Journal of Computational Physics 298 (2015) 636–651

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A new finite element and finite difference hybrid method for computing electrostatics of ionic solvated biomolecule

Jinyong Ying, Dexuan Xie *

Department of Mathematical Sciences, University of Wisconsin-Milwaukee, Milwaukee, WI, 53201, USA

ARTICLE INFO

Article history:
Received 24 November 2014
Received in revised form 10 June 2015
Accepted 24 June 2015
Available online 29 June 2015

Keywords:
Poisson–Boltzmann equation
Finite element method
Finite difference method
Biomolecular electrostatics
Domain decomposition
Multigrid preconditioning
Solvation free energy

ABSTRACT

The Poisson–Boltzmann equation (PBE) is one widely-used implicit solvent continuum model for calculating electrostatics of ionic solvated biomolecule. In this paper, a new finite element and finite difference hybrid method is presented to solve PBE efficiently based on a special seven-overlapped box partition with one central box containing the solute region and surrounded by six neighboring boxes. In particular, an efficient finite element solver is applied to the central box while a fast preconditioned conjugate gradient method using a multigrid V-cycle preconditioning is constructed for solving a system of finite difference equations defined on a uniform mesh of each neighboring box. Moreover, the PBE domain, the box partition, and an interface fitted tetrahedral mesh of the central box can be generated adaptively for a given PQR file of a biomolecule. This new hybrid PBE solver is programmed in `C`, `Fortran`, and `Python` as a software tool for predicting electrostatics of a biomolecule in a symmetric 1:1 ionic solvent. Numerical results on two test models with analytical solutions and 12 proteins validate this new software tool, and demonstrate its high performance in terms of CPU time and memory usage.

1. Introduction

Electrostatic interactions are important in understanding the structure and biological function of biomolecules, catalytic activity, ligand association, and pK_a values [1–4]. The Poisson–Boltzmann equation (PBE) is one widely-used implicit solvent continuum model for calculating electrostatics of biomolecules (protein or nucleic acids) in ionic solvent [5–7]. A lot of work has been done in the development of PBE numerical solvers and program packages based on finite difference, finite element, and boundary integral equation approaches [8–13]. The popular PBE program packages and web-based resources, such as APBS [10], DelPhi [14], PBEQ [12,15], and UHBD [16], have become powerful simulation aides in the study of biomolecular structure and function, biomolecule-ligand association, ion channel modeling, and computer-aided drug design.

To improve the accuracy of PBE numerical solutions, we recently developed a solution decomposition PBE solver (SDPB) using finite element and minimization techniques [13]. Instead of solving PBE directly, in SDPB, two other interface problems — one linear interface problem for Ψ and one nonlinear interface problem for $\tilde{\Phi}$ — are solved to yield a numerical solution u of PBE as a sum of a given function G with Ψ and $\tilde{\Phi}$. Since G collects all the singularity points of u, both Ψ and $\tilde{\Phi}$ become twice continuously differentiable within both the solute and solvent regions. Thus, their numerical approximations can be calculated much more easily and less expensively than directly calculating u, which is singular at each atomic position $\mathbf{r}_j$, to reach a required numerical accuracy. We note that in DelPhi, APBS and PBEQ, a finite difference method

* Corresponding author.
E-mail address: dxie@uwm.edu (D. Xie).

http://dx.doi.org/10.1016/j.jcp.2015.06.016
0021-9991/

defined on a uniform mesh accelerated by geometric multigrid techniques is the major algorithm for solving PBE directly. Its mesh size h may become very small to catch the solution behavior of PBE around each singularity point $\mathbf{r}_j$, making the finite difference system too large to be solved due to the memory limitation of computers. Hence, it is critical to solve PBE indirectly through computing Ψ and $\tilde{\Phi}$.

Another feature of SDPB is to use an unstructured interface fitted tetrahedral mesh to approximate the interface Γ between the solute and solvent regions. Such a treatment can yield a numerical solution of PBE in high accuracy in comparison to the case of a uniform finite difference mesh [17]. However, using an unstructured mesh may cause efficient geometric multigrid techniques no longer to work, and require extra memory locations to store the mesh data and the nonzero entries of the coefficient matrix of each involved linear system.

In order to confine the disadvantages while retaining the advantages of SDPB, in this paper, we present a new finite element and finite difference hybrid method based on the Schwartz domain decomposition approach [18,19]. This new hybrid PBE solver was motivated from the fact that the linear variational problem for determining the search direction p_k of the modified Newton minimization algorithm from SDPB can be reformulated as a linear elliptic interface boundary value problem (see Theorem 3.1). From this fact it implies that the modified Newton minimization algorithm can also be implemented based on the finite difference approach. The only change to be made is to solve the linear interface problems of Ψ (see (5)) and p_k (see (11)) by a finite difference method. To combine the advantages of finite element and finite difference approaches together, we propose an overlapped box iterative method for solving the linear interface problems of Ψ and p_k based on a special partition of the whole domain Ω into seven overlapped boxes, in which one central box contains the solute region D_p and is surrounded by the six neighboring boxes (see Fig. 1 for an illustration). Since each neighboring box is a part of the solvent region D_s only, the corresponding equations of Ψ and p_k are reduced to the regular linear elliptic boundary value problems. Thus, they can be solved by a finite difference method defined on a uniform mesh to enable the usage of efficient geometric multigrid techniques. To do so, we construct the preconditioned conjugate gradient (PCG) method using a multigrid V-cycle preconditioning, called PCG-MG, on each neighboring box while adopting the finite element solver from [13] to solve the interface variational problem on the central box. Using this special overlapped box iterative method, we modify SDPB as a finite element and finite difference hybrid method for solving PBE, which is expected to improve the performance of SDPB significantly in terms of CPU time and memory usage.

For a general purpose, in the paper, we describe the overlapped box iterative method for solving a general interface boundary value problem (see (12)), which contains the interface problems of Ψ and p_k as two particular cases. Here the number of boxes has been set as the smallest number of seven to make the overlapped box iterative method to have a fixed rate of convergence for a fixed value of relaxation parameters ω, since the convergence rate of an overlapped box iterative method may decay with the increment of the number of boxes. The central box has been ordered as the 7th box to employ all the most recent iterates from the six neighboring boxes to update the boundary value function of the interface problem on the central box. Moreover, we propose a scheme for this new hybrid PBE solver to adaptively generate Ω, the seven boxes of Ω, an interface fitted tetrahedral mesh of the central box, and a uniform mesh of each neighboring box according to a given PQR file of a biomolecule to be calculated. In this scheme, the same uniform mesh has been applied to the overlapped part of each box so that the data exchange between any two neighboring boxes can be done easily and efficiently.

We completed the program of our hybrid PBE solver in `C`, `Fortran`, and `Python`. The scheme for the generation of Ω, seven boxes, and meshes was programmed in `C`, along with three new parameters for a user to adjust the size of Ω, the size of the central box, and the mesh size h of the uniform mesh. We modified the molecular surface and volumetric mesh generation program package `GAMer` [20] to make it work for a rectangular domain. We then applied it to our `C` program for the generation of an interface fitted mesh of the central box (see Fig. 2). We programmed the PCG-MG method in `Fortran` without storing any mesh data or coefficient matrices of finite difference systems. In the multigrid V-cycle preconditioning, one forward Gauss–Seidel iteration and one backward Gauss–Seidel iteration were used to define the pre-smoother and post-smoother, respectively. The coarsest grid equation was solved simply by the successive over-relaxation (SOR) method. The PCG using the incomplete LU preconditioning (PCG-ILU) from the `PETSc` library [21] was used to solve each system of finite element equations on the central box. Furthermore, to speedup calculations, we wrote `Fortran` subroutines for calculating the values of G at the mesh nodes of Ω, and the values of ∇G at the mesh nodes of the central box. All of the `Fortran` subroutines and `C` programs were converted to `Python` modules by using the Fortran-to-Python interface generator `f2py` (*http://cens.ioc.ee/projects/f2py2e/*) and `SWIG` (*http://www.swig.org*), respectively. By using those modules, the hybrid PBE solver was programmed in `Python` as a part of the software package SDPB.

To validate our new hybrid PBE solver and program, we made numerical experiments on two test problems with analytical solutions: One is an interface test model problem proposed in [17], and the other one is the nonlinear Born Ball model problem used in [13]. Three nested meshes were constructed for these experiments. Numerical results showed that almost three fourths of the absolute error norms between the numerical and analytical solutions were reduced when the mesh size was reduced by half, which well validated our new hybrid method and program package.

We further demonstrated the performance of our new hybrid PBE solver for 12 proteins (the number of atoms up to 11,439) in terms of CPU time and memory usage. Numerical results showed that our new hybrid PBE solver took the same number of modified Newton iterations to satisfy the iteration termination rule as SDPB did. This confirms that the new hybrid PBE solver retains the convergence rate of SDPB. Due to the efficiency of our special overlapped box iterative method, the new hybrid PBE solver was found to improve the performance of SDPB significantly. In these numerical tests, the total CPU time of SDPB was reduced by 55% to 73%. For example, for a protein with PDB ID 2LZX on a mesh with

535,400 vertices, the total CPU time was reduced from about 127 seconds to about 34 seconds on one core of a Mac Pro workstation with 3.7 GHz Quad-Core Intel Xeon E5 processor, which included the mesh generation time.

Finally, as an application, we calculated the electrostatic solvation free energy E for 12 proteins using the new hybrid PBE solver. To demonstrate the numerical behavior of our hybrid method in such a calculation, we constructed five different meshes with the numbers of mesh nodes from about 75,000 to 1.3 million for four proteins. We also did the calculation using other two commonly-used molecular surfaces — solvent-accessible surface (SAS) and solvent-excluded surface (SES), which resulted in two more bumpy interfaces than the one generated from `GAMer` so that PBE became more difficult to solve numerically. In these tests, the values of E were found to be toward a limit with the increment of the number of mesh nodes. These further verified the convergence and robustness of the new hybrid PBE solver. In these tests, the numerical solution of PBE was also found to have significant changes in some areas of a molecular surface when the mesh was refined from a coarse mesh to a fine mesh. This indicates that using a large mesh to solve PBE may be critical since it can reveal some important details of the PBE solution that are hidden in the coarse mesh. Hence, it is important to develop a numerical PBE algorithm and program package that can work well for a large biomolecule on a large mesh. In the future, we plan to further improve the efficiency and capability of our hybrid PBE solver to make it a powerful tool for predicting the electrostatic solvation free energy for a large biomolecule on a large mesh domain.

