A SIMPLE AND EFFICIENT TECHNIQUE TO ACCELERATE THE COMPUTATION OF A NONLOCAL DIELECTRIC MODEL FOR ELECTROSTATICS OF BIOMOLECULE

JIAO LI  
School of Mathematics and Statistics  
Changsha University of Science and Technology  
Changsha, Hunan 410114 China

JINYONG YING*  
School of Mathematics and Statistics  
Central South University  
Changsha, Hunan 410083 China

(Communicated by Kok Lay Teo)

Abstract. The nonlocal modified Poisson-Boltzmann equation (NMPBE) is one important variant of a commonly-used dielectric continuum model, Poisson-Boltzmann equation (PBE). In this paper, we use a nonlinear block relaxation method to develop a new nonlinear solver for the nonlinear equation of \( \Phi \) and thus a new NMPBE solver, which is then programmed as a software package in C/C++, Fortran and Python for computing the electrostatics of a protein in a symmetric 1:1 ionic solvent. Numerical tests validate the new package and show that the new solver can improve the efficiency by at least 40% than the finite element NMPBE solver without compromising solution accuracy.

1. Introduction. The Poisson-Boltzmann equation (PBE) has been widely used as an implicit continuum model to predict electrostatic potentials [7, 11, 4, 8], which is crucial to understand the structure and the biological function of biomolecules. However, due to ignoring the ion size and the ion-ion correlations in the theory [3], the PBE model has some drawbacks. The size-modified PBEs [2, 12, 14], which include the ion size effect, and the nonlocal dielectric continuum models [1, 9, 10, 17], which take into account the polarization correlations of water molecules, were derived and proposed for the purpose of accurately modeling the electrostatic properties. Recently, the size-modified PBE [12] has been well analyzed [21, 13] and efficiently solved by the finite element method and its hybridized method [23, 24, 25] due to its PBE-like formulation. In this paper, we mainly focus on the algorithm construction for efficiently solving the nonlocal continuum model.

Nonlocal dielectric continuum models were firstly proposed in Dogonadze and Kornyshev’s early research in 1970s [9, 10, 17, 5]. However, how to numerically
solve these nonlocal models was a very difficult issue due to the involved integro-differential term \[1, 10\]. In 2004, Hildebrandt et al. [6] proposed a novel reformulation based on the Helmholtz decomposition, making it possible to solve a nonlocal model numerically. Since then, several nonlocal dielectric continuum models and numerical algorithms were developed by different techniques [6, 18, 20], including the first nonlinear nonlocal continuum model for a protein in an ionic solvent, which was called the nonlocal modified PBE (NMPBE), and its efficient finite element solver [19] validated by a nonlocal Poisson test model with available analytical solution [22]. In this paper, we propose a new numerical algorithm to further improve the efficiency of the NMPBE solver.

The difficulties for solving NMPBE come from the singular Dirac delta distributions for atomic charge terms, the exponential nonlinearity, and the nonlocal convolution term. To deal with the convolution term, the reformulation technique proposed in [19] is applied to sharply reduce the computation cost. Specifically, the nonlocal convolution term is treated as an unknown function so that NMPBE can be equivalently reformulated into a system of coupled equations, which is the key step to develop efficient numerical algorithms. Then the solution decomposition scheme reported in [19] is used to overcome the singularities, in which the solution \( u \) of NMPBE and its convolution \( w \) can be derived as a sum of three functions (i.e., \( G, \Psi, \) and \( \Phi \)), and their corresponding convolution components (\( G_w, \Psi_w, \) and \( \Phi_w \)), respectively. Here \( G \) and \( G_w \) are known in analytical expressions, \( (\Psi, \Psi_w) \) is a solution of a linear interface system, and \( (\Phi, \Phi_w) \) is a solution of a nonlinear interface system. Thus, instead of solving \( u \) directly, we solve the interface systems for \( \Psi \) and \( \Phi \), respectively.

For the finite element NMPBE solver reported in [19], it is noticed that most of its CPU time (up to 70%) was spent on solving the nonlinear equation of \( \Phi \). Here in order to solve the nonlinear problem, a modified Newton method was proposed and it needs to solve a Newton equation, whose number of unknowns is twice many as the number of mesh nodes. Due to this, it needs doubled memory and lots of CPU time to get a numerical solution. Hence, the NMPBE solver would be significantly improved in terms of efficiency if the computation time to get a numerical solution of the nonlinear equation is sharply reduced. Meanwhile, in the nonlinear system of \( (\Phi, \Phi_w) \), the equation for \( \Phi \) is nonlinear while the one for \( \Phi_w \) is actually linear.

Inspired by the above observations, we proposed a new nonlinear block relaxation method to improve the finite element NMPBE solver. More specifically, we decomposed the nonlinear system of \( (\Phi, \Phi_w) \) into a nonlinear subproblem of \( \Phi \) and a linear subproblem of \( \Phi_w \). Here we introduced a relaxation parameter \( \omega \) to accelerate the convergence of block iterations between \( \Phi \) and \( \Phi_w \). Then we use a finite element method to solve the linear problem of \( \Phi_w \) and the modified Newton minimization scheme to get a solution of the nonlinear problem of \( \Phi \), which has been proved to be globally convergent. Comparison tests on 12 proteins shows that the new nonlinear solver could improve the efficiency up to 52.4% without compromising solution accuracy, compared to the finite element NMPBE solver.

