, $R_s = 10$, $I_s = 100mM$, and $z = \pm 1, \pm 2, \pm 3$, ± 4 by linear, quadratic, cubic

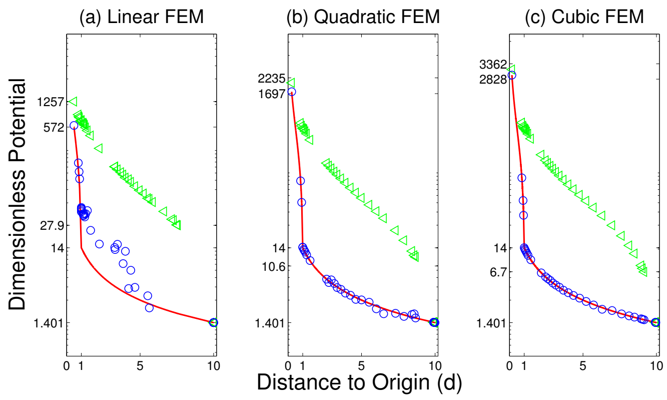

FIG. 5.1. *A comparison of the finite element solutions of the new LPBE (blue circles) and current LPBE (green triangles) models with the analytic solution (red line) of the nonlinear Born ball model* (5.1) *with* $z=2, R_p=1$, *and* $R_s=10$.

finite element methods based on a tetrahedral mesh with 2955 vertices. The relative error of the LPBE solutions u_L with respect to the analytical solution u was calculated by the formula $\|u_L-u\|_{L_2(\Omega)}/\|u\|_{L_2(\Omega)}$, and reported in Table 5.1. A comparison of the finite element solutions with the analytical solution is displayed in Figure 5.1 as a function of distance d. Here $d=(x^2+y^2+z^2)^{\frac{1}{2}}$ for a mesh node (x,y,z). For clarity, we only displayed a few number of mesh nodes in these plots.

TABLE 5.1. *Relative errors of the new and the current LPBE models for the nonlinear Born ball model* (5.1) *with* $R_p=1$ *and* $R_s=10$ *in a cubic finite element method.*

z	LPBE	New LPBE	z	LPBE	New LPBE
± 1	1.8938×10^{-3}	2.2083×10^{-4}	± 3	1.2740×10^{2}	2.2041×10^{-4}
± 2	2.9627×10^{-1}	2.1950×10^{-4}	± 4	7.7190×10^{4}	3.1570×10^{-4}

In the case of $z=\pm 1$, we see from Table 5.1 that both the new and the current LPBE models have small relative errors. This test validated our new solution decomposition schemes defined in Algorithms 1 and 2 and our program package.

However, from Figure 5.1 and Table 5.1 it can be seen that as $|z|$ was more than 1, the current LPBE model became very poor with large relative errors, in contrast, the new LPBE model retained a high accuracy of solution. These numerical results confirmed that our new LPBE model may significantly improve the solution accuracy of the current LPBE model in the case that $|\tilde{\Phi}|<1$ while $|u|>1$ within D_s.

5.2. Test cases of biomolecules

We made numerical tests on three proteins and one protein-DNA complex represented in PDB files 1CBN, 1SVR, 1A63, and 1AZQ, which were downloaded from the Protein Data Bank (*http://www.rcsb.org*). We also tested two highly charged biomolecules, one DNA and one protein called Sso7D, which were isolated from the PDB file 1AZQ. The spherical domains Ω of these six biomolecules have radii 90.2, 167.8, 128.2, 150.8, 103.6, and 153.4, respectively (in units Å). Their total charges are ranged from $0e_c$ to $14e_c$ as listed in Table 5.2. We converted these PDB files to PQR files by PDB2PQR (*http://www.poissonboltzmann.org/pdb2pqr*), and took each PQR

Table 5.2. *A comparison of the solution relative errors and solvation free energy errors between the new and the current LPBE models for biomolecules. Here, the solution relative error is defined in* (5.3)*, the solvation free energy error is defined in* (4.6)*, and a linear finite element method was used to solve both LPBE and NPBE.*