The remaining part of this paper is organized as follows: In Section 2, we define the PBE model and its solution decomposition. In Section 3, we briefly review the finite element PBE solver used in SDPB. In Section 4, we present our new overlapped box iterative method. In Section 5, we describe the new hybrid PBE solver. Finally, the numerical results are reported in Section 6.

2. PBE and solution decomposition

Let Ω be a sufficiently large bounded domain of $\mathbb{R}^3$ satisfying

$$\Omega = D_p \cup D_s \cup \Gamma,$$

where D_p denotes a solute region hosting a protein biomolecule with n_p atoms, D_s denotes a solvent region, and Γ is the interface between D_p and D_s. Under the implicit solvent approach, both D_p and D_s are treated as continuum media with dielectric constants ϵ_p and ϵ_s, respectively. For a symmetric 1:1 ionic solvent (i.e., a salt solution with sodium (Na^+) and chloride (Cl^-) ions), the electrostatic potential u (in unit k_BT/e_c) can be predicted by the following PBE boundary value problem:

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

where $\mathbf{r}_j$ and z_j are the position and charge number of the jth atom, respectively, g is a boundary function (e.g., g can be set as zero for a sufficiently large domain), $\partial\Omega$ denotes the boundary of Ω, $\delta_{\mathbf{r}_j}$ is the 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 two constants, which are defined by

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

in the SI units and the length unit having been transferred to angstrom (Å). Here, e_c, ϵ_0, k_B, T and N_A are the electron charge, the permittivity of vacuum, the Boltzmann constant, the absolute temperature, and the Avogadro number, respectively, and I_s is the ionic strength in mole/liter. For $T = 298.15$, and $I_s = 0.1$, we can get

$$\alpha = 7042.940010604046, \qquad \bar{\kappa}^2 = 0.8482715968170331. \tag{2}$$

From the PBE solution decomposition proposed in [13], the solution u of (1) can be constructed by

$$u = G + \Psi + \tilde{\Phi}, \tag{3}$$

where G is given by

$$G(\mathbf{r}) = \frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} \frac{z_j}{|\mathbf{r} - \mathbf{r}_j|}, \tag{4}$$

Ψ is a solution of the linear interface boundary value problem

$$\begin{cases} \Delta\Psi(\mathbf{r}) = 0, & \mathbf{r} \in D_p \cup D_s, \\ \Psi(\mathbf{s}^+) = \Psi(\mathbf{s}^-), & \mathbf{s} \in \Gamma, \\ \epsilon_s \frac{\partial \Psi(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial \Psi(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_p - \epsilon_s) \frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \Psi(\mathbf{s}) = g(\mathbf{s}) - G(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{5}$$

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

$$\begin{cases} \Delta\tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta\tilde{\Phi}(\mathbf{r}) + \bar{\kappa}^2 \sinh(\tilde{\Phi} + \Psi + G) = 0, & \mathbf{r} \in D_s, \\ \tilde{\Phi}(\mathbf{s}^+) = \tilde{\Phi}(\mathbf{s}^-), \quad \epsilon_s \frac{\partial\tilde{\Phi}(\mathbf{s}^+)}{\partial\mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial\tilde{\Phi}(\mathbf{s}^-)}{\partial\mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \tilde{\Phi}(\mathbf{s}) = 0, & \mathbf{s} \in \partial\Omega. \end{cases} \tag{6}$$

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

$$\nabla G(\mathbf{r}) = -\frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} z_j \frac{\mathbf{r} - \mathbf{r}_j}{|\mathbf{r} - \mathbf{r}_j|^3}. \tag{7}$$

As shown in [13,22], both Ψ and $\tilde{\Phi}$ are continuous in the whole domain Ω, and twice continuously differentiable within D_p and D_s. Thus, (5) and (6) are well defined, and can be solved numerically by the finite element and finite difference methods. From (3) it also implies that the PBE solution u has singularity points at the atomic positions $\mathbf{r}_j$ for $j = 1, 2, \ldots, n_p$, which causes difficulties in solving equation (1) directly. In fact, to catch the singular behavior of u, a mesh around each singularity point $\mathbf{r}_j$ must be dense enough, which may lead to a finite difference/finite element system with a huge number of unknowns, increasing the cost of computations sharply. In addition, solving (1) directly by a finite element or finite difference method needs to approximate each singular distribution δ_j as a discrete delta function $\delta_h(\mathbf{r} - \mathbf{r}_j)$. This may further reduce the numerical accuracy. Hence, using (3) is critical in solving PBE.

3. Finite element PBE solver

By the solution decomposition (3), an effective finite element PBE solver has been proposed in [13]. Its main part is a modified Newton minimization algorithm for solving (6) for $\tilde{\Phi}$, which we review in this section for clarity.

Let $\mathcal{M}$ denote a finite element function space as a subspace of the usual Sobolev function space $H^1(\Omega)$, and Ψ have been computed on $\mathcal{M}$. To solve (6) by the minimization algorithm, the interface boundary value problem (6) is reformulated as the following minimization problem:

$$J(\tilde{\Phi}) = \min_{v \in \mathcal{M}_0} J(v), \tag{8}$$

where $\mathcal{M}_0 = \{v \in \mathcal{M} \mid v = 0 \text{ on } \partial\Omega\}$, which is a subspace of the Sobolev function space $H_0^1(\Omega)$, and J is defined by

$$J(v) = \frac{1}{2} a(v, v) + \bar{\kappa}^2 \int_{D_s} \cosh(\Psi + G + v) d\mathbf{r}$$

with $a(u, v)$ being a bilinear functional given by

$$a(u, v) = \epsilon_p \int_{D_p} \nabla u \cdot \nabla v d\mathbf{r} + \epsilon_s \int_{D_s} \nabla u \cdot \nabla v d\mathbf{r}.$$

The modified Newton minimization algorithm is then defined as a sequence of iterates, $\{\tilde{\Phi}^{(k)}\}$, as follows:

$$\tilde{\Phi}^{(k+1)} = \tilde{\Phi}^{(k)} + \lambda_k p_k, \quad k = 0, 1, 2, \ldots, \tag{9}$$

where $\tilde{\Phi}^{(0)}$ is an initial guess, λ_k is a step length, and p_k is a search direction satisfying the Newton equation in the variational form: Find $p_k \in \mathcal{M}_0$ such that

$$J''(\tilde{\Phi}^{(k)})(p_k, v) = -J'(\tilde{\Phi}^{(k)})v \quad \forall v \in \mathcal{M}_0, \tag{10}$$

where $J'(\tilde{\Phi})$ is the first Fréchet-derivative of J at $\tilde{\Phi}$, which is a linear continuous functional on $H_0^1(\Omega)$ defined by

$$J'(\tilde{\Phi})v = a(\tilde{\Phi}, v) + \bar{\kappa}^2 \int_{D_s} \sinh(\Psi + G + \tilde{\Phi}) v dx \quad \forall v \in H_0^1(\Omega),$$

and $J''(\tilde{\Phi})$ is the second Fréchet-derivative of J at $\tilde{\Phi}$, which is a bilinear continuous functional on $H_0^1(\Omega)$ defined by

$$J''(\tilde{\Phi})(p, v) = a(p, v) + \bar{\kappa}^2 \int_{D_s} \cosh(\Psi + G + \tilde{\Phi}) p v dx \quad \forall p, v \in H_0^1(\Omega).$$

In the implementation, the initial iterate $\tilde{\Phi}^{(0)}$ can be selected as zero or a solution of a linearized equation of (6) [23]. A upper bound of 85 is set by default for truncating the value of the sum $\Psi + G + \tilde{\Phi}^{(k)}$ to avoid the possible overflow

problem of hyperbolic terms. Each Newton equation of (10) is solved numerically by the PCG-ILU with the absolute and relative residue errors less than a given tolerance (10^{-10} by default).

In the following theorem, we show that (10) can be reformulated as an interface boundary value problem.

Theorem 3.1. *Let $V = H_0^1(\Omega) \cap C^2(D_p) \cap C^2(D_s)$. If $p \in V$, then the variation problem (10) is equivalent to the boundary value problem*

$$-\Delta p(\mathbf{r}) = \Delta\tilde{\Phi}(\mathbf{r}), \quad \mathbf{r} \in D_p, \tag{11a}$$

$$-\epsilon_s \Delta p(\mathbf{r}) + \bar{\kappa}^2 \cosh(\Psi + G + \tilde{\Phi}) p = \epsilon_s \Delta\tilde{\Phi} - \bar{\kappa}^2 \sinh(\Psi + G + \tilde{\Phi}), \quad \mathbf{r} \in D_s, \tag{11b}$$

$$p(\mathbf{s}^+) = p(\mathbf{s}^-),\ \epsilon_s \frac{\partial p(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} - \epsilon_p \frac{\partial p(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial \tilde{\Phi}(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} - \epsilon_s \frac{\partial \tilde{\Phi}(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})}, \quad \mathbf{s} \in \Gamma, \tag{11c}$$

$$p(\mathbf{s}) = 0, \quad \mathbf{s} \in \partial\Omega, \tag{11d}$$

where $\tilde{\Phi}$ is a given function of V, Ψ is a given solution of (5), and G is defined in (4).