The remaining parts of this paper are organized as follows. In Section 2, we review the NMPBE model and its finite element solver. Then we propose a nonlinear block relaxation method for solving \( \Phi \) and the new algorithm for solving NMPBE in Section 3. Finally, the numerical results are reported in Section 4.
2. Review of finite element NMPBE solver. Given a protein immersed in an ionic solvent, we have the following domain partition

$$\mathbb{R}^3 = D_s \cup D_p \cup \Gamma,$$

where $D_p$ denotes a solute region hosting the protein with $n_p$ atoms, $D_s$ is the solvent region surrounding $D_p$, and $\Gamma$ is the interface between $D_s$ and $D_p$. In the implicit solvent approach, both $D_p$ and $D_s$ are treated as continuum media. Let $\lambda$ denote the polarization correlations of water molecules, $\epsilon_p > 0$ be the dielectric constant of $D_p$, $\epsilon_s$ and $\epsilon_\infty$ satisfying $0 < \epsilon_\infty < \epsilon_s$ be the dielectric constants of $D_s$ when $\lambda \to 0$ and $\lambda \to \infty$, respectively. Then for a 1:1 electrolyte (e.g., a salt solution with Sodium ($\text{Na}^+$) and chloride ($\text{Cl}^-$) ions), when distance is measured in angstrom (Å), we have the following dimensionless NMPBE [19]:

$$
\begin{aligned}
-\epsilon_s \Delta u(r) &= \alpha \sum_{j=1}^{n_p} z_j \delta_{r_j}, \quad r \in D_p, \\
-\epsilon_\infty \Delta u(r) &= (\epsilon_s - \epsilon_\infty) \nabla \cdot \int_{\mathbb{R}^3} Q_\lambda(r-r') \nabla u(r') dr' + \kappa^2 \sinh(u) = 0, \quad r \in D_s, \\
u(s^+) = u(s^-), \\
\epsilon_p \frac{\partial u(s^-)}{\partial n(s)} &= \epsilon_\infty \frac{\partial u(s^+)}{\partial n(s)} + (\epsilon_s - \epsilon_\infty) \int_{\mathbb{R}^3} Q_\lambda(s-r') \nabla u(r') dr' \cdot n(s), \quad s \in \Gamma, \\
u(r) &\to 0, |r| \to \infty,
\end{aligned}
$$

(1)

where $\alpha$ and $\kappa$ are constants defined by

$$\alpha = 10^{10} e_\infty^2 / (\epsilon_0 k_B T) \quad \text{and} \quad \kappa^2 = 2 \times 10^{-17} e_\infty^2 N_A I_s / (\epsilon_0 k_B T),$$

with $k_B$, $N_A$, $I_s$ and $T$ being the Boltzmann constant, the Avogadro number, the ionic strength in mole/liter and the absolute temperature, respectively. $r_j$ and $z_j$ are the position and charge number of the $j$th atom, respectively, $\delta_{r_j}$ is the Dirac delta distribution at point $r_j$, $n(s)$ denotes the unit outward normal vector of $D_p$, and $Q_\lambda(r)$ is the kernel function given by

$$Q_\lambda(r) = \frac{\exp(-|r|/\lambda)}{4\pi \lambda^2 |r|}.$$ 

Solving (1) directly would be a challenge due to the integro-differential term and the Dirac delta distributions. To avoid computing the nonlocal term directly, we firstly reformulate Equation (1). Since the kernel function $Q_\lambda$ satisfies the following equation [6, 19]:

$$-\lambda^2 \Delta Q_\lambda(r) + Q_\lambda(r) = \delta(r), \quad r \in \mathbb{R}^3,$$

setting $w(r) = (Q_\lambda * u)(r) = \int_{\mathbb{R}^3} Q_\lambda(r-r') u(r') dr'$ and using the properties of convolution and the kernel function $Q_\lambda(r)$, we have the following equation to characterize $w(r)$:

$$-\lambda^2 \Delta w(r) + w(r) - u(r) = 0, \quad r \in \mathbb{R}^3,$$

(2)
from which we can rewrite Equation (1) equivalently as a system of coupled equations:

\[
\begin{align*}
-\epsilon_p \Delta u(r) &= \alpha \sum_{j=1}^{n_p} z_j \delta_{\epsilon_j}, \quad r \in D_p, \\
-\epsilon \Delta u(r) + \frac{\epsilon - \epsilon_s}{\lambda^2} [u(r) - w(r)] + \kappa^2 \sinh(u(r)) &= 0, \quad r \in D_s, \\
-\lambda^2 \Delta w(r) + u(r) - w(r) &= 0, \quad r \in \Omega, \\
u(s^+) = u(s^-), \quad \epsilon_p \frac{\partial u(s^+)}{\partial n(s)} - \epsilon \frac{\partial u(s^-)}{\partial n(s)} &= (\epsilon_s - \epsilon) \frac{\partial \Psi(s)}{\partial n(s)} + \gamma \epsilon_p, \quad s \in \Gamma, \\
u(r) &\rightarrow 0, \quad w(r) \rightarrow 0, \quad |r| \rightarrow \infty.
\end{align*}
\]

Then, to handle singularities caused by the Dirac delta distributions, a solution \((u(r), w(r))\) of Equation (3) is represented as follows:

\[
u(r) = G(r) + \Psi(r) + \Phi(r), \quad w(r) = G_w(r) + \Psi_w(r) + \Phi_w(r),
\]

where \(G_w(r) = (G \ast Q_\lambda)(r), \Psi_w(r) = (\Psi \ast Q_\lambda)(r), \Phi_w(r) = (\Phi \ast Q_\lambda)(r), G(r)\) and its convolution \(G_w(r)\) are explicitly given by

\[
G(r) = \frac{\alpha}{4\pi \epsilon_p} \sum_{j=1}^{n_p} \frac{z_j}{|r - r_j|}, \quad G_w(r) = \frac{\alpha}{4\pi \epsilon_p} \sum_{j=1}^{n_p} \frac{1 - \exp(-|r - r_j|/\lambda)}{|r - r_j|}.
\]

