Spring 2024

Tools for Biomolecular Modeling and Simulation

Xin Yang
Southern Methodist University, xiny@smu.edu

TOOLS FOR BIOMOLECULAR MODELING AND SIMULATION

Approved by:

Dr. Weihua Geng
Associate Professor of Mathematics

Dr. Johannes Tausch
Professor of Mathematics

Dr. Benno Rumpf
Associate Professor of Mathematics

Dr. Peng Tao
Associate Professor of Chemistry

TOOLS FOR BIOMOLECULAR MODELING AND SIMULATION

A Dissertation Presented to the Graduate Faculty of the

Dedman College

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Department of Mathematics

by

Xin Yang

B.S., Environmental Engineering, Southern Methodist University
M.S., Computational and Applied Mathematics, Southern Methodist University

May 11, 2024

Acknowledgments

I am deeply grateful for the enriching opportunities and support that have shaped my academic journey, and I wish to express my heartfelt appreciation to the following individuals and organizations:

- First and foremost, I extend my sincere gratitude to Professor Weihua Geng, who has been my advisor since my undergraduate research studies. His unwavering support, mentorship, and profound influence on both my academic pursuits and personal growth have been invaluable. I vividly remember attending his class during my sophomore year, which marked a significant moment in my academic journey, sparking my interest in mathematics and research. His patience, encouragement, and passion have fostered in me a commitment to research characterized by thorough attention to detail.

- Additionally, I am equally appreciative of my esteemed committee members, Professor Johannes Tausch and Professor Benno Rumpf, for their invaluable guidance, insights, and contributions to my academic journey. Professor Tausch's mentorship regarding the expectations of a Ph.D. student, along with his course on Computational Electromagnetics, significantly enhanced my knowledge and comprehension in this field. Similarly, Professor Benno Rumpf's captivating course on Dynamical Systems fueled my enthusiasm for various mathematical topics. I am also thankful to Professor Peng Tao for generously serving on the committee.

- Furthermore, I extend special appreciation to Professor Daniel Reynolds for his enlightening classes, programming support, and invaluable guidance in the realm of

high-performance computing, which has significantly shaped my career trajectory. Additionally, I extend my gratitude to Professor Brandilyn Stigler for her course on Computational Algebraic Geometry, which left a lasting impression on me and ignited a passion for the subject. I deeply appreciate her mentorship, encouragement, and the pivotal role her reference letter played in my job search endeavors.

- As I look back on the beginning of my Ph.D. journey, I am thankful to Professor Yunkai Zhou for his recommendation letter, which played a crucial role in my admission to the Ph.D. program. I also want to express my gratitude to all the exceptional professors from my undergraduate studies, particularly Professors Patricia Taylor, Mark Boyd and John Easton. Their encouragement and support motivated me to pursue further education at Southern Methodist University in Dallas. Additionally, I deeply appreciate the faculty at the Hart Center for Engineering Leadership for their invaluable support in guiding my career path during both my undergraduate and graduate studies.

- I extend my appreciation to the Mathematics Department at Southern Methodist University for providing a conducive and supportive research environment. Special acknowledgment is also due to the Office of Information Technology and the O'Donnell Data Science and Research Computing Institute for their unwavering resources and support.

- Lastly, I am deeply grateful for the financial support of my family and the emotional support of my friends, especially Tianpeng Wu and Lizuo Liu, whose encouragement has been invaluable throughout my educational journey.

Yang, Xin B.S., Environmental Engineering, Southern Methodist University
M.S., Computational and Applied Mathematics, Southern Methodist University

Tools for Biomolecular Modeling and Simulation

Advisor: Dr. Weihua Geng
Doctor of Philosophy degree conferred May 11, 2024
Dissertation completed April 23, 2024

Electrostatic interactions play a pivotal role in understanding biomolecular systems, influencing their structural stability and functional dynamics. The Poisson-Boltzmann (PB) equation, a prevalent implicit solvent model that treats the solvent as a continuum while describes the mobile ions using the Boltzmann distribution, has become a standard tool for detailed investigations into biomolecular electrostatics. There are two primary methodologies: grid-based finite difference or finite element methods and body-fitted boundary element methods. This dissertation focuses on developing fast and accurate PB solvers, leveraging both methodologies, to meet diverse scientific needs and overcome various obstacles in the field.

Chapter 2 introduces the Polarizable Multipole Poisson-Boltzmann (PM-PB) model, which integrates the AMOEBA force field with a linear PB equation. We integrate this PM-PB model with a regularized Matched Interface and Boundary PB (MIB-PB) solver. Within the MIB-PB solver framework, we implement boundary conditions on a truncated computational domain and employ numerical approaches incorporating surface potential gradients. This implementation is validated using Kirkwood's analytical solutions. This integrated approach yields improved PB simulation outcomes with a more precise and realistic source term. Additionally, we propose an ongoing study exploring a sophisticated

Polarizable Multipole Nonlinear PB model (PM-NPB) in future investigations to address the polarization caused by dipole as well as the nonlinear effects from the Boltzmann term.

Chapter 3 introduces two parallelized solvers for solving the Boundary Integral Poisson-Boltzmann (BI-PB) equations: the Direct-Sum Boundary Integral (DSBI) PB solver, developed using KOKKOS on GPUs, and the Treecode-Accelerated Boundary Integral (TABI) PB solver, developed by our collaborators using the Message Passing Interface (MPI) on CPUs. These solvers undergo evaluation for their parallel performance in solving the BI-PB equations on selected proteins crucial for understanding COVID-19 transmission, treatment, and prevention. The parallelization techniques discussed in this chapter significantly enhance computational efficiency for macromolecular simulations. Additionally, we identify a threshold value, denoted as $n_b = 250,000$, under the current hardware conditions on ManeFrameIII provided, by the O'Donnell Data Science and Research Computing Institute at Southern Methodist University, which serves as a guideline for selecting the appropriate solver.