Biomolecules	Solution Relative Error		Solvation Energy Error	
(#atoms,#charges)	LPBE	New LPBE	LPBE	New LPBE
Ionic Strength $I_s = 15$ (mM)				
1CBN (642,0)	3.6399×10^{-3}	2.7676×10^{-4}	0.18	0.01
1A63 (2065,−1)	1.3436×10^{-2}	1.8460×10^{-3}	3.69	0.31
1SVR (1433,−2)	2.4958×10^{-2}	9.5078×10^{-3}	4.94	1.91
Sso7D (1095,+6)	6.0151×10^{-2}	2.0816×10^{-2}	5.82	2.67
1AZQ (1603,−8)	4.8493×10^{-2}	2.3098×10^{-2}	10.35	5.66
DNA (508,−14)	7.8425×10^{-2}	3.8014×10^{-2}	14.23	7.85
Ionic Strength $I_s = 150$ (mM)				
1CBN (642,0)	6.2170×10^{-3}	7.3755×10^{-4}	0.56	0.03
1A63 (2065,−1)	1.0467×10^{-2}	2.2177×10^{-3}	5.24	0.55
1SVR (1433,−2)	1.9084×10^{-2}	8.4096×10^{-3}	7.51	2.48
Sso7D (1095,+6)	1.7549×10^{-2}	4.5520×10^{-3}	4.63	1.04
1AZQ (1603,−8)	1.8594×10^{-2}	6.6302×10^{-3}	6.35	2.06
DNA (508,−14)	2.6143×10^{-2}	8.7803×10^{-3}	6.82	2.17
Ionic Strength $I_s = 350$ (mM)				
1CBN (642,0)	5.7253×10^{-3}	8.0342×10^{-4}	0.66	0.04
1A63 (2065,−1)	8.0312×10^{-3}	1.6956×10^{-3}	4.91	0.26
1SVR (1433,−2)	1.5331×10^{-2}	6.3296×10^{-3}	7.67	2.07
Sso7D (1095,+6)	1.0926×10^{-2}	2.1906×10^{-3}	4.07	0.65
1AZQ (1603,−8)	1.0769×10^{-2}	2.5676×10^{-3}	4.45	0.81
DNA (508,−14)	1.3946×10^{-2}	2.9812×10^{-3}	4.16	0.74

file as an input file of our program package. The finite element meshes were generated from GAMer [12] within our program package with 93,590 (1CBN), 94,340 (1A63), 89,791 (1SVR), 93,734 (Sso7D), 105,332 (DNA), and 173,548 (1AZQ) vertices, respectively. In the tests, we used the linear finite element method, $w=\tilde{\Phi}_{l0}$, and g of (2.5). The numerical results are reported in Tables 5.2 and 5.3 and Figurs 5.2 and 5.3.

Table 5.2 lists the values of the solution relative errors and solvation free energy errors for the new and the current LPBE models. Here, the solvation free energy error is defined in (4.6), and the solution relative error is defined by

$$\|u_i - u\|_{L_2(\Omega)}/\|u\|_{L_2(\Omega)}, \quad i=1,2, \tag{5.3}$$

where u_i denotes a LPBE finite element solution, and u is the corresponding NPBE finite element solution defined on the same finite element function space. The NPBE numerical solutions were calculated by a finite element program package of NPBE that we developed recently based on the solution decomposition (3.1) [11]. Table 5.2 shows that the new LPBE model is more accurate than the current LPBE model in the sense of Criteria 1 and 2.

Furthermore, we counted the total number n_T of the mesh vertices at which the solution relative errors of (5.3) were less than 1, 0.5, or 10^{-j} for $j=1$ to 5. We then calculated the percentage p of mesh vertices for the current and the new LPBE models by $p=100n_T/N_h$. Here N_h is the total number of vertices of the mesh. These

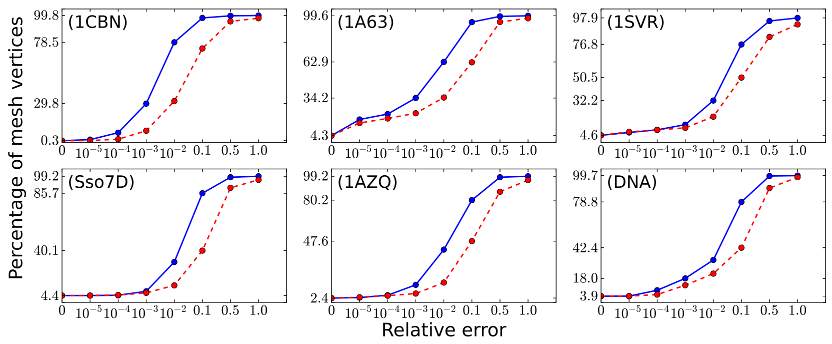

FIG. 5.2. *A comparison of the percentage of mesh vertices at which the solution relative errors* (5.3) *less than 1, 0.5, or* 10^{-j} *for* $j=1$ *to 5 for the new LPBE model (in solid line) with that for the current LPBE model (in dashed line). Here* $I_s=150$ mM.

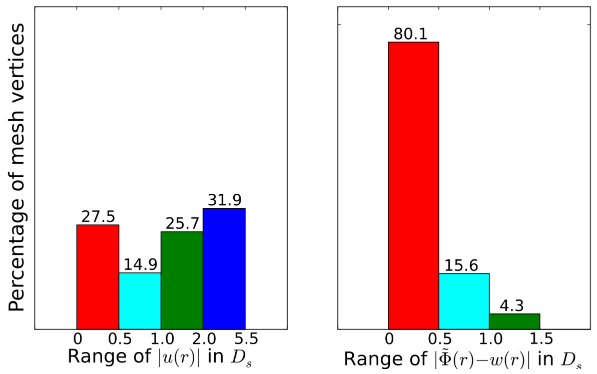

FIG. 5.3. *A comparison of the percentage of mesh vertices at which Condition* (1.1) *held (42.4% (27.5+14.9) on the left plot) with that Condition* (1.2) *held (95.7% (80.1 + 15.6) on the right plot) for the case of Sso7D using* $I_s=350$ mM.

percentage values were displayed on Figure 5.2, showing that the new LPBE solution is more accurate than the current LPBE solution for the majority of mesh vertices.

