August 2016

Domain Decomposition Based Hybrid Methods of Finite Element and Finite Difference and Applications in Biomolecule Simulations

Jinyong Ying
University of Wisconsin-Milwaukee

Domain Decomposition Based Hybrid Methods of finite element and finite difference and applications in biomolecule simulations

by

Jinyong Ying

A Dissertation Submitted in
Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
in Mathematics

at

The University of Wisconsin–Milwaukee

August 2016

ABSTRACT

Domain Decomposition Based Hybrid Methods of finite element and finite difference and applications in biomolecule simulations

by

Jinyong Ying

The University of Wisconsin-Milwaukee, 2016
Under the Supervision of Professor Dexuan Xie

The dielectric continuum models, such as Poisson Boltzmann equation (PBE), size modified PBE (SMPBE), and nonlocal modified PBE (NMPBE), are important models in predicting the electrostatics of a biomolecule in an ionic solvent. To solve these dielectric continuum models efficiently, in this dissertation, new finite element and finite difference hybrid methods are constructed by Schwartz domain decomposition techniques based on a special seven-box partition of a cubic domain. As one important part of these methods, a finite difference optimal solver — the preconditioned conjugate gradient method using a multigrid V-cycle preconditioner — is described in details and proved to have a convergence rate independent of mesh size in solving a symmetric positive definite linear system. These new hybrid algorithms are programmed in `Fortran`, `C`, and `Python` based on the efficient finite element library `DOLFIN` from the `FEniCS` project, and are well validated by test models with known analytical solutions. Comparison numerical tests between the new hybrid solvers and the corresponding finite element solvers are done to show the improvement in efficiency. Finally, as applications, solvation free energy and binding free energy calculations are done and then compared to the experiment data.

To
my wife Jiao Li
and my parents

TABLE of CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Being able to finish the dissertation, I owe acknowledgements to many people at University of Wisconsin-Milwaukee (UWM). First of all, I am greatly indebted to Prof. Dexuan Xie, who spent lots of time supervising the thesis work. I would like to express my utmost gratitude and heartfelt appreciation to him for his time and guidance with my work, which will be invaluable to me throughout my whole life. I am also extremely grateful to Prof. Hans Volkmer for his inspired discussions. Many thanks go to Prof. Bruce Wade, Prof. Kevin McLeod, Prof. Suzanne Boyd, Prof. Peter Hinow, and Prof. Boris Okun, who instructed me in graduate courses and spent time with me discussing various problems.

I would like to thank the Department of Mathematical Sciences at UWM for offering me the opportunities to teach some undergraduate courses. Meanwhile, I would also like to express my gratitudes to the course coordinator, Lawrence Schultz, who gave me constant encouragement during my first semester's teaching, giving me the faith and strength to make it through the toughest time.

I am also very much indebted to Prof. Yimin Zou and Prof. Dashan Fan, who constantly shared important information and provided suggestions when I needed help with my studies or other concerns of my daily life. I would like to thank Lisa Alzalde, Jane Miles, Kathleen Wehrheim, and other members of office staffs, who are really nice people and provided really great help during the four years at UWM.

I also would like to thank my officemate and dear friends at UWM for their friendships, kindnesses and help. I especially want to thank Yi Jiang, Xiaoying Lin, Rongcan Huang, Daoping Yu, Prof. Wei Wei, and Prof. Lei Wang. We had great time together and they helped me a lot in my daily life, which made me feel less lonely even though I am far away from my home country. I also would like to express my appreciation to the friends in China for their constant support.

I would like to take this chance to thank my wife Jiao Li too. Without her and her support, it would have been impossible to fully devote myself to this research and finish this dissertation. At the same time, I would like to express my gratitude to my parents and my sister in China for their constant encouragement and support, as well as to my previous thesis supervisors, Prof. Yuan Yuan at Memorial University of Newfoundland and Prof.

Shangjiang Guo at Hunan University, for their encouragement and help with my academic work.

Finally, I really appreciate the financial support for my four years' Ph.D work from Prof. Dexuan Xie (the National Science Foundation, USA, through grant DMS-1226259), the teaching assistantship from the Department of Mathematical Sciences, and the Distinguished Dissertation Fellowship from the graduate school at UWM.

Chapter 1

Introduction

1.1 Motivation and current research

The understanding of molecular interactions is significant for insights into the biological systems at the molecular scale [4, 5, 6]. Among the various components of molecular interactions, electrostatics of a biomolecule in an ionic solvent (as illustrated in Figure 1.1) is particularly important due to the long-range nature and influence on charged molecules. Understanding of its properties is a key in investigating many biomolecular processes, including protein structural stability, enzyme catalysis, biomolecular recognition, ligand binding, and protein folding problems [4, 7, 8].

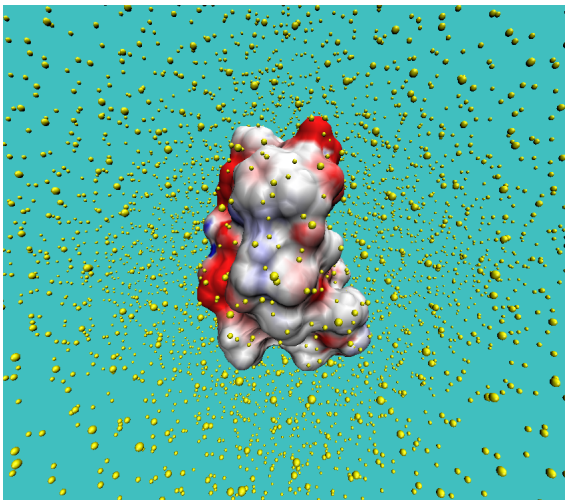


Figure 1.1: A protein with PDB ID 2LZX immersed in an ionic solvent. Here the yellow dots are the mobile ions in a solvent. The figure is generated using VMD.

To study the important electrostatic properties of a biomolecule, there are two commonly-used approaches, which mainly differ in the treatment of solution around the biomolecule. As the first one, the explicit solvent approach treats the water solvent in full molecular detail and accounts for all the degrees of freedom of water molecules. This class of methods can offer the detailed insight into the biomolecular interaction properties we are interested in. It has been widely adopted to the molecular dynamics for studying the physical movements of molecules [9]. However, the huge number of water molecules results in a high computational cost and thus limits the ability of these methods to calculate thermodynamic quantities for large biomolecular systems. To overcome these difficulties, the other one — the implicit solvent approach — had been well studied. Under this approach, the biomolecule is described in details with atomic partial charges on the positions of atoms, whereas the solvent, i.e., the water solution, and the solute domain that hosts the biomolecule are both treated as the dielectric continuum media with different dielectric constants, respectively. An interface between the solute and solvent regions is modeled by one commonly-used molecular surface (e.g., Gaussian surface, Solvent–excluded surface, or Solvent–accessible surface). This implicit solvent treatment turns out to greatly reduce the computational cost of the traditional explicit solvent approach, and has been applied to the studies of protein-protein or protein-ligand binding, scoring of protein conformations in the structure prediction, protein folding/unfolding, and ion channels [10, 11, 12, 13]. As the flagship of the implicit solvent approach, Poisson-Boltzmann equation (PBE) has become a well-established and popular dielectric continuum model in predictions of electrostatics of a biomolecule in numerous applications, including calculating and predicting the solvation free energy, the binding free energy, the pKa value, and the electrostatic force [5, 6, 14, 15, 16, 17].

Many work has been done in the development of the numerical PBE solvers. In the current popular PBE program packages, such as APBS [18], PBEQ [19, 20], DelPhi [21, 22], and UHBD [23], PBE was treated as a second-order jump-coefficient elliptic boundary value problem without considering any interface condition. It was mainly solved by the finite difference method based on a uniform Cartesian grid. The geometric multigrid techniques were then used to solve the finite difference linear systems optimally in the sense that the total number of floating-point operations is proportional to the number of unknowns. However, by ignoring the interface conditions, such a finite difference algorithm suffers serious problems

of solution accuracy and numerical stability [24, 25]. To overcome these problems, new finite difference schemes on uniform meshes with improved solution accuracy were developed in the last few decades, including the immersed boundary method [26, 27], the immersed interface method [28, 29, 30], the virtual node method [31], and the matched interface and boundary method [32]. The immersed interface method and the matched interface and boundary method have been used to solve PBE as an interface problem and develop the new PBE solvers, which are called PBSA [33] and MIBPB [34], respectively. Furthermore, the boundary element method, which recasts the linear partial differential equations into a boundary integral equation using a kernel function, also has been used to solve the linearized PBE [35]. Recently, the boundary element method has been generalized to the case where the kernel function is not known, which is called the kernel free boundary integral method [36], and it also has been applied to solve the nonlinear PBE [37].

Compared to the finite difference method and the boundary element method, the finite element method [38] can not only naturally incorporate the interface conditions into the formulation of a weak form, but also provide more flexibility for handling nonlinear equations. However, it was quite rare to apply the finite element method to solve PBE due to some difficulties of implementing this method. Unlike the finite difference method on a uniform mesh, an unstructured interface-fitted tetrahedral mesh is required to implement the finite element method in an effective way. Meanwhile, in order to assemble the stiffness matrix, it requires a large amount of memory and CPU time to initialize a mesh, including storing mesh data and computing the connectivity between vertex and cell and between facet and cell. Because of these technical issues, there were only a few groups working on developing finite element PBE solvers. Fortunately, in the last decade, an efficient finite element library `DOLFIN` from the `FEniCS` project was developed [39, 40]. Meanwhile, an efficient and powerful tetrahedral mesh generator `Tetgen` [41], written in `C++`, was released, whose implementation simply requires a surface triangulation as an input. Based on this progress, the finite element method became relatively easy to solve interface problems. Currently, Prof. Xie's group at UWM not only established a new simplified PBE mathematical theory [42], but also developed an efficient finite element PBE solver [43].

However, despite the variety of the numerous methods and PBE solvers, solving PBE is still too expensive in terms of computer memory and CPU time, especially for a large

molecular structure. In recent years, the domain decomposition method was a popular technique to compute the numerical solutions of partial differential equations (PDEs) on a large domain to obtain a satisfactory efficiency. The technique was firstly introduced by H. A. Schwartz [44, 45] in the nineteenth century to establish the solution's existence and uniqueness of a Poisson equation on an irregular domain due to the limitation of the Fourier transform method. With the rapid development of parallel computer architectures in 1980s, the domain decomposition method was re-visited and studied from the computational point of view [46, 47, 48, 49] to develop the parallel algorithms in scientific computing [50, 51, 52]. Currently, due to the flexibility in the treatment of complex geometries and the attractive performance on parallel computer architectures, many domain decomposition algorithms, such as the Schwarz, the Neumann-Neumann/FETI (Finite Element Tearing and Interconnection) and the Optimized Schwarz, have been developed and mathematically analyzed for linear systems [51, 53]. Meanwhile, the domain decomposition method also has been used to solve nonlinear problems (see [54, 55] and references therein). Using the domain decomposition method as a bridge, we recently combined finite element and finite difference methods together [1] to yield a new hybrid PBE solver. In this new hybrid method, a special seven overlapped box partition of a cubic domain was constructed so that the central box covers the interface and is surrounded by six neighboring boxes. The the finite element method is then used to deal with the interface problem within the central box while the finite difference method, which is enhanced by the use of the mesh free, the matrix free, and geometric multigrid techniques, is applied to each neighboring box to solve a boundary value problem efficiently. Due to the high efficiency of the new hybrid PBE solver, it is currently set as the default solver of a new released web-server SDPBS [56] for solving PBE. Using the same seven box partition, we developed another PBE hybrid model [57] constructed by combining nonlinear PBE finite element solver with a finite difference scheme for solving a linearized PBE, which provides another way to develop the efficient PBE solver. Furthermore, under this framework, it would be easy to develop a parallel PBE solver to compute the electrostatics for large microtubule and ribosome structures.

PBE has been proven to be an important dielectric continuum model in biophysics, biochemistry and structural biology. Nonetheless, PBE had some limitations [2, 58, 59, 60, 61, 62, 63, 64], mainly due to ignore the ion sizes and water polarization correlations. In PBE,

ions are treated as points without any volume; thus the ions with different sizes but equal valencies are treated to be identical. One well-known consequence of this simple treatment is that ion concentrations may exceed their maximally allowed values near highly charged biomolecular surfaces, such as those near enzyme active sites. Therefore, it is essential to consider these effects to develop more realistic models. To include the ion size effects, several attempts have been taken to modify PBE under assumptions that the water molecules and the mobile ions have different sizes by using different mechanics, such as the stern layers [65, 66], the Monte Carlo or mean field approach [67, 68, 69], the density functional theory [70], the generalized Poisson-Fermi distribution [71], and a simple statistical mechanics principle [72]. In literature, these models are all called size modified PBE (SMPBE). By the lattice gas formulation, Borukhov *et.al.* [58] proposed a size modified PDE in terms of a uniform ion size parameter, and Chu *et.al.* [59] generalized the case to allow two different ionic species to have different ion sizes. Furthermore, as the generalization of PBE case [42, 73], the solution existence and uniqueness of SMPBE were analyzed in [62] for the case of nonuniform ion sizes.

To numerically solve SMPBE, APBS [18] incorporated the solver reported in [59], which was constructed by a traditional finite difference method based on a uniform mesh for the case of two different ionic sizes. However, this solver also suffered the problems of solution accuracy and numerical stability [24, 25]. By the formulation from [58], a finite element solver using a two-term solution decomposition scheme was developed in [63], where the solution decomposition was used to treat the singularity of SMPBE caused by the Dirac Delta distributions, but numerical tests were done only on the simplest Born ion case. An augmented Lagrange multiplier method was used in [74] to solve SMPBE with the nonuniform ion sizes. However, it did not consider any interface condition. To develop a more effective and more reliable numerical SMPBE solver, a finite element program package for solving SMPBE [2] was developed in the Prof. Xie's group recently by using a new three-term solution decomposition scheme and was well tested numerically for both test models and protein cases with different net charges.

Besides the steric effect, how to reflect water molecules hydrogen bond network in a dielectric continuum model is also important especially for protein docking [75]. In the 1970s, it was firstly studied in Dogonadze *et.al.*'s nonlocal electrostatics. See paper [76] for

an overview of this nonlocal electrostatic study, [77] for applications in a primarily biological perspective, and [78] in the computational modeling perspective. Instead of modeling the solvent as a dielectric medium with a permittivity constant (like what is done in PBE and SMPBE), a convolution function is used to reflect the permittivity changes over the whole space and the spatial-frequency dependence of the dielectric function. As a result, the equation in the solvent region becomes an integro-differntial term, which is difficult to solve and thus the development of its numerical solvers is a key to explore its applications. Due to this difficulty, the early work mainly considered the simple Lorentz nonlocal model for the water solvent with charges near a half space or a dielectric sphere containing one central charge or multiple charges [77, 79]. This situation was changed by Hildebrandt *et.al.*' work in 2004 [80], which made it possible to numerically solve a nonlocal dielectric continuum model for a protein in water. Thereafter, numerous efforts were done to develop the solvers of the nonlocal dielectric continuum model for the pure water case by using the finite element, the finite difference, and the boundary element approaches [81, 82, 83, 84]. Furthermore, Prof. Xie from UWM proposed a nonlocal Poisson dielectric model for a protein in an ionic solvent [85] and a nonlocal modified Poisson-Boltzmann equation (NMPBE) [3], which is the first nonlinear nonlocal dielectric continuum model for computing electrostatics of an ionic solvated biomolecule. Moreover, in [3], an efficient NMPBE solver was developed by using the finite element method and a solution decomposition scheme. Prof. Xie used novel reformulation techniques that are different from the ones by Hildebrandt *et.al.* to solve NMPBE without involving any direct calculation of convolution.

1.2 Outline

As the continuation work of [1], in this dissertation, we develop new hybrid algorithms for efficiently solving both SMPBE and NMPBE. These new algorithms are the modifications of the corresponding finite element solvers proposed in [2] and [3] based on the hybrid techniques proposed in [1]. We then have programmed these two new hybrid algorithms in `C`, `Fortran` and `Python`, and numerically shown that the new hybrid algorithms can sharply improve the performance of their corresponding finite element solvers in terms of both computer CPU time and memory usage. The remaining parts of the dissertation are organized as follows:

In Chapter 2, we present the finite element and finite difference hybrid method for solving

a class of linear interface problems based on a special seven overlapped box partition. Then a mesh generation and domain partition scheme is given to automatically generate the domain partition, the uniform finite difference mesh, and the finite element mesh. Furthermore, an optimal finite difference solver using a multigrid preconditioner for the conjugate gradient method is constructed. A test example is given to show that the new hybrid method has a second-order convergence rate.

In Chapter 3, the dimensionless formulation of SMPBE is firstly given and then the new hybrid method is constructed to solve SMPBE. Numerical experiments are given to verify the new SMPBE program package, and to show the improvement of SMPBE from PBE using a dipole test in the prediction of the ion concentrations. Numerical tests demonstrate the high efficiency as well as the numerical stability of the new hybrid SMPBE solver. Moreover, a brief discussion of the differences and similarities between SMPBE and PBE are discussed from the numerical point of view.

In Chapter 4, the new hybrid method is constructed to solve the NMPBE model. For completeness, the reformulation of NMPBE into a system of coupled equations is presented. Then the reformulation of the system of equations on each neighboring box of the special seven box partition is given so that each discretized linear system is symmetric positive definite (SPD) in order to use the optimal finite difference solver — the preconditioned conjugate gradient method using a multigrid preconditioner. Numerical experiments are given to verify the new program package and to show the performance improvement compared to the finite element NMPBE solver. The binding free energy of a complex molecule (PDB ID 1d86) is calculated by the new hybrid solver to show that NMPBE can better match the experiment data than PBE.

The conclusions and some future work are given in the last chapter.

Chapter 2

Domain decomposition method for interface problems

In this chapter, a new finite element and finite difference hybrid method based on a special seven box partition will be constructed to solve a class of second-order linear interface elliptic boundary value problems. A scheme will be presented to automatically generate the domain partition, the uniform finite difference mesh, and the finite element mesh for an input. Meanwhile, a finite element solver and an optimal finite difference solver will be constructed. Lastly a test example will be given to show that the constructed hybrid method has a second-order convergence rate.

2.1 A class of linear interface elliptic problems

In this section, for a sufficiently large cubic domain Ω such that

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

we consider a class of second-order linear interface elliptic boundary value problems having the following form:

$$\begin{cases} -\epsilon_p \Delta w(\mathbf{r}) = f_p(\mathbf{r}), & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta w(\mathbf{r}) + \beta(\mathbf{r}) w(\mathbf{r}) = f_s(\mathbf{r}), & \mathbf{r} \in D_s, \\ w(\mathbf{s}^+) = w(\mathbf{s}^-), \quad \epsilon_s \dfrac{\partial w(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \dfrac{\partial w(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} + \zeta(\mathbf{s}), & \mathbf{s} \in \Gamma, \\ w(\mathbf{s}) = g(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{2.1}$$

where ϵ_p and ϵ_s are two given positive constants, region D_p is surrounded by region D_s while Γ is the interface between them, $\partial\Omega$ denotes the boundary of Ω, f_p, f_s, β, ζ, and g are given continuous functions, and we further assume that β is nonnegative.

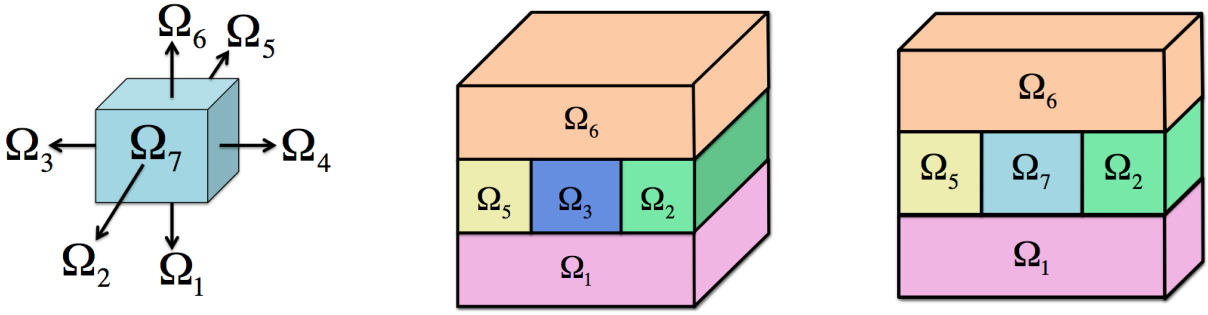
Due to the interface conditions, the nature choice to solve problem (2.1) would be the finite element method, provided an interface-fitted tetrahedra mesh. However, using an unstructured mesh, not only it needs extra CPU time to initialize a mesh for the computation purpose, including generating a mesh and computing the connectivity (between facet and cell, between vertex and cell, etc), but also it requires a large amount of memory to store the mesh data and the nonzero entries of each involved coefficient matrix. Meanwhile, it also causes the efficient geometric multigrid techniques no longer to work to make the finite element method less attractive than the finite difference method in real applications. To partially overcome the disadvantage of the finite element method, in this chapter, we will construct a new box iterative method using the Schwartz alternating method based on a special seven-box partition of a cubic domain and then propose a new finite element and finite difference hybrid method to efficiently solve the linear interface problem (2.1).

2.2 Overlapped box iterative method

In this section, we firstly present a special partition of a given cubic domain Ω. For a given region D_p, we select another cubic box D in Ω, and then divide Ω into seven overlapped boxes Ω_i for $i = 1, 2, \cdots, 7$ with Ω_7 being the central box satisfying

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

The position and ordering index of each box are illustrated in Figure 2.1. Clearly, the cubic domain Ω was decomposed into seven overlapped boxes, in which one central box contains the solute region D_p and is surrounded by six neighboring boxes; Furthermore, $\Omega_7 \setminus D$ gives the overlapped part of Ω_7 with its six neighboring boxes. Here the cubic domain is decomposed into seven boxes in order to avoid the domain singularity (reentrant corner), which would reduce the convergence rate of the multigrid method on uniform meshes [86], and meanwhile it is easier to implement the multigrid method in a rectangular box. Therefore, seven is the minimal partition number we can have for domain Ω to guarantee the rapid convergence of the domain decomposition method.



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

Figure 2.1: The special partition of 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 of Plot (b). Here the overlapping parts are not exposed in Plots (b) and (c) for clarity. These figures are from [1].

Based on the special domain partition proposed above and following the Schwartz domain decomposition scheme [87], in this section, we define an overlapped box iterative method for solving (2.1) by

$$w_i^{(k)} = (1-\omega)w_i^{(k-1)} + \omega\bar{w}_i \quad \text{on } \Omega_i \text{ for } i = 1, 2, \cdots, 7, \tag{2.2}$$

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:

$$\begin{cases} -\epsilon_s \Delta w(\mathbf{r}) + \beta(\mathbf{r})w = f_s(\mathbf{r}) & \text{in } \Omega_i, \\ w(\mathbf{s}) = w_j^{(k-1)}(\mathbf{s}) & \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = i+1 \text{ to } 7, \\ w(\mathbf{s}) = w_j^{(k)}(\mathbf{s}) & \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = 1 \text{ to } i-1, \\ w(\mathbf{s}) = g(\mathbf{s}) & \text{on } \partial\Omega_i \cap \partial\Omega, \end{cases} \tag{2.3}$$

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

$$\begin{cases} -\epsilon_p \Delta w(\mathbf{r}) = f_p(\mathbf{r}) & \text{in } D_p, \\ -\epsilon_s \Delta w(\mathbf{r}) + \beta(\mathbf{r})w = f_s(\mathbf{r}) & \text{in } \Omega_7 \cap D_s, \\ w(\mathbf{s}^+) = w(\mathbf{s}^-), \quad \epsilon_s \dfrac{\partial w(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \dfrac{\partial w(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} + \zeta(\mathbf{s}) & \text{on } \Gamma, \\ w(\mathbf{s}) = w_j^{(k)}(\mathbf{s}) & \text{on } \partial\Omega_7 \cap \Omega_j \text{ for } j = 1 \text{ to } 6, \end{cases} \tag{2.4}$$

where $\partial\Omega_i$ denotes the boundary of Ω_i. Here we order the central box Ω_7 as the last one so that the updates $w_i^{(k)}$ from the six neighboring boxes can be employed in the construction

of boundary condition on $\partial\Omega_7$ to yield a good boundary value problem for the sake of speeding up solving the interface problem and the convergence of the box iterative method. From the standard Schwartz domain decomposition theory [88, 89], it can be known that the overlapped domain decomposition scheme has a fixed convergence rate for a fixed value of the over-relaxation parameter ω. Therefore, for a specific problem in form of (2.1), we could determine the "optimal" relaxation parameter ω empirically. Meanwhile, in the implementation, the new iterative method stops as long as the following condition holds:

$$\sqrt{\sum_{i=1}^{7} ||w_i^{(k)} - w_i^{(k-1)}||^2} \leq \epsilon, \tag{2.5}$$

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

2.3 Finite element and finite difference hybrid solver

Applying the new box iterative method to solve (2.1), in principle, we could apply different methods in different boxes to solve problems (2.3) and (2.4). Specially, due to the construction of the central box Ω_7, we prefer to use the finite element method to naturally deal with the interface conditions; while the problem on each neighboring box is just the regular second-order elliptic problem, we use the finite difference method on uniform meshes to efficiently solve it. Therefore, this leads to a new finite element and finite difference hybrid method to solve (2.1).

In the following subsections, we will firstly present a scheme and a program for generating the domain Ω, the seven overlapped boxes $\{\Omega_i\}_{i=1}^7$, a special interface-fitted tetrahedral mesh of Ω_7, and a uniform finite difference mesh of Ω_i for $i = 1$ to 6. Then we will construct an efficient finite element scheme for solving the interface boundary value problem (2.4) and an optimal finite difference scheme for solving the boundary value problem (2.3) to guarantee the constructed hybrid method has better efficiency than the finite element method.