Chapter 4 explores the unique capabilities of the Eulerian Solvent Excluded Surface (ESES) software, which can generate both Eulerian and Lagrangian surfaces. By interfacing ESES outputs, i.e., the molecular surface representations on both Cartesian grids and triangular elements, we conduct a thorough numerical assessment of surface discretization quality using two recently developed Poisson-Boltzmann (PB) solvers: a finite difference based MIB-PB solver and a boundary integral based TABI-PB solver. Our numerical simulations demonstrate that ESES facilitates the convergence of both solvers respectively when employing the Eulerian and Lagrangian representations of the molecular surface it generates.

Chapter 5 presents a novel Poisson-Boltzmann-based Machine Learning (PB-ML) model designed to predict the electrostatic solvation free energies of biomolecules. Through meticulous evaluation of various PB solvers, we choose the most accurate MIB-PB solver for generating ML labels. Leveraging advanced techniques like the Multiscale Weighted Colored

Subgraph (MWCS) method and the Generalized Born (GB) model, we extract ML features essential for the prediction task. Subsequently, we assess the performance of several ML algorithms to determine the best one. The culmination of our research leads to the development of a PBML model trained on a dataset comprising over 4000 biomolecules using the DNN architecture. This model demonstrates superior efficiency and accuracy in predicting electrostatics compared to traditional grid-based PB solvers.

In summary, the preceding four chapters serve as the foundation of this dissertation, offering significant contributions to the advancement of PM-PB modeling, the parallelization of BI-PB solvers, and the integration of machine learning for accurate and efficient biomolecular electrostatic computations. Chapter 6 elaborates on software dissemination, while Chapter 7 discusses the dissertation contributions, along with outlining future avenues for research.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF TABLES

To my dad, Tian-Min Yang, who nurtured my intellectual curiosity and provided the means for my educational journey.

Chapter 1

Introduction

Electrostatic interactions play a pivotal role in the study of biomolecular systems, contributing significantly to their structural stability and functional dynamics. However, fully understanding the strength and characteristics of these interactions requires a deep comprehension of the physical and chemical properties of molecules in "aqueous" solutions.

To gain a comprehensive understanding of aqueous solutions, it is essential to employ models that consider both solute and solvent molecules, along with their interactions. The widely used molecular mechanics approach utilizes a "force field", incorporating nonbonded van der Waals and electrostatic terms to capture intermolecular interactions. Describing solvent effects accurately at the molecular level necessitates computing the interactions among a vast number of molecules and averaging them over numerous solvent configurations. Although explicit solvent models offer detailed insights into solvent-mediated biomolecular interactions, their computational efficiency diminishes for large systems due to the numerous solvent degrees of freedom. In contrast, implicit solvent models characterize solvent properties in an averaged or continuum manner, making them standard techniques for analyzing the energetics and dynamics of biomolecular systems.

In 1982, leveraging the advancements in comprehending the three-dimensional structure of proteins and the increased computational power, Warwicker and Watson [1] introduced a grid-based, finite difference approach for computing the electrostatic potential of nonspherical proteins by solving the Poisson–Boltzmann (PB) equation. This equation, derived from a continuum description of the solvent and counterion environment surrounding a biomolecule [2–5], is a nonlinear partial differential equation (PDE) that incorporates intri-

cate details about the biomolecular shape and charge distribution. Since their pioneering work, the PB equation has emerged as a standard method for conducting detailed investigations into biomolecular electrostatics. It has broad applications in biomolecular simulations, including protein structure [6], chromatin packing [7], protein pKa values [8–10], protein-membrane interactions, [11], binding energy [12], solvation-free energy [13], and ion channel profiling [14]. In addition to PB models, simpler approximate models have also been developed for continuum electrostatics. Notably, the generalized Born models, pioneered by Still et al. [15], have gained widespread popularity among these simpler models.

However, despite its widespread utility, the PB model is an elliptic interface problem with several numerical challenges:

1. Charge singularity: The potential exhibits singularity at atomic centers due to the singular point charge representation.

2. Geometric singularity: The solute-solvent boundary forms a complex interfacial molecular surface, characterized by cusps, sharp edges, sharp wedges and self-intersecting surfaces.

3. Discontinuous dielectric coefficients: Addressing the loss of regularity in the potential caused by the discontinuous dielectric coefficients across the interface requires an interface treatment. This complicates the scheme, decreasing the convergence speed of the iterative solver for the linear system.

4. Nonlinearity: Instability may arise from the nonlinearity caused by the strong ionic strength.

5. Unbounded condition: The model involves an infinite computational domain, necessitating significant memory resources.

There are two primary methodologies emerged to address these challenges:

1. Grid-based finite-difference or finite-element methods that discretize the entire computational domain, resulting in efficient and robust PB solvers such as APBS [16], AMBER [17,18], CHARMM [19], and Delphi [3,20]. In these schemes, singular charges are either interpolated to the grid or regularized using Green's function, while interface conditions are approximated, and the far-field boundary condition is enforced on a truncated domain. However, the inherent nature of discretizing the partial differential equation leads to a reduction in accuracy unless special treatments for interface [21,22] and singularity [23–27] are applied. Implementing these treatments requires more complicated schemes, potentially leading to a decrease in the convergence speed of the iterative solver for the linear system. Furthermore, discretizing both the solute and solvent domains imposes significant demands on computer power and resources.