Proof. We only show the derivation of (11) from the variational form (10) since the proof of the converse is easy. For any $v \in H_0^1(\Omega)$ satisfying $v = 0$ on D_s and $v \in \mathbb{C}_0^\infty(D_p)$, from (10) we can get

$$\int_{D_p} (\Delta p + \Delta\tilde{\Phi}) v d\mathbf{r} = 0 \quad \forall v \in \mathbb{C}_0^\infty(D_p),$$

from which it implies equation (11a).

Next, for any $v \in H_0^1(\Omega)$ satisfying $v = 0$ on D_p and $v \in \mathbb{C}_0^\infty(D_s)$, (10) can be reduced to the form

$$\epsilon_s \int_{D_s} \nabla p \cdot \nabla v d\mathbf{r} + \bar{\kappa}^2 \int_{D_s} \cosh(u) p v d\mathbf{r} = -\epsilon_s \int_{D_s} \nabla\tilde{\Phi} \cdot \nabla v d\mathbf{r} - \bar{\kappa}^2 \int_{D_s} \sinh(u) v d\mathbf{r},$$

where $u = G + \Psi + \tilde{\Phi}$. By Green's identity, the above equality can be reformulated as

$$-\epsilon_s \int_{D_s} \Delta p v d\mathbf{r} + \bar{\kappa}^2 \int_{D_s} \cosh(u) p v d\mathbf{r} = \epsilon_s \int_{D_s} \Delta\tilde{\Phi} v d\mathbf{r} - \bar{\kappa}^2 \int_{D_s} \sinh(u) v d\mathbf{r},$$

from which we can obtain equation (11b).

Furthermore, applying the Green's identity to the two terms of $a(\cdot,\cdot)$ for $v \in H_0^1(\Omega)$ in D_p and D_s, respectively, we can reformulate (10) as

$$\begin{aligned} &\int_\Gamma \left[\epsilon_p \frac{\partial p(\mathbf{s}^-)}{\partial \mathbf{n}} - \epsilon_s \frac{\partial p(\mathbf{s}^+)}{\partial \mathbf{n}} \right] v d\mathbf{s} - \epsilon_p \int_{D_p} \Delta p v d\mathbf{r} - \int_{D_s} \left[\epsilon_s \Delta p - \bar{\kappa}^2 \cosh(u) p \right] v d\mathbf{r} \\ &= \int_\Gamma \left[\epsilon_s \frac{\partial \tilde{\Phi}(\mathbf{s}^+)}{\partial \mathbf{n}} - \epsilon_p \frac{\partial \tilde{\Phi}(\mathbf{s}^-)}{\partial \mathbf{n}} \right] v d\mathbf{s} + \epsilon_p \int_{D_p} \Delta\tilde{\Phi} v d\mathbf{r} + \int_{D_s} \left[\epsilon_s \Delta\tilde{\Phi} - \bar{\kappa}^2 \sinh(u) \right] v d\mathbf{r}. \end{aligned}$$

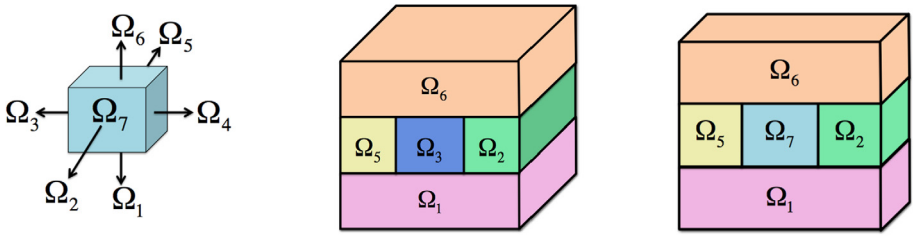
Applying (11a) and (11b) to the above identity leads to the interface condition (11c). The boundary condition (11d) is natural because of $p \in V$. This completes the proof. □

With Theorem 3.1, we can generate the search direction p_k by using a finite difference solver too. This motivates us to develop a hybrid method to combine the advantages of finite element and finite difference methods together in the numerical solution of PBE.

4. An overlapped box iterative method

The performance of the finite element PBE solver SDPB relies on that of an iterative scheme for solving the two linear interface boundary problems (5) and (11). To improve it, in this section, we present an overlapped box iterative method for computing Ψ and p_k. Clearly, both (5) and (11) are two special cases of the following interface boundary value problem:

$$\begin{cases} -\epsilon_p \Delta w(\mathbf{r}) = f_p(\mathbf{r}), & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta w(\mathbf{r}) + \beta(\mathbf{r}) w = f_s(\mathbf{r}), & \mathbf{r} \in D_s, \\ w(\mathbf{s}^+) = w(\mathbf{s}^-), \quad \epsilon_s \frac{\partial w(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial w(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} + \zeta(\mathbf{s}), & \mathbf{s} \in \Gamma, \\ w(\mathbf{s}) = g(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{12}$$



(a) Ordering of seven boxes (b) Left view of the box partition (c) Cross-section view

Fig. 1. Our special partition of the domain Ω into seven overlapped boxes Ω_i for $i = 1$ to 7. Plot (a) illustrates the positions and ordering numbers of the seven boxes. Plots (*b*) displays a view of the box partition from the left hand side. Plot (*c*) gives a view of the central box Ω_7 and its neighboring boxes on the cross-section generated by the half cutting. Here the overlapping parts are not exposed in Plots (b) and (c) for clarity.

where f_p, f_s, β, ζ and g are given functions, and β is nonnegative. For a general purpose, we describe the overlapped box iterative method for solving (12).

Let Ω be a cubic domain. For a given biomolecule region D_p, we select a cubic box D, and then partition Ω into seven overlapped boxes Ω_i for $i = 1, 2, \cdots, 7$ with Ω_7 being the central box satisfying that

$$D_p \subset D \subset \Omega_7, \qquad \Omega \setminus D = \cup_{j=1}^{6} \Omega_j.$$

The position and ordering index of each box are illustrated in Fig. 1. Clearly, $\Omega_7 \setminus D$ gives the overlapped part of Ω_7 with its six neighboring boxes.

Following the Schwartz domain decomposition scheme [24], we define the overlapped box iterative method for solving (12) by

$$w_i^{(k)} = (1-\omega) w_i^{(k-1)} + \omega \bar{w}_i \quad \text{on } \Omega_i \text{ for } i = 1, 2, \cdots, 7, \tag{13}$$

where $k = 1, 2, \cdots$, $w_i^{(0)}$ is an initial iterate, $\omega \in (1, 2)$ is the over-relaxation parameter, $\bar{w}_i$ with $i = 1$ to 6 denotes a solution of the elliptic boundary value problem:

$$\begin{cases} -\epsilon_s \Delta w(\mathbf{r}) + \beta(\mathbf{r}) w = f_s(\mathbf{r}) & \text{in } \Omega_i, \\ w(\mathbf{s}) = w_j^{(k-1)}(\mathbf{s}) & \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = i+1 \text{ to } 7, \\ w(\mathbf{s}) = w_j^{(k)}(\mathbf{s}) & \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = 1 \text{ to } i-1, \\ w(\mathbf{s}) = g(\mathbf{s}) & \text{on } \partial\Omega_i \cap \partial\Omega, \end{cases} \tag{14}$$

and $\bar{w}_7$ is a solution of the elliptic interface boundary value problem:

$$\begin{cases} -\epsilon_p \Delta w(\mathbf{r}) = f_p(\mathbf{r}) & \text{in } D_p, \\ -\epsilon_s \Delta w(\mathbf{r}) + \beta(\mathbf{r}) w = f_s(\mathbf{r}) & \text{in } \Omega_7 \cap D_s, \\ w(\mathbf{s}^+) = w(\mathbf{s}^-), \quad \epsilon_s \frac{\partial w(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial w(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} + \zeta(\mathbf{s}) & \text{on } \Gamma, \\ w(\mathbf{s}) = w_j^{(k)}(\mathbf{s}) & \text{on } \partial\Omega_7 \cap \Omega_j \text{ for } j = 1 \text{ to } 6. \end{cases} \tag{15}$$

Here $\partial\Omega_i$ denotes the boundary of Ω_i. Note that the updates $w_i^{(k)}$ from the six neighboring boxes have been employed in the construction of boundary condition on $\partial\Omega_7$ to yield a good boundary value function. In numerical test, we use the following simple termination rule:

$$\sqrt{\sum_{i=1}^{7} ||w_i^{(k)} - w_i^{(k-1)}||^2} \leq \epsilon, \tag{16}$$

where $||\cdot||$ is the Euclidean norm, and ϵ is set as 10^{-7} by default.

5. The finite element and finite difference hybrid PBE solver

In principle, different numerical methods can be applied to different boxes. Specially, in this section, we construct an efficient finite element scheme for solving the interface boundary problem (15) and a fast finite difference scheme for solving the boundary value problem (14). Moreover, we obtain a scheme and a program for generating the domain Ω, the seven overlapped boxes $\{\Omega_i\}_{i=1}^7$, an interface fitted tetrahedral mesh of Ω_7, and a uniform finite difference mesh of Ω_i for $i = 1$ to 6. Applying this special overlapped box method to the numerical calculation of Ψ and p_k, we modify the finite element PBE solver SDPB of [13] as a finite element and finite difference hybrid PBE solver.