\(\Psi\) and \(\Psi_w\) are solutions of the following linear interface system

\[
\begin{align*}
\Delta \Psi(r) &= 0, \quad r \in D_p, \\
-\epsilon \Delta \Psi(r) + \frac{\epsilon - \epsilon_s}{\lambda^2} [\Psi - \Psi_w] &= -\frac{\epsilon_s - \epsilon}{\lambda^2} [G - G_w], \quad r \in D_s, \\
-\lambda^2 \Delta \Psi_w(r) + \Psi_w(r) - \Psi(r) &= 0, \quad r \in \Omega, \\
\Psi(s^+) = \Psi(s^-), \quad \epsilon_p \frac{\partial \Psi(s^+)}{\partial n(s)} - \epsilon \frac{\partial \Psi(s^-)}{\partial n(s)} &= (\epsilon_s - \epsilon) \frac{\partial \Psi_w(s)}{\partial n(s)} + g_i(s), \quad s \in \Gamma, \\
\Psi(r) &\rightarrow 0, \quad \Psi_w(r) \rightarrow 0, \quad |r| \rightarrow \infty.
\end{align*}
\]

And \(\Phi\) and \(\Phi_w\) are solutions of the nonlinear interface boundary value system

\[
\begin{align*}
\Delta \Phi(r) &= 0, \quad r \in D_p, \\
-\epsilon \Delta \Phi(r) + \frac{\epsilon_s - \epsilon}{\lambda^2} [\Phi - \Phi_w] + \kappa^2 \sinh(\Phi + \Psi + G) &= 0, \quad r \in D_s, \\
-\lambda^2 \Delta \Phi_w(r) + \Phi_w(r) - \Phi(r) &= 0, \quad r \in \Omega, \\
\Phi(s^+) = \Phi(s^-), \quad \epsilon_p \frac{\partial \Phi(s^+)}{\partial n(s)} - \epsilon \frac{\partial \Phi(s^-)}{\partial n(s)} &= (\epsilon_s - \epsilon) \frac{\partial \Phi_w(s)}{\partial n(s)} + g_i(s), \quad s \in \Gamma, \\
\Phi(r) &\rightarrow 0, \quad \Phi_w(r) \rightarrow 0, \quad |r| \rightarrow \infty.
\end{align*}
\]

Here the function \(g_i\) is defined by

\[
g_i(s) = (\epsilon_s - \epsilon_p) \frac{\partial G(s)}{\partial n(s)} + (\epsilon_s - \epsilon) \frac{\partial G_w(s)}{\partial n(s)}
\]

for any \(s \in \Gamma\).

In computation, we need to truncate Equation (3) to a sufficient large domain \(\Omega = D_s \cup D_p \cup \Gamma\) with some prescribed boundary functions \(g(s)\) and \(g_w(s) = (g \ast Q_\lambda)(s)\) for \(s \in \partial \Omega\). Correspondingly, we solve systems (6) and (7) for \(\Psi\) and \(\Phi\) with boundary functions \((g(s) - G(s), g_w(s) - G_w(s))\) and \((0, 0)\), respectively.

In the implementation, we could simply set \(g = g_w = 0\) or use the analytical solution of the nonlocal Debye-Hückel equation reported in [19].

To solve the linear interface system (6) of \(\Psi\), we reformulate it as the following variational problem:

Find \((\Psi, \Psi_w) \in \mathcal{M} \times \mathcal{M}\) satisfying \(\Psi(s) = g(s) - G(s)\)

and \(\Psi_w(s) = g_w(s) - G_w(s)\) for all \(s \in \partial \Omega\) such that
A SIMPLE AND EFFICIENT TECHNIQUE TO ACCELERATE THE COMPUTATION ...

\[ A_1((\Psi, \Psi_w), (v_1, v_2)) = L_1((v_1, v_2)) \quad \forall (v_1, v_2) \in M_0 \times M_0, \quad (8) \]

where \( M \) is a finite element function space as a subspace of the usual Sobolev function space \( H^1(\Omega) \), \( M_0 = \{ v \in M \mid v = 0 \text{ on } \partial\Omega \} \), and the bilinear form \( A_1 \) and the linear form \( L_1 \) are given as follows

\[ A_1((\Psi, \Psi_w), (v_1, v_2)) = a(\Psi, v_1) + (\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla \Psi_w(\mathbf{r}) \cdot \nabla v_1(\mathbf{r})d\mathbf{r} \]
\[ + \lambda^2 \int_{\Omega} \nabla \Psi_w(\mathbf{r}) \cdot \nabla v_2(\mathbf{r})d\mathbf{r} + \int_{\Omega} (\Psi_w(\mathbf{r}) - \Psi(\mathbf{r}))v_2(\mathbf{r})d\mathbf{r}, \quad (9) \]

\[ L_1((v_1, v_2)) = (\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla G_w(\mathbf{r}) \cdot \nabla v_1(\mathbf{r})d\mathbf{r} + (\epsilon_p - \epsilon_\infty) \int_{D_s} \nabla G(\mathbf{r}) \cdot \nabla v_1(\mathbf{r})d\mathbf{r}, \quad (10) \]

where

\[ a(u, v) = \epsilon_p \int_{D_p} \nabla u \cdot \nabla v d\mathbf{r} + \epsilon_\infty \int_{D_s} \nabla u \cdot \nabla v d\mathbf{r}. \]