2. Body-fitted boundary element methods that discretize the molecular surface, e.g. [28–40]. These schemes leverage the fact that in many cases, the differential equations can be reformulated into a set of boundary integral equations. This reformulation reduces the dimensionality of the problem by one and circumvents the challenges associated with domains extending to infinite. These methods offer advantages such as analytically incorporating singular charges, interface conditions, and far-field conditions in the formulation, inherent in the nature of a boundary integral formulation. Furthermore, the boundary integral approach can benefit from efficient acceleration using fast algorithms like the fast multipole method [41] and treecode [42,43], as well as parallel computing capabilities such as CPU and GPU processing. Despite these advantages, boundary integral PB solvers face their own set of challenges, including the computational cost of solving a dense linear system and the complexity of addressing the nonlinear PB equation under the boundary integral formulation.

This dissertation is dedicated to the development of fast and accurate PB solvers, which serve as invaluable tools for theoretical and computational bio-scientists. We utilize these two methodologies, each with its advantages and limitations, tailored for different scientific needs and challenges.

In Chapter 2, we address the challenges (1-3) by employing a regularized Matched Interface and Boundary (MIB) method [24]. This approach involves decomposing the potential into two parts, allowing the singular component to be analytically captured by the Green's function while ensuring that other components remain bounded. Additionally, we integrate a polarizable and multipolar (PM) force field AMOEBA, which offers a more realistic and accurate charge density representation, with our MIB-PB solver to solve the simplified linear PB (LPB) equation. However, for highly charged biomolecules, the assumption of the LPB model, where the electrostatic potential is weak enough, becomes invalid. Hence, the nonlinear PB (NPB) model emerges as a superior option compared to the LPB model for coupling with the PM sources. Nonetheless, implementing the PM-NPB model for real protein systems presents several challenges due to its highly nonlinear, recursive nature, and strong singularity in sources. To address these challenges, we propose a new regularization formulation and a linearized iterative algorithm for **future** study.

Our attention turns to Boundary Integral PB (BI-PB) solvers for macromolecular systems, where grid-based discretization becomes impractical. These solvers circumvent the challenges (1-3). In Chapter 3, we delve into the application of acceleration techniques like treecode and parallel computing to solve the dense linear system arising from the discretized boundary integral equations. Moreover, we incorporate a preconditioning scheme [44] to tackle challenges arising from low-quality triangles generated by the molecular surface generator, which can increase the number of GMRES iterations needed to solve the linear system. We present our development of CPU and GPU acceleration for two types of BI-PB solvers:

the Treecode-Accelerated BI-PB (TABI-PB) solver and the Direct-Sum BI-PB (DSBI-PB) solver, followed by a comparative analysis.

As mentioned above, molecular surface generators play a critical role in determining the accuracy and efficiency of PB solvers by defining discretized solute-solvent interfaces. In Chapter 4, we evaluate the performance of a novel molecular surface generator, ESES, in handling surface singularities like cusps and self-intersecting surfaces. We evaluate its applicability for both MIB-PB and BI-PB solvers.

Lastly, considering the significant computational expense involved in generating highly accurate electrostatic potentials for large biomolecules, as outlined in Chapter 5, we introduce a novel PB machine learning (PB-ML) model. By harnessing the power of machine learning methodologies, our model represents a significant leap forward in the field, offering promising avenues for addressing increasingly intricate computational challenges.

Indeed, while the Poisson-Boltzmann equation provides valuable insights into the electrostatics of biomolecular systems, it is important to recognize its limitations. As an approximate theory derived from the mean field of the electrolyte system, the PB equation neglects counterion correlations and fluctuations. These omissions can significantly impact the energetics of highly charged biomolecular systems such as DNA and RNA. Therefore, it is essential to carefully consider the applicability of the PB model for systems with high charge densities and acknowledge its limitations in such scenarios.

1.1. Implicit Solvent Models

The implicit solvent models average over the configuration space of solvent and counterion species surrounding the biomolecule. These models retain the electrostatic interactions without explicitly modeling solvent molecules, thereby reducing the dimensionality of the solvent-solute system. Consequently, the solvent is represented as a polarizable continuum, while the counterion distribution is depicted as a mean field charge "cloud", necessitating careful

modeling. To initiate our discussion, we delve into the details of the Poisson-Boltzmann (PB) and Generalized Born (GB) models as outlined by Baker [16] in this section.

1.1.1. Poisson-Boltzmann Model

We start with the canonical Poisson's equation, which describes the dimensionless electrostatic potential $\phi(x)$ in some finite domain Ω, generated by a charge distribution $\rho(x)$ in a polarizable continuum with the dielectric constant $\epsilon(x)$ and the Dirichlet boundary condition $g(x)$ on the boundary Γ:

$$\begin{aligned} -\nabla \cdot \epsilon(x) \nabla \phi(x) &= \rho(x) \text{ for } x \in \Omega, \\ \text{where } \phi(x) &= g(x) \text{ for } x \in \Gamma. \end{aligned} \tag{1.1}$$

There are two types of charge distribution for consideration in a biomolecular system. One is the "fixed" charge distribution for the partial atomic charges:

$$\rho_{\mathrm{f}}(x) = \frac{4\pi e_{\mathrm{c}}^2}{kT} \sum_{i=1}^{N} q_i \delta\left(x - x_i\right), \tag{1.2}$$

which models N atomic partial charges as delta functions $\delta(x - x_i)$ located at the atom centers x_i with magnitudes q_i. The coefficients include the charge of an electron e_c, and the thermal energy of the system kT to ensure the dimensionless form of the potential. The other one is the "mobile" charge distribution, where the counterions' contributions are approximated by a Boltzmann distribution in a continuous manner:

$$\rho_{\mathrm{m}}(x) = \frac{4\pi e_{\mathrm{c}}^2}{kT} \sum_{j}^{n} c_j Q_j \exp\left[-Q_j \phi(x) - V_j(x)\right], \tag{1.3}$$

for n counterion species with charges Q_i, bulk concentrations c_j, and steric potentials V_j. This equation reduces to:

$$\rho_{\mathrm{m}}(x) = \bar{\kappa}^2(x) \sinh \phi(x), \tag{1.4}$$

for a one-to-one electrolyte such as NaCl, where the coefficient $\bar{\kappa}^2(x)$ describes ion accessibility and bulk ionic strength.