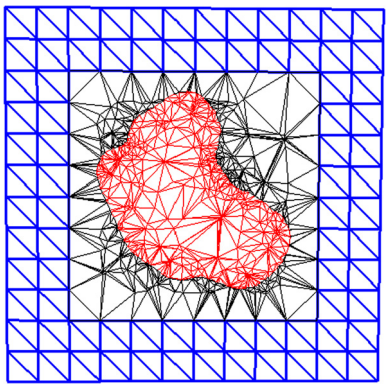


Fig. 2. A cross section of a tetrahedral mesh of Ω_7 on the xy-coordinate plane for a protein with PDB ID 2LZX. D_p is colored in red, and the overlapped part $\Omega_7 \setminus D$ in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.1. Domain partition and mesh generation scheme and program

Let a cubic region of D be given in the form $D = \prod_{i=1}^{3}(a_i, b_i)$ with each side length $b_i - a_i = L$ for $i = 1, 2, 3$. We define the mesh size h and two other mesh parameters τ and η satisfying $\tau < \eta$ by

$$h = L/2^n, \quad \tau = 2^m h, \quad \eta = \mu L/2, \tag{17}$$

where n, m, and μ are positive integers to be input by a user ($n = 3$, $m = 2$, and $\mu = 4$ by default) according to the memory limit of a computer and a demanded accuracy of PBE numerical solution. Using them, we construct a cubic domain of Ω on which PBE is to be calculated and the seven boxes Ω_i as follows:

$$\Omega = \prod_{i=1}^{3}(a_i - \eta, b_i + \eta), \qquad \Omega_7 = \prod_{i=1}^{3}(a_i - \tau, b_i + \tau),$$
$$\Omega_1 = (a_1 - \eta, b_1 + \eta) \times (a_2 - \eta, b_2 + \eta) \times (a_3 - \eta, a_3),$$
$$\Omega_2 = (a_1 - \eta, b_1 + \eta) \times (a_2 - \eta, a_2) \times (a_3 - \tau, b_3 + \tau),$$
$$\Omega_3 = (a_1 - \eta, a_1) \times (a_2 - \tau, b_2 + \tau) \times (a_3 - \tau, b_3 + \tau),$$
$$\Omega_4 = (b_1, b_1 + \eta) \times (a_2 - \eta, b_2 + \tau) \times (a_3 - \tau, b_3 + \tau),$$
$$\Omega_5 = (a_1 - \eta, b_1 + \eta) \times (b_2, b_2 + \eta) \times (a_3 - \tau, b_3 + \tau),$$
$$\Omega_6 = (a_1 - \eta, b_1 + \eta) \times (a_2 - \eta, b_2 + \eta) \times (b_3, b_3 + \eta).$$

We also construct a uniform finite difference mesh of $\Omega \setminus D$ with mesh nodes (x_i, y_j, z_k) being defined as

$$x_i = a_1 - \eta + ih, \quad y_j = a_2 - \eta + jh, \quad z_k = a_3 - \eta + kh,$$

from which a uniform finite difference mesh of each box Ω_i with $i = 1$ to 6 is produced. Furthermore, we construct a hybrid mesh of Ω_7 — an unstructured tetrahedral mesh on D to well approximate the interface Γ and a uniform tetrahedral mesh on the overlapped part $\Omega_7 \setminus D$ with the six neighboring boxes. In particular, the uniform mesh is simply made from the uniform finite difference mesh located on the overlapped part $\Omega_7 \setminus D$ of the central box Ω_7 through cutting each cubic grid cell into six tetrahedra. Clearly, any two neighboring boxes have been set to share the same mesh nodes on their overlapped parts. Hence, the data exchange between them can be done simply and efficiently.

We programmed the above scheme in `C`. In this program, a PQR file of a biomolecule is required as an input file. It can be generated from a PDB file of the biomolecule, which can be downloaded from the Protein Data Bank (PDB) (*http://www.rcsb.org/*), by the program tool `PDB2PQR` [25]. We made a modified version of the molecular surface and volumetric mesh generation program package `GAMer` [20] to enable it to work for both a spherical domain and a rectangular box domain. We then used it to generate a cubic region of D and an interface fitted tetrahedral mesh of Ω_7. See Fig. 2 for a mesh of Ω_7 generated from this program.

5.2. The finite element solver

Let $\mathcal{M}_7 \subset H^1(\Omega_7)$ be a linear finite element function space based on a tetrahedral mesh of Ω_7. We reformulate the interface boundary value problem (15) into the following finite element variational problem:

Find $\bar{w}_7 \in \mathcal{M}_7$ with $\bar{w}_7 = w_i^{(k)}$ on $\partial\Omega_7 \cap \Omega_i$ for $i = 1$ to 6 such that

$$b(\bar{w}_7, v) = l(v) \quad \forall v \in \mathcal{M}_{7,0}, \tag{18}$$

where $\mathcal{M}_{7,0} = \{v \in \mathcal{M}_7 \mid v = 0 \text{ on } \partial\Omega_7\}$, $b(w, v)$ is a symmetric bilinear functional defined by

$$b(w, v) = \epsilon_p \int_{D_p} \nabla w \cdot \nabla v d\mathbf{r} + \epsilon_s \int_{D_s \cap \Omega_7} \nabla w \cdot \nabla v d\mathbf{r} + \int_{D_s \cap \Omega_7} \beta(\mathbf{r}) w(\mathbf{r}) v(\mathbf{r}) d\mathbf{r},$$

and $l(v)$ is a linear functional defined by

$$l(v) = \int_{\Gamma} \zeta(\mathbf{s}) v(\mathbf{s}) d\mathbf{s} + \int_{D_s \cap \Omega_7} f_s(\mathbf{r}) v(\mathbf{r}) d\mathbf{r} + \int_{D_p} f_p(\mathbf{r}) v(\mathbf{r}) d\mathbf{r}.$$

Based on the finite element library `DOLFIN` [26], we wrote a finite element program in `Python` for solving (18) approximately using the PCG-ILU from the `PETSc` library [21]. Here both the relative and absolute residue error parameters are set as 10^{-8}.

5.3. The finite difference solver

Based on a uniform mesh of each box Ω_ν with mesh size h and $\nu = 1$ to 6, we approximate the boundary value problem (14) as a system of second-order centered finite difference equations as follows: For $i = 1$ to $N_{\nu,1} - 1$, $j = 1$ to $N_{\nu,2} - 1$, and $k = 1$ to $N_{\nu,3} - 1$,

$$\frac{\epsilon_s}{h^2}(6w_{i,j,k} - w_{i+1,j,k} - w_{i-1,j,k} - w_{i,j+1,k} - w_{i,j-1,k} - w_{i,j,k+1} - w_{i,j,k-1}) + \beta_{i,j,k} w_{i,j,k} = f_{s,i,j,k}, \tag{19}$$

where $N_{\nu,1}$, $N_{\nu,2}$, and $N_{\nu,3}$ denote the numbers of partitions on the x, y, z-axes, respectively, $w_{i,j,k}$ denotes a numerical value of w at the mesh node (x_i, y_j, z_k), $f_{s,i,j,k} = f_s(x_i, y_j, z_k)$, $\beta_{i,j,k} = \beta(x_i, y_j, z_k)$, and the boundary values are set at $i = 0$, $N_{\nu,1}$; $j = 0$, $N_{\nu,2}$; or $k = 0$, $N_{\nu,3}$. The finite difference system (19) is clearly symmetric positive definite. It can be optimally solved by PCG-MG in the sense that the total number of flops required to solve (19) is proportional to the total number of unknowns. In our multigrid V-cycle method, the pre and post smoothers are defined by one forward and one backward Gauss–Seidel iteration, respectively, the prolongation operator is the trilinear interpolation, and its adjoint is set as the restriction operator (i.e., the standard full weight operator [27]). Since the coarsest mesh only has a small number of mesh nodes, the coarsest gird equation is simply solved by the SOR method with its residual vector has norm less than 10^{-10}. The PCG-MG iteration does not stop until the relative residual norm of the finite difference system (19) is less than or equal to 10^{-8} by default.

We programmed the above PCG-MG in `Fortran` without using any extra array to store mesh data or the coefficient matrix A of the linear system (19). In the program, the values of a function on the uniform mesh of each finite difference box were stored by a three dimensional array. Thus, the product of the coefficient matrix A with a vector of d, which is the only operation of PCG that is related to A, is stored as a three dimensional array, Ad, and implemented by the following triple loop: For $i = 1$ to $N_{\nu,1} - 1$, $j = 1$ to $N_{\nu,2} - 1$, and $k = 1$ to $N_{\nu,3} - 1$,

$$(Ad)_{i,j,k} = \frac{\epsilon_s}{h^2}(6d_{i,j,k} - d_{i+1,j,k} - d_{i-1,j,k} - d_{i,j+1,k} - d_{i,j-1,k} - d_{i,j,k+1} - d_{i,j,k-1}) + \beta_{i,j,k} d_{i,j,k},$$

where the values of $d_{i,j,k}$ on the boundary mesh nodes have been set as zero.

5.4. New hybrid PBE solver

Our new finite element and finite difference hybrid PBE solver is a modification of the finite element PBE solver SDPB of [13]. For clarity, we describe it in Algorithm 1.

Algorithm 1. Let our special overlapped box iterative method be defined in (13) with the PCG-MG algorithm for solving a finite difference system of (14) on a uniform mesh of box Ω_i for $i = 1$ to 6 and the PCG-ILU algorithm for solving a finite element system of (15) on an interface fitted tetrahedral mesh of the central box Ω_7. The finite element and finite difference hybrid PBE solver is defined in five steps:

Step 1. Generate D, Ω, Ω_i for $i = 1$ to 7, a uniform mesh of $\Omega \setminus D$, and an interface fitted tetrahedral mesh of Ω_7 according to an input PQR file of a biomolecule to be calculated.
Step 2. Calculate the mesh node values of G on Ω and ∇G on Ω_7 according to (4) and (7), respectively.
Step 3. Solve the linear interface problem (5) for Ψ by our special overlapped box iterative method.
Step 4. Solve the nonlinear interface problem (6) for $\tilde{\Phi}$ by the modified Newton minimization scheme defined in (9) with p_k being calculated by our special overlapped box iterative method and the iteration termination rule being given by $\|\tilde{\Phi}^{(k+1)} - \tilde{\Phi}^{(k)}\| \le \epsilon$. Here $\epsilon = 10^{-7}$ by default.
Step 5. Construct a numerical solution u of PBE by the solution decomposition $u = G + \Psi + \tilde{\Phi}$.