2.3.1 Mesh generation scheme and program

To apply the new finite element and finite difference hybrid method to solve the problem, now we present the scheme to automatically generate the meshes as well as the domain partition. To do that, let a cubic region D be given in the form $D = \prod_{i=1}^{3}(a_i, b_i)$ with

each length $b_i - a_i = L$ for $i = 1, 2, 3$. We then define the mesh size h and two other mesh parameters τ and η satisfying $\tau < \eta$ by

$$h = L/2^n, \quad \tau = 2^m h, \quad \eta = \mu L/2, \tag{2.6}$$

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 the numerical solution. Using them, we construct a cubic domain Ω and the seven partitioned boxes Ω_i, $i = 1, 2, \cdots, 7$, indicated in Figure 2.1, as follows:

$$\Omega = \prod_{i=1}^{3}(a_i - \eta, b_i + \eta), \qquad \Omega_7 = \prod_{i=1}^{3}(a_i - \tau, b_i + \tau),$$

$$\Omega_1 = (a_1 - \eta, b_1 + \eta) \times (a_2 - \eta, b_2 + \eta) \times (a_3 - \eta, a_3),$$

$$\Omega_2 = (a_1 - \eta, b_1 + \eta) \times (a_2 - \eta, a_2) \times (a_3 - \tau, b_3 + \tau),$$

$$\Omega_3 = (a_1 - \eta, a_1) \times (a_2 - \tau, b_2 + \tau) \times (a_3 - \tau, b_3 + \tau),$$

$$\Omega_4 = (b_1, b_1 + \eta) \times (a_2 - \eta, b_2 + \tau) \times (a_3 - \tau, b_3 + \tau),$$

$$\Omega_5 = (a_1 - \eta, b_1 + \eta) \times (b_2, b_2 + \eta) \times (a_3 - \tau, b_3 + \tau),$$

$$\Omega_6 = (a_1 - \eta, b_1 + \eta) \times (a_2 - \eta, b_2 + \eta) \times (b_3, b_3 + \eta),$$

from which we can see that these two parameters τ and μ determine the overlapped parts among the seven boxes. We then 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 defined. 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. For simplicity, 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. Upon this fact, the data exchange between them in the box iterative method can be done easily and

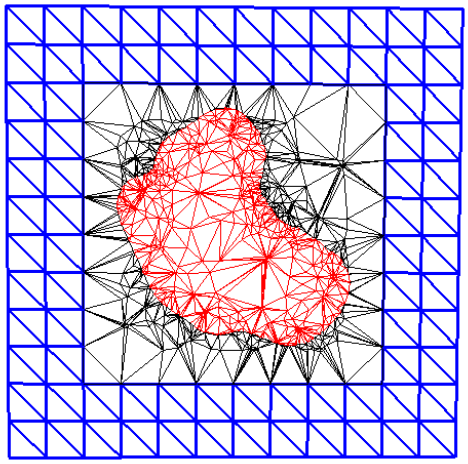


Figure 2.2: A cross section of a special tetrahedral mesh of Ω_7 on the xy-coordinate plane for a protein with PDB ID 2LZX. D_p is colored in red, and the overlapped part $\Omega_7 \setminus D$ in blue. This figure is from [1].

efficiently. We programmed the above mesh generation scheme in `C`. For clarity, the scheme to generate the seven-box partition, the finite difference uniform mesh on each neighboring box, and the finite element mesh on the central box is summarized as follows:

- **Mesh scheme to generate domain partition and meshes**
  - — Generate a surface triangulation mesh of the boundary Γ of the domain D_p;
  - — Generate a surface triangulation mesh of ∂D;
  - — Combine the surface triangulation meshes of Γ and ∂D to be a closed piecewise linear complex as the input for `Tetgen`;
  - — Call `Tetgen` to generate a tetrahedral mesh for the cubic domain D;
  - — Input the values of three parameters n, m, and μ to determine a cubic domain Ω and a uniform step size h, thus determining the uniform finite difference mesh on each neighboring box;
  - — "Add" the uniform mesh on the overlapped part $\Omega_7 \setminus D$ to the unstructured mesh of D to generate the finite element mesh on Ω_7.

In this program, the domain D_p will be an input (In application of electrostatic potential calculations, a PQR file of a biomolecule is required as an input file, which spedifies the D_p

domain. It can be generated from a PDB file of a biomolecule, which can be downloaded from the Protein Data Bank (PDB) (`http://www.rcsb.org/`), by the program tool `PDB2PQR` [90]). The surface triangulation mesh of the interface Γ is generated from the molecular surface and volumetric mesh generation program package `GAMer` [91], which uses an isosurface of a Gaussian function to approximate the molecular surface Γ of a biomolecule and then applies the Marching Cube method [92] to extract the surface triangulation. Then a function written in `C` is developed to generate the surface triangulation of ∂D. After combining these two surface triangulations and calling the `Tetgen` to generate a 3D mesh of the domain D, we next "add" the uniform meshes on the overlapped part $\Omega_7 \backslash D$ and "cut" each grid to 6 tetrahedra to generate a special 3D finite element mesh on Ω_7. See Figure 2.2 for a mesh of Ω_7 generated from this program.

2.3.2 Finite element solver

To solve the linear interface boundary value problem on the central box Ω_7, we will use the classic finite element method. Let $\mathcal{M}_7 \subset H^1(\Omega_7)$ be a linear Lagrange finite element function space based on a tetrahedral mesh of Ω_7. We reformulate the interface boundary value problem (2.4) into the following equivalent variational problem:

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

$$b(\bar{w}_7, v) = l(v) \quad \forall v \in \mathcal{M}_{7,0}, \tag{2.7}$$

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

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

We implement this finite element program in `Python` based on the efficient finite element library `DOLFIN` [93] from the `FEniCS` project. Here each assembled SPD linear finite element system is solved by calling the linear solver – the preconditioned conjugate gradient method using the incomplete LU preconditioning (PCG-ILU) – from the scientific computing library

PETSc [94], and by default both the relative and the absolute residue error parameters are set to 10^{-8}. Another option of the preconditioner for the PCG algorithm would be the algebraic multigird method. Numerical tests showed the latter is not as efficient as PCG-ILU, although it has the convergence rate independent of mesh sizes. Thus, in the following, we will use PCG-ILU to solve the finite element linear systems on the central box. When the finite element system is not SPD in the nonlocal case, then we will call GMRES-ILU from PETSc to solve the corresponding linear system.

2.3.3 Finite difference solver

To solve the regular second-order elliptic boundary value problem on each neighboring box, we adopt the finite difference method on uniform meshes. More specifically, we use the seven-point finite difference stencil to discretize the corresponding problem. That is, the equation on each neighboring box will be approximated by the following linear system: for $i = 1$ to $N_{\nu,1} - 1$, $j = 1$ to $N_{\nu,2} - 1$, and $k = 1$ to $N_{\nu,3} - 1$,

$$\begin{aligned} \frac{\epsilon_s}{h^2}(6w_{i,j,k} - w_{i+1,j,k} - w_{i-1,j,k} - w_{i,j+1,k} - w_{i,j-1,k} - w_{i,j,k+1} - w_{i,j,k-1}) \\ +\beta_{i,j,k}w_{i,j,k} = f_{s,i,j,k}, \end{aligned} \tag{2.8}$$

where $\nu = 1, 2, \cdots, 6$, $N_{\nu,1}$, $N_{\nu,2}$, 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 (2.8) is clearly SPD. Then we will construct an optimal solver to solve this SPD linear system.

Here we try to construct a preconditioned conjugate gradient method using the multigrid method as a preconditioner (PCG-MG). The scheme of PCG for solving a linear system in the matrix form $AU = F$ is well known (see page 297 in [87] for example). It is presented in Algorithm 1 for clarity.

Algorithm 1: the Preconditioned Conjugate Gradient method

$r_0 = F - Ax_0$
$d_0 = z_0 = Br_0$ // preconditioning
For $k = 1, 2, \cdots\cdots$ (until convergence):
    if $\|(d_{k-1}, d_{k-1})_A\| \leq 10^{-20}$:
        return iteration x_{k-1};
    $\alpha_k = (z_{k-1}, r_{k-1})/(d_{k-1}, d_{k-1})_A$
    $x_k = x_{k-1} + \alpha_k d_{k-1}$
    $r_k = r_{k-1} - \alpha_k A d_{k-1}$
    $z_k = Br_k$ // preconditioning
    $\beta_k = (z_k, r_k)/(z_{k-1}, r_{k-1})$
    $d_k = z_k + \beta_k d_k$

In Algorithm 1, A denotes a SPD coefficient matrix and $(x, y)_A = x^T Ay$. From this algorithm, it is easy to see that the two most time-consuming parts are the calculations of the matrix-vector product Ad and the solution of the preconditioning equation $B^{-1}z = r$ with B being a preconditioner (a matrix approximating A^{-1} in some sense). Meanwhile, the efficiency of the algorithm highly depends on the choice of the preconditioner B.

Multigird is a very popular numerical technique by a global correction of the fine grid solution approximation from solving the problem on a coarse mesh. And it has been well known that it is efficient to solve Poisson equations on uniform meshes. Meanwhile, it is also popular to use multigrid techniques to do the preconditioner, main advantage of which versus a pure multigrid solver is particularly clear for nonlinear problems, e.g., the eigenvalue problems. Here, to construct the preconditioner of the CG method, we use the standard multigrid V-cycle method. More specifically, in the 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 [86]). With the standard coarsening scheme, the coarsening of a mesh is stopped once one direction has only one interior point or an odd partition number. On each neighboring box Ω_ν, the following gives the detailed descriptions of functions in the programming.

[**PCG-MG Algorithm**] `PCG-MG`$(U_{fd,\nu}, F_{fd,\nu}, \epsilon)$: A_ν denotes the discretized matrix of the problem on the ν-th neighboring box, $U_{fd,\nu}$ is the solution vector, $F_{fd,\nu}$ is the corresponding right-hand side vector (RHSV), and ϵ is the given termination criterion.

1. Compute the norm of $F_{fd,\nu}$ and set $\epsilon_0 = \epsilon \|F_{fd,\nu}\|$;
2. Call `MVP`$(U_{fd,\nu})$ and return the vector $A_\nu U_{fd,\nu}$;
3. Compute the residue vector $r_\nu = F_{fd,\nu} - A_\nu U_{fd,\nu}$;
4. Call `Pre-MG`(r_ν) and return z_ν;
5. Set $d_\nu = z_\nu$, and $\tau_1 = (r_\nu, z_\nu)$;
6. Call `MVP`(d_ν), return $A_\nu d_{fd,\nu}$, and then calculate $\eta = (d_\nu, A_\nu d_{fd,\nu})$;
7. [Singularity check] If $\eta \leq 10^{-20}$, exit and return the solution $U_{fd,\nu}$;
8. Set $\alpha = \tau_1/\eta$, update $U_{fd,\nu} = U_{fd,\nu} + \alpha d_\nu$ and $r_\nu = r_\nu - \alpha A_\nu d_{fd,\nu}$;
9. [Convergence test] If $\|r_\nu\| \leq \epsilon_0$, exit and return the solution $U_{fd,\nu}$;
10. Call `Pre-MG`(r_ν) and return z_ν;
11. Set $\tau_2 = \tau_1$, $\tau_1 = (r_\nu, z_\nu)$, and $\beta = \tau_1/\tau_2$;
12. Update $d_\nu = z_\nu + \beta d_\nu$ and go back to step 6;

[**Matrix-Vector-Product**] `MVP`(u): For $i = 1$ to $N_{\nu,1} - 1$, $j = 1$ to $N_{\nu,2} - 1$, and $k = 1$ to $N_{\nu,3} - 1$,

$$(A_\nu u)_{i,j,k} = \frac{\epsilon_s}{h^2}(6u_{i,j,k} - u_{i+1,j,k} - u_{i-1,j,k} - u_{i,j+1,k} - u_{i,j-1,k} - u_{i,j,k+1} - u_{i,j,k-1}) + \beta_{i,j,k} u_{i,j,k}$$

with $u_{i,j,k}$ being zero if the point (x_i, y_j, z_k) is on the boundary of the neighboring box Ω_ν.

[**Multigrid V-cycle Preconditioning**] `Pre-MG`(r): Here we use one step multigrid V-cycle method as a preconditioner of the CG method to construct the preconditioned equation $z_\nu = M_K r_\nu$ on the neighboring box Ω_ν, where K denotes the number of levels of uniform meshes with uniform mesh sizes $h_1 = 2h_2 = \cdots = 2^{K-1} h_K$ on Ω_ν. Let b_l, z_l, and r_l denote the RHSV, the solution vector, and the residue vector on l-th level mesh, respectively. For this function, the initial guess is always taken to be zero (i.e., $z_L = 0$).

- For $l = K, \cdots, 2$:

  `Pre-smooth`(z_l, b_l, r_l);

  `Restriction`(b_{l-1}, r_l);

- `Coarsest_Solver`(z_1, b_1);

- For l=1, $K-1$:

  `Interpolation`(z_{l+1}, z_l);

  `Post-smooth`(z_{l+1}, b_{l+1});

- Return the solution vector z_K;

where

- `Pre-smooth`: given a RHSV b_l, we use one step Forward Gauss-Seidel iteration as the pre-smoother to update z_l with the zero initial guess. That is,

  $$z_l = (D_l - L_l)^{-1} b_l,$$

  where the discretized matrix on the l-th level uniform mesh $A_l = D_l - L_l - U_l$ with D_l, L_l, and U_l being the diagonal part, the negative lower part, and the negative upper part of the coefficient matrix A_l, respectively. Then the residue vector r_l is computed by

  $$r_l = b_l - A_l z_l;$$

- `Post-smooth`: given a RHSV b_l, we use one step Backward Gauss-Seidel iteration to update the solution z_l as follows:

  $$z_l = z_l + (D_l - U_l)^{-1}(b_l - A_l z_l);$$

- `Restriction`: given a residue vector r_l on the l-th level mesh, we construct the new RHSV b_{l-1} for the (l-1)-th level mesh with the 3D full weight method, which has the following stencil notation:

  $$\frac{1}{64}\left[\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\begin{bmatrix}2&4&2\\4&8&4\\2&4&2\end{bmatrix}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\right];$$

In the implementation, we have the following pseudo-code to do the restriction for a point (x_i, y_j, z_k) of the (l-1)-th level mesh, which is from a $3d$ array r_l on the l-th level mesh to update a $3d$ array b_{l-1} on the (l-1)-th level mesh :
On the plane $z = z_{2k}$,

$$\begin{aligned} b1 =& 8r_{l,2i,2j,2k} + 4(r_{l,2i-1,2j,2k} + r_{l,2i+1,2j,2k} + r_{l,2i,2j-1,2k} + r_{l,2i,2j+1,2k}) \\ &+ 2(r_{l,2i-1,2j-1,2k} + r_{l,2i-1,2j+1,2k} + r_{l,2i+1,2j-1,2k} + r_{l,2i+1,2j+1,2k}); \end{aligned}$$

On the plane $z = z_{2k+1}$,

$$\begin{aligned} b2 =& 4r_{l,2i,2j,2k+1} + 2(r_{l,2i-1,2j,2k+1} + r_{l,2i,2j-1,2k+1} + r_{l,2i,2j+1,2k+1} + r_{l,2i+1,2j,2k+1}) \\ &+ r_{l,2i-1,2j-1,2k+1} + r_{l,2i-1,2j+1,2k+1} + r_{l,2i+1,2j-1,2k+1} + r_{l,2i+1,2j+1,2k+1}; \end{aligned}$$

On the plane $z = z_{2k-1}$,

$$\begin{aligned} b3 =& 4r_{l,2i,2j,2k-1} + 2(r_{l,2i-1,2j,2k-1} + r_{l,2i,2j-1,2k-1} + r_{l,2i,2j+1,2k-1} + r_{l,2i+1,2j,2k-1}) \\ &+ r_{l,2i-1,2j-1,2k-1} + r_{l,2i-1,2j+1,2k-1} + r_{l,2i+1,2j-1,2k-1} + r_{l,2i+1,2j+1,2k-1}. \end{aligned}$$

And then we have

$$b_{l-1,i,j,k} = \frac{1}{64}[b1 + b2 + b3].$$

- `Interpolation`: we use the trilinear interpolation method [86, Page 72] to get the values of grid nodes on the fine mesh from the coarse mesh; In the implementation, to do the interpolation from a $3d$ array z_l to a $3d$ array z_{l+1}, we do the following steps to avoid the *if-else* sentence: (here we assume the starting index is zero (like `C` and `Python`) and use N_i, $i = 1, 2, 3$, to denote the partition number in x, y, z direction of l-th level mesh, respectively)

  1. For $i = 2$ to $N_1 - 1$ with increment 2, $j = 2$ to $N_2 - 1$ with increment 2, and $k = 2$ to $N_3 - 1$ with increment 2,
  $$z_{l+1,i,j,k} = z_{l,i/2,j/2,k/2};$$
  2. For $i = 1$ to $N_1 - 1$ with increment 2, $j = 2$ to $N_2 - 1$ with increment 2, and $k = 2$ to $N_3 - 1$ with increment 2,
  $$z_{l+1,i,j,k} = \frac{1}{2}(z_{l+1,i-1,j,k} + z_{l+1,i+1,j,k});$$
  3. For $i = 2$ to $N_1 - 1$ with increment

2, $j = 1$ to $N_2 - 1$ with increment 2, and $k = 2$ to $N_3 - 1$ with increment 2,

$$z_{l+1,i,j,k} = \frac{1}{2}(z_{l+1,i,j-1,k} + z_{l+1,i,j+1,k});$$

4. For $i = 2$ to $N_1 - 1$ with increment 2, $j = 2$ to $N_2 - 1$ with increment 2, and $k = 1$ to $N_3 - 1$ with increment 2,

$$z_{l+1,i,j,k} = \frac{1}{2}(z_{l+1,i,j,k-1} + z_{l+1,i,j,k+1});$$

5. For $i = 2$ to $N_1 - 1$ with increment 2, $j = 1$ to $N_2 - 1$ with increment 2, and $k = 1$ to $N_3 - 1$ with increment 2,

$$z_{l+1,i,j,k} = \frac{1}{2}(z_{l+1,i,j-1,k} + z_{l+1,i,j+1,k});$$

6. For $i = 1$ to $N_1 - 1$ with increment 2, $j = 2$ to $N_2 - 1$ with increment 2, and $k = 1$ to $N_3 - 1$ with increment 2,

$$z_{l+1,i,j,k} = \frac{1}{2}(z_{l+1,i,j,k-1} + z_{l+1,i,j,k+1});$$

7. For $i = 1$ to $N_1 - 1$ with increment 2, $j = 1$ to $N_2 - 1$ with increment 2, and $k = 2$ to $N_3 - 1$ with increment 2,

$$z_{l+1,i,j,k} = \frac{1}{2}(z_{l+1,i-1,j,k} + z_{l+1,i+1,j,k});$$

8. For $i = 1$ to $N_1 - 1$ with increment 2, $j = 1$ to $N_2 - 1$ with increment 2, and $k = 1$ to $N_3 - 1$ with increment 2,

$$z_{l+1,i,j,k} = \frac{1}{2}(z_{l+1,i-1,j,k} + z_{l+1,i+1,j,k}).$$

- `Coarsest_Solver`: given a RHSV b_1 and an initial guess z_1, we use the successive over-relaxation (SOR) method with the relaxation parameter $w = 1.7$ to solve the linear system on the coarsest uniform mesh. Here we stop SOR iterations whenever the residue vector has the norm value less than or equal to 10^{-10}. With the setup of domains Ω, Ω_7, D, and the uniform step-size h, we have symmetrically

  1. For the boxes Ω_1 and Ω_6, lengths of the edges in x, y, z directions are $(L + 2\eta), (L + 2\eta), \eta$, and the partition numbers are $(\mu + 1)2^n, (\mu + 1)2^n, \mu 2^{n-1}$, respectively;
  2. For the boxes Ω_2 and Ω_5, lengths of the edges in x, y, z directions are $(L + 2\eta), \eta, (L + 2\tau)$, and the partition numbers are $(\mu + 1)2^n, \mu 2^{n-1}, 2^n + 2^{m+1}$, respectively;
  3. For the boxes Ω_3 and Ω_4, lengths of the edges in x, y, z directions are $\eta, (L +$

$2\tau), (L + 2\tau)$, and the partition numbers are $\mu 2^{n-1}, 2^n + 2^{m+1}, 2^n + 2^{m+1}$, respectively;

Thus, if we set μ to be an even integer, for example, $n = 4, m = 2$ and $\mu = 4$, then for the top and bottom boxes, the front and rear ones, the left and right ones, we have $K = 5$, 4, 4, respectively, and on the coarsest mesh, only linear systems with sizes of $4 \times 4 \times 1$, $9 \times 3 \times 2$ or $3 \times 2 \times 2$ need to be solved. Obviously, these linear systems could be solved efficiently by the SOR method.

In the program, the PCG-MG iteration does not stop until the relative residual norm of the finite difference system (2.8) is less than or equal to 10^{-8} (i.e., $\epsilon = 10^{-8}$) by default. As expected, it turned out this PCG-MG linear solver is more efficient than the multigrid method to solve the Poisson-like equation (2.3). In the following, we give some theoretical results to guarantee the PCG-MG linear solver works and indeed is an optimal linear solver for solving (2.8) as we claimed.

Firstly, we present the following theorem to prove that one step multigrid V-cycle method as a preconditioner is SPD and it indeed works for the PCG algorithm.

Theorem 2.3.1. *The preconditioner M_K is SPD.*

Proof. For the equation on box Ω_ν, $\nu \in \{1, \cdots, 6\}$, A_l is the discretized matrix on the l-th level uniform mesh. Then A_l is obviously SPD. Since the multigird V-cycle method is just the nested two-level-grid iteration, we have the following recursion of the matrix M_l, $l = 1, \cdots, L$, as a preconditioner (i.e., initial guess is zero):

$$
\begin{aligned}
&M_1 = A_1^{-1}, \\
&M_l = H_{post}^l R_{pre}^l + R_{post}^l + H_{post}^l I_{l-1}^l M_{l-1} I_l^{l-1} (I_l - A_l R_{pre}^l), \quad l = 2, \cdots, K,
\end{aligned}
$$

where I_{l-1}^l is the prolongation operator, and I_l^{l-1} is the restriction operator, $R_{pre}^l = (D_l - L_l)^{-1}, R_{post}^l = (D_l - U_l)^{-1}, H_{post}^l = (D_l - U_l)^{-1} L_l$, $I_{l-1}^l = 8 * (I_l^{l-1})^T$, and I_l denotes the identity matrix on the l-th level uniform mesh.

Hence, we have

$$
\begin{aligned}
H_{post}^l R_{pre}^l + R_{post}^l &= (D_l - U_l)^{-1} L_l (D_l - L_l)^{-1} + (D_l - U_l)^{-1} \\
&= (D_l - U_l)^{-1} (L_l + D_l - L_l)(D_l - L_l)^{-1} \\
&= (D_l - U_l)^{-1} D_l (D_l - L_l)^{-1},
\end{aligned}
$$

which is obviously SPD.

Furthermore, we obtain

$$\begin{aligned} I_l - A_l R_{pre}^l &= I_l - (D_l - L_l - U_l)(D_l - L_l)^{-1} \\ &= I_l - I_l + U_l(D_l - L_l)^{-1} \\ &= U_l(D_l - L_l)^{-1}, \end{aligned}$$

which implies that

$$H_{post}^l I_{l-1}^l M_{l-1} I_l^{l-1}(I_l - A_l R_{pred}^l) = (D_l - U_l)^{-1} L_l I_{l-1}^l M_{l-1} I_l^{l-1} U_l (D_l - L_l)^{-1},$$

and again this is SPD if the matrix M_{k-1} is. Since M_1 is SPD, then by the mathematical induction, we know the matrix M_K is SPD. This completes the proof. □

Remark 2.3.2. In [95], the author proved a more general theorem that the multigrid method could be a preconditioner for the CG method. There it assumed the pre- and post-smoothing methods are the same and the numbers of smoothing steps are even. Here we just present a simpler proof to show the matrix generated by the one-step multigrid V-cycle method is SPD, which indeed works for the PCG algorithm. Furthermore, the author also showed numerically this kind of PCG-MG methods are more efficient than both the incomplete cholesky conjugate gradient method and the multigrid method, which is consistent with ours.

Theorem 2.3.1 theoretically guarantees that the constructed PCG-MG indeed works for linear SPD problem. Next, we prove that the PCG-MG solver is an optimal one in the sense that the convergence rate is independent of mesh size.

Theorem 2.3.3. *To solve a linear SPD system $AU = F$ with the defined multigrid preconditioner M_K, we have*

$$||r_k||_{M_K} \leq 2C^k ||r_0||_{M_K}$$

for some constant $C \in (0, 1)$ independent of the mesh size h. Here r_k denotes the residue vector at the k-th iterate, i.e., $r_k = F - AU_k$. That is, the convergence rate of the constructed PCG-MG solver is independent of mesh size.

Proof. From Theorem 2.3.1, we already know the preconditioner M_K using the multigrid V-cycle technique is SPD. By [87, Page 297], we then have exact the same algorithm as the one described in Algorithm 1 to solve the following preconditioned system $M_K Ax = M_K b$ under the A-norm instead of the usual l_2 norm. Indeed,

$$(M_K Ax, y)_A = (AM_K Ax, y) = (x, AM_K Ay) = (x, M_K Ay)_A,$$

and

$$(M_K Ax, x)_A \geq 0 \text{ for } \forall x.$$

Therefore, from the standard convergence theory of the CG algorithm [87], we have the following convergence result

$$||r_k||_{M_K} \leq 2 \left[\frac{\sqrt{cond(M_K A)} - 1}{\sqrt{cond(M_K A)} + 1} \right]^k ||r_0||_{M_K},$$

where the condition number is defined by

$$cond(M_K A) = \frac{\lambda_{max}(M_K A)}{\lambda_{min}(M_K A)}$$

under the A-norm.

To explore the upbound of this condition number, we firstly observed that for the multigrid V-cycle method,

$$\begin{aligned} U_{j+1} &= U_j + M_K(b - AU_j) \\ &= (I - M_K A)U_j + M_K b. \end{aligned}$$

From the multigrid analysis [96], we know the method is convergent independent of mesh size and thus we have

$$\rho(I - M_K A) < 1,$$

where

$$\rho(I - M_K A) = \sup_{v \neq 0} \frac{\|((I - M_K A)v, v)_A\|}{(v, v)_A},$$

from which it implies there exist $\alpha_0, \alpha_1 \in (0, 1)$, both of which are independent of the mesh size h, such that

$$-\alpha_0 \leq \frac{((I - M_K A)v, v)_A}{(v, v)_A} \leq \alpha_1$$

and thus

$$1 - \alpha_1 \leq \frac{(M_K A v, v)_A}{(v, v)_A} \leq 1 + \alpha_0.$$

Therefore, since $M_K A$ is SPD respect to the A-inner product, according to the property of the Rayleigh quotient, we have

$$\lambda_{max}(M_K A) \leq \sup_{v \neq 0} \frac{(M_K A v, v)_A}{(v, v)_A} \leq 1 + \alpha_0,$$

and

$$\lambda_{min}(M_K A) \geq \inf_{v \neq 0} \frac{(M_K A v, v)_A}{(v, v)_A} \geq 1 - \alpha_1.$$

Thus, we obtain

$$cond(M_K A) = \frac{\lambda_{max}(M_K A)}{\lambda_{min}(M_K A)} \leq \frac{1 + \alpha_0}{1 - \alpha_1} \equiv C'$$

and

$$||r_k||_{M_K} \leq 2C^k ||r_0||_{M_K} \quad \text{with} \quad C \equiv \frac{\sqrt{C'} - 1}{\sqrt{C'} + 1}.$$

This completes the proof. □

Due to these two theorems, the constructed finite difference solver can optimally solve each elliptic boundary value problem on each neighboring box, which makes it possible to significantly improve the efficiency compared to the finite element method. Furthermore, to maximize the efficiency of the solver, we programmed the PCG-MG solver in `Fortran` without storing the mesh data or the coefficient matrix A of linear system (2.8), and pre-allocated all the necessary temporary memories required in the PCG or the multigrid V-cycle algorithm, which would be shared and repeatedly used in the box iterative method on the six neighboring boxes.