Up to this point, we obtain the Poisson-Boltzmann equation for a one-to-one electrolyte considering both fixed and mobile distributions as:

$$\begin{gathered} -\nabla \cdot \epsilon(x)\nabla\phi(x) + \bar{\kappa}^2(x)\sinh\phi(x) = \frac{4\pi e_c^2}{kT}\sum_i q_i\delta\left(x - x_i\right) \text{ for } x \in \Omega, \\ \text{where } \phi(x) = g(x) \text{ for } x \in \Gamma. \end{gathered} \tag{1.5}$$

This nonlinear PB equation can be simplified to a linearized PB equation by taking the first-order approximation of $\sinh\phi(x) \approx \phi(x)$:

$$\begin{gathered} -\nabla \cdot \epsilon(x)\nabla\phi(x) + \bar{\kappa}^2(x)\phi(x) = \frac{4\pi e_c^2}{kT}\sum_i q_i\delta\left(x - x_i\right) \text{ for } x \in \Omega, \\ \text{where } \phi(x) = g(x) \text{ for } x \in \Gamma. \end{gathered} \tag{1.6}$$

The coefficients of the PB model capture the biomolecular structure. Specifically, the atomic positions and radii are part of the definitions of $\epsilon(x)$ and $\bar{\kappa}^2(x)$ through different molecular surface definitions. We use the SES representation (further elaborated in Chapter 4), where $\epsilon(x)$ is defined as discontinuous across the biomolecular surface with solute dielectric values inside and bulk solvent values outside the surface. As illustrated in Fig. 1.1(a), for a system of charges at positions $\mathbf{r}_i$ for $i = 1, ..., N_c$ with N_c as the total number of charges, the PB model is:

$$-\nabla \cdot \epsilon(\mathbf{r})\nabla\phi(\mathbf{r}) + \bar{\kappa}^2(\mathbf{r})\phi(\mathbf{r}) = \sum_{i=1}^{N_c} q_i\delta(\mathbf{r} - \mathbf{r}_i), \tag{1.7}$$

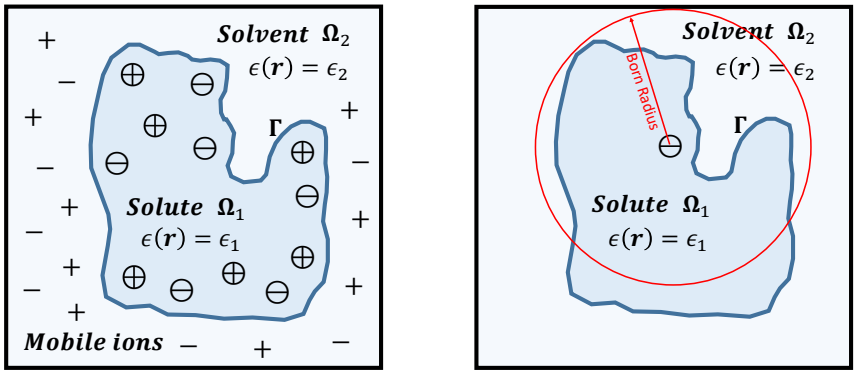
with

$$\epsilon(\mathbf{r}) = \begin{cases} \epsilon_1 & \text{for } \mathbf{r} \in \Omega_1, \\ \epsilon_2 & \text{for } \mathbf{r} \in \Omega_2, \end{cases} \tag{1.8}$$

and $\bar{\kappa}$ is the screening parameter with the relation $\bar{\kappa}^2 = \epsilon_2\kappa^2$, where κ is the inverse Debye length measuring the ionic effective length. The resulting interface conditions on the molecular surface are:

$$\phi_1(\mathbf{r}) = \phi_2(\mathbf{r}), \quad \epsilon_1 \frac{\partial \phi_1(\mathbf{r})}{\partial \nu} = \epsilon_2 \frac{\partial \phi_2(\mathbf{r})}{\partial \nu}, \quad \mathbf{r} \in \Gamma \tag{1.9}$$

where ϕ_1 and ϕ_2 are the limit values when approaching the interface from the inside and the outside the solute domain, and ν is the outward unit normal vector on Γ. The far-field boundary condition is $\lim_{|\mathbf{r}|\to\infty} \phi(\mathbf{r}) = 0$.



(a) (b)

Figure 1.1: An illustration of the PB model and the GB model. (a): the PB model in which the molecular surface Γ separates the computational domain into the solute region Ω_1 and solvent region Ω_2; (b): the GB model in which the molecule is represented by the superposition of spherical cavities with Born radii and centered charges (one is shown here).

1.1.2. Generalized Born Model

The Generalized Born (GB) model is based on the Born ion model, which characterizes the electrostatic potential and solvation energy of a spherical ion. It approximates the electrostatic properties of small molecules using an analytical expression derived from the Born ion model [45]. Unlike the PB model, the GB model fails to capture the molecular structure and ion distributions in detail. However, it offers a fast approach, suitable for our

PB-ML model in Chapter 5, to evaluating the electrostatics of the solvated biomolecules [16]. Below, we outline the derivations of the GB model.