Then the remaining difficulty to solve NMPBE is how to solve the nonlinear problem of \( \Phi \). To do so, the modified Newton method was adapted in the finite element solver. That is, to get the solution \( (\Phi(\mathbf{r}), \Psi_w(\mathbf{r})) \) of (7), we have a sequence of iterates, \( \{(\Phi^k, \Phi_w^k)\} \), as follows:

\[ \Phi^{k+1} = \Phi^k + \lambda_k p_k, \quad \Phi_w^{k+1} = \Phi_w^k + \lambda_k q_k \quad k = 0, 1, 2, \ldots, \quad (11) \]

where \((\Phi^0, \Phi_w^0)\) is the initial guess, \( \lambda_k \) is a step length determined by a line search algorithm, \((p_k, q_k)\) is a pair of search directions satisfying the following variational problem:

Find \((p_k, q_k) \in M_0 \times M_0\) such that

\[ A((p_k, q_k), (v_1, v_2)) = L((v_1, v_2)) \quad \forall (v_1, v_2) \in M_0 \times M_0, \quad (12) \]

where the bilinear functional \( A \) is defined by

\[ A((p_k, q_k), (v_1, v_2)) = a(p_k, v_1) + (\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla q_k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r})d\mathbf{r} \]
\[ + \lambda^2 \int_{\Omega} \nabla q_k(\mathbf{r}) \cdot \nabla v_2(\mathbf{r})d\mathbf{r} + \int_{\Omega} (q_k(\mathbf{r}) - p_k(\mathbf{r}))v_2(\mathbf{r})d\mathbf{r} \]
\[ + \nu^2 \int_{D_s} p_k(\mathbf{r})v_1(\mathbf{r}) \cosh(\Phi^k + \Psi + G)d\mathbf{r}, \quad (13) \]

and the linear functional \( L \) is given as follows

\[ L((v_1, v_2)) = - \left[ (\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla \Phi^k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r})d\mathbf{r} + \epsilon_p \int_{D_p} \nabla \Phi^k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r})d\mathbf{r} \right. \]
\[ + \epsilon_\infty \int_{D_s} \nabla \Phi^k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r})d\mathbf{r} + \nu^2 \int_{D_s} v_1(\mathbf{r}) \sinh(\Phi^k + \Psi + G)d\mathbf{r} \] \[ \left. \right] \quad (14) \]

In the implementation, the initial guess \( \Phi^0 \) can be selected as zero for simplicity or a solution of the local PBE. An upper bound of 85 is set by default for truncating the value of the sum \( \Psi + G + \Phi^k \) to avoid the possible overflow problem of hyperbolic terms. Each Newton equation of (12) is solved by GMRES-ILU from the PETSc library with the absolute and the relative residue errors less than a given tolerance \((10^{-8})\) by default. And the Newton iteration stops until the following termination rule holds

\[ \| \Phi^{k+1} - \Phi^k \| \leq \epsilon \quad \text{or} \quad \| \Phi_w^{k+1} - \Phi_w^k \| \leq \epsilon. \]

Here \( \| \cdot \| \) denotes the Euclidean norm and \( \epsilon \) is a given tolerance.
3. Nonlinear block relaxation method. For the finite element NMPBE solver, the main drawback is that it may take up to 70% of the total CPU time to obtain the nonlinear solution of $\Phi$. Therefore, it would be significantly improved in terms of efficiency if the computation time to get the numerical solution of the nonlinear equation is sharply reduced. In this section, we try to propose a new nonlinear block relaxation method for solving $\Phi$ so that we can have a new efficient NMPBE solver.

3.1. Nonlinear block relaxation method for solving $\Phi$. For solving the nonlinear problem of $\Phi$, the modified Newton method needs solve the Newton equation (12) and it has twice many number of unknowns as the number of mesh nodes. Hence it needs more memory and possibly deteriorates the convergence of the linear solver for solving each Newton equation, which results in more computation cost. Meanwhile, in Equation (7), the equation of $\Phi$ is nonlinear while the one of $\Phi^W$ is linear. Inspired by these observations, we develop the following nonlinear block relaxation method.

Instead of solving $\Phi$ by the modified Newton method in [19], we construct the following scheme to get a numerical solution. Given a pair of initial guesses $(\Phi_0, \Phi_0^W)$, we update them by the following equations:

$$\Phi^k = \omega \Phi_k + (1 - \omega) \Phi^{k-1}, \quad \Phi^W_k = \omega \Phi_{w,k} + (1 - \omega) \Phi^{W,k-1}, \quad k = 1, 2, \ldots, (15)$$

where $\omega \in (0, 2)$ is a relaxation parameter, $\bar{\Phi}^k$ is a solution of the following nonlinear variational equation

$$a(\Phi, v) + \kappa^2 \int_D \sinh(\Psi + G + \Phi)vd\mathbf{r} = (\epsilon_\infty - \epsilon_s) \int_D \nabla \Phi^{k-1}_w \cdot \nabla v d\mathbf{r} \quad \forall v \in H^1_0(\Omega),$$

and $\bar{\Phi}^{W,k}$ is a solution of the following linear variational equation

$$\lambda^2 \int_\Omega \nabla \Phi^W \cdot \nabla v d\mathbf{r} + \int_\Omega \Phi^W vd\mathbf{r} = \int_\Omega \Phi^{W,k} d\mathbf{r} \quad \forall v \in H^1_0(\Omega).$$