2.4 Validation tests for the hybrid method

We programmed the new hybrid method for solving the class of interface problems (2.1) in `Fortran`. Before we move to the application of this new method, we firstly consider the following linear interface boundary value problem with available analytical solution reported

in [24] for the verification purpose:

$$\begin{cases} \Delta\Psi(\mathbf{r}) = 0 & \text{in } D_p, \\ -\epsilon_s \Delta\Psi(\mathbf{r}) = f_s(\mathbf{r}) & \text{in } D_s, \\ \Psi(\mathbf{s}^-) = \Psi(\mathbf{s}^+), \epsilon_p \dfrac{\partial \Psi(\mathbf{s}^-)}{\partial \mathbf{n}} = \epsilon_s \dfrac{\partial \Psi(\mathbf{s}^+)}{\partial \mathbf{n}} + (\epsilon_s - \epsilon_p) \dfrac{\partial G(\mathbf{s})}{\partial \mathbf{n}} & \text{on } \Gamma, \\ \Psi(\mathbf{s}) = U(\mathbf{s}) & \text{on } \partial\Omega, \end{cases} \tag{2.9}$$

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 by

$$G(\mathbf{r}) = \frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} \frac{z_j}{|\mathbf{r} - \mathbf{r}_j|}, \tag{2.10}$$

and

$$\frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} = \nabla G \cdot \mathbf{n}$$

with ∇G being given by

$$\nabla G(\mathbf{r}) = -\frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} z_j \frac{\mathbf{r} - \mathbf{r}_j}{|\mathbf{r} - \mathbf{r}_j|^3}. \tag{2.11}$$

Furthermore, f_s is given by

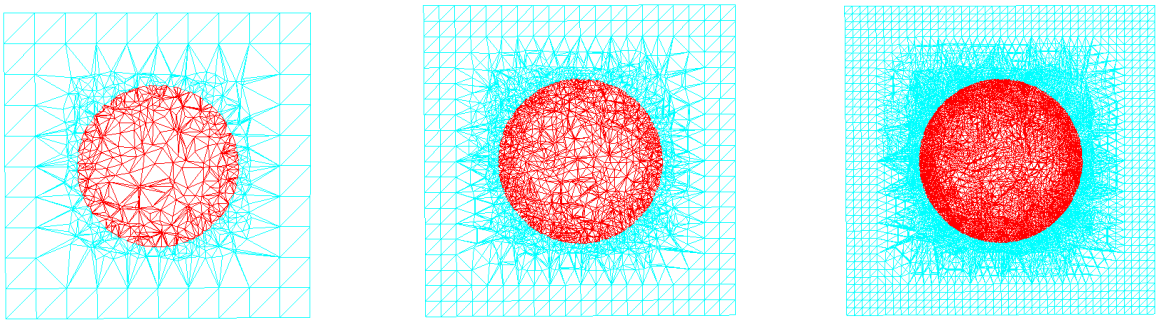
$$f_s(\mathbf{r}) = \frac{\alpha(\epsilon_p - \epsilon_s)}{4\pi a^2 \epsilon_p} \sum_{j=1}^{n_p} z_j \left[\left(\frac{7|\mathbf{r}|^2 - 5\mathbf{r}\cdot\mathbf{r}_j}{|\mathbf{r} - \mathbf{r}_j|^3} - 6\frac{(|\mathbf{r}|^2 - \mathbf{r}\cdot\mathbf{r}_j)^2}{|\mathbf{r} - \mathbf{r}_j|^5} \right) \cos\left(\frac{|\mathbf{r}|^2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

and U is given by

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

For this test model, it is easy to see that the analytical solution is $\Psi(\mathbf{r}) = 0$ for $\mathbf{r} \in D_p$ and $\Psi(\mathbf{r}) = U(\mathbf{r})$ for $\mathbf{r} \in D_s$.

In numerical tests, we set $\epsilon_p = 2.0$, $\epsilon_s = 80.0$, $\alpha = 1.0$, $a = 1$, and $A = 3$. The charge number z_j and the atomic position $\mathbf{r}_j$ were obtained from a PQR file of a protein with PDB



(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

Figure 2.3: Cross sections of 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.

ID 2LZX, which has 488 atoms (i.e., $n_p = 488$). Here we divided each atomic position $\mathbf{r}_j$ by 17 to rescale it into the unit spherical region D_p without modifying the protein structure.

Furthermore, we set $D = (-1.5, 1.5)^3$, $\tau = 0.375$, and $\eta = 1.5$, from which we obtained the seven overlapped 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, 57515, 448773, respectively. For these nested meshes, the corresponding three unstructured tetrahedral meshes of Ω_7, denoted by Ω_{7,h_j} for $j = 1, 2, 3$, were found to have 3933, 30839, and 243069 vertices, respectively. Meanwhile, we got the mesh sizes of these three uniform meshes on $\Omega \setminus D$ and the finite element meshes on Ω_7, which are

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

and

$$h_{7,1} = 0.7389, h_{7,2} = 0.4058, \text{ and } h_{7,3} = 0.2307,$$

respectively. Here the mesh size $h_{7,j}$ of the unstructured mesh on Ω_{7,h_j} is defined as the largest edge length among all tetrahedra. Figure 2.3 displays the cross section views of these nested tetrahedral meshes. The numerical results were reported in Table 2.1. Here the absolute error E_a between the analytical solution Ψ and the numerical solution Ψ_h is defined

Mesh of Ω	Number of Mesh Nodes	Absolute Error E_a of (2.12)	Order of Convergence
Ω_{h_1}	7,515	0.3248	
Ω_{h_2}	57,515	0.0791	2.037
Ω_{h_3}	448,773	0.0199	1.991

Table 2.1: Errors of the new hybrid method on three nested meshes for solving the test model (2.9).

as follows

$$E_a(\Psi) = \sqrt{\int_{\Omega_7} |\Psi(\mathbf{r}) - \Psi_h(\mathbf{r})|^2 d\mathbf{r} + \sum_{\mathbf{r}^j \in (\Omega - \Omega_7)_h} h^3 [\Psi(\mathbf{r}^j) - \Psi_h(\mathbf{r}^j)]^2}, \tag{2.12}$$

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

From Table 2.1 it can be seen that the absolute error E_a of the constructed overlapped box iterative method on the special seven box partition was reduced about three fourths when the mesh size h was reduced approximately by half, due to use the seven-point finite difference stencil and linear Lagrange finite element space in the hybrid method, resulting in a second order convergence rate approximately and well matching the mathematical theory. This well validated the program of the new hybrid method. As for the properties that the convergence rates of both the new box iterative method and the optimal finite difference solver are independent of mesh size, they will be verified using the SMPBE and NMPBE test models in the following chapters.

Chapter 3

Size modified Poisson Boltzmann equation

It has been well known that PBE has some drawbacks in applications due to ignore the ion sizes and the polarization correlations of water molecules. In this chapter, we will consider the new dielectric continuum model incorporating the steric effects. Then the new hybrid method will be applied to solve SMPBE for the uniform ion size case. Numerical results will be given to show the improvement of SMPBE in predicting the ion concentrations. Meanwhile, the comparison results on six proteins with different net charges and atom numbers will be done to show the significant efficiency improvement of the new hybrid solver. For the nonuniform ion size case, there is no explicit PDE form available and instead a PDE-constrained problem must be solved. The development of its numerical solver will be considered to be future work.

3.1 Review of the finite element SMPBE solver

Recently, a finite element SMPBE program package using a new three-term solution decomposition scheme has been proposed in [2], which turned out to work efficiently for both test models and proteins with different net charges. Although this finite element solver has shown sharp improvements in terms of efficiency and accuracy compared to other SMPBE program packages, we intent to develop a new hybrid SMPBE solver to further improve the efficiency using the new hybrid method. In this section, we will firstly give the dimensionless formulation of SMPBE and then give a short review of the important techniques used in the finite element solver.

Under the SI system with the length unit of angstrom (Å), when we measure the electrostatic potentials u in units k_BT/e_c, the dimensionless SMPBE is given as follows [2, 62]:

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha \sum_{j=1}^{n_p} z_j \delta_{\mathbf{r}_j}, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta u(\mathbf{r}) + \dfrac{\kappa^2 \sinh(u)}{1 + 2M\Lambda^3 \cosh(u)} = 0, & \mathbf{r} \in D_s, \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), \quad \epsilon_s \dfrac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \dfrac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ u(\mathbf{s}) = g(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{3.1}$$

where $\mathbf{r}_j$ and z_j are the position and charge number of the jth atom, respectively, g is a boundary function (SMPBE is a second-order jump interface elliptic problem on $\mathbb{R}^3$ and equation (3.1) is a boundary value problem after truncating the computational domain to Ω. Therefore, it is critical to choose the appropriate boundary value function g in order to reduce the "truncation" error. The common way is to set the function g to zero or the size modified Multiple Debye-Hückel (MDH) function for a sufficiently large domain Ω), $\partial\Omega$ denotes the boundary of Ω, $\delta_{\mathbf{r}_j}$ is the Dirac Delta distribution at the point $\mathbf{r}_j$, $\mathbf{n}(\mathbf{s})$ is the unit outward normal vector of D_p, Λ^3 denotes the uniform ion sizes, and the constants α, κ, and M are defined by

$$\alpha = \frac{10^{10} e_c^2}{\epsilon_0 k_B T}, \quad \kappa^2 = 2I_s \frac{10^{-17} N_A e_c^2}{\epsilon_0 k_B T}, \quad \text{and} \quad M = 10^{-27} N_A I_s$$

with N_A and I_s being the Avogadro number and the ionic strength in mole/liter, respectively. For $T = 298.15$ and $I_s = 0.1$, α, κ, and M can be estimated by

$$\alpha \approx 7042.94, \quad \kappa^2 \approx 0.848272, \quad \text{and} \quad M \approx 6.022 \times 10^{-5}, \tag{3.2}$$

which will be used in the numerical tests.

According to [2], a solution decomposition scheme is used to isolate the singularities caused by the Dirac Delta distributions, which splits the solution u of equation (3.1) into three components as follows

$$u = G + \Psi + \tilde{\Phi}, \tag{3.3}$$

where G is a function given by (2.10), Ψ is a solution of the following linear interface boundary

value problem

$$\begin{cases} \Delta\Psi(\mathbf{r}) = 0, & \mathbf{r} \in D_p \cup D_s, \\ \Psi(\mathbf{s}^+) = \Psi(\mathbf{s}^-), & \mathbf{s} \in \Gamma, \\ \epsilon_s \dfrac{\partial \Psi(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \dfrac{\partial \Psi(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_p - \epsilon_s) \dfrac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \Psi(\mathbf{s}) = g - G(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{3.4}$$

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

$$\begin{cases} \Delta\tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta\tilde{\Phi}(\mathbf{r}) + \dfrac{\kappa^2 \sinh(G + \Psi + \tilde{\Phi})}{1 + 2M\Lambda^3 \cosh(G + \Psi + \tilde{\Phi})} = 0, & \mathbf{r} \in D_s, \\ \tilde{\Phi}(\mathbf{s}^+) = \tilde{\Phi}(\mathbf{s}^-), \quad \epsilon_s \dfrac{\partial \tilde{\Phi}(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \dfrac{\partial \tilde{\Phi}(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \tilde{\Phi}(\mathbf{s}) = 0, & \mathbf{s} \in \partial\Omega. \end{cases} \tag{3.5}$$

Here $\dfrac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} = \nabla G \cdot \mathbf{n}$ with ∇G being given by (2.11). Hence, to get a numerical solution u, we only need to solve (3.4) for Ψ and then (3.5) for $\tilde{\Phi}$. An effective finite element scheme has been given in [2, Algorithms 1 and 2]. For completeness, we give it a short review.

In the finite element scheme, we denote by $\mathcal{M}$ a finite element space of the usual Sobolev function space $H^1(\Omega)$. A finite element solution Ψ of (3.4) is firstly found by solving the following equivalent linear variational problem:

Find a $\Psi \in \mathcal{M}$ with $\Psi|_{\partial\Omega} = g - G$ such that

$$a(\Psi, v) = (\epsilon_p - \epsilon_s) \int_{D_s} \nabla G(\mathbf{r}) \cdot \nabla v(\mathbf{r}) d\mathbf{r} \quad \forall v \in \mathcal{M}_0, \tag{3.6}$$

where $a(u, v)$ is a bilinear form defined by

$$a(u, v) = \epsilon_p \int_{D_p} \nabla u(\mathbf{r}) \cdot \nabla v(\mathbf{r}) d\mathbf{r} + \epsilon_s \int_{D_s} \nabla u(\mathbf{r}) \cdot \nabla v(\mathbf{r}) d\mathbf{r}, \tag{3.7}$$

where $\mathcal{M}_0 = \{v \in \mathcal{M} \mid v = 0 \text{ on } \partial\Omega\}$, a subspace of the Sobolev function space $H_0^1(\Omega)$. And the nonlinear equation (3.5) is then reformulated as the equivalent minimization problem:

$$J(\tilde{\Phi}) = \min_{v \in \mathcal{M}_0} J(v), \tag{3.8}$$

with J being defined by

$$J(v) = \frac{1}{2} a(v, v) + \frac{\kappa^2}{2M\Lambda^3} \int_{D_s} \ln(1 + 2M\Lambda^3 \cosh(U + v)) d\mathbf{r},$$

where $U = G + \Psi$, which has been pre-calculated. The above minimization problem is then solved by the Newton iterative method:

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

where $\tilde{\Phi}^{(k)}$ denotes the k-th iterate of the Newton minimization method, $\tilde{\Phi}^{(0)}$ is an initial guess, λ_k is a step length, and p_k is a search direction generated from the Newton equation in the variational form: Find a $p_k \in \mathcal{M}_0$ such that

$$J''(\tilde{\Phi}^{(k)})(p_k, v) = -J'(\tilde{\Phi}^{(k)})v \quad \forall v \in \mathcal{M}_0, \tag{3.9}$$

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

$$J'(\tilde{\Phi})v = a(\tilde{\Phi}, v) + \kappa^2 \int_{D_s} \frac{\sinh(U + \tilde{\Phi})}{1 + 2M\Lambda^3 \cosh(U + \tilde{\Phi})} v d\mathbf{r} \quad \forall v \in H_0^1(\Omega),$$

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

$$J''(\tilde{\Phi})(p, v) = a(p, v) + \kappa^2 \int_{D_s} \frac{2M\Lambda^3 + \cosh(U + \tilde{\Phi})}{(1 + 2M\Lambda^3 \cosh(U + \tilde{\Phi}))^2} p v d\mathbf{r} \quad \forall p, v \in H_0^1(\Omega).$$

Similar to the finite element PBE program package, in the implementation, the initial iterate $\tilde{\Phi}^{(0)}$ can be selected as zero or a solution of a linearized equation of (3.5). Meanwhile, to deal with the strong nonlinearity, an 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 problems of the hyperbolic terms. Each Newton equation of (3.9) is solved numerically by PCG-ILU with the absolute and the relative residue errors less than a given tolerance (10^{-10} by default).

3.2 Reformulation of the Newton equation

As the key step to apply the hybrid method, we firstly have the following theorem to get a PDE reformulation of the Newton equation (3.9).

Theorem 3.2.1. *If $p \in V$, where $V = H_0^1(\Omega) \cap H^2(D_p) \cap H^2(D_s)$, then the variation problem* (3.9) *is equivalent to the following boundary value problem*

$$-\Delta p(\mathbf{r}) = \Delta\tilde{\Phi}(\mathbf{r}), \quad \mathbf{r} \in D_p, \tag{3.10a}$$

$$-\epsilon_s \Delta p(\mathbf{r}) + \kappa^2 \frac{2M\Lambda^3 + \cosh(U+\tilde{\Phi})}{(1+2M\Lambda^3\cosh(U+\tilde{\Phi}))^2} p = \epsilon_s \Delta\tilde{\Phi} - \kappa^2 \frac{\sinh(U+\tilde{\Phi})}{1+2M\Lambda^3\cosh(U+\tilde{\Phi})}, \quad \mathbf{r} \in D_s, \tag{3.10b}$$

$$p(\mathbf{s}^+) = p(\mathbf{s}^-),\ \epsilon_s \frac{\partial p(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} - \epsilon_p \frac{\partial p(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_p \frac{\partial \tilde{\Phi}(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} - \epsilon_s \frac{\partial \tilde{\Phi}(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})}, \quad \mathbf{s} \in \Gamma, \tag{3.10c}$$

$$p(\mathbf{s}) = 0, \quad \mathbf{s} \in \partial\Omega, \tag{3.10d}$$

where $\tilde{\Phi}$ is a given function of V, Ψ is a given solution of (3.4) *and function G is defined in* (2.10).

Proof. We only show the derivation of (3.10) from the variational form (3.9) 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 (3.9) 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 (3.10a).

Next, for any $v \in H_0^1(\Omega)$ satisfying $v = 0$ on D_p and $v \in \mathbb{C}_0^\infty(D_s)$, (3.9) can be reduced to the form

$$\begin{aligned} &\epsilon_s \int_{D_s} \nabla p \cdot \nabla v d\mathbf{r} + \kappa^2 \int_{D_s} \frac{2M\Lambda^3 + \cosh(U+\tilde{\Phi})}{(1+2M\Lambda^3\cosh(U+\tilde{\Phi}))^2} p v d\mathbf{r} \\ =& -\epsilon_s \int_{D_s} \nabla\tilde{\Phi} \cdot \nabla v d\mathbf{r} - \kappa^2 \int_{D_s} \frac{\sinh(U+\tilde{\Phi})}{1+2M\Lambda^3\cosh(U+\tilde{\Phi})} v d\mathbf{r}. \end{aligned}$$

By the Green's identity, the above equality can be reformulated as

$$\begin{aligned} &-\epsilon_s \int_{D_s} \Delta p v d\mathbf{r} + \kappa^2 \int_{D_s} \frac{2M\Lambda^3 + \cosh(U+\tilde{\Phi})}{(1+2M\Lambda^3\cosh(U+\tilde{\Phi}))^2} p v d\mathbf{r} \\ =&\epsilon_s \int_{D_s} \Delta\tilde{\Phi} v d\mathbf{r} - \kappa^2 \int_{D_s} \frac{\sinh(U+\tilde{\Phi})}{1+2M\Lambda^3\cosh(U+\tilde{\Phi})} v d\mathbf{r}, \end{aligned}$$

from which we can obtain equation (3.10b).

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 (3.9) as

$$\begin{aligned}
&\int_\Gamma \left[\epsilon_p \frac{\partial p(\mathbf{s}^-)}{\partial \mathbf{n}} - \epsilon_s \frac{\partial p(\mathbf{s}^+)}{\partial \mathbf{n}}\right] v d\mathbf{s} - \epsilon_p \int_{D_p} \Delta p v d\mathbf{r} - \\
&\int_{D_s} \left[\epsilon_s \Delta p - \kappa^2 \frac{2M\Lambda^3 + \cosh(U+\tilde{\Phi})}{(1+2M\Lambda^3\cosh(U+\tilde{\Phi}))^2} p\right] v d\mathbf{r} \\
= &\int_\Gamma \left[\epsilon_s \frac{\partial \tilde{\Phi}(\mathbf{s}^+)}{\partial \mathbf{n}} - \epsilon_p \frac{\partial \tilde{\Phi}(\mathbf{s}^-)}{\partial \mathbf{n}}\right] v d\mathbf{s} + \epsilon_p \int_{D_p} \Delta \tilde{\Phi} v d\mathbf{r} + \\
&\int_{D_s} \left[\epsilon_s \Delta \tilde{\Phi} - \kappa^2 \int_{D_s} \frac{\sinh(U+\tilde{\Phi})}{1+2M\Lambda^3\cosh(U+\tilde{\Phi})}\right] v d\mathbf{r}.
\end{aligned}$$

Applying (3.10a) and (3.10b) to the above identity leads to the interface condition (3.10c). The boundary (3.10d) condition is natural because of $p \in V$. This completes the proof. □

3.3 New hybrid solver for SMPBE

Applying the hybrid method to solve the linear interface problem (3.4) for Ψ and (3.10) for the search direction p_k, we can modify the Algorithms in [2] to solve SMPBE. For clarity, the new algorithm to efficiently solve SMPBE is given as follows:

Algorithm for Solving SMPBE. Let the special overlapped box iterative method be defined in (2.2) with the PCG-MG algorithm for solving a finite difference system of (2.3) on a uniform mesh of the box Ω_i for $i = 1$ to 6 and the PCG-ILU algorithm for solving a finite element system of (2.4) on an interface-fitted tetrahedral mesh of the central box Ω_7. A numerical solution u of equation (3.1) is calculated approximately in the following five steps:

Step 1. Construct an interface-matched tetrahedral mesh for the central box Ω_7 and a uniform mesh for each neighboring box Ω_i for $i \neq 7$ with a mesh size $h > 0$.

Step 2. Calculate G on each box and ∇G on Ω_7 according to (2.10) and (2.11), respectively.

Step 3. Calculate Ψ of (3.4) by the overlapped box iterative method.

Step 4. Calculate $\tilde{\Phi}$ by the modified Newton method in the following steps:

(a) Set $k = 0$ and $\tilde{\Phi}^{(0)} = 0$ (by default).

(b) Calculate the search direction p_k by the overlapped box iterative method.

(c) Find the steplength λ_k by a line search algorithm (starting with $\lambda_k = 1$).

(d) Define the modified Newton iterate $\tilde{\Phi}^{(k+1)}$ by $\tilde{\Phi}^{(k+1)} = \tilde{\Phi}^{(k)} + \lambda_k p_k$.

(e) Check the convergence: If $\|\tilde{\Phi}^{(k+1)} - \tilde{\Phi}^{(k)}\| \leq 10^{-7}$ (by default), then $\tilde{\Phi}^{(k+1)}$ is set as a solution $\tilde{\Phi}$ of the nonlinear interface problem (3.5); otherwise, increase k by 1 and go back to (b).

Step 5. Construct a numerical solution u of SMPBE by the solution decomposition $u = G + \Psi + \tilde{\Phi}$.

We programmed the new algorithm for solving SMPBE in `C`, `Fortran`, and `Python` as a software package. Similar to the finite element SMPBE solver, the main program of the software was written in `Python` based on the state-of-the-art finite element library `DOLFIN` from the `FEniCS` project [39, 40]. Each finite element equation is produced by `DOLFIN`, and solved by PCG-ILU from the `PETSc` library [97]. The input of the program is a PQR file of a biomolecule, which contains the positions $\mathbf{r}_j$, the charge numbers z_j, and the radii of atoms as well as the related hydrogen atoms. We further wrote a `Fortran` subroutine to speed up the calculations of the values of functions G and ∇G at each mesh point. All

`Fortran` subroutines and `C` programs were converted to the Python external modules by the Fortran-to-Python interface generator `f2py` (`http://cens.ioc.ee/projects/f2py2e/`) and `SWIG` (`http://www.swig.org`), respectively. Thus, they can be directly called by the `Python` main program.

3.4 Numerical results

In this section, we report the numerical experiments we made using the new `Python` program. For simplicity, we set $\epsilon_p = 2.0$, $\epsilon_s = 80.0$, $\Lambda = 3.11$, $T = 298.15$, $I_s = 0.1$, and all the numerical tests were done by using the default values of the parameters (except the parameters m, n, μ of (2.6)) on one processor of a Mac Pro Workstation with the 3.7 GHZ Quad-Core Intel Xeon E5 and 64 GB memory.

3.4.1 Validation tests

Using the superposition principle and the rotational symmetry, we have obtained the analytical solution, a simple series expression in terms of Legendre polynomials, of a Poisson equation with a spherical solute domain containing arbitrary number of changes [57]. Using this analytical solution U, we artificially construct the following SMPBE test model:

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha \sum_{n=1}^{n_p} z_j \delta_{\mathbf{r}_j} & \text{in } D_p, \\ -\epsilon_s \Delta u(\mathbf{r}) + \dfrac{\kappa^2 \sinh(u)}{1 + 2M\Lambda^3 \cosh(u)} = F_s(\mathbf{r}) & \text{in } D_s, \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), \quad \epsilon_s \dfrac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}} = \epsilon_p \dfrac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}} & \text{on } \Gamma, \\ u(\mathbf{r}) = U(\mathbf{r}) & \text{on } \partial\Omega, \end{cases} \tag{3.11}$$

where $D_p = \{\mathbf{r} \mid \|\mathbf{r}\| < a\}$ with $a > 0$, $\Gamma = \{\mathbf{r} \mid \|\mathbf{r}\| = a\}$, Ω is a cubic domain, $D_s = \Omega - D_p - \Gamma$ is nonempty, and $F_s(\mathbf{r}) = \kappa^2 \sinh(U(\mathbf{r}))[1 + 2M\Lambda^3 \cosh(U(\mathbf{r}))]^{-1}$, which can be understood as an excess charge density function. For this SMPBE test model, clearly, $U(\mathbf{r})$ is still the analytical solution of (3.11).

In numerical tests, we set $a = 1$, $\Omega = (-6, 6)^3$, and constructed an overlapped box partition of Ω using $D = (-2, 2)^3$, $\tau = 1$, and $\eta = 4$, which gave $\Omega_7 = (-3, 3)^3$. The over-relaxation parameter ω of the overlapped box iterative method was set to 1.275 and

1.225 in solving (3.4) for Ψ and (3.10) for p_k, respectively. Three nested meshes with the uniform mesh sizes $h = 0.25$, $h/2$ (0.125), and $h/4$ (0.0625) were constructed for testing the convergence behavior of the hybrid solver. Their numbers of mesh points were found to be 120887, 940247, and 7412989, respectively, including the numbers of mesh points from the finite element meshes of the central box Ω_7, which were 18863, 145223, and 1136605, respectively.