To speedup the solution process, we programmed the second step of Algorithm 1 as a `Fortran` subroutine. Using the Fortran-to-Python interface generator `f2py` (*http://cens.ioc.ee/projects/f2py2e/*), we converted this `Fortran` subroutine and ones for PCG-MG to `Python` modules. We also converted the `C` programs for domain and mesh generation into a `Python` module by `SWIG` (*http://www.swig.org*). With these modules, we programmed Algorithm 1 in `Python`, and added it to SDPB as another PBE solver. The parameters for defining our hybrid method were also added to the parameter file of the software package SDPB, including the one for selecting the new hybrid PBE solver.

6. Numerical tests

In this section, we report the numerical experiments we made using our new `Python` program of Algorithm 1. In these numerical tests, two test models with analytical solutions were used to validate our special overlapped box iterative method and hybrid PBE solver, respectively. Performance comparisons were then done between the new hybrid program and SDPB for 12 proteins. As an application of the hybrid PBE solver, we finally calculated the electrostatic solvation free energy, and tested the numerical behavior of the hybrid PBE solver in terms of different mesh sizes. For simplicity, all the numerical tests were done by using $\epsilon_p = 2.0$, $\epsilon_s = 80.0$, $I_s = 0.1$, and the default values of other parameters (except the parameters m, n, μ of (17)) on one processor of a Mac Pro Workstation with the 3.7 GHz Quad-Core Intel Xeon E5 and 64 GB memory.

6.1. Validation tests for the overlapped box iterative method

To validate our overlapped box iterative method, we consider the following interface boundary value problem

$$\begin{cases} \Delta\Psi(\mathbf{r}) = 0 & \text{in } D_p, \\ -\epsilon_s \Delta\Psi(\mathbf{r}) = f_s(\mathbf{r}) & \text{in } D_s, \\ \Psi(\mathbf{s}^-) = \Psi(\mathbf{s}^+),\ \epsilon_p \frac{\partial \Psi(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_s \frac{\partial \Psi(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_s - \epsilon_p)\frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} & \text{on } \Gamma, \\ \Psi(\mathbf{s}) = U(\mathbf{s}) & \text{on } \partial\Omega, \end{cases} \tag{20}$$

where $D_p = \{\mathbf{r} \mid |\mathbf{r}| < a\}$, $\Gamma = \{\mathbf{r} \mid |\mathbf{r}| = a\}$, $\Omega = (-A, A)^3$ is a cubic domain, $D_s = \Omega - D_p - \Gamma$, G is given in (4), f_s is given by

$$f_s(\mathbf{r}) = \frac{\alpha(\epsilon_p - \epsilon_s)}{4\pi a^2 \epsilon_p} \sum_{j=1}^{n_p} z_j \left[\left(\frac{7|\mathbf{r}|^2 - 5\mathbf{r}\cdot\mathbf{r}_j}{|\mathbf{r} - \mathbf{r}_j|^3} - 6\frac{(|\mathbf{r}|^2 - \mathbf{r}\cdot\mathbf{r}_j)^2}{|\mathbf{r} - \mathbf{r}_j|^5} \right) \cos\left(\frac{|\mathbf{r}|^2 - a^2}{a^2} \right) - \frac{2|\mathbf{r}|^2(|\mathbf{r}|^2 - \mathbf{r}\cdot\mathbf{r}_j)}{a^2|\mathbf{r} - \mathbf{r}_j|^3} \sin\left(\frac{|\mathbf{r}|^2 - a^2}{a^2} \right) \right],$$

and U is given by

$$U(\mathbf{r}) = \frac{\alpha(\epsilon_s - \epsilon_p)}{8\pi\epsilon_p\epsilon_s} \sin\left(\frac{|\mathbf{r}|^2 - a^2}{a^2} \right) \sum_{j=1}^{n_p} z_j \frac{(\mathbf{r} - \mathbf{r}_j)\cdot\mathbf{r}}{|\mathbf{r} - \mathbf{r}_j|^3}.$$

The analytical solution of (20) is that $\Psi(\mathbf{r}) = 0$ for $\mathbf{r} \in D_p$ and $\Psi(\mathbf{r}) = U(\mathbf{r})$ for $\mathbf{r} \in D_s$ [17].

In numerical tests, we set $\alpha = 1.0$, $a = 1$, and $A = 3$. The charge numbers z_j and atomic positions $\mathbf{r}_j$ were obtained from a PQR file of a protein with the PDB ID 2LZX, which has 488 atoms (i.e., $n_p = 488$). Here we divided $\mathbf{r}_j$ by 17 to rescale it into the unit spherical region D_p. The structure of this protein was preserved within the unit ball.

Furthermore, we set $D = (-1.5, 1.5)^3$, $\tau = 0.375$, and $\eta = 1.5$, from which we obtained the seven boxes Ω_i for $i = 1$ to 7. Next, we generated three nested meshes of Ω, denoted by Ω_{h_j} for $j = 1, 2, 3$, whose total numbers of mesh nodes were found to be 7515, 57,515, 448,773, respectively. In the region $\Omega \setminus D$, we got the mesh sizes of the three uniform meshes as follows:

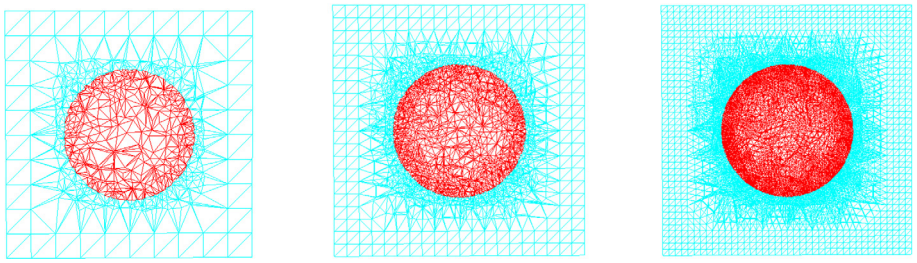
$$h_1 = 0.375, \quad h_2 = h_1/2 = 0.1875, \quad \text{and} \quad h_3 = h_2/2 = 0.09375.$$

The corresponding three unstructured tetrahedral meshes of Ω_7, denoted by Ω_{7,h_j} for $j = 1, 2, 3$, had 3933, 30,839, and 243,069 vertices, respectively. The mesh size h_j of the unstructured mesh Ω_{7,h_j} is defined as the largest edge length among all the tetrahedra, which was found to be $h_1 = 0.7389$, $h_2 = 0.4058$, and $h_3 = 0.2307$. Fig. 3 displays cross section views of these three nested tetrahedral meshes. The numerical results were reported in Table 1. Here the absolute error E_a between the analytical solution Ψ and numerical solution Ψ_h is defined by

$$E_a(\Psi) = \sqrt{\int_{\Omega_7} |\Psi(\mathbf{r}) - \Psi_h(\mathbf{r})|^2 d\mathbf{r} + \sum_{\mathbf{r}^j \in (\Omega - \Omega_7)_h} h^3 [\Psi(\mathbf{r}^j) - \Psi_h(\mathbf{r}^j)]^2}, \tag{21}$$

where $\mathbf{r}^j$ denotes the jth mesh node of a uniform finite difference mesh on $\Omega \setminus \Omega_7$.

From Table 1 it can be seen that the absolute error E_a of our special overlapped box iterative method was reduced about three fourths when the mesh size h was reduced by a half, resulting in a second order of convergence rate almost.



(a) Ω_{7,h_1} with 3,933 vertices (b) Ω_{7,h_2} with 30,839 vertices (c) Ω_{7,h_3} with 243,069 vertices

Fig. 3. Cross section views of the three nested tetrahedral meshes Ω_{7,h_j} for $j = 1, 2, 3$ of the central box Ω_7. Here the spherical protein region D_p is colored in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Convergence of our special overlapped box iterative method for solving the test model (20).

Mesh of Ω	Number of mesh nodes	Absolute error E_a of (21)	Order of convergence
Ω_{h_1}	7515	0.3248	
Ω_{h_2}	57,515	0.0791	2.037
Ω_{h_3}	448,773	0.0199	1.991

These test results confirm the convergence rate of our new overlapped box iterative method as what is expected from the mathematical theory. Hence, our hybrid program is well validated.

6.2. Validation tests for the hybrid PBE solver

To validate the hybrid PBE solver, we made numerical experiments on the following nonlinear Born Ball model:

$$\begin{cases} -\epsilon_p \Delta u = \alpha z \delta, & \text{in } D_p \\ -\epsilon_s \Delta u + \bar{\kappa}^2 \sinh u = f_s, & \text{in } D_s \\ u(s^+) = u(s^-), \quad \epsilon_s \frac{\partial u(s^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial u(s^-)}{\partial \mathbf{n}(\mathbf{s})} & \text{on } \Gamma \\ u(\mathbf{r}) = \frac{\alpha z}{4\pi\epsilon_s |\mathbf{r}|} & \text{on } \partial\Omega, \end{cases} \tag{22}$$

where $\alpha = 7042.94$, $f_s(\mathbf{r}) = \bar{\kappa}^2 \sinh(\frac{\alpha z}{4\pi\epsilon_s|\mathbf{r}|})$, $D_p = \{\mathbf{r} \mid |\mathbf{r}| < 1\}$, $\Gamma = \{\mathbf{r} \mid |\mathbf{r}| = 1\}$, $\Omega = (-6, 6)^3$, and $D_s = \Omega - D_p - \Gamma$. The analytical solution of (22) is given by

$$u = \begin{cases} \frac{\alpha z}{4\pi a}(\frac{1}{\epsilon_s} - \frac{1}{\epsilon_p}) + \frac{\alpha z}{4\pi\epsilon_p|\mathbf{r}|} & \text{in } D_p, \\ \frac{\alpha z}{4\pi\epsilon_s|\mathbf{r}|} & \text{in } D_s. \end{cases} \tag{23}$$

In the numerical tests, we set $D = (-2, 2)^3$, $\tau = 1$, $\eta = 4$, and $z = 1$, which gave $\Omega_7 = (-3, 3)^3$. The over-relaxation parameter ω of our overlapped box iterative method was set as 1.275 and 1.21 in solving (5) for Ψ and (11) for p_k, respectively. The initial iterate $\tilde{\Phi}^{(0)}$ was selected as a numerical solution of a new linearized equation of (6) [23]. Using the mesh sizes $h_1 = 0.25$, $h_2 = h_1/2$, and $h_3 = h_2/2$, we constructed three nested meshes Ω_{h_i} with the total numbers of mesh nodes were 120,887, 940,247, 7,412,989, respectively, including 18,863, 145,223, 1,136,605 mesh nodes from the meshes of Ω_7.