Here the updated function value $\Phi^k(r)$ has been used in solving the equation of $\Phi^W$ to accelerate the convergence of the method. For solving the nonlinear equation (16), we adapt the following minimization method. Firstly the variational problem of (16) is equivalently reformulated as the following minimization problem:

$$J(\bar{\Phi}_k) = \min_{v \in M_0} J(v),$$

where the energy functional $J$ is defined by

$$J(v) = \frac{1}{2} a(v, v) + (\epsilon_s - \epsilon_{\infty}) \int_{D_s} \nabla v \cdot \nabla \Phi^{k-1}_w d\mathbf{r} + \kappa^2 \int_{D_s} \cosh(\Psi + G + v)d\mathbf{r}.$$

Then a modified Newton minimization algorithm defines a sequence of iterates, $\{\Phi^m_k\}$, as follows:

$$\bar{\Phi}^{m+1}_k = \bar{\Phi}^m_k + \lambda_k^m p_k^m, \quad m = 0, 1, 2, \ldots, (19)$$

where $\bar{\Phi}^0_k$ is the initial guess, $\lambda_k^m$ is a step length, and $p_k^m$ is a search direction satisfying the Newton equation in the variational form: Find $p_k^m \in M_0$ such that

$$J''(\bar{\Phi}_k^m)(p_k^m, v) = -J'(\bar{\Phi}_k^m)v \quad \forall v \in M_0,$$
where \( J'(\bar{\Phi}_m^k) \) is the first Fréchet-derivative of \( J \) at \( \bar{\Phi}_m^k \), which is a linear continuous functional on \( H_0^1(\Omega) \) defined as follows: \( \forall v \in H_0^1(\Omega) \),
\[
J'(\bar{\Phi}_m^k)v = a(\bar{\Phi}_m^k, v) + (\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla v \cdot \nabla \Phi^{k-1}_w dr + \kappa^2 \int_{D_s} \sinh(\Psi + G + \Phi^m_k) v dr,
\]
and \( J''(\bar{\Phi}_m^k) \) is the second Fréchet-derivative of \( J \) at \( \bar{\Phi}_m^k \), which is a bilinear continuous functional on \( H_0^1(\Omega) \) defined by
\[
J''(\bar{\Phi}_m^k)(p, v) = a(p, v) + \kappa^2 \int_{D_s} \cosh(\Psi + G + \Phi^m_k) pv dr \quad \forall p, v \in H_0^1(\Omega).
\]

In the implementation, the initial guess \( \Phi^{(0)} \) of the nonlinear block relaxation method can be selected as zero for simplicity or a solution of the local PBE model as before. And the initial guess for solving each nonlinear equation (16) of \( \Phi^k \) is simply set as \( \Phi^{k-1} \). To avoid the possible overflow problem of hyperbolic terms, an upper bound of 85 is also set by default for truncating the value of the sum \( \Psi + G + \Phi^m_k \). And each Newton equation of (20) can be solved numerically by the conjugate gradient method with incomplete LU preconditioning due to the symmetry and coercivity of \( J''(\bar{\Phi}_m^k)(p, v) \). Here we set the absolute and relative residue errors as \( 10^{-8} \) by default.

Given a tetrahedral mesh \( \Omega_h \) on \( \Omega \), we denote the true solutions on the defined finite element function space by \( \Phi_h \) and \( \Phi_{w,h} \), respectively. Then we have the following result.

**Theorem 3.1.** If the iterate sequences \( \{\Phi^k\} \subset H_0^1(\Omega_h) \) and \( \{\Phi^k_w\} \subset H_0^1(\Omega_h) \) are convergent, then \( \Phi^k \to \Phi_h \) and \( \Phi^k_w \to \Phi_{w,h} \) as \( k \to \infty \).

**Proof.** It is known that each equivalence class in \( H_0^1(\Omega_h) \) has a continuous representative. Since \( \{\Phi^k\} \subset H_0^1(\Omega_h) \) is convergent, we have
\[
\int_{\Omega} |\Phi^m - \Phi^n|^2 \leq ||\Phi^m - \Phi^n||^2_{H_0^1(\Omega_h)} \to 0 \quad \text{as} \quad n, m \to \infty.
\]
Thus, on the bounded domain \( \Omega \), it is easy to have a upper bound for \( \{\Phi^k\} \), which is also true for \( \{\Phi^k_w\} \). Then applying the dominated convergence theorem to variational problems corresponding to problems (16) and (17), respectively, we have \( \forall v \in H_0^1(\Omega_h) \)
\[
\int_{\Omega} a(\Phi, v) + \kappa^2 \int_{D_s} \sinh(\Psi + G + \Phi_h) v dx = (\epsilon_\infty - \epsilon_s) \int_{D_s} \nabla \Phi_{w,h} \cdot \nabla v dr
\]
and
\[
\lambda^2 \int_{\Omega} \nabla \Phi_{w,h} \cdot \nabla v dr + \int_{\Omega} \Phi_{w,h} v dr = \int_{\Omega} \Phi_h v dr.
\]
Adding the above two equations together, we obtain the variation form corresponding to Equation (7). Since Equation (7) has a unique weak solution [16] in \( H_0^1(\Omega_h) \), the conclusion is followed.