Table 3.1 reports the numerical results for two validation tests. In the first case, the unit ball region D_p has only one central charge ($n_p = 1$). Such a test model is often referred to as a Born ball test model. Its analytical solution u can be given in the closed form as follows

$$u(\mathbf{r}) = \begin{cases} \dfrac{\alpha}{4\pi}(\dfrac{1}{\epsilon_s} - \dfrac{1}{\epsilon_p}) + \dfrac{\alpha}{4\pi\epsilon_p|\mathbf{r}|} & \text{in } D_p, \\ \dfrac{\alpha}{4\pi\epsilon_s|\mathbf{r}|} & \text{in } D_s. \end{cases} \tag{3.12}$$

In the second case ($n_p = 488$), we assign the 488 atomic charges of a protein molecule (PDB ID: 2LZX) to the unit ball region D_p through dividing each atomic position $\mathbf{r}_j$ by 19. The structure of the protein is preserved in the unit ball. In Table 3.1, for those two cases, the numbers of iterations for PCG-MG, PCG-ILU, and the overlapped box iterative method were their averages defined as total iteration numbers over the total number of the linear systems solved.

Mesh size h	Error $\dfrac{\|u-u_h\|_{l^2(\Omega)}}{\|u\|_{l^2\Omega}}$	PCG-MG Iter. on Ω_i ($i = 1$ to 6)	PCG-ILU Iter. on Ω_7	Hybrid Box Iter. on Ω	Newton Iter.	Order
Case 1: The region D_p containing one central charge only						
0.25	5.77×10^{-2}	$8.54 \approx 9$	$7.89 \approx 8$	$11.8 \approx 12$	5	-
0.25/2	1.56×10^{-2}	$8.13 \approx 8$	$13.74 \approx 14$	$11.4 \approx 11$	10	1.89
0.25/4	3.62×10^{-3}	$8.28 \approx 8$	$22.98 \approx 23$	$11.0 \approx 11$	11	2.11
Case 2: The region D_p containing 488 point charges from a protein (2LZX)						
0.25	9.10×10^{-2}	$8.94 \approx 9$	$7.97 \approx 8$	$11.7 \approx 12$	5	-
0.25/2	2.34×10^{-2}	$8.95 \approx 9$	$13.36 \approx 13$	$12.2 \approx 12$	5	1.96
0.25/4	5.31×10^{-3}	$9.61 \approx 10$	$25.11 \approx 25$	$11.0 \approx 11$	5	2.14

Table 3.1: Performance of the hybrid SMPBE solver for the SMPBE test model (3.11) in relative solution errors and average iteration numbers (Iter.).

From Table 3.1 it can be seen that the errors were reduced almost by three fourths as the mesh size h was decreased approximately by half, indicating that the convergence order of

the hybrid SMPBE solver is around 2. The number of Newton iterations was only up to 11, indicating the new hybrid modified Newton iterative algorithm for computing $\tilde{\Phi}$ retained a fast convergence rate of the modified Newton method. The average numbers of PCG-MG iterations were about 9 for these three different mesh sizes, numerically confirming that the PCG-MG solver has a convergence rate independent of the mesh size h. Furthermore, the average numbers of the box iterations were around 11, showing it also has a convergence rate independent of mesh size.

In these tests, the numbers of PCG-ILU iterations were small, showing the efficiency of PCG-ILU for solving each finite element linear system on the central box Ω_7. Although it increased with the reduction of the mesh size, PCG-ILU was found to take much less CPU runtime than a PCG using an algebraic multigrid preconditioner, called *amg_hypre*, from `PETSc`. The test problem sizes might not be large enough to take the advantage of an algebraic multigrid preconditioner.

3.4.2 Ion concentrations of a dipole test model

The SMPBE/PBE test model with D_p being a unit ball only containing a central charge is a common test model to demonstrate that SMPBE is a better dielectric continuum model than PBE in prediction of the ion concentrations. We did tests on it using the hybrid solver and got the same results as the ones reported in [2]. Furthermore, we did tests on a more interesting dipole model, in which D_p consists of two overlapped balls with the same radius r and two opposite central charges.

From the derivations of PBE [42] and SMPBE [62], for a salt solution consisting of sodium (Na^+) and chloride (Cl^-) ions, the concentrations C_{Na} and C_{cl} of sodium (Na^+) and chloride (Cl^-) ions are estimated (in mole per liter) by:

$$C_{Na} = \begin{cases} \dfrac{I_s e^{-u}}{1 + 2M\Lambda^3 \cosh u} & \text{for SMPBE,} \\ I_s e^{-u} & \text{for PBE,} \end{cases}$$
$$C_{cl} = \begin{cases} \dfrac{I_s e^{u}}{1 + 2M\Lambda^3 \cosh u} & \text{for SMPBE,} \\ I_s e^{u} & \text{for PBE,} \end{cases} \tag{3.13}$$

where I_s is a ionic strength in *mole/liter*, and constant M is given in (3.2). In the numerical tests, we set $r = 1.5$ Å, a positive charge of $+3e_c$ at the center $(1, 0, 0)$, and a negative

charge of $-3e_c$ at $(-1, 0, 0)$. We then constructed an overlapped box partition and meshes with $D = (-4, 4)^3$, $\mu = 2$, $n = 5$ and $m = 2$ according to the formulas given in Section 2.3 to get $\Omega = (-12, 12)^3$ and $\Omega_7 = (-5, 5)^3$. The meshes of Ω and Ω_7 had 900740 and 56988 mesh points, respectively. A cross-section view of the finite element mesh of Ω_7 on the xy-plane is given in Figure 3.1. The boundary value function g was set to zero here.

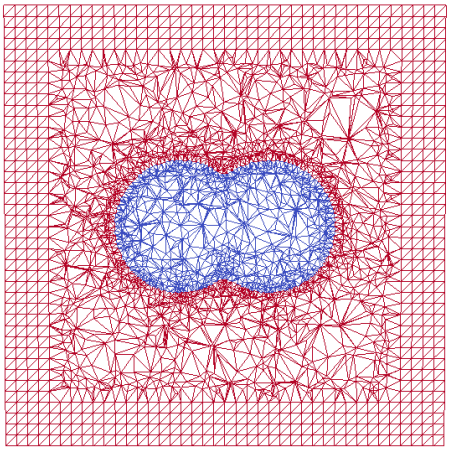


Figure 3.1: A cross section on the xy plane of the mesh on Ω_7 for the dipole test case.

Figure 3.2 displays the electrostatic field E on the xy coordinate plane, which we calculated by the formula $E = -\nabla u$ using a numerical solution u produced by the hybrid SMPBE solver. From Figure 3.2 it can be seen that the electrostatic field lines emanated from the positive charged sphere and extended radially toward the negative charged sphere. Since the two balls have an identical quantity of charge, their abilities to alter the space surrounding them are the same. Hence, the electrostatic field around them should occur in a symmetric pattern. As shown in the figure, these basic features of the electrostatic field lines were well captured by the numerical solution, which partially validated the SMPBE program package.

Figure 3.3 displays the two concentrations C_{Na} and C_{cl} predicted by the hybrid SMPBE solver on the xy coordinate plane according to the formulas given in (3.13). From the figure, we can see they have reasonably reached the saturation value, 55.2, claimed in Physics (i.e., $10^{27}/(N_A\Lambda^3) \approx 55.2$). Meanwhile, the predicted values of C_{Na} and C_{cl} were distributed symmetrically around the surface of these two balls, well matching the law of electrostatic attraction.

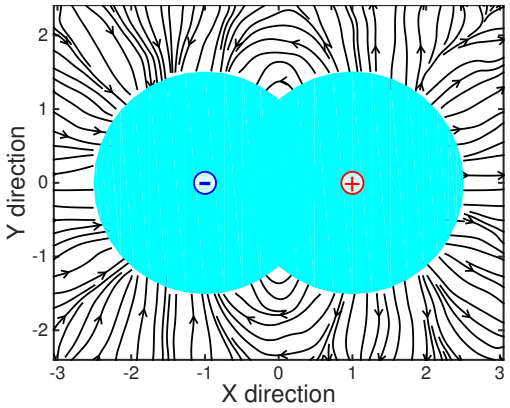
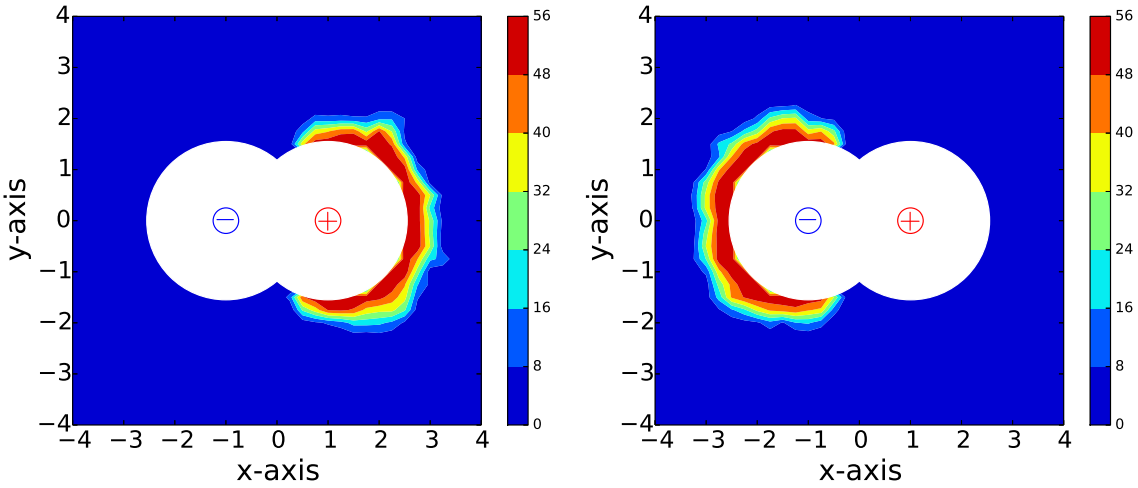


Figure 3.2: The electrostatic field $E = -\nabla u$ calculated by the numerical solution u of the new hybrid SMPBE solver.



(a) Concentration C_{cl} of anions Cl^- (b) Concentration C_{Na} of cations Na^+

Figure 3.3: The concentrations of anions (Cl^-) and cations (Na^+) predicted by the hybrid SMPBE solver for the dipole model on the xy-plane.

We also repeated the above tests using $\Lambda = 0$, i.e., PBE case, to compare the obtained results. Due to ignore the ion sizes, unreasonable amount of ions were cumulated near the surface of the dipole and the concentrations C_{Na} and C_{cl} were found to be unreasonably large (over 1000, which exceeds the physical maximal number already) around the spherical surfaces. From this test case, we can easily see the improvement of SMPBE in terms of predicting the ion concentrations.

3.4.3 Performance tests on proteins

We made numerical experiments on the six proteins tested in [2] to show that the hybrid SMPBE program package can improve the performance of the finite element program package significantly. These six proteins have PDB ID 1CBN, 1SVR, 4PTI, 1AZQ, 1D3X and 1TC3, respectively, which can be downloaded from the Protein Data Bank. Their PDB files were then converted to the PQR files by `PDB2PQR` [90]. These six protein molecules have 642, 1433, 892, 1603, 756, and 2124 atoms, and $0e_c$, $-2e_c$, $+6e_c$, $-8e_c$, $-21e_c$, and $-35e_c$ net charges, respectively. Meanwhile, for these six proteins, the boxes of D produced from the mesh generation program are listed as follows:

$$D = (-9.5, 28.1) \times (-9.1, 28.5) \times (-11.7, 25.9) \text{ for 1CBN},$$
$$D = (-24.8, 27.3) \times (-30.4, 21.7) \times (-24.5, 27.6) \text{ for 1SVR},$$
$$D = (-7.8, 38.4) \times (-2.3, 43.9) \times (-18.6, 27.6) \text{ for 4PTI},$$
$$D = (-20.9, 40.3) \times (-19.6, 41.6) \times (-19.2, 42.0) \text{ for 1AZQ},$$
$$D = (-21.1, 20.7) \times (-21.6, 20.2) \times (-20.4, 21.4) \text{ for 1D3X},$$
$$D = (-30.3, 51.0) \times (91.4, 172.7) \times (-8.1, 73.2) \text{ for 1TC3}.$$

In numerical tests, we used $\mu = 4$, $m = 2$, and $n = 3$ to construct a cubic domain of Ω and an overlapped box partition of Ω according to (2.6) for each protein. The over-relaxation parameter ω was set to 1.225 and 1.015 in solving (3.4) for Ψ and (3.10) for p_k, respectively. The boundary value function g was set to zero, and the initial iterate $\tilde{\Phi}^{(0)}$ as a numerical solution of the linearized SMPBE reported in [2].

Because the convergence rate of the PCG-MG solver is independent of the mesh size h, the hybrid solver particularly works well on a finer mesh. To demonstrate this feature, we used $m = 3$ and $n = 4$ to refine the meshes, and then repeated all the tests on these proteins.

Furthermore, we repeated all the tests by the finite element SMPBE program package. Here each tetrahedral mesh of Ω was produced from the corresponding mesh of each protein used by the hybrid solver. That is, each gird of the uniform finite difference mesh was divided into six tetrahedra, making the mesh for the finite element solver have the same number of mesh points on Ω as the mesh used in the hybrid solver.

Table 3.2 reports these numerical results. In the case of the hybrid solver, the construction of a finite element mesh and the calculation of ∇G were done on Ω_7 only. Hence, they took

Number of mesh points	Find G & ∇G		Find Ψ		Find $\tilde{\Phi}$		Total Time	
	Hybrid	FE	Hybrid	FE	Hybrid	FE	Hybrid	FE
Protein with PDB ID 1CBN (642 atoms and $0e_c$ net charge)								
77,969	0.42	0.76	0.91	1.99	2.32	8.02	4.17	11.33
537,953	2.63	5.23	4.71	14.97	8.31	50.21	17.81	74.09
Protein with PDB ID 1SVR (1433 atoms and $-2e_c$ net charge)								
89,850	1.40	2.37	1.77	2.53	5.05	10.71	9.14	16.25
550,170	6.96	14.49	5.96	15.46	11.40	60.10	27.02	93.84
Protein with PDB ID 4PTI (892 atoms and $+6e_c$ net charge)								
81,356	0.74	1.33	1.20	2.12	3.35	9.47	5.92	13.54
541,329	4.19	8.84	5.47	14.96	10.21	50.45	22.30	77.97
Protein with PDB ID 1AZQ (1603 atoms and $-8e_c$ net charge)								
89,089	1.54	2.62	1.69	2.40	4.73	10.54	8.87	16.20
549,279	7.75	16.14	5.94	15.27	13.21	57.29	29.76	92.49
Protein with PDB ID 1D3X (756 atoms and $-21e_c$ net charge)								
82,897	0.64	1.15	1.32	2.15	4.14	10.65	6.77	14.54
542,878	3.58	7.54	5.86	14.76	12.78	57.48	24.59	83.53
Protein with PDB ID 1TC3 (2124 atoms and $-35e_c$ net charge)								
101,944	2.54	3.99	2.90	2.91	10.05	16.81	16.90	24.45
564,871	10.88	21.98	7.58	15.65	23.50	85.29	45.45	126.82

Table 3.2: A comparison of the performance in CPU time (in seconds) of the hybrid solver (Hybrid) with that of the finite element solver (FE) reported in [2] in the calculation of component functions G, Ψ and $\tilde{\Phi}$ of SMPBE solution u, including the total CPU time (exclusion of the time for finite element mesh generation).

less CPU time than the ones from the finite element solver. Due to the efficiency of the overlapped box iterative method, we can see that the hybrid solver reduced the total CPU runtime of the finite element SMPBE solver from 31% to 63% on the coarse meshes and 64% to 76% on the fine meshes, showing that the hybrid solver can significantly improve the performance of the finite element one, especially in the case of fine meshes. Here for solving the nonlinear problem Ψ, the newton iterations for each protein are at most one step difference for those two solvers.

3.4.4 Electrostatic solvation free energy calculations

As a variant of PBE, one important application of SMPBE is to predict the electrostatic solvation free energy of a biomolecule in an ionic solvent, which measures the energy changes

of a biomolecule from a vacuum state to a solvated state. This quantity is commonly used in the community to measure the "quality" of a developed solver by comparing either to the available experiment data or the analytical values of some test models (i.e., the Born Ball model (`http://www.poissonboltzmann.org/examples/The_Born_ion/`)). Since it is the energy difference in two different states, as did in APBS [18], in order to "carlibrate" this term, we need compute the numerical solution twice corresponding to the vacuum state and the solvated state, respectively, and then obtain the difference. However, according to the solution decomposition (3.3), without computing the numerical solutions twice, we can directly estimate the electrostatic solvation energy E by the formula

$$E = \frac{N_A}{4184} \cdot \frac{k_B T}{2} \sum_{j=1}^{n_p} z_j \left(\Psi(\mathbf{r}_j) + \tilde{\Phi}(\mathbf{r}_j) \right) \text{ kilocalorie per mole (kcal/mol)}, \tag{3.14}$$

which reflects another advantage of the solution decomposition scheme to solve SMPBE. To compute this quantity, the numerical solutions Ψ_h and $\tilde{\Phi}_h$ are found from solving (3.4) and (3.5) to obtain an approximation E_h of E based on a mesh Ω_h of Ω. Therefore, a convergent and numerically stable SMPBE 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 verify the solver's numerical stability, we calculated $\triangle E$ by the formula (3.14) for a set of 216 biomolecules with atom numbers varying from 506 to 69711 (including proteins, protein-protein complexes, and nucleic acids) downloaded from Prof. Ray Luo's website

`http://ray10.bio.uci.edu/ray1/`.

To do the tests, each domain Ω was selected to be 2 times larger than the box D (i.e., $\mu = 2$), and the zero boundary condition was used. For each biomolecule, we calculated $\triangle E$ using six successively refined meshes. The averages of the mesh point numbers over the 216 meshes of the whole domain Ω were found to be 49979, 74946, 99081, 170005, 443704, and 981550 for the six sets of meshes, respectively. Since the analytical value of $\triangle E$ is unknown, we took the numerical value of $\triangle E$ calculated from the finest mesh as the reference to calculate a relative error of $\triangle E_h$. To simplify the display of the numerical results, we calculated the average of the 216 relative errors for each set of meshes, which were reported in Figure 3.4. From Figure 3.4 we can see that the relative errors were changed less than 0.004 only, implying that this new solver has satisfied performance in the numerical stability and the convergence behavior in calculations of the electrostatic solvation free energy.

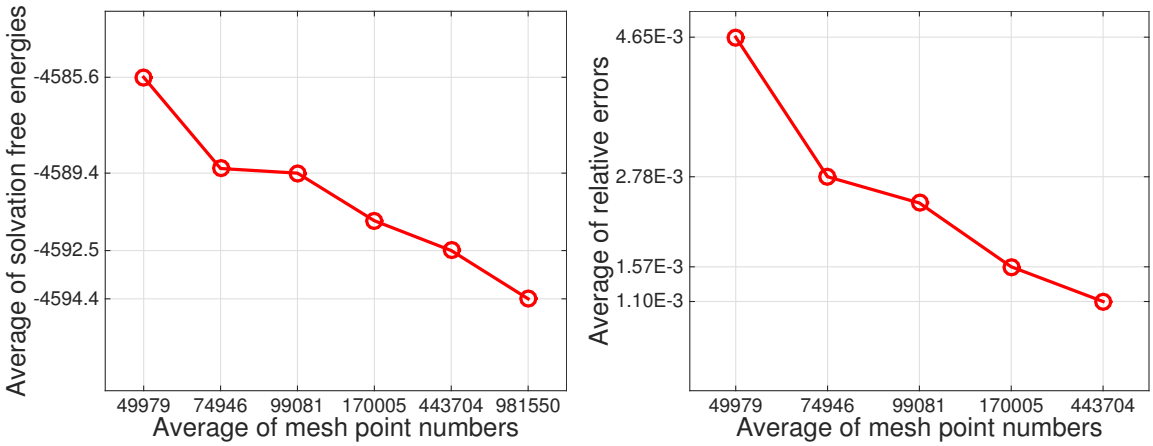


Figure 3.4: Numerical behaviors of the new SMPBE solver on the six sets of meshes in calculations of electrostatic solvation free energies for the 216 biomolecules.

3.4.5 Binding free energy calculations

Similar to the classic PBE [98, 99], SMPBE can also be used to study the salt dependence of the binding free energy for a complex molecule. Many experiment data are available from the literature, which make them extremely valuable for the validation of a dielectric model and related numerical solvers. In this section, we will use the application of SMPBE in the bind free energy calculation to further validate the new program package.

For a complex C consisting of molecule components A and B, the binding free energy $E_b(I_s)$ is defined by

$$E_b(I_s) = E(C, I_s) - E(A, I_s) - E(B, I_s),$$

which measures the difference of electrostatic free energies before and after binding two components together. Here $E(X, I_s)$ denotes an electrostatic free energy of a molecule X in a solvent with an ionic strength I_s. From the counterion condensation theory [100, 101], instead of quantifying this binding energy, it is known that E_b can be transformed by the variable change, $\xi = \ln I_s$, to a linear function of ξ as follows

$$E_b = m\xi + b, \tag{3.15}$$

where m and b are constants to be determined, and m can be determined experimentally. Notice, some authors used slightly different term to characterize this linear relationship. For

example, in [102], a scaled slope m_s from the chemical point of view is further defined by

$$m_s = -m/(N_A k_B T).$$

To simulate the binding free energy calculations, we made tests on a DNA-drug complex represented in PDB ID 1D86 using the PQR files from [100]. A chemical experimental value, -1.51, of m_s was given in [102, Table 3]. In these numerical tests, the electrostatic free energy $E(X, I_s)$ was computed according to the formula (3.14).

To produce a numerical prediction of the scaled slope m_s, we calculated the binding free energy E_b using the following 11 different values of I_s:

$$I_{s,j} = e^{\xi_j} \quad \text{with} \quad \xi_j = -3 + 0.2j \quad \text{for } j = 0, 1, 2, \ldots, 9, 10,$$

and then produced the best-fitted line by a linear regression program, which is downloaded from the APBS website

`http://www.poissonboltzmann.org/examples/Protein-Rna_Tutorial/`

to yield the predicted value of m_s.

Mesh set index	Number of mesh points		
	Mesh for complex	Mesh for DNA	Mesh for drug
1	85195	86435	73051
2	154705	155903	137797
3	259903	261512	233983
4	411583	413961	367030
5	613983	616328	542747
6	4252682	4255163	4163790

Table 3.3: Mesh data for the six sets of meshes used in the calculation of the binding free energy for a DNA-drug complex represented in PDB ID 1D86.

Moreover, we constructed six different sets of meshes and repeated the above calculation in order to study the sensitivity of the solver to the discretization error. Here each set contains three meshes: the first one for the complex 1D86, the second one for its DNA component, and the third one for its drug component. In each mesh, a domain of Ω was set to be four times larger than the domain of D, i.e., $\mu = 4$. The interface Γ produced from the first mesh were shared by other five meshes to ensure the same interface problem was solved

on each mesh. Here the boundary condition is set to zero for the sufficiently large domain of each mesh set.

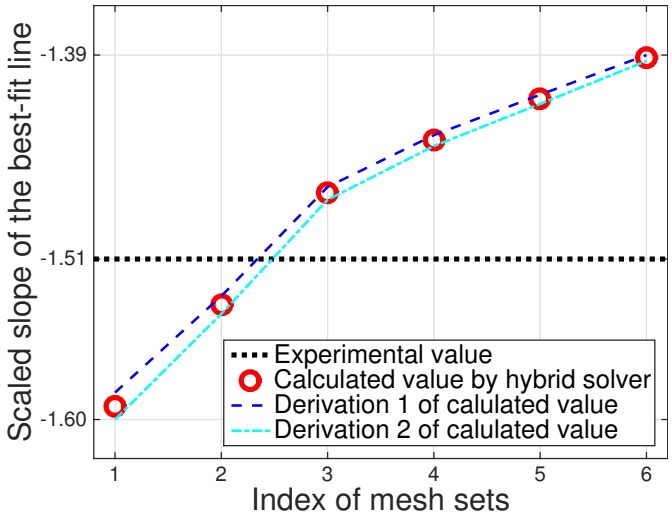


Figure 3.5: The scaled slops m_s of the six best-fitted lines calculated by the hybrid SMPBE solver for a DNA-drug complex represented in PDB ID 1D86 based on the six sets of meshes listed in Table 3.3.

From Figure 3.5 it can be seen that the numerical values of the scaled slop m_s increased from -1.6 to -1.39, as the mesh set index was changed from 1 to 6 (or a mesh became finer and finer). The deviations from the best-fitted line calculations were all very small. These tests showed that the hybrid solver behaved stably in the numerical calculation of the binding free energy. Furthermore, from Figure 3.6, it can be seen that the predicted binding free energies perfectly describe its linear relationship with the log function of the ionic strength I_s as claimed in theory. In Figure 3.7, we displayed the electrostatic free energy values of $E(X, I_s)$ calculated by the hybrid solver using $I_s = e^{-2.2} \approx 0.11$, which showed a tendency of convergence for X being the DNA-drug complex, DNA component, or drug component, respectively. These important properties make it easy for us to produce a good predicted value of m_s on a properly constructed mesh set.

3.4.6 Comparisons between PBE and SMPBE

SMPBE is the variant of PBE by considering the steric effects. And from the numerical results, we have clearly seen that the improvement of the new dielectric continuum model in

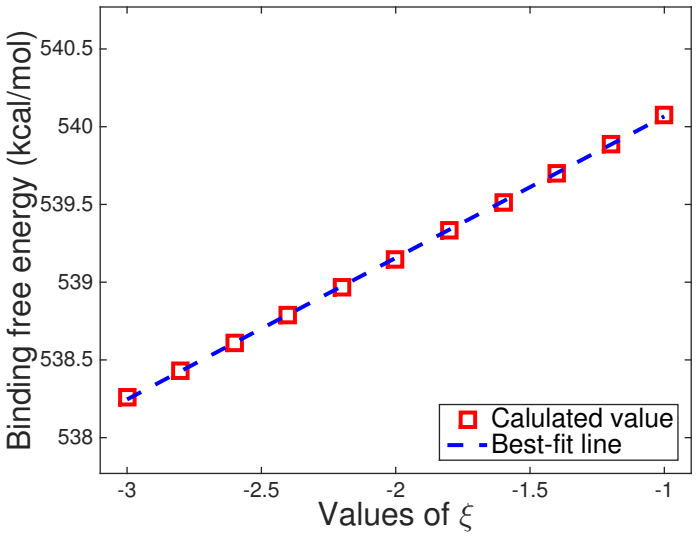


Figure 3.6: The best-fitted line obtained on the second set of meshes.