The potential energy of a system of point charges in free space is the work done by external force in moving a charge q_i from the infinity to $\mathbf{r}_i$, where the potential is $\phi(\mathbf{r}_i)$, without acceleration,

$$W_i = q_i\phi(\mathbf{r}_i) = \frac{q_i}{4\pi\epsilon_0}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{q_j}{|\mathbf{r}_i-\mathbf{r}_j|}. \tag{1.10}$$

Note that $\lim_{|\mathbf{r}|\to\infty}\phi(\mathbf{r}) = 0$. Adding each charge in succession,

$$W = \frac{1}{4\pi\epsilon_0}\sum_{i=1}^{n}\sum_{j<i}\frac{q_iq_j}{|\mathbf{r}_i-\mathbf{r}_j|} = \frac{1}{8\pi\epsilon_0}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}\frac{q_iq_j}{|\mathbf{r}_i-\mathbf{r}_j|}. \tag{1.11}$$

For a continuous charge distribution $\rho(\mathbf{r}_i)$, the potential is:

$$\phi(\mathbf{r}) = \frac{1}{4\pi\epsilon_0}\int\frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}', \tag{1.12}$$

and then the work is:

$$W = \frac{1}{8\pi\epsilon_0}\int\int\frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}d\mathbf{r}' = \frac{1}{2}\int\rho(\mathbf{r})\phi(\mathbf{r})d\mathbf{r}. \tag{1.13}$$

Apply the Poisson's equation and integration by parts, the work is then:

$$W = \frac{1}{2}\int(-\epsilon_0\nabla^2\phi)(\phi)d\mathbf{r} = \frac{\epsilon_0}{2}\int|\nabla\phi|^2\,d\mathbf{r} = \frac{\epsilon_0}{2}\int|\mathbf{E}|^2\,d\mathbf{r} = \frac{1}{2}\int\mathbf{E}\cdot\mathbf{D}d\mathbf{r}. \tag{1.14}$$

Now consider assembling a charge to the center of a sphere of origin with ϵ_1 for $r < a$ and ϵ_2 for $r > a$ with radius a. The work is:

$$G_i = \frac{1}{8\pi\epsilon}\int\mathbf{D}\cdot\mathbf{D}d\mathbf{r} \approx \frac{1}{8\pi}\int_{\Omega_1}\frac{q_i^2}{r^4\epsilon_1}d\mathbf{r} + \frac{1}{8\pi}\int_{\Omega_2}\frac{q_i^2}{r^4\epsilon_2}d\mathbf{r}, \tag{1.15}$$

where Ω_1 and Ω_2 are the solute and solvent domains, $\epsilon_1 \in \Omega_1$ is the permittivity of solute, and $\epsilon_2 \in \Omega_2$ is the permittivity of solvent. The electrostatic free energy is:

$$\Delta G_{\text{solv},i} = \frac{1}{8\pi}(\frac{1}{\epsilon_2} - \frac{1}{\epsilon_1}) \int_{\Omega_2} \frac{q_i^2}{r^4} d\mathbf{r} = (\frac{1}{\epsilon_2} - \frac{1}{\epsilon_1}) \frac{q^2}{2a_i}. \tag{1.16}$$

As the molecules are treated as collection of atoms,

$$\Delta G_{\text{solv}} \approx \sum_i^N \frac{q_i^2}{2a_i}(\frac{1}{\epsilon_2} - \frac{1}{\epsilon_1}) + \frac{1}{2}\sum_i^N \sum_{j\neq i}^N \frac{q_i q_j}{r_{ij}}(\frac{1}{\epsilon_2} - \frac{1}{\epsilon_1}) \approx (\frac{1}{\epsilon_1} - \frac{1}{\epsilon_2})\frac{1}{2}\sum_{ij} \frac{q_i q_j}{f_{ij}^{\text{GB}}}, \tag{1.17}$$

where

$$f_{ij}^{\text{GB}} = \sqrt{r_{ij}^2 + R_i R_j \exp\left(-\frac{r_{ij}^2}{4R_i R_j}\right)}, \tag{1.18}$$

with the effective Born radius R_i for ith atom, and the distance r_{ij} between atoms i and j. The effective Born radii R_i is calculated by the following boundary integral:

$$R_i^{-1} = \Big(- \frac{1}{4\pi} \oint_\Gamma \frac{\mathbf{r} - \mathbf{r}_i}{|\mathbf{r} - \mathbf{r}_i|^6} \cdot \mathrm{d}\mathbf{S} \Big)^{1/3}. \tag{1.19}$$

1.2. Motivation

As mentioned earlier, the PB model presents several challenges. A primary concern lies in accurately and efficiently modeling molecular surfaces, as they significantly impact the stability of PB solvers. We choose the solvent excluded surface (SES) definition for all our PB models due to its C^1-continuous surface representation, as discussed further in Chapter 4. While existing SES software packages, such as MSMS [46], efficiently offer SESs in the Lagrangian representation, we require the conversion of its Lagrangian representation into the Eulerian form for the utilization of finite difference based PB solvers. This is where the Eulerian Solvent Excluded Surface (ESES) software becomes essential, as it provides both Lagrangian and Eulerian representations. We integrate ESES's outputs on both Cartesian

grids and triangular elements with the Eulerian (MIB) and Lagrangian (BI) PB solvers respectively to assess its efficacy.

Another concern arises regarding the source term in the PB model. In describing electrostatic interactions among highly charged biomolecules, the traditional partial charge representation for the source term lacks accuracy as it overlooks crucial polarization effects, i.e., the redistribution of the electron density in the presence of an external electric field. To address this limitation, we adopt a Polarizable Multipole (PM) source representation, leading to the utilization of our PM-PB model, and to capture the singularities from this source representation, we couple it with a regularized Matched Interface and Boundary (MIB) method.