Table 2 reports the average errors of the numerical solutions u_h, the numbers of the modified Newton iterations, and the average numbers of PCG-MG iterations on each finite difference box, PCG-ILU iterations on the finite element box Ω_7, and overlapped box iterations on the domain Ω. Each average number of iterations was calculated by

$$\frac{1}{K}\sum_{j=1}^{K} n_j, \tag{24}$$

where n_j denotes the total number of iterations determined by an iteration termination rule in solving the jth linear system, and K is the total number of linear systems solved to yield a numerical solution of PBE.

From Table 2 it can be seen that the errors were reduced from 7.55×10^{-4} to 7.66×10^{-5} while the number of modified Newton iterations was only up to 10 as the mesh size h was decreased from 0.25 to 0.0625. These results validated the numerical solutions generated from our hybrid PBE program and indicated that our modified Newton iterations retained the fast rate of convergence. As what the multigrid theory claims, the average number of PCG-MG iterations was found to be around 7 in three different mesh sizes, confirming our PCG-MG method to have a convergence rate independent of mesh size h. Furthermore, PCG-ILU for solving each system of finite element equations on the central box Ω_7 was found to be very efficient. In fact, we also used the PCG using an algebraic multigrid preconditioning from the PETSc library to solve

Table 2
Convergence performance of our finite element and finite difference hybrid PBE solver for the nonlinear Born test model (22). Here the number of iterations (iter.) for PCG-MG, PCG-ILU, and the box iterative method is an average number defined in (24).

Mesh of Ω	Number N_h of mesh nodes	Average error $\|u-u_h\|_{l^2(\Omega)}/N_h$	PCG-MG iter. on Ω_i $(i\neq 7)$	PCG-ILU iter. on Ω_7	Box method iter. on Ω	Modified Newton iter. for finding $\tilde{\Phi}$
Ω_{h_1}	120,887	7.55×10^{-4}	$7.16\approx7$	$7.3\approx7$	$12.2\approx12$	7
Ω_{h_2}	940,247	1.69×10^{-4}	$7.17\approx7$	$12.5\approx13$	$11.7\approx12$	10
Ω_{h_3}	7,412,989	7.66×10^{-5}	$7.33\approx7$	$22.5\approx23$	$12.4\approx12$	9

Table 3
Some basic information on the 12 proteins used for numerical tests. Here, N_{Ω_7} and N_Ω denote the numbers of mesh nodes on Ω_7 and Ω, respectively, n_p is the number of atoms, and $\rho = 100N_{\Omega_7}/N_\Omega\%$.

Index	PDB ID	n_p	Cubic region $D=(a_1,b_1;a_2,b_2;a_3,b_3)$	N_{Ω_7}	N_Ω	Percentage ρ
1	2LZX	488	(−16.5, 17.0; −16.1, 17.4; −16.1, 17.4)	39,896	535,400	7.5%
2	1AJJ	513	(−8.8, 28.9; −11.4, 26.3; −15.7, 21.9)	42,817	538,321	8.0%
3	1FXD	811	(−12.5, 31.4; −24.8, 19.1; −13.6, 30.3)	45,345	540,849	8.4%
4	1HPT	852	(−14.8, 31.7; −13.7, 32.9; −5.7, 40.8)	47,716	543,220	8.8%
5	4PTI	892	(−7.8, 38.4; −2.3, 43.9; −18.6, 27.6)	45,825	541,329	8.5%
6	1SVR	1433	(−24.8, 27.3; −30.4, 21.7; −24.5, 27.6)	54,666	550,170	9.9%
7	1A63	2065	(−29.9, 37.0; −34.3, 32.6; −33.9, 33.0)	62,506	558,010	11.2%
8	1CID	2783	(−47.2, 27.9; 0.6, 75.8; −4.8, 70.3)	62,870	558,374	11.3%
9	1A7M	2803	(−43.3, 40.0; −41.0, 42.3; −40.6, 42.8)	68,415	563,919	12.1%
10	2AQ5	6024	(−21.5, 50.6; 17.0, 89.1; −15.0, 57.1)	82,317	577,821	14.2%
11	1F6W	8243	(−38.7, 47.7; −37.9, 48.5; −21.1, 65.3)	79,182	574,686	13.8%
12	1C4K	11,439	(19.2, 135.1; −26.9, 88.9; −16.5, 99.4)	77,607	573,111	13.5%

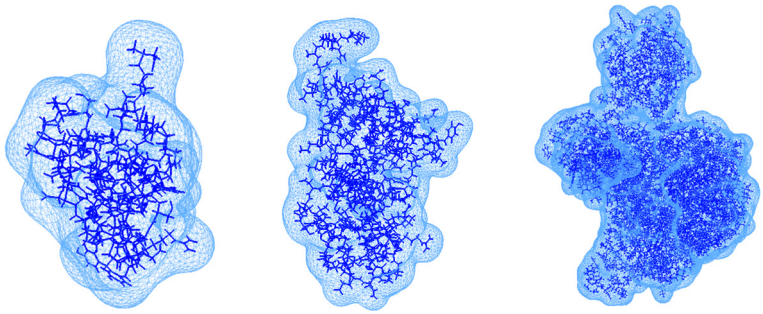


Fig. 4. Molecular surfaces generated from the Gaussian blurring approach [20] for three proteins (PDB ID 1HPT, 1A63, and 1C4K) with their molecular structures given in solid lines.

each linear system of Ω_7. It was found to take more CPU time than PCG-ILU although its number of iterations was smaller and independent of h.

6.3. Performance for protein cases

We made numerical tests on 12 proteins to demonstrate the performance of our hybrid PBE program in terms of CPU time. The PDB files of these proteins were downloaded from the PDB website and then converted to PQR files by the tool `PDB2PQR` with the `CHARMM` force field. From our program a cubic region D was generated for each protein as listed in Table 3.

We then set $\mu=4$, $m=3$, and $n=4$ to get Ω, Ω_i for $i=1$ to 7, the mesh size h, the finite difference mesh of $\Omega\setminus D$, and the interface fitted tetrahedral mesh of Ω_7. Here, each interface Γ was generated by our modified version of `GAMer` based on the Gaussian blurring approach [20]. It was found to be close to a molecular structure as demonstrated in Fig. 4 for three protein molecules (1HPT, 1A63, and 1C4K). Additionally, the over-relaxation parameter ω of our overlapped box iterative method was set to 1.215 and 1.015 in solving (5) for Ψ and (11) for p_k, respectively. The boundary value function g and an initial iterate $\tilde{\Phi}^{(0)}$ were set as zero for simplicity. As comparison, we used SDPB to repeat all the calculations based on the same meshes used by the hybrid solver.

Table 4 compares the performance of the new hybrid PBE solver (Hybrid) with that of SDPB in terms of CPU time. In the case of Hybrid, the construction of a finite element (FE) mesh and the calculation of ∇G were done on Ω_7 only. Hence, they took less CPU time than the case of SDPB. Due to the efficiency of the overlapped box iterative method, the total CPU time was sharply reduced by about 55% to 73%. In the tests of these 12 proteins, the same numbers of modified Newton iterations were found for both Hybrid and SDPB, which were 17, 21, 19, 14, 15, 21, 46, 17, 15, 26, 25, and 46, respectively, indicating

Table 4
A comparison of the performance of our new hybrid PBE solver (Hybrid) with that of the finite element PBE solver SDPB proposed in [13] in computer CPU runtime measured in seconds.

PDB ID (# mesh nodes)	Find FE mesh		Find G & ∇G		Find Ψ		Find $\tilde{\Phi}$		Total time	
	Hybrid	SDPB	Hybrid	SDPB	Hybrid	SDPB	Hybrid	SDPB	Hybrid	SDPB
2LZX (535,400)	5.15	10.68	2.27	4.79	4.67	13.65	22.36	97.31	34.71	127.24
1AJJ (538,321)	5.59	11.03	2.43	5.08	4.97	13.7	27.92	123.47	41.19	154.09
1FXD (540,849)	6.05	11.49	3.85	8.05	5.19	13.81	26.26	107.23	41.71	141.39
1HPT (543,220)	7.01	12.49	4.06	8.48	4.86	14.36	18.16	83.18	34.48	119.33
4PTI (541,329)	6.68	12.34	4.22	9.00	5.13	14.05	19.00	85.42	35.43	121.66
1SVR (550,170)	9.18	14.72	7.02	14.55	5.78	14.50	30.60	122.82	53.14	167.44
1A63 (558,010)	12.47	18.31	10.32	21.28	6.45	14.36	83.54	293.54	113.56	347.85
1CID (558,374)	12.72	18.41	13.96	28.75	6.39	15.14	26.38	101.7	60.42	164.87
1A7M (563,919)	12.91	18.45	14.33	28.92	7.03	15.43	24.42	91.76	59.67	155.41
2AQ5 (577,821)	23.87	29.34	32.24	63.57	8.99	15.84	56.1	163.71	123.18	273.35
1F6W (574,686)	27.19	32.99	43.43	87.07	8.59	15.07	51.1	164.07	132.92	300.13
1C4K (573,111)	39.48	45.17	60.23	120.82	8.33	15.31	89.51	275.1	201.01	457.31

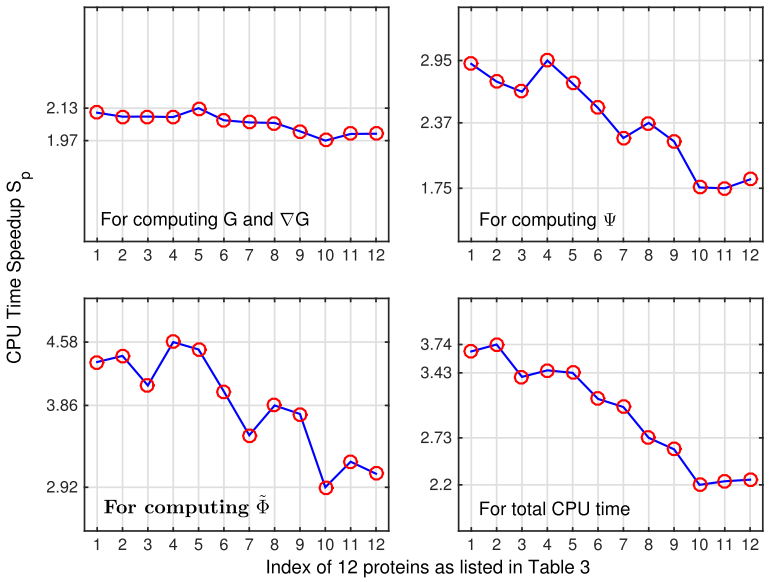


Fig. 5. The CPU time speedup S_p produced by our hybrid PBE solver using the data of Table 4.

that the modification we made on SDPB did not affect the convergence rate of SDPB but accelerated the calculation of Ψ and p_k significantly.