Furthermore, it is easy to see that if
\[
||\Phi^k - \Phi^{k-1}|| \to 0 \quad \text{and} \quad ||\Phi^k_w - \Phi^{k-1}_w|| \to 0,
\]
then we know the iterates \( \Phi^k \) and \( \Phi^k_w \) generated by (15) are convergent. Thus, based on Theorem 3.1, we could set up the following inequalities as a termination rule for the nonlinear block relaxation method.
\[
||\Phi^k - \Phi^{k-1}|| \leq \epsilon \quad \text{or} \quad ||\Phi^k_w - \Phi^{k-1}_w|| \leq \epsilon. \quad (21)
\]
Here $\| \cdot \|$ denotes the Euclidean norm and $\epsilon$ is set as $10^{-6}$ by default.

Meanwhile, the termination rule for the modified Newton minimization algorithm for solving nonlinear equation (16) of $\Phi$ is set as

$$\| J'(\Phi^m_k) \| \leq \epsilon_0 \quad \text{or} \quad \| \Phi^m_k - \Phi^{m-1}_k \| \leq \epsilon_0,$$

where $\epsilon_0$ is set adaptively as

$$\max \{ \min \{ 10^{-1}, \| \Phi^k - \Phi^{k-1} \| / 10 \}, 10^{-6} \},$$

since it is unnecessary to solve the nonlinear problem of (16) accurately at the first few steps due to the error from the nonlinear block relaxation method.

3.2. New algorithm for solving NMPBE. Based on the nonlinear block relaxation method for solving $\Phi$, we can modify the finite element NMPBE solver to a new one. For clarity, this new solver is presented in Algorithm 1.

Algorithm 1 (The new NMPBE solver). Suppose the space $\Omega$ is sufficiently large. A solution $u$ of Equation (3) is calculated approximately in the following five steps:

Step 1. Construct an interface-matched tetrahedral mesh for a given protein.

Step 2. Calculate $G$ and $G_w$ as well as $\nabla G$ and $\nabla G_w$ on $\Omega$ according to (5).

Step 3. Solve the linear variational system (8) for $\Psi$.

Step 4. Solve the nonlinear problem (7) of $\Phi$ by the nonlinear block relaxation method with the termination rule (21).

Step 5. Construct $u$ by the solution decomposition $u = G + \Psi + \Phi$.

The finite element method is chosen to solve each linear system and then the new NMPBE solver is programmed in C\C++, Fortran and Python based on the finite element NMPBE solver [19]. The modified version of the molecular surface and volumetric mesh generation program package GAMer [25] was adapted to generate the interface-matched tetrahedral mesh. We wrote Fortran subroutines for efficiently calculating the values of $G$ and $G_w$ and their gradients at the finite element mesh nodes. In the end, all of these Fortran subroutines and C\C++ programs were converted to Python modules by f2py (http://cens.ioc.ee/projects/f2py2e/) and SWIG (http://www.swig.org), respectively. With these modules, the main program of the new solver was programmed in python based on the state-of-the-art finite element library DOLFIN [15] from the FEniCS project.

4. Numerical results. In this section, we report the numerical experiments using our new Python program of Algorithm 1. For simplicity, we set $\epsilon_p = 2.0$, $\epsilon_s = 80.0$, $\epsilon_\infty = 1.8$, $\lambda = 15.0$, $T = 298.15$, $I_s = 0.1$, and all the numerical tests were done by using the default values of the parameters on a processor of a Macbook Pro with the 2.7 GHZ Dual-Core Intel I5 and 8 GB memory. Without explicit statement, the boundary condition $g$ is always set to be the solution of the nonlocal Debye-Hückel equation.

Numerical tests were made on the following 12 proteins (varying number of atoms up to 11,439, see Table 1) to validate the new program package as well as demonstrating the efficiency improvements in terms of CPU time, compared to the finite element NMPBE solver. Here PDB (Protein Data Bank) files of these 12 proteins were downloaded from the PDB website (https://www.rcsb.org) and then converted to PQR files by the tool PDB2PQR (http://nbcr-222.ucsd.edu/pdb2pqr_2.1.1/).
We called our modified version of GAMer [25] to generate the finite element meshes for computation. Here, each interface $\Gamma$ is generated by using the Gaussian blurring approach. It has been shown that the molecular surface generated from GAMer is close to the commonly-used solvent-accessible surface (SAS) in [24]. Additionally, the over-relaxation parameter $\omega$ of the nonlinear block relaxation method was set as 0.88. The boundary value function $g$ and an initial iterate $(\Phi^0, \Phi^0_w)$ of the nonlinear block relaxation method were set as the nonlocal Debye-Hückel solution and the solution of the local PBE model, respectively. For comparison purpose, keeping all the same settings, we called the finite element NMPBE solver in [19] to repeat the calculations. Thus, for the modified Newton method to solve problem (12), the termination rule was also set as

$$\|\Phi^{k+1} - \Phi^k\| \leq 10^{-6} \quad \text{or} \quad \|\Phi_w^{k+1} - \Phi_w^k\| \leq 10^{-6},$$

which is exactly the same as (21).