Given the computational challenges associated with solving dense linear systems in boundary integral PB (BI-PB) solvers, particularly considering the problem size after discretizing molecular surfaces for various proteins, we have devised two BI-PB solvers optimized for different hardware configurations: a Direct-Sum Boundary Integral (DSBI) PB solver utilizing Kokkos on GPUs, and a Treecode-Accelerated Boundary Integral (TABI) PB solver employing MPI on CPUs. These parallel PB solvers are capable of delivering both global solvation energy and local surface potential in practical applications.

Despite these advancements, effectively and accurately solving PB models for large-scale modeling remains computationally expensive. Furthermore, insights gained from electrostatic analysis of one biomolecule may not directly apply to others, necessitating separate electrostatic analyses for different proteins or even the same protein with different protonation states or conformations. To address this challenge, we propose developing an efficient machine learning based tool. Leveraging the capability of the PB model to capture biomolecular structure, we plan to design a PB-based ML model for predicting the electrostatic properties based on the biomolecular structure. As the dataset grows in the future, there is a potential to incorporate experimental label data for training the ML model.

Chapter 2

A Regularized Matched Interface and Boundary Poisson-Boltzmann Solver with Polarizable Force Fields

In this chapter, we introduce a Polarizable Multipole Poisson-Boltzmann (PM-PB) model, which integrates the polarizable and multipolar Atomic Multipole Optimized Energetics for Biomolecular Applications (AMOEBA) force field with a linear PB equation. Our aim is to mitigate inaccuracies stemming from the traditional partial charge representation of charge density in the PB equation's source term. As the polarizable multipole source representation involves the Dirac delta functions and their derivatives, we utilize a regularized Matched Interface and Boundary (MIB) method [26] to analytically regularize the singular source term in the PB model, maintaining second-order accuracy by handling interface conditions. Initially, we incorporate the MIB-PB solver with the multipole source term and perform preliminary validation of this method on spherical cavities and several test proteins. The inclusion of polarizable dipoles is deferred to **future** implementations.

Additionally, we propose a more sophisticated PM Nonlinear PB (PM-NPB) model, given that PM source models amplify electrostatic interactions, resulting in substantial nonlinear effects in the Boltzmann term. We tackle the associated challenges, such as high nonlinearity, recursion, and strong singularity in sources, by proposing a new regularization formulation and a linearized iterative algorithm.

2.1. Introduction

The implicit solvent models are parameterized based on the force fields, and these force fields significantly influence the accuracy of molecular simulations. The most commonly

used force fields, known as the classical force fields, describe the charge distribution using the partial charge model [3,47]. In this model, point charges are positioned at atomic centers and are represented in terms of Dirac delta functions. These force fields have demonstrated success in numerous applications [48].

However, when describing electrostatic interactions among highly charged biomolecules, this discrete charge representation is recognized as an important source of the PB modeling errors [49], and it fails to capture the important polarization effects [50], i.e., the redistribution of the electron density in the presence of an external electric field. Instead, more sophisticated force fields are employed, which take into account not only the monopoles but also higher-order multipole components capable of polarization, to provide a more realistic and precise charge representation, leading to an improved accuracy in the simulations. Among these newer force fields, the AMOEBA force field [48,51] is the most popular one. It utilizes point multipoles, extending up to the quadrupole moment, to characterize the charge distribution and allows the dipole moment to incorporate a polarizable component. As a result, this model demonstrates notably improved agreement with experimental and high-level *ab initio* findings across various domains, including cluster structures, energetics, bulk thermodynamics, and structural characteristics for water [51], organic molecules [52], and proteins [53]. This force field is accessible through several packages, such as Tinker [54–56], OpenMM [57] and AMBER [58].

Besides the errors in PB source modeling, irregular interfaces and geometric singularities also pose challenges by introducing numerical instability and impeding the convergence of the PB solvers. To tackle these issues, we utilize the Matched Interface and Boundary (MIB) method, which rigorously enforces the flux jump conditions at the solvent-solute interface, ensuring highly accurate biomolecular electrostatics in continuum electric environments. However, when the mesh size nears half of the van der Waals radius, the accuracy of MIB decreases due to the overlap between grid points carrying the interface jump conditions

and those carrying the distributed singular charges. To address this limitation, we employ a regularization formulation that divides the PB solution into "regular" and "singular" parts, by adopting a Green's function formulation for the molecular surfaces of proteins with potential geometric singularities. This approach efficiently redistributes the impact of charge singularities into a set of interface jump conditions, which can be determined from solving the corresponding Laplace equation with given singular sources. Hence, we are capable of addressing geometric and charge singularities with equal efficacy.

Furthermore, for highly charged biomolecules [59], the electrostatic interactions are intensified, leading to amplified nonlinear effects on the Boltzmann term. Consequently, the assumption of the LPB model, which presumes weak electrostatic potentials, becomes untenable. Therefore, the nonlinear Poisson-Boltzmann (NPB) model emerges as a superior choice over the LPB model as an implicit solvent theory for interfacing with PM sources. However, implementing the PM-NPB model for real protein systems presents computational challenges due to its highly nonlinear, recursive self-consistent, and strongly singular nature in sources. To tackle these challenges, we propose a theoretical framework for **future** investigations: utilizing a Green's function-based decomposition method to analytically eliminate the charge singularities arising from the Dirac delta function and its derivatives, combined with an iterative algorithm to linearize the self-consistency recursion. This approach reduces the computational burden by solving a linearized Poisson-Boltzmann (LPB) equation in each iteration.

2.2. Polarizable Multipole Poisson-Boltzmann Model

Ideally, the representation of charge density should accurately reflect the electron density distribution obtained from computationally intensive quantum mechanical (QM) calculations [60, 61]. To achieve this, we employ a recently developed polarizable multipole (PM) model utilizing the AMOEBA force field [51, 53]. This approach enhances the accuracy of modeling electron density and polarization while preserving the atomic representation.