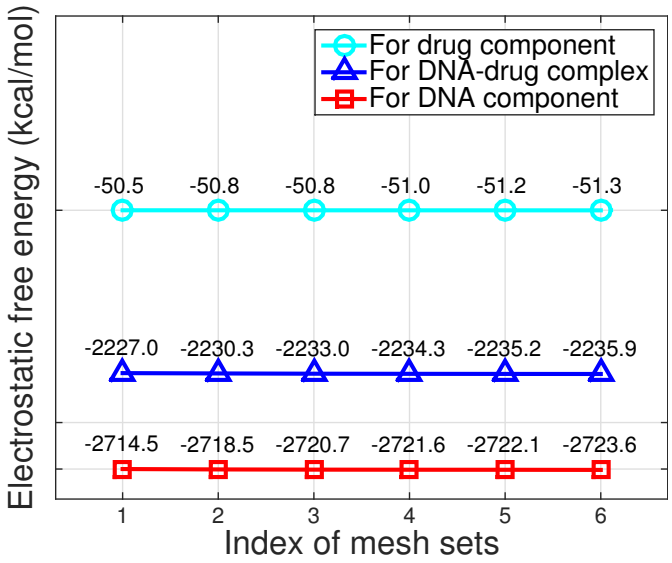


Figure 3.7: Electrostatic free energies (numbers on the top of each curve) of the complex with PDB ID 1D86, its DNA component, and its drug part calculated by the hybrid solver on the six mesh sets listed in Table 3.3 using $I_s \approx 0.11$.

terms of the ion concentrations (See Section 3.4.2). However, surprisingly, it was found out that the numerical solutions of SMPBE and PBE are close to each other, and the differences in the ion concentrations were caused by the different formulas of the ion concentration (i.e., one has upper bound and one has not, see (3.13)). Next we try to repeat the numerical

simulation of the binding free energy calculation in previous subsection to further explore the difference and the similarity between these two dielectric continuum models.

With the six sets of meshes, the same domain setup, and the same boundary value function used in previous subsection, we carried out the binding free energy calculations using the hybrid PBE solver reported in [1] and then compared to the results in last subsection.

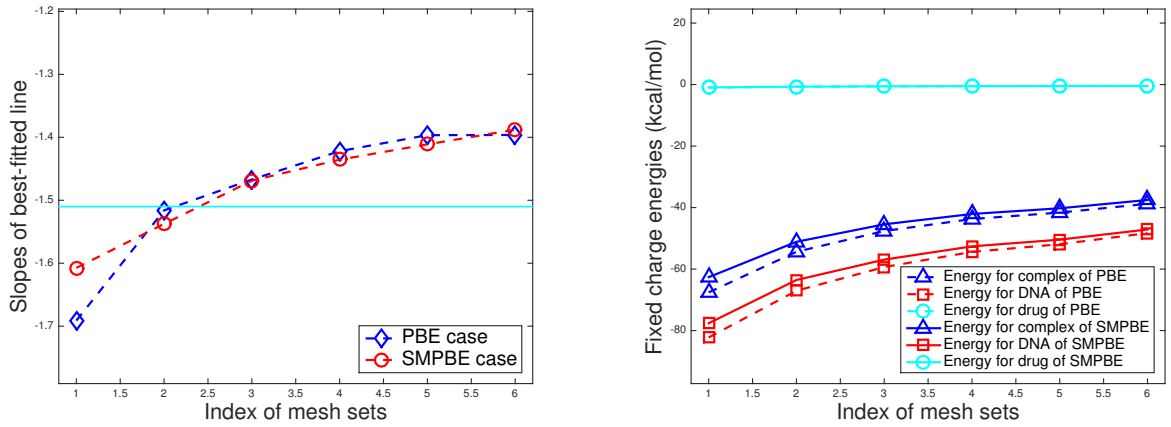


Figure 3.8: Comparisons of binding free energy calculations using SMPBE and PBE for the complex with PDB ID 1D86.

From Figure 3.8, it can been seen that the best-fitted lines obtained from the hybrid SMPBE and PBE solvers on different mesh sets are close. Meanwhile, the electrostatic free energies for each component on the six sets of meshes are also very close. With these observations, it seems indeed the numerical solutions of SMPBE and PBE do not have much differences. As the uniform ion sizes are introduced to result in a "bounded" Boltzmann distribution, the SMPBE solver is much faster than the PBE one in solving the nonlinear problem of $\tilde{\Phi}$ due to the "less" nonlinearity of the Boltzmann distribution in SMPBE. Therefore, it seems we can treat the uniform ion sizes Λ^3 in SMPBE as a scale factor to reduce the difficulty to solve PBE. In [60], Wang *et.al.* reported their comparison results using PBE and nonuniform SMPBE as well as the molecular dynamics simulations for lipid bilayers, in which they claimed:

"The SMPBE appears to reproduce the molecular dynamics simulation results better than the PBE only under specific parameter sets, but in general, it performs no better than the Stern layer correction of the PBE."

This is partially consistent with the above observations. Nonetheless, we will firstly develop a accurate numerical solver by solving the nonuniform SMPBE, and then carry out further comparisons before we can make any conclusion. This is considered to be future work.

Chapter 4

Nonlocal modified Poisson Boltzmann equation

Besides the dielectric continuum model considering the steric effects, there are nonlocal dielectric continuum models, which took the polarization correlations of water molecules into account. This nonlocal theoretical framework was firstly studied in Dogonadze *etc*'s nonlocal electrostatics. But due to the integro-differential term in the equation to reflect the structural effects, for a long time developing a numerical solver for the nonlocal case had been an intimidating work. Remarkably, Hildebrandt *et.al.* [80] proposed a novel reformulation in 2004, making it possible to solve the improved continuum model numerically. Thereafter, numerous efforts had been devoted to develop fast solvers of the nonlocal continuum model for the pure water case using the finite element, the finite difference, and the boundary element approaches [81, 82, 83, 84]. However, due to the complexity caused by the integro-differential term, it is not trivial to develop the nonlocal dielectric continuum model for a biomolecule in an ionic solvent. Currently, the only available nonlocal model for the ionic solvent case, which is called nonlocal modified PBE (NMPBE), was proposed in paper [3]. In this chapter, following the formulation of NMPBE and the techniques used in the finite element NMPBE solver, the new hybrid method will also be applied to solve this case. Numerical results will show the efficiency improvement of the new solver and demonstrate the improvement of the new nonlocal dielectric model.

4.1 Dimensionless formulation

Following the description in [3], for the case of a symmetric 1:1 electrolyte solution, when the distance is measured in angstrom (Å), we have the following so-called dimensionless NMPBE (i.e., the electrostatic potentials have units $(k_BT)/e_c$):

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha \sum_{j=1}^{n_p} z_j \delta_{\mathbf{r}_j}, & \mathbf{r} \in D_p, \\ -\epsilon_\infty \Delta u(\mathbf{r}) - (\epsilon_s - \epsilon_\infty)\nabla \cdot \int_{\mathbb{R}^3} Q_\lambda(\mathbf{r}-\mathbf{r}')\nabla u(\mathbf{r}')d\mathbf{r}' + \kappa^2 \sinh(u(\mathbf{r})) = 0, & \mathbf{r} \in D_s, \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), & \\ \epsilon_p \dfrac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_\infty \dfrac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_s - \epsilon_\infty) \int_{\mathbb{R}^3} Q_\lambda(\mathbf{r}-\mathbf{r}')\nabla u(\mathbf{r}')d\mathbf{r}' \cdot \mathbf{n}(\mathbf{s}), & \mathbf{s} \in \Gamma, \\ u(\mathbf{r}) \to 0, & |\mathbf{r}| \to \infty, \end{cases} \tag{4.1}$$

where α and κ are two constants given in (3.2), and the kernel function Q_λ is given by

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

To numerically solve (4.1), it is difficult task due to the singularities from the Dirac Delta distributions, the strong nonlinearity from the hyperbolic functions, and the integro-differential term in the solvent region. To overcome the singularities, a solution decomposition scheme would be used. Same as in [1, 3, 42], we split the solution u of NMPBE into a sum of three components: G with an explicit formula, a solution Ψ of a linear interface problem, and a solution $\tilde{\Phi}$ of a nonlinear interface problem. Thus, instead of solving u directly, we compute G and solve two interface problems of Ψ and $\tilde{\Phi}$. Therefore, the efficiency of the NMPBE solver would mainly depends on the solvers for solving Ψ and $\tilde{\Phi}$, respectively. Furthermore, since both equations of Ψ and $\tilde{\Phi}$ contain the integro-differential terms, another reformulation technique [3, 80] different from the one proposed by Hildebrandt *et.al.* is applied to sharply reduce the computational cost. That is, set a new variable and a new equation for the convolution term by the fact that the integro-differential term has a Yukawa-type kernel. Therefore, in the end we need solve the systems of equations for both Ψ and $\tilde{\Phi}$ to obtain the numerical solution u of NMPBE, thus avoiding the assemble of the dense matrix and sharply reduce the computational cost. And in order to deal with the strong nonlinearity, as did in the PBE case, an upper bound will be provided to truncate the input value of the hyperbolic functions. All of these techniques have been used in the finite

element NMPBE solver proposed recently in [3]. For clarity, we give them a short review firstly.

4.2 Finite element NMPBE solver

Here we make a review of the important techniques to make equation (4.1) numerically solvable and the modified Newton scheme used to solve the nonlinear interface problem proposed in the finite element NMPBE solver reported in [3], which computes the numerical solution on a sufficient large domain

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

with a prescribed boundary function g on the domain boundary $\partial\Omega$.

Solving (4.1) directly would be a challenge due to the involvements of the integro-differential part and the Dirac Delta distributions. To avoid computing the convolution part, since the kernel Q_λ is a Yukawa-type function satisfying the following equation [3, 80]:

$$-\lambda^2 \Delta Q_\lambda(\mathbf{r}) + Q_\lambda(\mathbf{r}) = \delta(\mathbf{r}), \quad \mathbf{r} \in \mathbb{R}^3, \tag{4.2}$$

setting $w(\mathbf{r}) = (Q_\lambda * u)(\mathbf{r}) = \int_{\mathbb{R}^3} Q_\lambda(\mathbf{r} - \mathbf{r}')u(\mathbf{r}')d\mathbf{r}'$ and according to the property of the convolution, we have

$$D^k(Q_\lambda * u) = (D^k Q_\lambda) * u = Q_\lambda * D^k u,$$

where D^k denotes the multiple-index derivative. Therefore, doing the convolution on both sides of (4.2) with the function u, we further have

$$\begin{aligned} -\lambda^2 (u * \Delta Q_\lambda)(\mathbf{r}) + (u * Q_\lambda)(\mathbf{r}) &= u(\mathbf{r}), \\ (u * \Delta Q_\lambda)(\mathbf{r}) &= \frac{1}{\lambda^2}\left[(u * Q_\lambda)(\mathbf{r}) - u(\mathbf{r})\right], \\ \nabla \cdot (\nabla u * Q_\lambda)(\mathbf{r}) &= \frac{1}{\lambda^2}\left[(u * Q_\lambda)(\mathbf{r}) - u(\mathbf{r})\right], \end{aligned}$$

from which we have the following equation to characterize $w(\mathbf{r})$:

$$-\lambda^2 \Delta w(\mathbf{r}) + w(\mathbf{r}) - u(\mathbf{r}) = 0, \quad \mathbf{r} \in \mathbb{R}^3. \tag{4.3}$$

Therefore, we can rewrite equation (4.1) to the following system of PDEs:

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha \sum_{j=1}^{n_p} z_j \delta_{\mathbf{r}_j}, & \mathbf{r} \in D_p, \\ -\epsilon_\infty \Delta u(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2} \left[u(\mathbf{r}) - w(\mathbf{r}) \right] + \kappa^2 \sinh(u(\mathbf{r})) = 0, & \mathbf{r} \in D_s, \\ -\lambda^2 \Delta w(\mathbf{r}) + w(\mathbf{r}) - u(\mathbf{r}) = 0, & \mathbf{r} \in \Omega, \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), \quad \epsilon_p \dfrac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_\infty \dfrac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_s - \epsilon_\infty) \dfrac{\partial w(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ u(\mathbf{s}) = g(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{4.4}$$

where the boundary condition of function $w(\mathbf{s})$ on $\partial\Omega$ will be discussed later.

Equation (4.4) make it possible to avoid the computation of the convolution parts, which sharply reduces the complexity of the computations. Furthermore, to deal with the singularities caused by the Dirac Delta distributions, according to [3], the solution $(u(\mathbf{r}), w(\mathbf{r}))$ of system (4.4) is decomposed as follows:

$$u(\mathbf{r}) = G(\mathbf{r}) + \Psi(\mathbf{r}) + \tilde{\Phi}(\mathbf{r}), \quad w(\mathbf{r}) = G_w(\mathbf{r}) + \Psi_w(\mathbf{r}) + \tilde{\Phi}_w(\mathbf{r}), \tag{4.5}$$

where

$$\begin{aligned} G_w(\mathbf{r}) &= (G * Q_\lambda)(\mathbf{r}), \\ \Psi_w(\mathbf{r}) &= (\Psi * Q_\lambda)(\mathbf{r}), \\ \tilde{\Phi}_w(\mathbf{r}) &= (\tilde{\Phi} * Q_\lambda)(\mathbf{r}), \end{aligned}$$

$G(\mathbf{r})$ and its convolution $G_w(\mathbf{r})$ are given explicitly as follows

$$\begin{aligned} G(\mathbf{r}) &= \frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} \frac{z_j}{|\mathbf{r} - \mathbf{r}_j|}, \\ G_w(\mathbf{r}) &= \frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} z_j \frac{1 - \exp(-|\mathbf{r} - \mathbf{r}_j|/\lambda)}{|\mathbf{r} - \mathbf{r}_j|}, \end{aligned} \tag{4.6}$$

Ψ and Ψ_w are solutions of the linear interface boundary value problems

$$\begin{cases} \Delta \Psi(\mathbf{r}) = 0, & \mathbf{r} \in D_p, \\ -\epsilon_\infty \Delta \Psi(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2} \left[\Psi - \Psi_w \right] = -\dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2} \left[G - G_w \right], & \mathbf{r} \in D_s, \\ -\lambda^2 \Delta \Psi_w(\mathbf{r}) + \Psi_w(\mathbf{r}) - \Psi(\mathbf{r}) = 0, & \mathbf{r} \in \Omega, \\ \Psi(\mathbf{s}^+) = \Psi(\mathbf{s}^-), \quad \epsilon_p \dfrac{\partial \Psi(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} - \epsilon_\infty \dfrac{\partial \Psi(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = (\epsilon_s - \epsilon_\infty) \dfrac{\partial \Psi_w(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} + g_\Gamma(\mathbf{s}), & \mathbf{s} \in \Gamma, \\ \Psi(\mathbf{s}) = g(\mathbf{s}) - G(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{4.7}$$

and $\tilde{\Phi}$ and $\tilde{\Phi}_w$ are solutions of the nonlinear interface boundary value problems

$$\begin{cases} \Delta\tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r} \in D_p, \\ -\epsilon_\infty \Delta\tilde{\Phi}(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2}\left[\tilde{\Phi} - \tilde{\Phi}_w\right] + \kappa^2 \sinh(\tilde{\Phi} + \Psi + G) = 0, & \mathbf{r} \in D_s, \\ -\lambda^2 \Delta\tilde{\Phi}_w(\mathbf{r}) + \tilde{\Phi}_w(\mathbf{r}) - \tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r} \in \Omega, \\ \tilde{\Phi}(\mathbf{s}^+) = \tilde{\Phi}(\mathbf{s}^-), \quad \epsilon_p \dfrac{\partial\tilde{\Phi}(\mathbf{s}^-)}{\partial\mathbf{n}(\mathbf{s})} - \epsilon_\infty \dfrac{\partial\tilde{\Phi}(\mathbf{s}^+)}{\partial\mathbf{n}(\mathbf{s})} = (\epsilon_s - \epsilon_\infty)\dfrac{\partial\tilde{\Phi}_w(\mathbf{s})}{\partial\mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \tilde{\Phi}(\mathbf{s}) = 0, & \mathbf{s} \in \partial\Omega. \end{cases} \tag{4.8}$$

Here

$$\frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} = \nabla G \cdot \mathbf{n}$$

with ∇G being given by (2.11), ∇G_w is given by

$$\nabla G_w(\mathbf{r}) = -\frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} z_j \frac{1 - \exp(-\frac{|\mathbf{r}-\mathbf{r}_j|}{\lambda}) - \frac{|\mathbf{r}-\mathbf{r}_j|}{\lambda}\exp(-\frac{|\mathbf{r}-\mathbf{r}_j|}{\lambda})}{|\mathbf{r} - \mathbf{r}_j|^3} \cdot (\mathbf{r} - \mathbf{r}_j), \quad \mathbf{r} \in \Omega, \tag{4.9}$$

and the function g_Γ is defined as

$$g_\Gamma(\mathbf{s}) = (\epsilon_\infty - \epsilon_p)\frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_s - \epsilon_\infty)\frac{\partial G_w(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, \quad \mathbf{s} \in \Gamma. \tag{4.10}$$

With the solution decomposition and the novel reformulations, the main difficult part is how to solve the nonlinear equation for $\tilde{\Phi}$. Here we make a review of the modified Newton scheme used in [3] for solving (4.8). 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 get the solution $(\tilde{\Phi}(\mathbf{r}), \tilde{\Phi}_w(\mathbf{r}))$ of (4.8), according to the modified Newton algorithm, we have a sequence of iterates, $\{(\tilde{\Phi}^{(k)}, \tilde{\Phi}_w^{(k)})\}$, as follows:

$$\tilde{\Phi}^{(k+1)} = \tilde{\Phi}^{(k)} + \lambda_k p_k, \quad \tilde{\Phi}_w^{(k+1)} = \tilde{\Phi}_w^{(k)} + \lambda_k q_k \quad k = 0, 1, 2, \ldots, \tag{4.11}$$

where $(\tilde{\Phi}^{(0)}, \tilde{\Phi}_w^{(0)})$ is an initial guess pair, λ_k is a step length determined by a line search algorithm, (p_k, q_k) is a pair of search directions satisfying the following equivalent variational problem:

Find a $(p_k, q_k) \in \mathcal{M}_0 \times \mathcal{M}_0$ such that

$$A((p_k, q_k), (v_1, v_2)) = L((v_1, v_2)) \quad \forall (v_1, v_2) \in \mathcal{M}_0 \times \mathcal{M}_0, \tag{4.12}$$

where $\mathcal{M}_0 = \{v \in \mathcal{M} \mid v = 0 \text{ on } \partial\Omega\}$, the bilinear functional A on $\mathcal{M}_0 \times \mathcal{M}_0$ at current iterate $(\tilde{\Phi}^{(k)}, \tilde{\Phi}_w^{(k)})$ is defined by

$$\begin{aligned} A((p_k, q_k), (v_1, v_2)) =& \epsilon_p \int_{D_p} \nabla p_k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} + \epsilon_\infty \int_{D_s} \nabla p_k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \\ &+ (\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla q_k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} + \lambda^2 \int_\Omega \nabla q_k(\mathbf{r}) \cdot \nabla v_2(\mathbf{r}) d\mathbf{r} \\ &+ \int_\Omega (q_k(\mathbf{r}) - p_k(\mathbf{r})) v_2(\mathbf{r}) d\mathbf{r} \\ &+ \kappa^2 \int_{D_s} p_k(\mathbf{r}) v_1(\mathbf{r}) \cosh(\tilde{\Phi}^{(k)} + \Psi + G) d\mathbf{r}, \end{aligned} \tag{4.13}$$

and the linear functional L is given in the following

$$\begin{aligned} L((v_1, v_2)) = -\Bigg[& (\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla \tilde{\Phi}_w^{(k)}(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \\ & + \epsilon_p \int_{D_p} \nabla \tilde{\Phi}^{(k)}(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \\ & + \epsilon_\infty \int_{D_s} \nabla \tilde{\Phi}^{(k)}(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \\ & + \kappa^2 \int_{D_s} v_1(\mathbf{r}) \sinh(\tilde{\Phi}^{(k)} + \Psi + G) d\mathbf{r} \Bigg]. \end{aligned} \tag{4.14}$$

In the implementation, the initial iterate pair $(\tilde{\Phi}^{(0)}, \tilde{\Phi}_w^{(0)})$ can be selected simply as zero or a solution of a linearized equation. An upper bound of 85 is set as default for truncating the value of the sum $\Psi + G + \tilde{\Phi}^{(k)}$ to avoid the possible overflow problem of the hyperbolic terms. Each Newton equation of (4.12) is solved numerically by GMRES-ILU due to the asymmetry of the assembled matrix from the `PETSc` library with the absolute and the relative residue errors less than a given tolerance (10^{-8} by default). As for the boundary function g, we will discuss it later for the nonlocal case.

4.3 Reformulation of the Newton equation

Same as SMPBE, as the key step to apply the finite element and finite difference hybrid method, in this section, we present the Newton equation' reformulation from the variational form to the equivalent PDE form for the nonlocal case.

Theorem 4.3.1. *let $V_1 = H_0^1(\Omega) \cap H^2(D_p) \cap H^2(D_s)$ and $V_2 = H_0^1(\Omega) \cap C^2(\Omega)$, then the search direction pair $(p_k, q_k) \in V_1 \times V_2$, $k = 1, 2, 3, \cdots$, satisfies the following system of PDEs:*

$$-\Delta p_k(\mathbf{r}) = \Delta\tilde{\Phi}^{(k)}(\mathbf{r}), \quad \mathbf{r} \in D_p, \tag{4.15a}$$

$$\begin{aligned} -\epsilon_\infty \Delta p_k(\mathbf{r}) + \frac{\epsilon_s - \epsilon_\infty}{\lambda^2}(p_k(\mathbf{r}) - q_k(\mathbf{r})) + \kappa^2 \cosh(\tilde{\Phi}^{(k)} + \Psi + G) p_k(\mathbf{r}) \\ = \epsilon_\infty \Delta\tilde{\Phi}^{(k)}(\mathbf{r}) + (\epsilon_s - \epsilon_\infty)\Delta\tilde{\Phi}_w^{(k)} - \kappa^2 \sinh(\tilde{\Phi}^{(k)} + \Psi + G), \quad \mathbf{r} \in D_s, \end{aligned} \tag{4.15b}$$

$$-\lambda^2 \Delta q_k(\mathbf{r}) + q_k(\mathbf{r}) - p_k(\mathbf{r}) = 0, \quad \mathbf{r} \in \Omega, \tag{4.15c}$$

$$p_k(\mathbf{s}^+) = p_k(\mathbf{s}^-), \quad q_k(\mathbf{s}^+) = q_k(\mathbf{s}^-), \quad \mathbf{s} \in \Gamma,$$

$$\epsilon_\infty \frac{\partial\tilde{\Phi}^{(k)}(\mathbf{s}^+)}{\partial \mathbf{n}} - \epsilon_p \frac{\partial\tilde{\Phi}^{(k)}(\mathbf{s}^-)}{\partial \mathbf{n}} + (\epsilon_s - \epsilon_\infty)\frac{\partial\tilde{\Phi}_w^{(k)}(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, \quad \mathbf{s} \in \Gamma, \tag{4.15d}$$

$$p_k(\mathbf{s}) = 0, \quad q_k(\mathbf{s}) = 0, \quad \mathbf{s} \in \partial\Omega, \tag{4.15e}$$

where the current Newton iterates $\tilde{\Phi}^{(k)}$ and $\tilde{\Phi}_w^{(k)}$ are the given functions of V_1 and V_2, respectively, Ψ is a given solution of (4.7), and G is defined in (4.6).

Proof. For any $v_1 = 0$ on Ω and any $v_2 \in C_0^\infty(\Omega) \subset V_2$, from (4.12) we have

$$\lambda^2 \int_\Omega \nabla q_k(\mathbf{r}) \cdot \nabla v_2(\mathbf{r}) d\mathbf{r} + \int_\Omega (q_k(\mathbf{r}) - p_k(\mathbf{r})) v_2(\mathbf{r}) d\mathbf{r} = 0,$$

from which, according to the Green's identity, we obtain

$$-\lambda^2 \Delta q_k(\mathbf{r}) + q_k(\mathbf{r}) - p_k(\mathbf{r}) = 0, \quad \mathbf{r} \in \Omega,$$

which is (4.15c).

Now for any $v_2 = 0$ on Ω and any $v_1 \in H_0^1(\Omega) \subset V_1$ satisfying $v_1 = 0$ on D_s and $v_1 \in \mathbb{C}_0^\infty(D_p)$, from (4.12) we can get

$$\int_{D_p} (\Delta p_k + \Delta\tilde{\Phi}^{(k)}) v_1 d\mathbf{r} = 0 \quad \forall v_1 \in \mathbb{C}_0^\infty(D_p),$$

from which it implies equation (4.15a).

Next, for any $v_2 = 0$ on Ω and any $v_1 \in H_0^1(\Omega)$ satisfying $v_1 = 0$ on D_p and $v_1 \in \mathbb{C}_0^\infty(D_s)$,

equation (4.12) is reduced to

$$\begin{aligned}
&\epsilon_\infty \int_{D_s} \nabla p_k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} + (\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla q_k(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \\
&+ \kappa^2 \int_{D_s} p_k(\mathbf{r}) v_1(\mathbf{r}) \cosh(\tilde{\Phi}^{(k)} + \Psi + G) d\mathbf{r} = \\
-&\left[(\epsilon_s - \epsilon_\infty) \int_{D_s} \nabla \tilde{\Phi}_w^{(k)}(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} + \epsilon_\infty \int_{D_s} \nabla \tilde{\Phi}^{(k)}(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \right. \\
&\left. + \kappa^2 \int_{D_s} v_1(\mathbf{r}) \sinh(\tilde{\Phi}^{(k)} + \Psi + G) d\mathbf{r} \right].
\end{aligned}$$

By the Green's identity, the above equality can be reformulated as

$$\begin{aligned}
&- \epsilon_\infty \int_{D_s} \Delta p_k(\mathbf{r}) v_1(\mathbf{r}) d\mathbf{r} - (\epsilon_s - \epsilon_\infty) \int_{D_s} \Delta q_k(\mathbf{r}) v_1(\mathbf{r}) d\mathbf{r} \\
&+ \kappa^2 \int_{D_s} p_k(\mathbf{r}) v_1(\mathbf{r}) \cosh(\tilde{\Phi}^{(k)} + \Psi + G) d\mathbf{r} = \\
&(\epsilon_s - \epsilon_\infty) \int_{D_s} \Delta \tilde{\Phi}_w^{(k)}(\mathbf{r}) v_1(\mathbf{r}) d\mathbf{r} + \epsilon_\infty \int_{D_s} \Delta \tilde{\Phi}^{(k)}(\mathbf{r}) v_1(\mathbf{r}) d\mathbf{r} \\
&- \kappa^2 \int_{D_s} v_1(\mathbf{r}) \sinh(\tilde{\Phi}^{(k)} + \Psi + G) d\mathbf{r},
\end{aligned}$$

from which we obtain

$$\begin{aligned}
&-\epsilon_\infty \Delta p_k(\mathbf{r}) - (\epsilon_s - \epsilon_\infty) \Delta q_k + \kappa^2 \cosh(\tilde{\Phi}^{(k)} + \Psi + G) p_k(\mathbf{r}) = \\
&(\epsilon_s - \epsilon_\infty) \Delta \tilde{\Phi}_w^{(k)}(\mathbf{r}) + \epsilon_\infty \Delta \tilde{\Phi}^{(k)}(\mathbf{r}) - \kappa^2 \sinh(\tilde{\Phi}^{(k)} + \Psi + G),
\end{aligned}$$

thus implying (4.15b) using (4.15c).