Fig. 5 displays the CPU time speedup S_p as a function of the protein ordering index. Here S_p was calculated as the ratio of the CPU times spent by SDPB and our new hybrid solver according to the data of Table 4. The ordering indices were set in Table 3, in which we also listed the number of mesh points within Ω_7 and its percentage ρ of the total number of mesh points of domain Ω. From this figure we can see that the speedup S_p varied with the percentage ρ. It was reduced from 3.74 to 2.2 for the total CPU time when ρ was increased from 7.5% to 14.2%. In these tests, we used the same parameter values to construct a uniform mesh of the region $\Omega \setminus D$ for each protein for simplicity of implementation, causing the increment of ρ when the size of region Ω_7 became larger for a larger protein.

Furthermore, a large amount of memory locations was saved clearly by the new program package. For example, in the case of 2LZX, about 1 GB was saved by only counting the mesh data storage. Hence, our hybrid PBE solver not only speeds up the numerical solution of PBE but also can be applied to a larger biomolecule in a larger solvent region than SDPB.

6.4. Electrostatic solvation free energy calculation

One important application of PBE is to predict the electrostatic solvation free energy of a biomolecule. According to the solution decomposition (3), we can estimate the electrostatic solvation energy E by the formula

$$E = \frac{N_A}{4184} \cdot \frac{k_B T}{2} \sum_{j=1}^{n_p} z_j \left(\Psi(\mathbf{r}_j) + \tilde{\Phi}(\mathbf{r}_j) \right) \quad \text{in kilocalorie per mole (kcal/mol).} \tag{25}$$

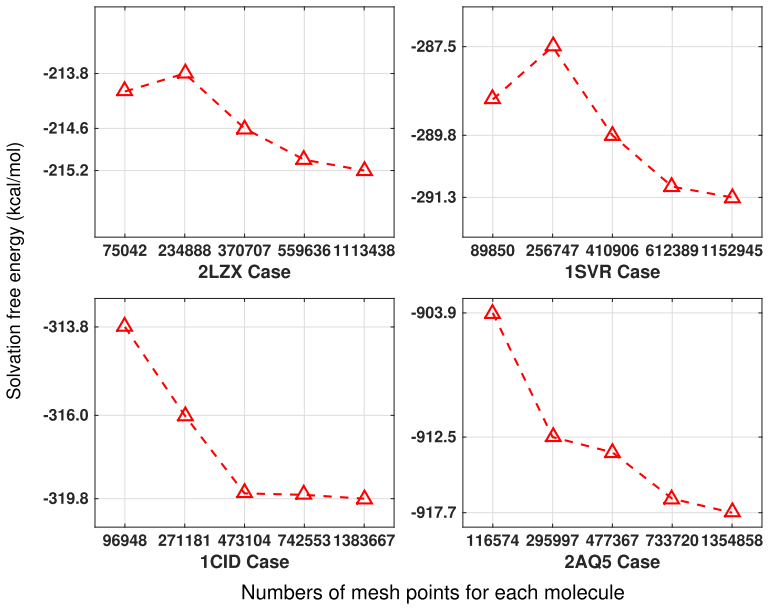
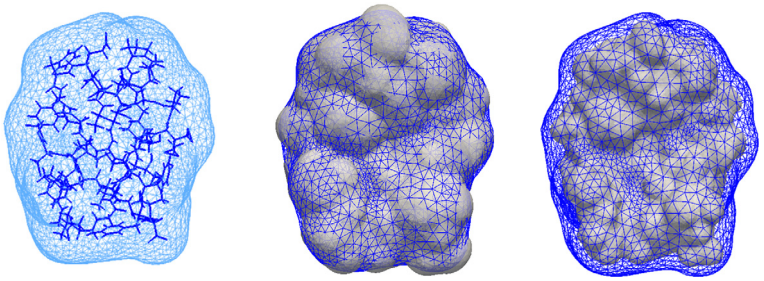


Fig. 6. Solvation free energies calculated by our hybrid PBE solver. Here the numbers on the x-axis are the numbers of mesh nodes used for calculation.



(a) Gaussian surface (b) Gaussian vs. SAS (c) Gaussian vs. SES

Fig. 7. A comparison of the Gaussian surface with the solvent-accessible surface (SAS) in Plot (b) and the solvent-excluded surface (SES) in Plot (c) for the protein with the PDB ID 2LZX. Here, the triangle meshes indicate the Gaussian surface, and the solid lines of Plot (a) represent a molecular structure of the protein.

In calculation, the numerical solutions Ψ_h and $\tilde{\Phi}_h$ are found from solving (5) and (6) to obtain an approximation E_h of E based on a mesh Ω_h of Ω. A convergent and numerically stable PBE solver is expected to yield a sequence of E_h that may vary around E but in a small range as $h \to 0$.

To study the numerical behavior of our new hybrid PBE solver, we constructed five different meshes for each of the four selected proteins considered in the previous subsection. The numbers of mesh nodes of these five meshes are displayed in Fig. 6 as the labeling numbers of the x-axis for each protein. They have been arranged in the increasing ordering from about 75,000 to 1,384,000. Each new mesh was constructed by adding new mesh nodes/tetrahedra to the current mesh, and the triangular surface mesh of the interface Γ was kept without any change. Here the triangular surface mesh was generated from our hybrid program with enough mesh nodes as a good approximation of Γ. We calculated the values of E_h using the hybrid PBE solver and plotted them in Fig. 6.

From Fig. 6 it can be seen that E_h varied in a small range toward a limit as the number of mesh nodes was increased, indicating a convergence tendency of our hybrid PBE solver. These test results also well demonstrate the numerical stability of our hybrid PBE solver.

Furthermore, in Table 5, we compared the electrostatic solvation free energies calculated by using the Gaussian surface with the ones by using two other commonly-used molecular surfaces — the solvent-accessible surface (SAS) and solvent-excluded surface (SES) for 12 proteins. In these tests, the SAS and SES were generated by a free software program called `EDTSurf` [28]. A comparison of these three different surfaces was given in Fig. 7 for the protein represented in the PDB ID 2LZX, showing that the shapes of these three surfaces are nearly similar but both SAS and SES are more bumpy than the Gaussian surface. Since the surface triangulations of SAS and SES generated from `EDTSurf` had much more mesh points than the surface triangulations of the Gaussian surface from `GAMer`, we used $m = 2$ and $n = 3$ for the tests of using SAS and SES while retaining $m = 3$ and $n = 4$ for the case of using Gaussian surface. From Table 5 it can be seen that the

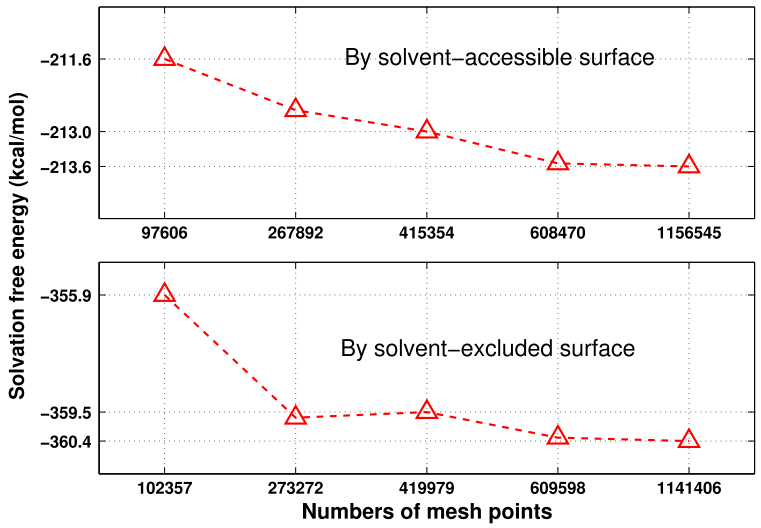


Fig. 8. A comparison of electrostatic solvation free energies calculated by our hybrid PBE solver using the interface Γ defined by SAS with that by SES for protein (2LZX) on different meshes. Here numbers on the x-axis are the numbers of mesh nodes.