In AMOEBA, permanent multipoles such as dipoles and quadrupoles (see Appendix A.4) are expressed at atomic centers using derivatives of delta functions. The polarization of the solute is taken into account to compute induced dipoles. These additional dipole and quadrupole moments, representing lone pairs and π bonds respectively, contribute to a more precise depiction of electrostatic interactions and properties across diverse physical and chemical environments [50]. Notably, in this framework, only dipole moments are regarded as polarizable, while quadrupoles are treated as non-polarizable for simplicity.

As a result, this model demonstrates substantially enhanced agreement with experimental and high-level *ab initio* outcomes across a range of systems, encompassing cluster configurations, energetics, bulk thermodynamic properties, and structural metrics for water [51], organic molecules [52], and proteins [53]. Furthermore, the AMOEBA force field maintains a straightforward atomic architecture, facilitating its seamless substitution for the traditional partial charge model as the source in PB computations.

2.2.1. Polarizable Multipole Sources of the AMOEBA

Consider a protein containing N_c atoms. Each atomic center, denoted as $\mathbf{r}_n = (x_n, y_n, z_n)$, possesses permanent atomic monopole, dipole, and quadrupole moments, expressed as:

$$\mathbf{M}^n = [q^n, d_x^n, d_y^n, d_z^n, \Theta_{xx}^n, \Theta_{xy}^n, \ldots, \Theta_{zz}^n]^t, \tag{2.1}$$

where the superscript t denotes the transpose. As the permanent multipoles including dipoles and quadrupoles are in terms of derivatives of delta functions, the permanent charge at $\mathbf{r}_n$ in vacuum can be written as [51, 53]:

$$\rho^n(\mathbf{r}) = q^n\delta(\mathbf{r}-\mathbf{r}_n) + d_\alpha^n\partial_\alpha\delta(\mathbf{r}-\mathbf{r}_n) + \Theta_{\alpha\beta}^n\partial_{\alpha\beta}\delta(\mathbf{r}-\mathbf{r}_n), \tag{2.2}$$

where the subscripts α and β represent the x, y, or z components of a position vector or the corresponding differentiations. Based on this charge density representation, the Coulomb potential, which is the fundamental solution to Poisson's equation in Eq. (1.1):

$$G(\mathbf{r}) = \frac{1}{4\pi\epsilon_1} \sum_{n=1}^{N_c} \frac{q^n}{|\mathbf{r} - \mathbf{r}_n|}, \tag{2.3}$$

can then be expanded into terms of Green's function as:

$$\begin{aligned} G(\mathbf{r}) &= \frac{1}{4\pi\epsilon_1} \sum_{n=1}^{N_c} q^n \left[\frac{1}{|\mathbf{r} - \mathbf{r}_n|} - r_{n,\alpha} \frac{(r_\alpha - r_{n,\alpha})}{|\mathbf{r} - \mathbf{r}_n|^3} + r_{n,\alpha} r_{n,\beta} \frac{3\,(r_\alpha - r_{n,\alpha})\,(r_\beta - r_{n,\beta})}{2\,|\mathbf{r} - \mathbf{r}_n|^5} \right] \\ &= \frac{1}{4\pi\epsilon_1} q \left(\frac{1}{R} \right) - d_\alpha \nabla_\alpha \left(\frac{1}{R} \right) + \frac{1}{3} \Theta_{\alpha\beta} \nabla_\alpha \nabla_\beta \left(\frac{1}{R} \right), \end{aligned} \tag{2.4}$$

where R denotes the scalar distance, and the multipole moments are defined as:

$$q = \sum_{n=1}^{N_c} q^n, \tag{2.5a}$$

$$d_\alpha = \sum_{n=1}^{N_c} r_{n,\alpha} q^n, \tag{2.5b}$$

$$\Theta_{\alpha\beta} = \sum_{n=1}^{N_c} \frac{3}{2} r_{n,\alpha} r_{n,\beta} q^n - \frac{1}{2} r_n r_n \delta_{\alpha\beta}. \tag{2.5c}$$

Note that the total Coulomb potential in Eq. (2.4) is "additive" for all permanent multipoles $\mathbf{M} = [\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^{N_c}]^t$.

2.2.2. Polarization in the Vacuum Phase

The polarization is "non-additive", even in a vacuum, meaning that the total polarization cannot be simply summed from individual atomic polarizations. Based on the Green's

function, we propose to compute the induced dipole moment $\boldsymbol{\mu}^n$ at position $\mathbf{r}_n$, given by:

$$\boldsymbol{\mu}^n = \alpha^n \mathbf{E}^n = \alpha^n \left(\sum_{m \neq n} \nabla G^m(\mathbf{r}_n) + \sum_{m \neq n} \mathbf{T}_{nm} \boldsymbol{\mu}^m \right), \tag{2.6}$$

where α^n represents the isotropic atomic polarizability for the n-th atom and $\mathbf{E}^n$ denotes the total field at each site $\mathbf{r}_n$ arising from contributions of permanent multipole sites and induced dipoles. The gradient of the Coulomb potential due to the m-th atom, denoted as $\nabla G^m(\mathbf{r})$, can be analytically derived, while the tensor coefficient $\mathbf{T}_{nm}$ is a 3-by-3 interaction tensor provided in the AMOEBA force field, incorporating masking and Thole damping for a distance s_{nm} [62]:

$$\mathbf{T}_{nm} = \begin{bmatrix} \partial^2/\partial x_n \partial x_m & \partial^2/\partial x_n \partial y_m & \partial^2/\partial x_n \partial z_m \\ \partial^2/\partial y_n \partial x_m & \partial^2/\partial y_n \partial y_m & \partial^2/\partial y_n \partial z_m \\ \partial^2/\partial z_n \partial x_m & \partial^2/\partial z_n \partial y_m & \partial^2/\partial z_n \partial z_m \end{bmatrix} \frac{1}{s_{nm}}. \tag{2.7}$$