Meanwhile, we compare the solutions obtained by these two solvers as follows

$$E_h = \frac{\|\Phi_{new} - \Phi_{fe}\|_{L_2}}{\|\Phi_{fe}\|_{L_2}},$$

where $\Phi_{new}$ is the numerical solution of $\Phi$ obtained by the new NMPBE solver, and $\Phi_{fe}$ is the one from the finite element solver. Furthermore, to demonstrate the efficiency improvements, we calculate the CPU time speedup $S_p$ defined as follows

$$S_p = \frac{T_{\Phi,fe}}{T_{\Phi,new}},$$

where $T_{\Phi,fe}$ and $T_{\Phi,new}$ denote the CPU time for solving the nonlinear system of (7) (do not count the CPU time for computing the initial guess pair $(\Phi^0, \Phi^0_w)$) spent by the finite element solver and the new solver, respectively. All of these results are reported in Table 2 and Figure 1.

From Table 2 it can be seen that for the used mesh sets, the efficiency of the finite element NMPBE solver on the 12 proteins is significantly improved without compromising any solution accuracy due to the efficiency improvement of the nonlinear solver for solving $\Phi$. Surprisingly, for several protein cases, the total CPU time is reduced to the one needed for solving the nonlinear system (7) in the finite element NMPBE solver. Meanwhile, the nonlinear block relaxation method only used approximately 11 iterations to reach the termination rule, showing its good convergent rate. And from Figure 1, it is obviously seen that the nonlinear block relaxation method accelerates the computation of (7) at least by 40% in these tests.

To explore further, taking the protein with PDB ID 1A63 as an example, we list out more detailed information about the Newton iteration number and the iteration

<table>
<thead>
<tr>
<th>Index</th>
<th>PDB ID</th>
<th>$n_p$</th>
<th>Index</th>
<th>PDB ID</th>
<th>$n_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2LZX</td>
<td>488</td>
<td>7</td>
<td>1A63</td>
<td>2065</td>
</tr>
<tr>
<td>2</td>
<td>1AJJ</td>
<td>513</td>
<td>8</td>
<td>1CID</td>
<td>2783</td>
</tr>
<tr>
<td>3</td>
<td>1FXD</td>
<td>811</td>
<td>9</td>
<td>1A7M</td>
<td>2803</td>
</tr>
<tr>
<td>4</td>
<td>1HPT</td>
<td>852</td>
<td>10</td>
<td>2AQ5</td>
<td>6024</td>
</tr>
<tr>
<td>5</td>
<td>4PTI</td>
<td>892</td>
<td>11</td>
<td>1F6W</td>
<td>8243</td>
</tr>
<tr>
<td>6</td>
<td>1SVR</td>
<td>1433</td>
<td>12</td>
<td>1C4K</td>
<td>11439</td>
</tr>
</tbody>
</table>

Table 1. Basic information of the 12 proteins used for numerical tests. Here $n_p$ is the number of atoms.
Table 2. Comparison of the performance of our new NMPBE solver (New) with that of the finite element NMPBE solver (FE) [19] in CPU time measured in seconds. Here Iter. Number denotes the iteration number needed in the nonlinear block relaxation method and $E_h$ is computed by (22), and residual norm means the norm of Equation (7)’s residual.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Number of Mesh Nodes</th>
<th>Iter. Number</th>
<th>Find $\Phi$ New</th>
<th>Total Time New</th>
<th>Relative error $E_h$</th>
<th>Residual norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>2LZX</td>
<td>26349</td>
<td>11</td>
<td>15.23</td>
<td>27.61</td>
<td>2.1 x 10^{-8}</td>
<td>1.42 x 10^{-4}</td>
</tr>
<tr>
<td>1AJJ</td>
<td>31910</td>
<td>11</td>
<td>26.60</td>
<td>48.25</td>
<td>3.9 x 10^{-8}</td>
<td>3.69 x 10^{-5}</td>
</tr>
<tr>
<td>1FXD</td>
<td>34469</td>
<td>12</td>
<td>23.19</td>
<td>42.48</td>
<td>1.2 x 10^{-8}</td>
<td>8.03 x 10^{-4}</td>
</tr>
<tr>
<td>1HPT</td>
<td>48229</td>
<td>10</td>
<td>32.33</td>
<td>58.78</td>
<td>3.3 x 10^{-8}</td>
<td>1.11 x 10^{-4}</td>
</tr>
<tr>
<td>4PTI</td>
<td>39468</td>
<td>10</td>
<td>25.85</td>
<td>46.52</td>
<td>1.5 x 10^{-8}</td>
<td>8.95 x 10^{-5}</td>
</tr>
<tr>
<td>1SVR</td>
<td>61074</td>
<td>11</td>
<td>55.17</td>
<td>90.59</td>
<td>2.6 x 10^{-8}</td>
<td>4.25 x 10^{-5}</td>
</tr>
<tr>
<td>1A63</td>
<td>22054</td>
<td>11</td>
<td>13.52</td>
<td>27.09</td>
<td>1.6 x 10^{-8}</td>
<td>1.82 x 10^{-4}</td>
</tr>
<tr>
<td>1CID</td>
<td>19872</td>
<td>10</td>
<td>11.07</td>
<td>21.68</td>
<td>1.9 x 10^{-8}</td>
<td>1.09 x 10^{-4}</td>
</tr>
<tr>
<td>1A7M</td>
<td>20883</td>
<td>10</td>
<td>11.63</td>
<td>22.42</td>
<td>3.2 x 10^{-8}</td>
<td>3.16 x 10^{-4}</td>
</tr>
<tr>
<td>2AQ5</td>
<td>38151</td>
<td>11</td>
<td>29.58</td>
<td>53.69</td>
<td>2.8 x 10^{-8}</td>
<td>1.61 x 10^{-4}</td>
</tr>
<tr>
<td>1F0W</td>
<td>49006</td>
<td>11</td>
<td>46.77</td>
<td>86.41</td>
<td>2.3 x 10^{-8}</td>
<td>7.05 x 10^{-4}</td>
</tr>
<tr>
<td>1C4K</td>
<td>72046</td>
<td>11</td>
<td>70.04</td>
<td>118.93</td>
<td>3.7 x 10^{-8}</td>
<td>1.69 x 10^{-4}</td>
</tr>
</tbody>
</table>

Time speedup $S_p$ defined in (23) achieved by our new NMPBE solver for the 12 protein tests on two mesh sets. Initial mesh sets denote the meshes used in Table 2 and Refined mesh sets mean the ones used in Table 3.