Furthermore, applying the Green's identity to the bilinear form A for any $(v_1, v_2) \in V_1 \times V_2$ and the above obtained equalities, we additionally have

$$\begin{aligned}
&\int_\Gamma \left[\epsilon_p \frac{\partial p_k(\mathbf{s}^-)}{\partial \mathbf{n}} - \epsilon_\infty \frac{\partial p_k(\mathbf{s}^+)}{\partial \mathbf{n}} \right] v_1(\mathbf{s}) d\mathbf{s} - (\epsilon_s - \epsilon_\infty) \int_\Gamma \frac{\partial q_k(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} v_1(\mathbf{s}) d\mathbf{s} \\
&= \int_\Gamma \left[\epsilon_\infty \frac{\partial \tilde{\Phi}^{(k)}(\mathbf{s}^+)}{\partial \mathbf{n}} - \epsilon_p \frac{\partial \tilde{\Phi}^{(k)}(\mathbf{s}^-)}{\partial \mathbf{n}} \right] v_1(\mathbf{s}) d\mathbf{s} + (\epsilon_s - \epsilon_\infty) \int_\Gamma \frac{\partial \tilde{\Phi}_w^{(k)}(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} v_1(\mathbf{s}) d\mathbf{s},
\end{aligned}$$

which implies the interface condition (4.15d). The boundary condition (4.15e) is natural because of $(p_k, q_k) \in V_1 \times V_2$. This completes the proof. □

Remark 4.3.2. In this theorem, we not only reformulate the Newton equation from the variational form into a system of PDEs, but also provide a new linearized NMPBE, which is the generalization of the local case reported in [103]. With the solution decomposition, we obtain the solution of the new linearized NMPBE after one iteration of the modified Newton scheme with the zero initial guess. Therefore, in the modified Newton scheme, we can simply set the initial guess to zero function.

4.4 New NMPBE solver

As mentioned in previous section, to get a numerical solution u of NMPBE, we essentially only need to solve two systems of linear interface problems: (4.7) for Ψ and (4.15) for search direction p_k in order to solve $\tilde{\Phi}$ with the modified Newton scheme. Based on the special box partition defined in Chapter 2, we could similarly define the following box iterative method for the nonlocal case to solve Ψ and p_k, respectively.

To solve Ψ, we have the following iteration scheme: For $m = 1, 2, 3, \cdots$,

$$\begin{aligned} \Psi_i^{(m)} &= (1-\omega)\Psi_i^{(m-1)} + \omega\widehat{\Psi}_i, \\ \Psi_{w,i}^{(m)} &= (1-\omega)\Psi_{w,i}^{(m-1)} + \omega\widehat{\Psi}_{w,i} \end{aligned} \tag{4.16}$$

on Ω_i for $i = 1, 2, \cdots, 7$, where $\Psi_i^{(0)}$ and $\Psi_{w,i}^{(0)}$ are initial iterates, $\omega \in (1, 2)$ is the over-relaxation parameter, $\widehat{\Psi}_i$ and $\widehat{\Psi}_{w,i}$ with $i = 1$ to 6 denote solutions of the system of the following boundary value problems:

$$\begin{cases} -\epsilon_\infty \Delta\Psi(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2}\left[\Psi - \Psi_w\right] = -\dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2}\left[G - G_w\right], & \mathbf{r} \in \Omega_i, \\ -\lambda^2\Delta\Psi_w(\mathbf{r}) + \Psi_w(\mathbf{r}) - \Psi(\mathbf{r}) = 0, & \mathbf{r} \in \Omega_i, \\ \Psi(\mathbf{s}) = \Psi_j^{(m-1)}(\mathbf{s}), \quad \Psi_w(\mathbf{s}) = \Psi_{w,j}^{(m-1)}(\mathbf{s}) & \\ \qquad\qquad \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = i+1 \text{ to } 7, & \\ \Psi(\mathbf{s}) = \Psi_j^{(m)}(\mathbf{s}), \quad \Psi_w(\mathbf{s}) = \Psi_{w,j}^{(m)}(\mathbf{s}) & \\ \qquad\qquad \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = 1 \text{ to } i-1, & \\ \Psi(\mathbf{s}) = g(\mathbf{s}) - G(\mathbf{s}), \quad \Psi_w(\mathbf{s}) = g_w(\mathbf{s}) - G_w(\mathbf{s}) \quad \text{on } \partial\Omega_i \cap \partial\Omega & \end{cases} \tag{4.17}$$

with $g_w(\mathbf{r}) = (g * Q_\lambda)(\mathbf{r})$, and $\widehat{\Psi}_7$ and $\widehat{\Psi}_{w,7}$ are solutions of the system of the linear interface

problems:

$$\begin{cases} \Delta\Psi(\mathbf{r}) = 0, & \mathbf{r} \in D_p, \\ -\epsilon_\infty \Delta\Psi(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2}\left[\Psi - \Psi_w\right] = -\dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2}\left[G - G_w\right], & \mathbf{r} \in D_s \cap \Omega_7, \\ -\lambda^2 \Delta\Psi_w(\mathbf{r}) + \Psi_w(\mathbf{r}) - \Psi(\mathbf{r}) = 0, & \mathbf{r} \in \Omega_7, \\ \Psi(\mathbf{s}^+) = \Psi(\mathbf{s}^-), & \\ \epsilon_p \dfrac{\partial\Psi(\mathbf{s}^-)}{\partial\mathbf{n}(\mathbf{s})} - \epsilon_\infty \dfrac{\partial\Psi(\mathbf{s}^+)}{\partial\mathbf{n}(\mathbf{s})} = (\epsilon_s - \epsilon_\infty)\dfrac{\partial\Psi_w(\mathbf{s})}{\partial\mathbf{n}(\mathbf{s})} + g_\Gamma(\mathbf{s}), & \mathbf{s} \in \Gamma, \\ \Psi(\mathbf{s}) = \Psi_j^{(m)}(\mathbf{s}), \quad \Psi_w(\mathbf{s}) = \Psi_{w,j}^{(m)}(\mathbf{s}) \qquad \text{on } \partial\Omega_7 \cap \Omega_j, & j = 1 \text{ to } 6. \end{cases} \tag{4.18}$$

After G, Ψ, $\tilde{\Phi}^{(m)}$ and $\tilde{\Phi}_w^{(m)}$ are computed, we then construct another overlapped box iterative method for computing the search direction pair (p_k, q_k) of problem (4.15): For $m = 1, 2, 3, \cdots$,

$$\begin{aligned} p_i^{(m)} &= (1-\omega)p_i^{(m-1)} + \omega\widehat{p}_i, \\ q_i^{(m)} &= (1-\omega)q_i^{(m-1)} + \omega\widehat{q}_i \end{aligned} \tag{4.19}$$

on Ω_i for $i = 1, 2, \cdots, 7$, where $p_i^{(0)}$ and $q_i^{(0)}$ are initial iterates, $\omega \in (1, 2)$ is the over-relaxation parameter, $\widehat{p}_i$ and $\widehat{q}_i$ with $i = 1$ to 6 denote solutions of the system of the following boundary value problems:

$$\begin{cases} -\epsilon_\infty \Delta p(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2}(p(\mathbf{r}) - q(\mathbf{r})) + \kappa^2 \cosh(\tilde{\Phi}^{(m)} + \Psi + G)p(\mathbf{r}) & \\ \qquad = \epsilon_\infty \Delta\tilde{\Phi}^{(m)}(\mathbf{r}) + (\epsilon_s - \epsilon_\infty)\Delta\tilde{\Phi}_w^{(m)} - \kappa^2 \sinh(\tilde{\Phi}^{(m)} + \Psi + G), & \mathbf{r} \in \Omega_i, \\ -\lambda^2 \Delta q(\mathbf{r}) + q(\mathbf{r}) - p(\mathbf{r}) = 0, & \mathbf{r} \in \Omega_i, \\ p(\mathbf{s}) = p_j^{(m-1)}(\mathbf{s}), \quad q(\mathbf{s}) = q_j^{(m-1)}(\mathbf{s}) & \\ \qquad \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = i+1 \text{ to } 7, & \\ p\mathbf{s}) = p_j^{(m)}(\mathbf{s}), \quad q(\mathbf{s}) = q_j^{(m)}(\mathbf{s}) & \\ \qquad \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = 1 \text{ to } i-1, & \\ p(\mathbf{s}) = 0, \quad q(\mathbf{s}) = 0 \quad \text{on } \partial\Omega_i \cap \partial\Omega, & \end{cases} \tag{4.20}$$

and $\widehat{p}_7$ and $\widehat{q}_7$ are solutions of the system of the following linear interface problems:

$$\begin{cases} -\Delta p(\mathbf{r}) = \Delta\tilde{\Phi}^{(m)}(\mathbf{r}), & \mathbf{r} \in D_p, \\ -\epsilon_\infty \Delta p(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2}(p(\mathbf{r}) - q(\mathbf{r})) + \kappa^2 \cosh(\tilde{\Phi}^{(m)} + \Psi + G)p(\mathbf{r}) & \\ \quad = \epsilon_\infty \Delta\tilde{\Phi}^{(m)}(\mathbf{r}) + (\epsilon_s - \epsilon_\infty)\Delta\tilde{\Phi}_w^{(m)} - \kappa^2 \sinh(\tilde{\Phi}^{(m)} + \Psi + G), & \mathbf{r} \in D_s \cap \Omega_7, \\ -\lambda^2 \Delta q(\mathbf{r}) + q(\mathbf{r}) - p(\mathbf{r}) = 0, & \mathbf{r} \in \Omega_7, \\ p(\mathbf{s}^+) = p(\mathbf{s}^-), \quad q(\mathbf{s}^+) = q(\mathbf{s}^-), & \mathbf{s} \in \Gamma, \\ \epsilon_p \dfrac{\partial p(\mathbf{s}^-)}{\partial \mathbf{n}} - \epsilon_\infty \dfrac{\partial p(\mathbf{s}^+)}{\partial \mathbf{n}} - (\epsilon_s - \epsilon_\infty)\dfrac{\partial q(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} = & \\ \epsilon_\infty \dfrac{\partial \tilde{\Phi}^{(m)}(\mathbf{s}^+)}{\partial \mathbf{n}} - \epsilon_p \dfrac{\partial \tilde{\Phi}^{(m)}(\mathbf{s}^-)}{\partial \mathbf{n}} + (\epsilon_s - \epsilon_\infty)\dfrac{\partial \tilde{\Phi}_w^{(k)}(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ p(\mathbf{s}) = p_j^{(m)}(\mathbf{s}), \quad q(\mathbf{s}) = q_j^{(m)}(\mathbf{s}) \quad \text{on } \partial\Omega_7 \cap \Omega_j \text{ for } j = 1 \text{ to } 6. & \end{cases} \tag{4.21}$$

In the implementation, we use the following as the termination rule of the overlapped box iterative method:

$$\sqrt{\sum_{i=1}^{7} ||\Psi_i^{(k)} - \Psi_i^{(k-1)}||^2 + \sum_{i=1}^{7} ||\Psi_{w,i}^{(k)} - \Psi_{w,i}^{(k-1)}||^2} \leq \epsilon$$

for equation (4.16) and

$$\sqrt{\sum_{i=1}^{7} ||p_i^{(k)} - p_i^{(k-1)}||^2 + \sum_{i=1}^{7} ||q_i^{(k)} - q_i^{(k-1)}||^2} \leq \epsilon$$

for equation (4.19), here $||\cdot||$ is the Euclidean norm, and ϵ is set to 10^{-7} by default.

4.4.1 Finite element solver for the nonlocal case

To solve the interface problem (4.18) of Ψ on Ω_7, we reformulate it as the following variational problem:

$$\text{Find a } \ (\Psi, \Psi_w) \in \mathcal{M} \times \mathcal{M} \ \text{ satisfying}$$

$$\Psi(\mathbf{s}) = g(\mathbf{s}) - G(s) \text{ and } \Psi_w(\mathbf{s}) = g_w(\mathbf{s}) - G_w(\mathbf{s}) \text{ for all } \mathbf{s} \in \partial\Omega \text{ such that}$$

$$A_1((\Psi, \Psi_w), (v_1, v_2)) = L_1((v_1, v_2)) \quad \forall (v_1, v_2) \in \mathcal{M}_0 \times \mathcal{M}_0, \tag{4.22}$$

where $\mathcal{M}$ is a finite element function space as a subspace of the usual Sobolev space $H^1(\Omega_7)$, $\mathcal{M}_0 = \{v \in \mathcal{M} \mid v = 0 \text{ on } \partial\Omega_7\}$, the bilinear form A_1 is defined by

$$\begin{aligned} A_1((\Psi, \Psi_w), (v_1, v_2)) = \quad & \epsilon_p \int_{D_p} \nabla\Psi(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} + \epsilon_\infty \int_{D_s \cap \Omega_7} \nabla\Psi(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \\ & + (\epsilon_s - \epsilon_\infty) \int_{D_s \cap \Omega_7} \nabla\Psi_w(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \\ & + \lambda^2 \int_{\Omega_7} \nabla\Psi_w(\mathbf{r}) \cdot \nabla v_2(\mathbf{r}) d\mathbf{r} \\ & + \int_{\Omega_7} (\Psi_w(\mathbf{r}) - \Psi(\mathbf{r})) v_2(\mathbf{r}) d\mathbf{r}, \end{aligned} \tag{4.23}$$

and the linear form L_1 is given as follows

$$\begin{aligned} L_1((v_1, v_2)) = \quad & (\epsilon_s - \epsilon_\infty) \int_{D_s \cap \Omega_7} \nabla G_w(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r} \\ & + (\epsilon_p - \epsilon_\infty) \int_{D_s \cap \Omega_7} \nabla G(\mathbf{r}) \cdot \nabla v_1(\mathbf{r}) d\mathbf{r}. \end{aligned} \tag{4.24}$$

While for the search direction p_k of equation (4.21) on the central box Ω_7, we also reformulate it into the following variational problem:

$$\text{Find a } (p_k, q_k) \in \mathcal{M}_0 \times \mathcal{M}_0 \text{ such that}$$

$$A((p_k, q_k), (v_1, v_2)) = L((v_1, v_2)) \quad \forall (v_1, v_2) \in \mathcal{M}_0 \times \mathcal{M}_0, \tag{4.25}$$

where the bilinear functional A and the linear functional L are given in (4.13) and (4.14) by replacing Ω and D_s by Ω_7 and $\Omega_7 \cap D_s$, respectively.

We programmed the finite element solver based on the efficient finite element library `DOLFIN` [93] to solve the variation problems (4.22) and (4.25), and we call the GMRES-ILU solver from the `PETSc` library [94] to solve each assembled finite element linear system due to its asymmetry. Here both the relative and the absolute residue error parameters are set to 10^{-8} by default.

4.4.2 Optimal finite difference solver for the nonlocal case

On each neighboring box Ω_i, $i = 1, 2, \cdots, 6$, same as the PBE and SMPBE cases, we use the seven-point finite difference stencil to discretize and approximate the system of second-order elliptic boundary value problems on a defined uniform mesh. We have the following results about the reformulation of equations (4.17) of Ψ so that the constructed optimal finite difference solver PCG-MG works for the discretized linear system in this nonlocal case.

Theorem 4.4.1. *For the following system of boundary value problems:*

$$\begin{cases} -\dfrac{\lambda^2\epsilon_\infty}{\epsilon_s-\epsilon_\infty}\Delta\Psi(\mathbf{r})+\Psi-\Psi_w=G_w-G, & \mathbf{r}\in\Omega_i,\\ -\lambda^2\Delta\Psi_w(\mathbf{r})+\Psi_w(\mathbf{r})-\Psi(\mathbf{r})=0, & \mathbf{r}\in\Omega_i,\\ \Psi(\mathbf{s})=\Psi_j^{(m-1)}(\mathbf{s}),\quad \Psi_w(\mathbf{s})=\Psi_{w,j}^{(m-1)}(\mathbf{s}) & \\ \qquad\qquad \text{on } \partial\Omega_i\cap\Omega_j \text{ if } \partial\Omega_i\cap\Omega_j\neq\varnothing \text{ for } j=i+1 \text{ to } 7, & \\ \Psi(\mathbf{s})=\Psi_j^{(m)}(\mathbf{s}),\quad \Psi_w(\mathbf{s})=\Psi_{w,j}^{(m)}(\mathbf{s}) & \\ \qquad\qquad \text{on } \partial\Omega_i\cap\Omega_j \text{ if } \partial\Omega_i\cap\Omega_j\neq\varnothing \text{ for } j=1 \text{ to } i-1, & \\ \Psi(\mathbf{s})=g(\mathbf{s})-G(\mathbf{s}),\quad \Psi_w(\mathbf{s})=g_w(\mathbf{s})-G_w(\mathbf{s}) \quad \text{on } \partial\Omega_i\cap\partial\Omega, & \end{cases} \tag{4.26}$$

the discretized matrix M using the seven-point finite difference stencil on a uniform mesh of each neighboring box Ω_i, $i=1,2,\cdots,6$, is SPD.

Proof. Let (x_i,y_j,z_k) be a mesh point of a uniform finite difference mesh defined on the neighboring box Ω_i. Suppose Ψ and Ψ_w are the solution vectors on the defined uniform mesh, with the unknown ordering (Ψ,Ψ_w), we then have

$$M\begin{bmatrix}\Psi\\ \Psi_w\end{bmatrix}=\begin{bmatrix}M_{11} & M_{12}\\ M_{21} & M_{22}\end{bmatrix}\begin{bmatrix}\Psi\\ \Psi_w\end{bmatrix},$$

where M_{11} and M_{22} denote the corresponding discretized matrice of the first equation and the second equation with respect to the unknowns Ψ and Ψ_w, respectively, and

$$M_{12}=M_{21}=-I$$

with I being the identity matrix.

Obviously, both M_{11} and M_{22} are symmetric because we use the centered finite difference method to discretize the Laplace operator. Therefore, we know matrix M is symmetric. To see M is positive definite, for the mesh point (x_i,y_j,z_k), we have

$$\begin{aligned}\left(\begin{bmatrix}M_{11} & M_{12}\end{bmatrix}\begin{bmatrix}\Psi\\ \Psi_w\end{bmatrix}\right)_{i,j,k}=&\frac{\lambda^2\epsilon_\infty}{\epsilon_s-\epsilon_\infty}\left[6\Psi_{i,j,k}-\Psi_{i-1,j,k}-\Psi_{i+1,j,k}-\Psi_{i,j-1,k}\right.\\ &\left.-\Psi_{i,j+1,k}-\Psi_{i,j,k-1}-\Psi_{i,j,k+1}\right]+\Psi_{i,j,k}-\Psi_{w,i,j,k}\end{aligned}$$

or

$$\begin{aligned}\left(\begin{bmatrix}M_{21} & M_{22}\end{bmatrix}\begin{bmatrix}\Psi\\ \Psi_w\end{bmatrix}\right)_{i,j,k}=&\lambda^2\left[6\Psi_{w,i,j,k}-\Psi_{w,i-1,j,k}-\Psi_{w,i+1,j,k}-\Psi_{w,i,j-1,k}\right.\\ &\left.-\Psi_{w,i,j+1,k}-\Psi_{w,i,j,k-1}-\Psi_{w,i,j,k+1}\right]+\Psi_{w,i,j,k}-\Psi_{i,j,k},\end{aligned}$$

where $\Psi_{i,j,k}$ and $\Psi_{w,i,j,k}$ denote the function values of Ψ and Ψ_w at the point (x_i, y_j, z_k), respectively. This implies that matrix M is diagonally dominant, where the strict diagonal dominance occurs at the boundary points. Since the matrix M is obviously irreducible, then we know M is nonsingular. Furthermore, since $\epsilon_\infty < \epsilon_s$ and M is symmetric, by the Gershgorin circle theorem [87], all real eigenvalues of the matrix M are positive. Therefore, M is SPD and we complete the proof. □

Remark 4.4.2. This theorem provides the reformulation of the equations of Ψ on each neighboring box so that the optimal solver PCG-MG can be applied to solve the discretized linear system. Here we can also modify the second equation about Ψ_w by multiplying $\dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2}$ on both sides of (4.17) to get another discretized matrix M, which is also SPD.

Similarly, for solving the search direction p_k on the neighboring box Ω_i, $i = 1, 2, \cdots, 6$, we also need do the reformation of equations (4.20) as follows

$$\begin{cases} -\dfrac{\lambda^2\epsilon_\infty}{\epsilon_s-\epsilon_\infty}\Delta p(\mathbf{r}) + (p(\mathbf{r}) - q(\mathbf{r})) + \dfrac{\lambda^2\kappa^2}{\epsilon_s-\epsilon_\infty}\cosh(\tilde{\Phi}^{(m)} + \Psi + G)p(\mathbf{r}) \\ \quad = \dfrac{\lambda^2\epsilon_\infty}{\epsilon_s-\epsilon_\infty}\Delta\tilde{\Phi}^{(m)}(\mathbf{r}) + \lambda^2\Delta\tilde{\Phi}_w^{(m)} - \dfrac{\lambda^2\kappa^2}{\epsilon_s-\epsilon_\infty}\sinh(\tilde{\Phi}^{(m)} + \Psi + G), & \mathbf{r} \in \Omega_i, \\ -\lambda^2\Delta q(\mathbf{r}) + (q(\mathbf{r}) - p(\mathbf{r})) = 0, & \mathbf{r} \in \Omega_i, \\ p(\mathbf{s}) = p_j^{(m-1)}(\mathbf{s}), \quad q(\mathbf{s}) = q_j^{(m-1)}(\mathbf{s}) \\ \qquad\qquad \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = i+1 \text{ to } 7, \\ p\mathbf{s}) = p_j^{(m)}(\mathbf{s}), \quad q(\mathbf{s}) = q_j^{(m)}(\mathbf{s}) \\ \qquad\qquad \text{on } \partial\Omega_i \cap \Omega_j \text{ if } \partial\Omega_i \cap \Omega_j \neq \varnothing \text{ for } j = 1 \text{ to } i-1, \\ p(\mathbf{s}) = 0, \quad q(\mathbf{s}) = 0 \quad \text{on } \partial\Omega_i \cap \partial\Omega. \end{cases} \tag{4.27}$$

Based on the above reformulations, now we can apply the optimal finite difference solver PCG-MG and the new hybrid method to modify the finite element NMPBE solver for the improving-the-efficiency purpose. For clarity, the new algorithm using the hybrid method to solve NMPBE is given in the following:

Algorithm for Solving NMPBE. Let the special overlapped box iterative method be defined in (4.16) and (4.19) with the PCG-MG algorithm for solving the finite difference systems of (4.26) and (4.27) on a uniform mesh of the box Ω_i for $i = 1$ to 6 and the GMRES-ILU algorithm for solving the finite element systems of (4.18) and (4.21) on an interface-fitted tetrahedral mesh of the central box Ω_7. A solution u of (4.4) is calculated approximately in the following five steps:

Step 1. Construct an interface-matched tetrahedral mesh for the central box Ω_7 and a uniform mesh for each neighboring box Ω_i for $i \neq 7$ with a mesh size $h > 0$.

Step 2. Calculate G and G_w on each box, and compute ∇G and ∇G_w on Ω_7 according to (4.6), (2.11) and (4.9), respectively.

Step 3. Calculate Ψ by the overlapped box iterative method (4.16). Here (4.17) and (4.18) are approximated as the finite difference and the finite element linear systems and then solved by PCG-MG and GMRES-ILU, respectively, until the relative or the absolute residual norm less than 10^{-8}.

Step 4. Calculate $\tilde{\Phi}$ by the modified Newton method in the following steps:

(a) Set $k = 0$ and $\tilde{\Phi}^{(0)} = 0$ (by default).

(b) Calculate the search direction pair (p_k, q_k) by the overlapped box iterative method (4.19). Here (4.20) and (4.21) are approximated as the finite difference and the finite element linear systems and then solved by PCG-MG and GMRES-ILU, respectively, until the relative or the absolute residual norm less than 10^{-8}.

(c) Find a steplength λ_k by a line search algorithm (starting with $\lambda_k = 1$).

(d) Define the modified Newton iterate $\tilde{\Phi}^{(k+1)}$ and $\tilde{\Phi}_w^{(k+1)}$ by $\tilde{\Phi}^{(k+1)} = \tilde{\Phi}^{(k)} + \lambda_k p_k$ and $\tilde{\Phi}_w^{(k+1)} = \tilde{\Phi}_w^{(k)} + \lambda_k q_k$.

(e) Check the convergence test: If $\|\tilde{\Phi}^{(k+1)} - \tilde{\Phi}^{(k)}\| \leq 10^{-7}$ and $\|\tilde{\Phi}_w^{(k+1)} - \tilde{\Phi}_w^{(k)}\| \leq 10^{-7}$ (by default), then $\tilde{\Phi}^{(k+1)}$ is set to a solution $\tilde{\Phi}$ of the nonlinear interface problem (4.8); otherwise, increase k by 1 and go back to (b).

Step 5. Construct a numerical solution u of SMPBE by the solution decomposition $u = G + \Psi + \tilde{\Phi}$.