In fact, for all the six biomolecules, the numbers of vertices within D_s at which condition (1.2) held were found to be much larger than that at which (1.1) held. For example, Figure 5.3 gives the case of Sso7D using $I_s=350\ mM$, from which we see that Condition (1.2) held on 95.7 percent of the vertices in D_s (right plot) while Condition (1.1) held only on 42.4 percent of the vertices in D_s (left plot).

Table 5.3 reports the computer performance of our program package for the six biomolecules. It includes the computer CPU time data for mesh generation and the calculations of G, ∇G, Ψ, $\tilde{\Phi}_{l0}$, and $\tilde{\Phi}_l$. From this table it can be seen that all the linear finite element equations were solved efficiently in less than 19 seconds only. The new LPBE model took only a few seconds (less than 5% of the total CPU time) more than the current LPBE model. (Note that the complexities of the new and current LPBE models are almost the same in the case of $w=0$). However, more than a half of the total CPU time was spent on mesh generation. This indicates the importance

TABLE 5.3. *CPU time data (seconds) of our program package for biomolecules in linear finite element method on one 2.8 GHz Intel Core i7 processor of a MacBook Pro.*

Biomolecule	1CBN	1A63	1SVR	Sso7D	1AZQ	DNA
Solve (3.6) for $\tilde{\Phi}_{l0}$	2.53	2.42	2.33	2.54	6.01	2.94
Solve (4.3) for $\tilde{\Phi}_l$	2.62	2.37	2.23	2.52	5.91	2.69
Solve (3.3) for Ψ	8.18	8.23	8.05	8.49	18.67	9.54
Find G and ∇G	1.92	6.24	4.13	3.30	8.91	1.72
Generate mesh	31.03	70.47	51.16	63.68	70.88	24.22
Total time for LPBE	45.15	91.63	68.56	80.36	110.64	39.79
Total time for New LPBE	47.78	94.00	70.80	82.88	116.55	42.48

of developing a fast mesh generator for a finite element program package.

In the future, we plan to do more studies on the new LPBE model and look for a fast mesh generator to further improve the performance of our LPBE finite element program package.

Acknowledgements. The authors thank the two anonymous referees' valuable comments.

REFERENCES

[1] A. Logg, K.-A. Mardal, and G.N. Wells (Editors), *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS book.* Vol. 84, *Lecture Notes in Computational Science and Engineering*, Springer, 2012.

[2] L. Chen, M. J. Holst, and J. Xu. The finite element approximation of the nonlinear Poisson-Boltzmann equation. *SIAM J. Numer. Anal.*, 45(6):2298–2320, 2007.

[3] I Chern, J. Liu, and W. Wang. Accurate evaluation of electrostatics for macromolecules in solution. *Methods Appl. Anal.*, 10(2):309–328, 2003.

[4] F. Fogolari, P. Zuccato, G. Esposito, and P. Viglino. Biomolecular electrostatics with the linearized Poisson-Boltzmann equation. *Biophys. J.*, 76(1):1–16, 1999.

[5] W. Geng and R. Krasny. A treecode-accelerated boundary integral Poisson-Boltzmann solver for electrostatics of solvated biomolecules. *J. Comput. Phys.*, 247: 62–78, 2013.

[6] M.J. Holst, *The Poisson-Boltzmann Equation: Analysis and multilevel numerical solution*, Citeseer, 1994.

[7] M. Holst, J. A. McCammon, Z. Yu, Y. Zhou, and Y. Zhu. Adaptive finite element modeling techniques for the Poisson-Boltzmann equation. *Commun. Comput. Phys.*, 11(1):179, 2012.

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

[9] B. Lu, X. Cheng, and J. A. McCammon. New-version-fast-multipole-method accelerated electrostatic interactions in biomolecular systems. *J. Comput. Phys.*, 226:1348–1366, 2007.

[10] D. Xie and J. Li. A new analysis of electrostatic free energy minimization and Poisson-Boltzmann equation for protein in ionic solvent. *Nonlinear Analysis: Real World Applications, 2014*, 21:185–196, 2015.

[11] D. Xie. New solution decomposition and minimization schemes for Poisson-Boltzmann equation in calculation of biomolecular electrostatics. *J. Comput. Phys.*, 275:294–309, 2014.

[12] Z. Yu, M.J. Holst, Y. Cheng, and J.A. McCammon. Feature-preserving adaptive mesh generation for molecular shape modeling and simulation. *J. Mol. Graph. Model*, 26(8):1370–1380, 2008.

[13] Z. Zhou, P. Payne, M. Vasquez, N. Kuhn, and M. Levitt. Finite-difference solution of the Poisson-Boltzmann equation: Complete elimination of self-energy. *J. Comput. Chem.*, 17(11):1344–1351, 1996.