To further show the improvement of the new NMPBE solver, we repeated the above numerical tests on the 12 proteins by constructing a refined mesh sets with much more number of mesh nodes, and the numerical results are reported in Table 3.
A SIMPLE AND EFFICIENT TECHNIQUE TO ACCELERATE THE COMPUTATION ...

Figure 2. Newton Steps and the iteration numbers of the linear solver to solve Φ for case 1A63 in the new and the finite element NMPBE program packages. The left plot presents the number of block relaxation iteration (the x-axis), the Newton steps in each block iteration (the numbers above the x-axis), and the concrete/average iteration number (the points/the solid lines) of the linear solver in the new program package. The right plot presents the Newton steps (the x-axis) and the iteration numbers of the linear solver in the finite element one.

From Table 3, we can see that the nonlinear block relaxation method still performs well, or even better, and improves the efficiency at least by 42% in this case without compromising the solution accuracy. However, it is interesting to note that the block iteration numbers were almost reduced by one compared to the case using the first mesh set, which indicates that its convergence rate may depend on the mesh size and this will be explored in the future.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Number of Mesh Nodes</th>
<th>Iter. Number</th>
<th>Find Φ New</th>
<th>Total Time New</th>
<th>FE</th>
<th>Relative error $E_h$</th>
<th>Residual norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>2LZX</td>
<td>535400</td>
<td>10</td>
<td>167.0</td>
<td>291.2</td>
<td>311.2</td>
<td>435.3</td>
<td>$3.5 \times 10^{-8}$</td>
</tr>
<tr>
<td>1AJJ</td>
<td>538321</td>
<td>10</td>
<td>223.9</td>
<td>347.3</td>
<td>387.4</td>
<td>600.8</td>
<td>$3.5 \times 10^{-8}$</td>
</tr>
<tr>
<td>1FXD</td>
<td>540849</td>
<td>11</td>
<td>201.5</td>
<td>346.2</td>
<td>363.1</td>
<td>507.8</td>
<td>$1.7 \times 10^{-8}$</td>
</tr>
<tr>
<td>1HIPT</td>
<td>543220</td>
<td>9</td>
<td>186.8</td>
<td>399.6</td>
<td>332.3</td>
<td>545.0</td>
<td>$3.3 \times 10^{-8}$</td>
</tr>
<tr>
<td>4PTI</td>
<td>541329</td>
<td>9</td>
<td>173.4</td>
<td>328.7</td>
<td>319.5</td>
<td>474.7</td>
<td>$2.3 \times 10^{-8}$</td>
</tr>
<tr>
<td>1SVR</td>
<td>550170</td>
<td>10</td>
<td>229.7</td>
<td>411.0</td>
<td>415.3</td>
<td>596.7</td>
<td>$2.0 \times 10^{-8}$</td>
</tr>
<tr>
<td>1A63</td>
<td>558010</td>
<td>11</td>
<td>253.3</td>
<td>442.8</td>
<td>573.1</td>
<td>762.7</td>
<td>$2.4 \times 10^{-8}$</td>
</tr>
<tr>
<td>1CID</td>
<td>558374</td>
<td>10</td>
<td>263.0</td>
<td>389.0</td>
<td>409.4</td>
<td>595.4</td>
<td>$2.7 \times 10^{-8}$</td>
</tr>
<tr>
<td>1A7M</td>
<td>563919</td>
<td>11</td>
<td>242.9</td>
<td>471.7</td>
<td>442.6</td>
<td>671.4</td>
<td>$4.8 \times 10^{-8}$</td>
</tr>
<tr>
<td>2AQ5</td>
<td>577821</td>
<td>10</td>
<td>296.7</td>
<td>566.8</td>
<td>637.7</td>
<td>907.8</td>
<td>$3.5 \times 10^{-8}$</td>
</tr>
<tr>
<td>1GOW</td>
<td>574686</td>
<td>11</td>
<td>332.1</td>
<td>597.3</td>
<td>707.8</td>
<td>973.0</td>
<td>$3.7 \times 10^{-8}$</td>
</tr>
<tr>
<td>1C4K</td>
<td>573111</td>
<td>14</td>
<td>396.6</td>
<td>698.3</td>
<td>940.8</td>
<td>1242.5</td>
<td>$3.5 \times 10^{-8}$</td>
</tr>
</tbody>
</table>

Table 3. Comparison of the performance of our new NMPBE solver (New) on the refined meshes with that of the finite element NMPBE solver (FE) [19] in CPU time measured in seconds.

Acknowledgments. The first author was financially supported by the China NSF (NSFC 11501053 and NSFC 5170713) and the Education Department of Hunan (15C0026). The second author was supported by the China NSF (NSFC 11701576) and the initiative research fund from Central South University.
REFERENCES


Received March 2018; revised June 2018.

*E-mail address*: lijiaoflying@163.com
*E-mail address*: ying48882345@gmail.com