We programmed above algorithm in `C`, `Fortran`, and `Python` as a software package. Similar to the finite element program package, the main program of the software was written in `Python` based on the state-of-the-art finite element library `DOLFIN` from the `FEniCS` project [39, 40]. Each finite element equation is produced by `DOLFIN`, and solved by GMRE-ILU from the `PETSc` library [97]. Also we wrote a `Fortran` subroutine for speeding up calculating the function values of G and G_w at the mesh points of Ω, and the function values of ∇G and ∇G_w at the mesh points of the central box Ω_7. All the `Fortran` subroutines and `C` programs were converted to the Python external modules by `f2py` and `SWIG`, respectively. With these modules, the hybrid solver were integrated as a `Python` program.

4.4.3 Selection of boundary value function

In this subsection, we address how to choose the boundary value functions $g(\mathbf{s})$ and $g_w(\mathbf{s})$ on $\partial\Omega$, respectively. In cases of PBE and SMPBE, g is commonly approximated by zero function or the analytical solution of the Debye-Hückel equation [104, 105] if the domain Ω is sufficiently large, latter of which is called MDH boundary condition. For the nonlocal case, we could also simply set $g(\mathbf{s}) = 0$ and $g_w(\mathbf{s}) = 0$ similarly with a sufficiently large domain Ω.

Meanwhile, the local MDH function has been extended to the nonlocal case by considering the generalized nonlocal Debye-Hückel equation, detailed of which will be reported in Yi Jiang's thesis, as follows:

$$-\epsilon_\infty \Delta u(\mathbf{r}) + \frac{\epsilon_s - \epsilon_\infty}{\lambda^2}[u(\mathbf{r}) - (u * Q_\lambda)(\mathbf{r})] + \kappa^2 u(\mathbf{r}) = \frac{10^{10} e_c^2}{\epsilon_0 k_B T} \sum_{j=1}^{n_p} z_j \delta_{\mathbf{r}_j}, \quad \mathbf{r} \in \mathbb{R}^3,$$

with $u(\mathbf{r}) \to 0$ as $|\mathbf{r}| \to \infty$. Its solution u and convolution $\hat{u} = u * Q_\lambda$ have been found in the analytical expressions: For all $\mathbf{r} \neq \mathbf{r}_j$ with $j = 1, 2, \ldots, n_p$,

$$\begin{aligned} u(\mathbf{r}) =& \frac{10^{10} e_c^2}{4\pi\epsilon_\infty(\tau_2 - \tau_1)\epsilon_0 k_B T} \sum_{j=1}^{n_p} \frac{z_j}{|\mathbf{r} - \mathbf{r}_j|} \left(\tau_2 e^{-\eta_1|\mathbf{r}-\mathbf{r}_j|} - \tau_1 e^{-\eta_2|\mathbf{r}-\mathbf{r}_j|}\right), \\ \hat{u}(\mathbf{r}) =& \frac{10^{10} e_c^2 \tau_1 \tau_2}{4\pi\epsilon_\infty(\tau_2 - \tau_1)\epsilon_0 k_B T} \sum_{j=1}^{n_p} \frac{z_j}{|\mathbf{r} - \mathbf{r}_j|} \left(e^{-\eta_1|\mathbf{r}-\mathbf{r}_j|} - e^{-\eta_2|\mathbf{r}-\mathbf{r}_j|}\right), \end{aligned} \tag{4.28}$$

where τ_1, τ_2, η_1, and η_2 are defined by

$$\begin{aligned}
\tau_1 &= \frac{\kappa^2\lambda^2 + \epsilon_s - 2\epsilon_\infty - \xi}{2(\epsilon_s - \epsilon_\infty)}, \\
\tau_2 &= \frac{\kappa^2\lambda^2 + \epsilon_s - 2\epsilon_\infty + \xi}{2(\epsilon_s - \epsilon_\infty)}, \\
\eta_1 &= \frac{1}{\lambda}\sqrt{\frac{\kappa^2\lambda^2 + \epsilon_s + \xi}{2\epsilon_\infty}}, \\
\eta_2 &= \frac{1}{\lambda}\sqrt{\frac{\kappa^2\lambda^2 + \epsilon_s - \xi}{2\epsilon_\infty}}, \\
\xi &= \sqrt{(\kappa^2\lambda^2 + \epsilon_s)^2 - 4\epsilon_\infty\lambda^2\kappa^2}.
\end{aligned}$$

As $\epsilon_\infty \to \epsilon_s$, from (4.28) it can yield the local Debye-Hückel equation's solution

$$u = \frac{\alpha}{4\pi\epsilon_s}\sum_{j=1}^{n_p}\frac{z_j}{|\mathbf{r} - \mathbf{r}_j|}e^{-\frac{\kappa}{\sqrt{\epsilon_s}}|\mathbf{r} - \mathbf{r}_j|},$$

which shows the solution is the generalization of MDH and thus is called NMDH. Without explicit statement, the boundary functions $g(\mathbf{s})$ and $g_w(\mathbf{s})$ are always set to the NMDH in the following numerical experiments.

4.5 Numerical results

In this section, we report the numerical experiments we made using the new `Python` program package. 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 (except the parameters m, n, μ of (2.6)) on one processor of a Mac Pro Workstation with the 3.7 GHZ Quad-Core Intel Xeon E5 and 64 GB memory.

4.5.1 Validation test

Firstly, we made numerical experiments in an artificially-constructed test model:

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha \sum_{j=1}^{n_p} z_j \delta_{\mathbf{r}_j}, & \mathbf{r} \in D_p, \\ -\epsilon_\infty \Delta u(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2} [u(\mathbf{r}) - w(\mathbf{r})] + \kappa^2 \sinh(u(\mathbf{r})) = \kappa^2 \sinh(U(\mathbf{r})), & \mathbf{r} \in D_s, \\ -\lambda^2 \Delta w(\mathbf{r}) + w(\mathbf{r}) - u(\mathbf{r}) = 0, & \mathbf{r} \in \Omega, \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), \quad \epsilon_p \dfrac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_\infty \dfrac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_s - \epsilon_\infty) \dfrac{\partial w(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ u(\mathbf{s}) = U(\mathbf{s}), \quad w(\mathbf{s}) = W(\mathbf{s}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{4.29}$$

where

$$D_p = \{\mathbf{r} \in \mathbb{R}^3 : \|\mathbf{r}\| < a\},$$

$$\Gamma = \{\mathbf{r} \in \mathbb{R}^3 : \|\mathbf{r}\| = a\},$$

Ω is a region containing D_p, $D_s = \Omega - D_p - \Gamma$, and the functions $U(\mathbf{r})$ and $W(\mathbf{r})$ are the analytical solutions in the simple series form in terms of Legendre polynomials and modified spherical Bessel functions of the following nonlocal Poisson test model reported in [106]:

$$\begin{cases} -\epsilon_p \Delta U(\mathbf{r}) = \alpha \sum_{j=1}^{n_p} z_j \delta_{\mathbf{r}_j}, & \mathbf{r} \in D_p, \\ -\epsilon_\infty \Delta U(\mathbf{r}) + \dfrac{\epsilon_s - \epsilon_\infty}{\lambda^2} [U(\mathbf{r}) - W(\mathbf{r})] = 0, & \mathbf{r} \in \mathbb{R}^3 - D_p - \Gamma, \\ -\lambda^2 \Delta W(\mathbf{r}) + W(\mathbf{r}) - U(\mathbf{r}) = 0, & \mathbf{r} \in \mathbb{R}^3, \\ U(\mathbf{s}^+) = U(\mathbf{s}^-), \quad \epsilon_p \dfrac{\partial U(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} = \epsilon_\infty \dfrac{\partial U(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_s - \epsilon_\infty) \dfrac{\partial W(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ U(\mathbf{r}) \to 0, \quad W(\mathbf{r}) \to 0 & \text{as } \mathbf{r} \to \infty. \end{cases}$$

To make the program package work for this test model, we only need to modify the system of equations related to $\tilde{\Phi}$. That is, after applying the solution decomposition, we only alter the $\tilde{\Phi}$ equation (4.8) by adding one right-hand side term $\kappa^2 \sinh(U(\mathbf{r}))$ in the solvent domain. Correspondingly, we add additional term

$$\kappa^2 \int_{D_s} \sinh(U(\mathbf{r})) v_1(\mathbf{r}) d\mathbf{r}$$

in the linear form L of (4.14), and the term $\kappa^2 \sinh(U(\mathbf{r}))$ on the right-hand size of equation (4.15b) for the search direction pair (p_k, q_k). All of these changes are easy to make based on

the `Python` program package of the algorithm. And obviously, the analytical solution $u(\mathbf{r})$ of (4.29) is just $U(\mathbf{r})$. For details of the explicit formulas of the solutions U and W, please see paper [106].

To carry out the computation and to avoid the crash of the program due to the overflow problems from the hyperbolic functions in the equation, here we set $\alpha = 1.0$ and $a = 1.0$. The charge numbers z_j and the atomic positions $\mathbf{r}_j$ were obtained from a PQR file of a protein with PDB ID 4PTI, which has 892 atoms (i.e., $n_p = 892$). Here we divided $\mathbf{r}_j$, $j = 1$ to 892, by 54 to rescale it into the unit spherical region D_p with the preserved protein structure within the unit ball. To setup the domain Ω and the finite difference uniform meshes, we set $D = (-2, 2)^3$, $\tau = 1$, and $\eta = 4$, which gave $\Omega_7 = (-3, 3)^3$ and $\Omega = (-6, 6)^3$. In order to calculate the analytical solutions of $U(\mathbf{r})$ and $W(\mathbf{r})$ reported in [106], we truncate both the series to the finite sum of $N = 20$ terms. The over-relaxation parameter ω of the overlapped box iterative method were both set to 1.25 in solving (4.16) for (Ψ, Ψ_w) and (4.19) for (p_k, q_k). Then four nested meshes with the finite difference mesh size $h = 1.0$, $h/2$ (0.5), $h/4$ (0.25), and $h/8$ (0.125), respectively, were constructed for testing the convergence behavior of the new NMPBE solver. Here the numbers of mesh points of these four nested meshes were 4124, 31145, 242017, and 1908033, respectively, including the numbers of mesh points from the finite element meshes of the central box Ω_7, which were 2270, 17717, 139993, and 1113009, respectively. The numerical results are reported in the following Table.

Mesh size h	Relative Error $\frac{\|u - u_h\|_{l^2(\Omega)}}{\|u\|_{l^2(\Omega)}}$	Convergence order	PCG-MG Iter. on Ω_i ($i = 1$ to 6)	New Box Iter. on Ω
0.1	6.18×10^{-3}	–	$2.29 \approx 2$	$10.0 \approx 10$
0.5	1.58×10^{-3}	1.97	$3.07 \approx 3$	$12.3 \approx 12$
0.25	3.98×10^{-4}	1.99	$3.27 \approx 3$	$11.3 \approx 11$
0.125	1.13×10^{-4}	1.82	$3.22 \approx 3$	$13.0 \approx 13$

Table 4.1: Performance of the new NMPBE solver for the test model (4.29) in the relative error and the average number of iterations (Iter.) on the four nested meshes.

Notice the analytical solution of (4.29) is linearly proportional to α, the relative error thus is independent of α, and the numerical results in Table 4.1 show the high accuracy of the new solver for solving the test model. Meanwhile, from the relative errors, it is obtained that the convergence order is approximately around 2, well matching the mathematical theory.

Furthermore, with different mesh sizes, the average iterations of both PCG-MG and the box iterative method almost kept the same, numerically confirming that PCG-MG and the box iterative method have the convergence rates independent of mesh sizes as claimed in the mathematical theory. All of these observations validated the new program package.

4.5.2 Performance tests on proteins

For clarity, we made numerical tests on the 12 proteins (varying atom numbers up to 11,439) used in the PBE comparison tests from [1] to demonstrate the improved performance of the new NMPBE solver in terms of CPU time in comparison with the finite element NMPBE solver. Here the PDB files of these proteins were downloaded from the PDB website and then converted to the PQR files by the tool `PDB2PQR`. The region D was generated for each protein as listed in Tables 4.2 and 4.3.

Index	PDB ID	Cubic Region $D = (a_1, b_1; a_2, b_2; a_3, b_3)$
1	2LZX	$(-16.5, 17.0; -16.1, 17.4; -16.1, 17.4)$
2	1AJJ	$(-8.8, 28.9; -11.4, 26.3; -15.7, 21.9)$
3	1FXD	$(-12.5, 31.4; -24.8, 19.1; -13.6, 30.3)$
4	1HPT	$(-14.8, 31.7; -13.7, 32.9; -5.7, 40.8)$
5	4PTI	$(-7.8, 38.4; -2.3, 43.9; -18.6, 27.6)$
6	1SVR	$(-24.8, 27.3; -30.4, 21.7; -24.5, 27.6)$
7	1A63	$(-29.9, 37.0; -34.3, 32.6; -33.9, 33.0)$
8	1CID	$(-47.2, 27.9; 0.6, 75.8; -4.8, 70.3)$
9	1A7M	$(-43.3, 40.0; -41.0, 42.3; -40.6, 42.8)$
10	2AQ5	$(-21.5, 50.6; 17.0, 89.1; -15.0, 57.1)$
11	1F6W	$(-38.7, 47.7; -37.9, 48.5; -21.1, 65.3)$
12	1C4K	$(19.2, 135.1; -26.9, 88.9; -16.5, 99.4)$

Table 4.2: The ordering index and the region D produced by the hybrid solver package for the 12 proteins.

We also then set $\mu = 4$, $m = 3$ and $n = 4$ to get Ω, Ω_i for $i = 1$ to 7, the uniform finite difference mesh size h, the uniform finite difference mesh of $\Omega \setminus D$, and the interface-fitted tetrahedral mesh of Ω_7. Here, each interface Γ was generated by the molecular surface and volumetric mesh generation program package `GAMer` based on the Gaussian blurring approach [91]. It has been shown that the molecular surface generated from `GAMer` is close to the commonly-used solvent-accessible surface (SAS) in [1]. Furthermore, according to recent

Index	PDB ID	n_p	N_{Ω_7}	N_Ω	Percentage ρ
1	2LZX	488	39896	535400	7.5%
2	1AJJ	513	42817	538321	8.0%
3	1FXD	811	45345	540849	8.4%
4	1HPT	852	47716	543220	8.8%
5	4PTI	892	45825	541329	8.5%
6	1SVR	1433	54666	550170	9.9%
7	1A63	2065	62506	558010	11.2%
8	1CID	2783	62870	558374	11.3%
9	1A7M	2803	68415	563919	12.1%
10	2AQ5	6024	82317	577821	14.2%
11	1F6W	8243	79182	574686	13.8%
12	1C4K	11439	77607	573111	13.5%

Table 4.3: Some basic information on the 12 proteins used for the 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 $\rho = 100N_{\Omega_7}/N_\Omega\%$.

studies [107], the Gaussian surface could be similar to the solvent-excluded surface, SAS, or the Van der Waals surface through properly setting the two controlling parameters (the decay rate and the isosurface value) of the Gaussian function. Therefore it is enough to use the Gaussian surface to do the comparison tests. Additionally, the over-relaxation parameter ω of the overlapped box iterative method was set to 1.215 and 1.015 in solving (4.16) for (Ψ, Ψ_w) and (4.19) for (p_k, q_k), respectively. The boundary value function g and the initial iterate $(\tilde{\Phi}^{(0)}, \tilde{\Phi}_w^{(0)})$ of the modified Newton scheme were set to the nonlocal Debye-Hückel solution and the solution of local PBE, respectively. For the fair comparison purpose, we used the finite element NMPBE solver to repeat all the calculations with the same boundary condition, initial guess and the terminal criteria of the modified Newton scheme. Here, by dividing each cubic grid cell in each neighboring box into 6 tetrahedra, we obtain the finite element meshes with the exact the same mesh points on Ω. Meanwhile, we compare the solutions obtained from these two solvers by the following formula:

$$E_h = \frac{\|u_{new} - u_{fe}\|_{l_2}}{\|u_{fe}\|_{l_2}}, \tag{4.30}$$

where u_{new} is a numerical solution of the new NMPBE solver and u_{fe} is the one obtained from the finite element solver. Furthermore, we calculate the CPU time speedup S_p to demonstrate the efficiency improvements. Here S_p was calculated as the ratio of the total

CPU times spent by the finite element solver and the new one. All of these results are reported in Table 4.4 and Figure 4.1.

PDB ID	Find G & G_w		Find Ψ		Find $\tilde{\Phi}$		Total Time		Relative error E_h
	New	FE	New	FE	New	FE	New	FE	
2LZX	7.32	22.44	11.54	30.51	80.95	414.17	100.76	471.55	3.4×10^{-4}
1AJJ	7.84	23.81	12.54	31.27	119.5	569.94	140.89	629.49	1.9×10^{-4}
1FXD	12.47	37.75	14.35	32.23	94.97	454.68	123.13	529.22	3.2×10^{-4}
1HPT	13.18	39.7	13.57	32.52	92.52	499.71	120.68	576.5	1.8×10^{-5}
4PTI	13.64	41.35	13.79	32.6	85.43	434.07	114.29	512.52	1.9×10^{-4}
1SVR	22.63	67.53	16.76	30.63	123.58	561.53	165.05	664.3	3.3×10^{-5}
1A63	33.49	98.57	20.67	33.87	183.82	694.7	240.85	831.82	3.0×10^{-5}
1CID	45.08	133.44	21.29	33.94	116.7	503.43	186.6	675.5	2.5×10^{-5}
1A7M	46.58	135.28	24.33	38.66	150.66	585.71	225.24	765.11	1.5×10^{-5}
2AQ5	104.81	297.77	32.38	40.86	242.35	743.41	386.61	1086.88	4.2×10^{-5}
1F6W	141.44	405.47	30.21	37.18	249.54	742.46	430.43	1189.91	2.5×10^{-5}
1C4K	194.81	562.92	30.62	37.66	285.71	988.71	523.56	1594.14	2.2×10^{-5}

Table 4.4: A comparison of the performance of the new NMPBE solver (New) with that of the finite element NMPBE solver (FE) proposed in [3] in computer CPU runtime measured in seconds.

Due to the restriction of the finite element method to the central box Ω_7, we calculated the gradient of G and its convolution G_w only on Ω_7, explaining the less cost in computing these function values. Meanwhile, the computational costs to solve for Ψ and $\tilde{\Phi}$ were significantly reduced because of the new box iterative method, resulting in the remarkable efficiency improvement in terms of the total CPU time. The last column gave the relative errors E_h of these 12 proteins to demonstrate that the new solver merely significantly improve the efficiency without compromising the accuracy of the finite element solver. Furthermore, Figure 4.1 presented the speedup S_p for these 12 test proteins. Same with PBE and SMPBE cases, the speedup S_p varied with the percentage ρ listed in Table 4.3, which was reduced from 4.8 to 2.8 for the total CPU time when ρ was increased from 7.5% to 14.2%. Finally, we also noticed the Newton iteration numbers are all quite close for these two different solvers, implying a fair efficiency comparison between these two solvers.

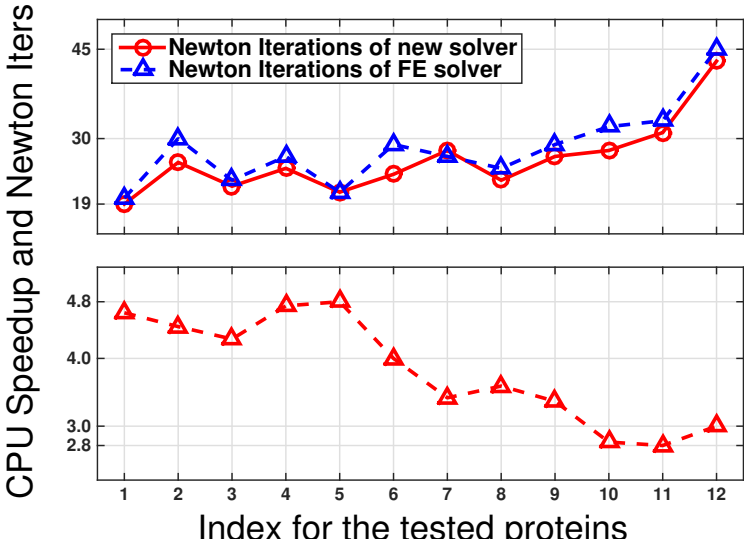


Figure 4.1: The modified Newton iteration numbers of the new and the finite element (FE) NMPBE solvers for the 12 test proteins and the CPU time speedup S_p produced by the new NMPBE solver using the data of Table 4.4.

4.5.3 Electrostatic solvation free energy calculations

Same as PBE and SMPBE, one important application related to NMPBE is to predict the electrostatic solvation free energy of a biomolecule. According to the solution decomposition (4.5), we can estimate the electrostatic solvation energy E using the same formula in (3.14). Here we again use this quantity to testify the numerical stability of the new solver. To do so, we constructed five different meshes for each of the four selected proteins considered in previous subsection. The numbers of mesh nodes of these five meshes are displayed in Figure 4.2 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 to make sure the same interface problem was solved. Here the triangular surface mesh was generated from the mesh generation program with enough mesh nodes as a good approximation of the interface Γ. We calculated the values of E_h using the new NMPBE solver and plotted them in Figure 4.2.

From Figure 4.2, with the increase of mesh points, which means the decrease of the mesh size h, all calculated electrostatic solvation free energies E_h have small perturbations. Particularly, for the 2LZX case, the values had only 0.4 kcal/mol difference from the coarsest

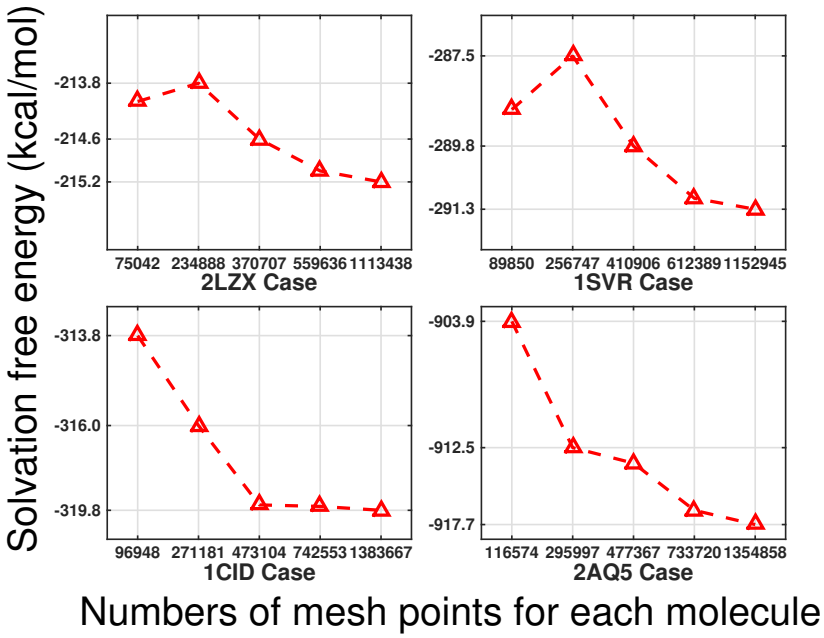


Figure 4.2: Solvation free energies calculated by the new NMPBE solver. Here the numbers on the x-axis are the numbers of mesh nodes used for calculations.

mesh to the finest one, resulting in only 0.2% deviation. These demonstrate the numerical stability of the new solver, which guarantees the accuracy of the predicted electrostatic solvation free energy.

4.5.4 Application in binding free energy calculations

Similar to the classic PBE [98, 99] and SMPBE, NMPBE can also be used to study the salt dependence of the binding free energy for a complex molecule. The method to numerically calculate the binding energy is given in Section 3.4.5. Here we will use the nonlocal dielectric continuum model to compute the scaled slope of the case, a DNA-drug complex represented in PDB ID 1D86, and then compare to the experiment data ($m_s = -1.51$). To produce a prediction to the scaled slope m_s, we also calculate the binding free energy E_b using the following 11 different values of I_s:

$$I_{s,j} = e^{\xi_j} \quad \text{with} \quad \xi_j = -3 + 0.2j \quad \text{for } j = 0, 1, 2, \ldots, 9, 10.$$

We then use these binding free energies to generate a best-fitted line by the same linear regression program used in SMPBE case to yield the predicted value of m_s.

Moreover, we use the six different sets of meshes in section 3.4.5 to repeat the above calculation in order to study the sensitivity of the new NMPBE solver to the discretization error. Furthermore, to compare the difference between the local model and the nonlocal model to see the improvements of the nonlocal continuum model, i.e., PBE and NMPBE, using the above 6 sets of constructed meshes, we repeat the calculations by the hybrid PBE solver in [1] to calculate m_s again. All of these numerical results are reported in Figures 4.3 and 4.4.

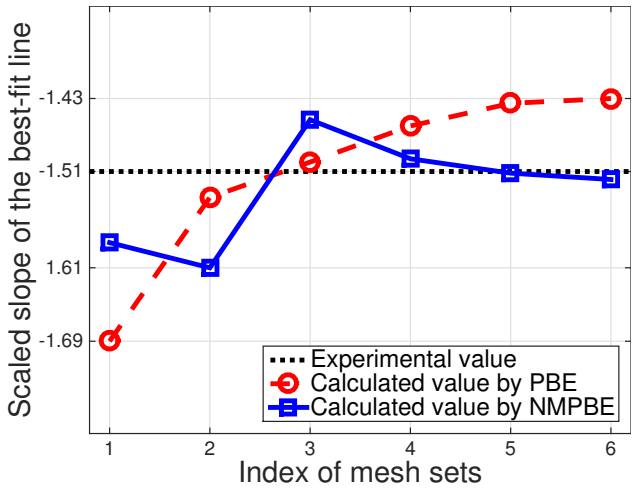


Figure 4.3: The scaled slopes m_s of the six best-fitted lines calculated by the hybrid PBE and NMPBE solvers for a DNA-drug complex represented in PDB ID 1D86 based on the six sets of meshes listed in Table 3.3.