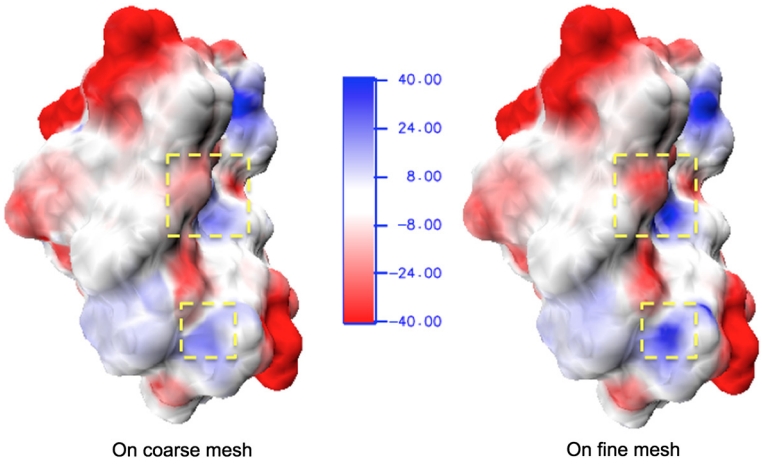


Fig. 9. A comparison of the PBE solution u calculated by our hybrid PBE solver on a coarse mesh with 75,042 mesh nodes with that on a fine mesh with 1,113,438 mesh nodes on a molecular surface of the protein with PDB ID 2LZX. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

free energy values generated by using SAS and Gaussian surface were close due to the small differences between the SAS and the Gaussian surface (see Plot (b) of Fig. 7). Because of a significant difference between SES and SAS (or the Gaussian surface), as shown in Plot (c) of Fig. 7, it was unsurprised to see that there existed big differences between the solvation free energies by SES and SAS.

Besides, to show the convergence behavior of our new hybrid solver in the case of using SAS and SES, we constructed two sets of tetrahedra meshes of Ω_7 by `Tetgen` [29], respectively, for the protein with PDB ID 2LZX. Each set consisted of five meshes. The solvation free energies were then calculated by our hybrid PBE solver, and reported in Fig. 8. Here the number of mesh nodes of each mesh was labelled on the x-axis. In spite of the bumpy surfaces of SAS and SES, the solvation free energies were found to vary stably, in a small range from -213 to -211 for the case of SAS and -360 to -356 for the case of SES.

Finally, in Fig. 9, we compared the PBE numerical solution u_h calculated by using the coarsest mesh with that by the finest mesh for the protein with PDB ID 2LZX. The values of u_h were displayed on the molecular surface in different colors. From this plot we can see that some values of u_h were changed significantly (as marked out in two yellow dashed squares) due to the mesh being refined. This indicates that solving PBE in a higher accuracy on a larger mesh may be critical for some applications, from which we may discover some new areas of the molecular surface on which the PBE solution u varies intensively.

Table 5
Electrostatic solvation free energies E calculated by our hybrid PBE solver using the interface Γ defined by the Gaussian surface, the solvent-accessible surface, or the solvent-excluded surface. Here, N_{Ω_7} and N_Ω are the numbers of mesh points within the central box Ω_7 and the whole domain Ω, respectively, and E is in units [kcal/mol].

Protein PDB ID	Number of atoms	Gaussian surface			Solvent-accessible surface			Solvent-excluded surface		
		E	N_{Ω_7}	N_Ω	E	N_{Ω_7}	N_Ω	E	N_{Ω_7}	N_Ω
2LZX	488	−214.872	39,896	535,400	−211.57	33,598	97,606	−355.918	38,349	102,357
1AJJ	513	−312.754	42,817	538,321	−328.148	37,465	101,473	−598.918	44,624	108,632
1FXD	811	−1332.21	45,345	540,849	−1323.75	51,679	115,687	−1741.69	61,977	125,985
1HPT	852	−120.527	47,716	543,220	−126.853	54,672	118,680	−419.945	66,331	130,339
4PTI	892	−330.939	45,825	541,329	−339.317	53,049	117,057	−648.255	66,752	130,760
1SVR	1433	−287.214	54,666	550,170	−299.047	78,803	142,811	−810.147	91,109	155,117
1A63	2065	−479.341	62,506	558,010	−507.616	113,817	177,825	−1245.5	132,977	196,985
1CID	2783	−312.622	62,870	558,374	−333.671	125,473	189,481	−1038.81	157,081	221,089
1A7M	2803	−468.621	68,415	563,919	−500.275	132,500	196,508	−1117.43	160,942	224,950
2AQ5	6024	−905.338	82,317	577,821	−986.396	229,570	293,578	−2540.78	175,222	239,230
1F6W	8243	−946.924	79,182	574,686	−1040.58	240,096	304,104	−2944.56	217,372	281,380
1C4K	11,439	−3213.46	77,607	573,111	−3293.37	200,296	264,304	−6292.98	260,303	324,311

Acknowledgements

This work was partially supported by the National Science Foundation, USA, through grant DMS-1226259, and the UWM Research Growth Initiative. The authors appreciated the anonymous reviewers' valuable comments.

References

[1] B. Honig, A. Nicholls, Classical electrostatics in biology and chemistry, Science 268 (1995) 1144–1149.
[2] M. Neves-Petersen, S. Petersen, Protein electrostatics: a review of the equations and methods used to model electrostatic equations in biomolecules — applications in biotechnology, Biotechnol. Annu. Rev. 9 (2003) 315–395.
[3] C.L. Vizcarra, S.L. Mayo, Electrostatics in computational protein design, Curr. Opin. Chem. Biol. 9 (2005) 622–626.
[4] P. Ren, J. Chun, D.G. Thomas, M.J. Schnieders, M. Marucho, J. Zhang, N.A. Baker, Biomolecular electrostatics and solvation: a computational perspective, Q. Rev. Biophys. 45 (04) (2012) 427–491.
[5] F. Fogolari, A. Brigo, H. Molinari, The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology, J. Mol. Recognit. 15 (6) (2002) 377–392.
[6] B. Lu, Y. Zhou, M. Holst, J. McCammon, Recent progress in numerical methods for the Poisson–Boltzmann equation in biophysical applications, Commun. Comput. Phys. 3 (5) (2008) 973–1009.
[7] L. Xiao, C. Wang, R. Luo, Recent progress in adapting Poisson–Boltzmann methods to molecular simulations, J. Theor. Comput. Chem. 13 (3) (2014) 1430001 (19 pages).
[8] A.H. Boschitsch, M.O. Fenley, A fast and robust Poisson–Boltzmann solver based on adaptive Cartesian grids, J. Chem. Theory Comput. 7 (5) (2011) 1524–1540.
[9] D. Chen, Z. Chen, C. Chen, W. Geng, G. Wei, MIBPB: a software package for electrostatic analysis, J. Comput. Chem. 32 (4) (2011) 756–770.
[10] N.A. Baker, D. Sept, S. Joseph, M. Holst, J.A. McCammon, Electrostatics of nanosystems: application to microtubules and the ribosome, Proc. Natl. Acad. Sci. USA 98 (18) (2001) 10037–10041.
[11] B. Lu, X. Cheng, J. Andrew McCammon, "New-version-fast-multipole-method" accelerated electrostatic calculations in biomolecular systems, J. Comput. Phys. 226 (2) (2007) 1348–1366.
[12] W. Im, D. Beglov, B. Roux, Continuum solvation model: computation of electrostatic forces from numerical solutions to the Poisson–Bolztmann equation, Comput. Phys. Commun. 111 (1998) 59–75.
[13] D. Xie, New solution decomposition and minimization schemes for Poisson–Boltzmann equation in calculation of biomolecular electrostatics, J. Comput. Phys. 275 (2014) 294–309.
[14] W. Rocchia, E. Alexov, B. Honig, Extending the applicability of the nonlinear Poisson–Boltzmann equation: multiple dielectric constants and multivalent ions, J. Phys. Chem. B 105 (2001) 6507–6514.
[15] S. Jo, M. Vargyas, J. Vasko-Szedlar, B. Roux, W. Im, PBEQ-solver for online visualization of electrostatic potential of biomolecules, Nucleic Acids Res. 36 (suppl. 2) (2008) W270–W275.
[16] M.E. Davis, J.D. Madura, B.A. Luty, J.A. McCammon, Electrostatics and diffusion of molecules in solution: simulations with the University of Houston Browian dynamics program, Comput. Phys. Commun. 62 (1991) 187–197.
[17] Y. Jiang, J. Ying, D. Xie, A Poisson–Boltzmann equation test model for protein in spherical solute region and its applications, Mol. Based Math. Biol. 2 (2014) 86–97, open Access.
[18] J. Xu, Iterative methods by space decomposition and subspace correction, SIAM Rev. 34 (4) (1992) 581–613.
[19] J. Xu, J. Zou, Some nonoverlapping domain decomposition methods, SIAM Rev. 40 (4) (1998) 857–914.
[20] Z. Yu, M. Holst, Y. Cheng, J. McCammon, Feature-preserving adaptive mesh generation for molecular shape modeling and simulation, J. Mol. Graph. Model. 26 (8) (2008) 1370–1380.
[21] S. Balay, J. Brown, K. Buschelman, W. Gropp, D. Kaushik, M. Knepley, L. McInnes, B. Smith, H. Zhang, PETSc Web page, http://www.mcs.anl.gov/petsc, 2012.
[22] D. Xie, J. Li, A new analysis of electrostatic free energy minimization and Poisson–Boltzmann equation for protein in ionic solvent, Nonlinear Anal., Real World Appl. 21 (2015) 185–196.
[23] J. Li, D. Xie, A new linear Poisson–Boltzmann equation and finite element solver by solution decomposition approach, Commun. Math. Sci. 13 (2) (2015) 315–325.
[24] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.
[25] T. Dolinsky, J. Nielsen, J. McCammon, N. Baker, PDB2PQR: an automated pipeline for the setup of Poisson–Boltzmann electrostatics calculations, Nucleic Acids Res. 32 (suppl. 2) (2004) W665.

[26] J. Hoffman, J. Jansson, A. Logg, G. Wells, et al., Dolfin, URL: http://www.fenics.org/dolfin.
[27] U. Trottenberg, C.W. Oosterlee, A. Schuller, Multigrid, Academic Press, 2000.
[28] D. Xu, Y. Zhang, Generating triangulated macromolecular surfaces by Euclidean distance transform, PLoS ONE 4 (12) (2009) e8140.
[29] H. Si, TetGen, a Delaunay-based quality tetrahedral mesh generator, ACM Trans. Math. Softw. 41 (2) (2015) 11:1–11:36.