From these two figures, we can see that, with the same mesh sets, due to the consideration of water molecule correlations (hydrogen bond network), the numerical results of NMPBE is closer to the experiment data than those of PBE (See Figure 4.3). Meanwhile, it is remarkably surprising to observe that the numerical results of NMPBE have the obvious convergence behavior to the experiment data. Meanwhile, for case $I_s \approx 0.11$, Figure 4.4 also demonstrates the numerical stability of the new NMPBE solver. Furthermore, using the first mesh set, we mapped the calculated electrostatic potentials with units k_BT/e_c to the surfaces of the drug, the DNA, and the complex, respectively, and compared the differences using VMD (`http://www.ks.uiuc.edu/Research/vmd/`) before and after the binding process. As

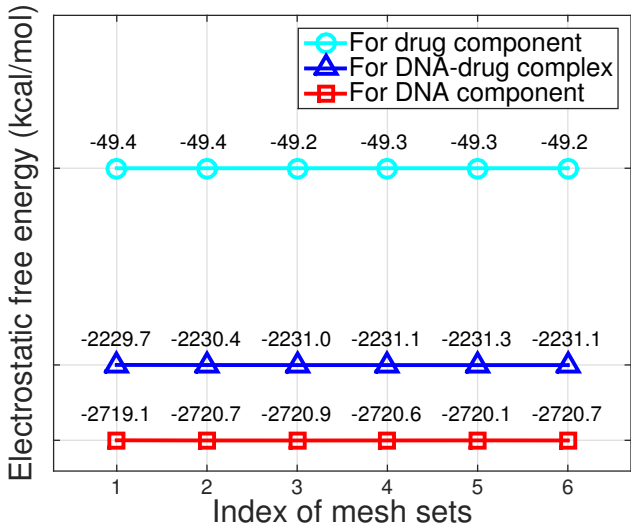


Figure 4.4: Electrostatic free energies (numbers on the top of each curve) of the complex, the DNA, and the drug calculated by the NMPBE hybrid solver on the same six mesh sets using $I_s \approx 0.11$.

illustrated in Figure 4.5, we can see that the DNA has negative electrostatic potentials at the binding site while the drug provides the positive ones, and after the binding process, they both are almost neutralized, which matched the physical energy theory and also validated the new solver. All of these observations just further guarantee the stability and reliability of the NMPBE solver in practical applications.

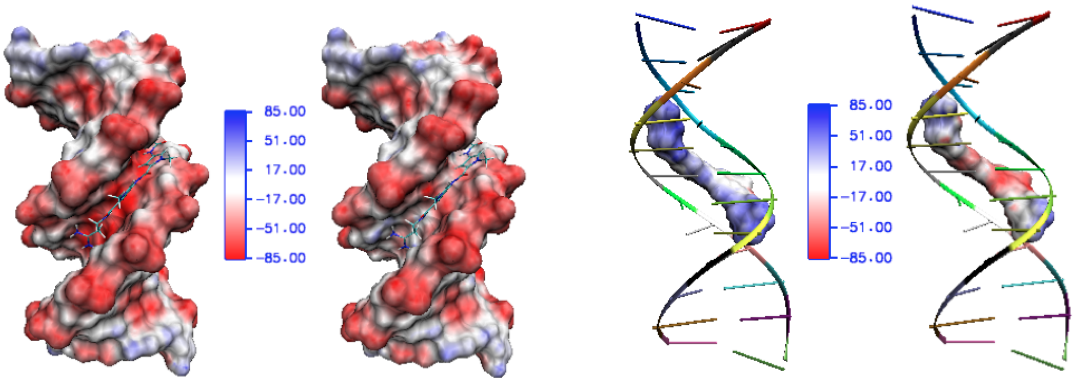


Figure 4.5: The electrostatic potentials mapping on the DNA surface and the drug surface, respectively, before and after binding for the complex with PDB ID 1D86. Here the unit of the electrostatic energy is k_BT/e_c, the drug structure is represented in CPK in left plot, and the ribbon represents the DNA structure in right plot.

Chapter 5

Conclusions and future work

In this dissertation, combining the Schwartz domain decomposition, the finite element, the finite difference, and the multigrid techniques, a new hybrid method based on a special seven box partition was constructed and then was applied to solve SMPBE and NMPBE for predicting the electrostatics of a biomolecule in an ionic solvent as the continuation work of [1]. The following gives a summary of the main work I have done in this dissertation and in my Ph.D period:

- A new box iterative method was constructed on a special seven box partition of a cubic domain, from which a new finite element and finite difference hybrid method was developed.

- Given a cubic domain, a scheme to automatically generate the seven box partition, the finite element mesh, and the uniform finite difference mesh was proposed and programmed in `C`; Meanwhile, a 3D mesh generator to produce the special finite element mesh was implemented in `C`.

- A linear solver – the preconditioned conjugate gradient method with a multigrid V-cycle preconditioner – was constructed and proved to be an optimal one to solve the second order elliptic problems on each neighboring box; Meanwhile, it was programmed in `Fortran`.

- Using the solution decomposition schemes, the reformulation techniques, and the modified Newton techniques, the new constructed method has been successfully applied to solve the dielectric continuum models (PBE, SMPBE, and NMPBE), resulting in

new hybrid solvers with improved efficiency; Furthermore, all new solvers have been programmed in the mixed of `C`, `Fortran`, and `Python` by combining together the readability of scripted language and the high efficiency of complied languages based on the state-of-the-art finite element library `DOLFIN` from the `FEniCS` project and the scientific computing toolkit `PETSc`; Numerous numerical experiments showed that new solvers have better efficiency in terms of CPU time compared to the corresponding finite element solvers. Meanwhile, related applications of the dielectric continuum models (solvation free energy and binding free energy calculations) were studied.

Based on the current work, the future work could include:

- For the uniform ion sizes, we have a PDE form similar to PBE. However, if the nonuniform ion sizes were considered, then we must solve a PDE-constrained minimization problem. The iterative algorithm to solve a PDE-constrained problem would be very different, and currently there is no published solver available. To fill in this gap, colleagues in our group and me will consider this nonuniform case and try to develop an efficient and accurate solver to compute the electrostatics with different ion sizes. Furthermore, the computed results with the nonuniform ion sizes will be compared to the results of PBE to further study the differences and the similarities of these two dielectric continuum models other than the improvements in the ion concentrations.

- One major advantage of the domain decomposition method is easy to parallel. To further reduce the computational cost, with the MPI library, the parallelized versions of those new hybrid solvers will be developed and released for publication.

BIBLIOGRAPHY

[1] J. Ying, D. Xie, A new finite element and finite difference hybrid method for computing electrostatics of ionic solvated biomolecule, Journal of Computational Physics 298 (2015) 636–651.

[2] J. Li, D. Xie, An effective minimization protocol for solving a size-modified Poisson-Boltzmann equation for biomolecule in ionic solvent, International Journal of Numerical Analysis and Modeling 12 (2) (2015) 286–301.

[3] D. Xie, Y. Jiang, A nonlocal modified Poisson-Boltzmann equation and finite element solver for computing electrostatics of biomolecules, Journal of Computational Physics, accepted, 2016.

[4] F. Fogolari, A. Brigo, H. Molinari, The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology, Journal of Molecular Recognition 15 (6) (2002) 377–392.

[5] B. Honig, A. Nicholls, Classical electrostatics in biology and chemistry, Science 268 (1995) 1144–1149.

[6] N. A. Baker, Poisson-Boltzmann methods for biomolecular electrostatics, Methods in Enzymology 383 (2004) 94–118.

[7] C. L. Vizcarra, S. L. Mayo, Electrostatics in computational protein design, Current Opinion in Chemical Biology 9 (6) (2005) 622–626.

[8] M. T. Neves-Petersen, S. B. Petersen, Protein electrostatics: a review of the equations and methods used to model electrostatic equations in biomolecules–applications in biotechnology, Biotechnology Annual Review 9 (2003) 315–395.

[9] J. Haile, Molecular dynamics simulation, Vol. 18, Wiley, New York, 1992.

[10] B. Roux, T. Simonson, Implicit solvent models, Biophysical Chemistry 78 (1) (1999) 1–20.

[11] B. Roux, S. Berneche, W. Im, Ion channels, permeation, and electrostatics: insight into the function of Kcsa, Biochemistry 39 (44) (2000) 13295–13306.

[12] W. Im, J. Chen, C. L. Brooks, Peptide and protein folding and conformational equilibria: theoretical treatment of electrostatics and hydrogen bonding with implicit solvent models, Advances in Protein Chemistry 72 (2005) 173–198.

[13] M. Feig, C. L. Brooks, Recent advances in the development and application of implicit solvent models in biomolecule simulations, Current Opinion in Structural Biology 14 (2) (2004) 217–224.

[14] P. Ren, J. Chun, D. G. Thomas, M. J. Schnieders, M. Marucho, J. Zhang, N. A. Baker, Biomolecular electrostatics and solvation: a computational perspective, Quarterly Revews of Biophysics 45 (04) (2012) 427–491.

[15] S.-w. W. Chen, B. Honig, Monovalent and divalent salt effects on electrostatic free energies defined by the nonlinear Poisson-Boltzmann equation: application to DNA binding reactions, The Journal of Physical Chemistry B 101 (44) (1997) 9113–9118.

[16] G. Arya, Q. Zhang, T. Schlick, Flexible histone tails in a new mesoscopic oligonucleosome model, Biophysical Journal 91 (1) (2006) 133–150.

[17] G. Arya, T. Schlick, Role of histone tails in chromatin folding revealed by a mesoscopic oligonucleosome model, Proceedings of the National Academy of Sciences 103 (44) (2006) 16236–16241.

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

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

[20] S. Jo, M. Vargyas, J. Vasko-Szedlar, B. Roux, W. Im, PBEQ-Solver for online visualization of electrostatic potential of biomolecules, Nucleic Acids Research 36 (suppl 2) (2008) W270–W275.

[21] N. Smith, S. Witham, S. Sarkar, J. Zhang, L. Li, C. Li, E. Alexov, DelPhi web server v2: incorporating atomic-style geometrical figures into the computational protocol, Bioinformatics 28 (12) (2012) 1655–1657.

[22] S. Sarkar, S. Witham, J. Zhang, M. Zhenirovskyy, W. Rocchia, E. Alexov, DelPhi Web Server: a comprehensive online suite for electrostatic calculations of biological macromolecules and their complexes, Communications in Computational Physics 13 (1) (2013) 269.

[23] 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 brownian dynamics program, Computer Physics Communications 62 (2) (1991) 187–197.

[24] Y. Jiang, J. Ying, D. Xie, A Poisson-Boltzmann equation test model for protein in spherical solute region and its applications, Molecular Based Mathematical Biology 2 (1) (2014) 86–97.

[25] Y. Zhou, M. Feig, G.-W. Wei, Highly accurate biomolecular electrostatics in continuum dielectric environments, Journal of Computational Chemistry 29 (1) (2008) 87–97.

[26] C. S. Peskin, The immersed boundary method, Acta numerica 11 (0) (2002) 479–517.

[27] Y. Liu, Y. Mori, Roperties of discrete delta functions and local convergence of the immersed boundary method, SIAM Journal on Numerical Analysis 50 (6) (2012) 2986–3015.

[28] Z. Li, T. Lin, X. Wu, New cartesian grid methods for interface problems using the finite element formulation, Numerische Mathematik 96 (1) (2003) 61–98.

[29] R. J. Leveque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM Journal on Numerical Analysis 31 (4) (1994) 1019–1044.

[30] Z. Li, K. Ito, The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains, Vol. 33, SIAM, 2006.

[31] J. L. Hellrung, L. Wang, E. Sifakis, J. M. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, Journal of Computational Physics 231 (4) (2012) 2015–2048.

[32] S. Yu, Y. Zhou, G.-W. Wei, Matched interface and boundary (MIB) method for elliptic problems with sharp-edged interfaces, Journal of Computational Physics 224 (2) (2007) 729–756.

[33] J. Wang, Q. Cai, Z.-L. Li, H.-K. Zhao, R. Luo, Achieving energy conservation in Poisson–Boltzmann molecular dynamics: accuracy and precision with finite-difference algorithms, Chemical Physics Letters 468 (4) (2009) 112–118.

[34] D. Chen, Z. Chen, C. Chen, W. Geng, G.-W. Wei, MIBPB: a software package for electrostatic analysis, Journal of Computational Chemistry 32 (4) (2011) 756–770.

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

[36] W. Ying, C. S. Henriquez, A kernel-free boundary integral method for elliptic boundary value problems, Journal of Computational Physics 227 (2) (2007) 1046–1074.

[37] W. Ying, A kernel-free boundary integral method for the nonlinear Poisson-Boltzmann equation.

[38] S. C. Brenner, R. Scott, The mathematical theory of finite element methods, Vol. 15, Springer, 2008.

[39] A. Logg, K.-A. Mardal, G. Wells, Automated solution of differential equations by the finite element method: the FEniCS book, Vol. 84, Springer Science & Business Media, 2012.

[40] A. Logg, G. N. Wells, DOLFIN: automated finite element computing, ACM Transactions on Mathematical Software (TOMS) 37 (2) (2010) 20.

[41] H. Si, Tetgen, A quality tetrahedral mesh generator and three-dimensional Delaunay triangulator. Version 1.

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

[43] D. Xie, New solution decomposition and minimization schemes for Poisson-Boltzmann equation in calculation of biomolecular electrostatics, Journal of Computational Physics 275 (2014) 294–309.

[44] H. A. Schwarz, Ueber einige Abbildungsaufgaben., Journal für die reine und angewandte Mathematik 70 (1869) 105–120.

[45] H. A. Schwarz, Gesammelte mathematische abhandlungen, Vol. 2, American Mathematical Soc., 1972.

[46] I. Babuška, The Schwarz algorithm in partial differential equations of mathematical physics, Czech. Math. J 83 (8) (1958) 328–343.

[47] J. H. Bramble, J. E. Pasciak, J. P. Wang, J. Xu, Convergence estimates for product iterative methods with applications to domain decomposition, Mathematics of Computation 57 (195) (1991) 1–21.

[48] M. Dryja, O. Widlund, An additive variant of the Schwarz alternating method for the case of many subregions, Ultracomputer Research Laboratory, Univ., Courant Inst. of Mathematical Sciences, Division of Computer Science, 1987.

[49] K. Miller, Numerical analogs to the Schwarz alternating procedure, Numerische Mathematik 7 (2) (1965) 91–103.

[50] P.-L. Lions, On the Schwarz alternating method. I, in: First international symposium on domain decomposition methods for partial differential equations, Paris, France, 1988, pp. 1–42.

[51] T. Mathew, Domain decomposition methods for the numerical solution of partial differential equations, Vol. 61, Springer Science & Business Media, 2008.

[52] A. Toselli, O. Widlund, Domain decomposition methods: algorithms and theory, Vol. 3, Springer, 2005.

[53] V. Dolean, P. Jolivet, F. Nataf, An introduction to domain decomposition methods: algorithms, theory, and parallel implementation, Vol. 144, SIAM, 2015.

[54] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, D. P. Young, Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation, SIAM Journal on Scientific Computing 19 (1) (1998) 246–265.

[55] X.-C. Cai, D. E. Keyes, Nonlinearly preconditioned inexact newton algorithms, SIAM Journal on Scientific Computing 24 (1) (2002) 183–200.

[56] Y. Jiang, Y. Xie, J. Ying, D. Xie, Z. Yu, SDPBS web server for calculation of electrostatics of ionic solvated biomolecules, Molecular Based Mathematical Biology 3 (1) (2015) 179–196.

[57] D. Xie, J. Ying, A new box iterative method for a class of nonlinear interface problems with application in solving Poisson-Boltzmann equation, Journal of Computational and Applied Mathematics 307 (2016) 319–334.

[58] I. Borukhov, D. Andelman, H. Orland, Steric effects in electrolytes: a modified Poisson-Boltzmann equation, Physical Review Letters 79 (3) (1997) 435.

[59] V. B. Chu, Y. Bai, J. Lipfert, D. Herschlag, S. Doniach, Evaluation of ion binding to DNA duplexes using a size-modified Poisson-Boltzmann theory, Biophysical Journal 93 (9) (2007) 3202–3209.

[60] N. Wang, S. Zhou, P. M. Kekenes-Huskey, B. Li, J. A. McCammon, Poisson–Boltzmann versus size-modified Poisson–Boltzmann electrostatics applied to lipid bilayers, The Journal of Physical Chemistry B 118 (51) (2014) 14827–14832.

[61] I. Borukhov, D. Andelman, H. Orland, Adsorption of large ions from an electrolyte solution: a modified Poisson–Boltzmann equation, Electrochimica Acta 46 (2) (2000) 221–229.

[62] B. Li, Continuum electrostatics for ionic solutions with non-uniform ionic sizes, Nonlinearity 22 (4) (2009) 811–833.

[63] J. H. Chaudhry, S. D. Bond, L. N. Olson, Finite element approximation to a finite-size modified Poisson-Boltzmann equation, Journal of Scientific Computing 47 (3) (2011) 347–364.

[64] S. Lamperski, C. Outhwaite, Exclusion volume term in the inhomogeneous Poisson-Boltzmann theory for high surface charge, Langmuir 18 (9) (2002) 3423–3424.

[65] O. Stern, Z elektrochem, Angew Phys Chem 30 (1924) 508–516.

[66] D. Henderson, Recent progress in the theory of the electric double layer, Progress in Surface Science 13 (3) (1983) 197–224.

[67] A. G. Volkov, D. Deamer, D. Tanelian, V. Markin, Electrical double layers at the oil/water interface, Progress in Surface Science 53 (1) (1996) 1–134.

[68] R. D. Coalson, A. M. Walsh, A. Duncan, N. Ben-Tal, Statistical mechanics of a coulomb gas with finite size particles: a lattice field theory approach, The Journal of Chemical Physics 102 (11) (1995) 4584–4594.

[69] K. Andresen, R. Das, H. Y. Park, H. Smith, L. W. Kwok, J. S. Lamb, E. Kirkland, D. Herschlag, K. Finkelstein, L. Pollack, Spatial distribution of competing ions around DNA in solution, Physical Review Letters 93 (24) (2004) 248103.

[70] Z. Tang, L. Scriven, H. Davis, A three-component model of the electrical double layer, The Journal of Chemical Physics 97 (1) (1992) 494–503.

[71] G. Tresset, Generalized Poisson-Fermi formalism for investigating size correlation effects with multiple ions, Physical Review E 78 (6) (2008) 061506.

[72] A. H. Boschitsch, P. V. Danilov, Formulation of a new and simple nonuniform size-modified Poisson-Boltzmann description, Journal of Computational Chemistry 33 (11) (2012) 1152–1164.

[73] B. Li, Minimization of electrostatic free energy and the Poisson-Boltzmann equation for molecular solvation with implicit solvent, SIAM Journal on Mathematical Analysis 40 (6) (2009) 2536–2566.

[74] S. Zhou, Z. Wang, B. Li, Mean-field description of ionic size effects with nonuniform ionic sizes: a numerical approach, Physical Review E 84 (2) (2011) 021901.

[75] G. R. Smith, M. J. Sternberg, Prediction of protein–protein interactions by docking methods, Current Opinion in Structural Biology 12 (1) (2002) 28–35.

[76] M. V. Fedorov, A. A. Kornyshev, Unravelling the solvent response to neutral and charged solutes, Molecular Physics 105 (1) (2007) 1–16.

[77] A. Rubinstein, S. Sherman, Influence of the solvent structure on the electrostatic interactions in proteins, Biophysical Journal 87 (3) (2004) 1544–1557.

[78] J. P. Bardhan, Gradient models in molecular biophysics: progress, challenges, opportunities, Journal of the Mechanical Behavior of Materials 22 (5-6) (2013) 169–184.

[79] M. V. Basilevsky, D. F. Parsons, Nonlocal continuum solvation model with exponential susceptibility kernels, The Journal of Chemical Physics 108 (21) (1998) 9107–9113.

[80] A. Hildebrandt, R. Blossey, S. Rjasanow, O. Kohlbacher, H.-P. Lenhof, Novel formulation of nonlocal electrostatics, Physical Review Letters 93 (10) (2004) 108104.

[81] A. Hildebrandt, Biomolecules in a structured solvent (2005).

[82] S. Weggler, V. Rutka, A. Hildebrandt, A new numerical method for nonlocal electrostatics in biomolecular simulations, Journal of Computational Physics 229 (11) (2010) 4059–4074.

[83] J. P. Bardhan, A. Hildebrandt, A fast solver for nonlocal electrostatic theory in biomolecular science and engineering, in: Proceedings of the 48th Design Automation Conference, ACM, 2011, pp. 801–805.

[84] D. Xie, Y. Jiang, P. Brune, L. R. Scott, A fast solver for a nonlocal dielectric continuum model, SIAM Journal on Scientific Computing 34 (2) (2012) B107–B126.

[85] D. Xie, Y. Jiang, L. R. Scott, Efficient algorithms for a nonlocal dielectric model for protein in ionic solvent, SIAM Journal on Scientific Computing 35 (6) (2013) B1267–B1284.

[86] U. Trottenberg, C. W. Oosterlee, A. Schuller, Multigrid, Academic press, 2000.

[87] Y. Saad, Iterative methods for sparse linear systems, SIAM, 2003.

[88] J. Xu, Iterative methods by space decomposition and subspace correction, SIAM review 34 (4) (1992) 581–613.

[89] J. Xu, J. Zou, Some nonoverlapping domain decomposition methods, SIAM review 40 (4) (1998) 857–914.

[90] T. J. Dolinsky, J. E. Nielsen, J. A. McCammon, N. A. Baker, PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations, Nucleic Acids Research 32 (suppl 2) (2004) W665–W667.

[91] Z. Yu, M. J. Holst, Y. Cheng, J. A. McCammon, Feature-preserving adaptive mesh generation for molecular shape modeling and simulation, Journal of Molecular Graphics and Modelling 26 (8) (2008) 1370–1380.

[92] W. E. Lorensen, H. E. Cline, Marching cubes: a high resolution 3d surface construction algorithm, ACM Siggraph Computer Graphics 21 (4) (1987) 163–169.

[93] J. Hoffman, J. Jansson, A. Logg, G. Wells, et al., DOLFIN, URL: http://www. fenics. org/dolfin.

[94] S. Balay, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, H. Zhang,

PETSc Web page, http://www.mcs.anl.gov/petsc (2014).
URL http://www.mcs.anl.gov/petsc

[95] O. Tatebe, The multigrid preconditioned conjugate gradient method, in: NASA Conference Publication, NASA, 1993, pp. 621–634.

[96] J. H. Bramble, Multigrid methods, Vol. 294, CRC Press, 1993.

[97] S. Balay, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.4, Argonne National Laboratory (2013).
URL http://www.mcs.anl.gov/petsc

[98] C. Bertonati, B. Honig, E. Alexov, Poisson-Boltzmann calculations of nonspecific salt effects on protein-protein binding free energies, Biophysical Journal 92 (6) (2007) 1891–1899.

[99] D. Sitkoff, K. A. Sharp, B. Honig, Accurate calculation of hydration free energies using macroscopic solvent models, The Journal of Physical Chemistry 98 (7) (1994) 1978–1988.

[100] M. O. Fenley, R. C. Harris, B. Jayaram, A. H. Boschitsch, Revisiting the association of cationic groove-binding drugs to DNA using a Poisson-Boltzmann approach, Biophysical Journal 99 (3) (2010) 879–886.

[101] G. S. Manning, The molecular theory of polyelectrolyte solutions with applications to the electrostatic properties of polynucleotides, Quarterly Reviews of Biophysics 11 (02) (1978) 179–246.

[102] K. J. Breslauer, D. P. Remeta, W.-Y. Chou, R. Ferrante, J. Curry, D. Zaunczkowski, J. G. Snyder, L. A. Marky, Enthalpy-entropy compensations in drug-DNA binding studies, Proceedings of the National Academy of Sciences 84 (24) (1987) 8922–8926.

[103] J. Li, D. Xie, A new linear Poisson-Boltzmann equation and finite element solver by solution decomposition approach, Communications in Mathematical Sciences 13 (2) (2015) 315–325.

[104] 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.

[105] W. Rocchia, Poisson-Boltzmann equation boundary conditions for biological applications, Mathematical and Computer Modelling 41 (10) (2005) 1109–1118.

[106] D. Xie, H. W. Volkmer, J. Ying, Analytical solutions of nonlocal poisson dielectric models with multiple point charges inside a dielectric sphere, Physical Review E 93 (4) (2016) 043304.

[107] T. Liu, M. Chen, B. Lu, Parameterization for molecular Gaussian surface and a comparison study of surface mesh generation, Journal of Molecular Modeling 21 (5) (2015) 1–14.

CURRICULUM VITAE

Jinyong Ying

Place of birth: Zhejiang, China

Education

Ph.D in Mathematics, University of Wisconsin-Milwaukee, US, 2016

M.S in Mathematics, Hunan University, China, 2010

B.S in Information and Computational Sciences, Hunan University, China, 2008

Employment

- TA, Department of Mathematics and Statistics, Memorial University of Newfoundland, Canada, 2010–2012
- TA and RA, Department of Mathematical Sciences, University of Wisconsin-Milwaukee, US, 2012–2015

Awards

1. Distinguished Dissertation Fellowship, UW-Milwaukee — Sep 2015 to May 2016
2. Research Excellence Award, UW-Milwaukee — Sep 2015 to May 2016
3. IMA workshop Travel Support, University of Minnesota — July 2015
4. Research Excellence Award, UW-Milwaukee — June 2015 to August 2015
5. Mark Lawrence Teply Award, UW-Milwaukee — May 2015
6. Morris and Miriam Marden Award, UW-Milwaukee — May 2014
7. Chancellor's Graduate Student Award, UW-Milwaukee — Sep 2012

Publications

1. D. Xie, H. Volkmer, and J. Ying, Analytical solutions of two nonlocal Poisson dielectric test models with spherical solute region containing multiple point charges, Physical Review E, 93 (11) (2016) 043304.

2. D. Xie, and J. Ying, A new box iterative method for a class of nonlinear interface problems with application in solving Poisson-Boltzmann equation, Journal of Computational and Applied Mathematics, 307 (2016): 319-334.

3. Y. Jiang, Y. Xie, J. Ying, D. Xie, and Z. Yu, SDPBS web server for calculation of electrostatics of ionic solvated biomolecules, Molecular Based Mathematical Biology, 3 (2015): 179-196.

4. J. Ying, and D. Xie, A new finite element and finite difference hybrid method for computing electrostatics of ionic solvated biomolecule, Journal of Computational Physics, 298 (2015): 636-651.

5. J. Ying, and Y. Yuan, Codimension one and two bifurcations in a symmetrical ring network with delay, Journal of Mathematical Analysis and Applications, 425 (2015): 1155-1176.

6. Y. Jiang, J. Ying, and D. Xie, A Poisson-Boltzmann equation test model for protein in spherical solute region and its applications, Molecular Based Mathematical Biology, 2 (2014): 86-97.

7. J. Ying, and Y. Yuan, Pattern formation in a symmetrical network with delay, Nonlinear Analysis: Real World Applications, 14(2) (2013): 1102-1113.

8. J. Ying, S. Guo, and Y. He, Multiple periodic solutions in a delay-coupled system of neural oscillators, Nonlinear Analysis: Real World Applications, 12(5) (2011): 2767-2783.