As shown in Fig. 2.1, the electric field $\mathbf{E}^n$ in Eq. (2.6) consists of both the polarization by other permanent multipoles, which is known as direct induction, and the polarization by other induced dipoles, which is known as mutual induction. The mutual polarization described in Eq. (2.6) represents a "self-consistent process" [62]. From a mathematical perspective, this self-consistent process can be viewed as an iterative procedure: the n-th induced dipole $\boldsymbol{\mu}^n$ depends on all other induced dipoles $\boldsymbol{\mu}^m$, while the new value of $\boldsymbol{\mu}^n$ computed using Eq. (2.6) will, in turn, influence other induced dipoles $\boldsymbol{\mu}^m$. To achieve an equilibrium state for electrostatic analysis, the self-consistent process must be iteratively computed.

We propose a non-iterative solution for Eq. (2.6). By rearranging the terms and combining the second term on the right-hand side (RHS) with the left-hand side (LHS), Eq.

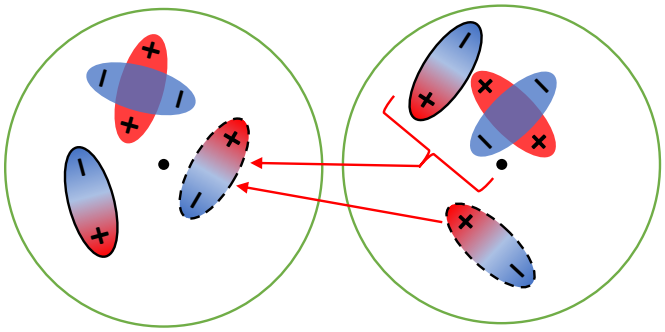


Figure 2.1: In vacuum, induced dipoles are determined by two polarizations: direct induction by other permanent multipoles and mutual induction by other induced dipoles. In the AMOEBA force field, all multipoles are defined at atomic centers. Here, they are placed off-centers for illustration purpose.

(2.6) can be reformulated into a linear system. The equilibrium state of the self-consistent mutual polarization in vacuum can then be determined by solving this linear system directly. We denote the solution as $\boldsymbol{\mu}^{(\mathrm{V})}$, where the superscript (V) denotes vacuum. The Coulomb potential G^{V} induced by $\boldsymbol{\mu}^{(\mathrm{V})}$ can be represented using Green's functions:

$$G^{\mathrm{V}}(\mathbf{r}) = \sum_{n=1}^{N_c} \frac{r_\alpha - r_{n,\alpha}}{|\mathbf{r} - \mathbf{r}_n|^3} \mu_\alpha^{(\mathrm{V}),n}, \quad E_{\mathrm{elec}}^{(\mathrm{V})} = \frac{1}{2} k_B T \int \left(G(\mathbf{r}) + G^{\mathrm{V}}(\mathbf{r})\right) \sum_{n=1}^{N_c} \rho^n(\mathbf{r})\, d\mathbf{r} \tag{2.8}$$

Once the Coulomb potential for permanent multipoles and induced dipoles, G and G^{V} respectively, are determined, the electrostatic energy in vacuum $E_{\mathrm{elec}}^{(\mathrm{V})}$ can be computed based on Eq. (2.8).

2.2.3. Polarization in the Solvated Phase

For discussing the situation in the solvated phase, consider a macromolecule with a low dielectric ϵ_1 immersed in a solvent with a high dielectric ϵ_2. We propose to characterize this polarization with three components as shown in Fig. 2.2: the direct induction by other permanent multipoles, the mutual induction by other induced dipoles, and the polarization

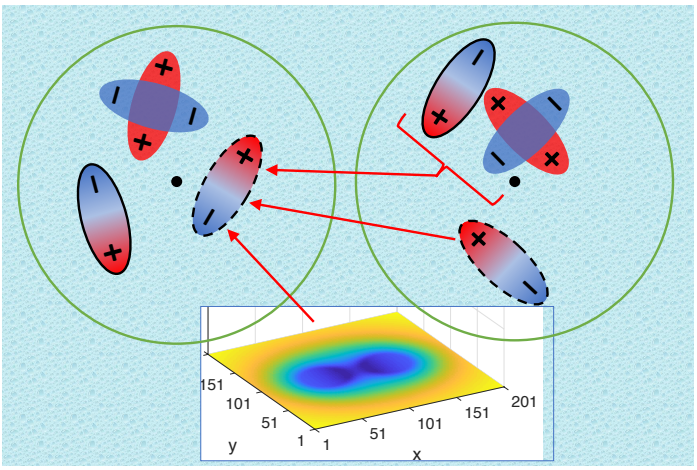


Figure 2.2: In solvent, besides direct induction and mutual induction, induced dipoles are also subject to polarization produced by the reaction field. In the AMOEBA force field, all multipoles are defined at atomic centers. Here, they are placed off-centers for illustration purpose.

induced by the total solvation reaction field ϕ_{RF}, so that the induced dipole is:

$$\boldsymbol{\mu}^n = \alpha^n \left(\sum_{m \neq n} \nabla G^m(\mathbf{r}_n) + \sum_{m \neq n} \mathbf{T}_{nm} \boldsymbol{\mu}^m - \nabla \phi_{\mathrm{RF}}(\mathbf{r}_n) \right), \tag{2.9}$$

where the reaction potential ϕ_{RF} is the difference between the electrostatic potential ϕ and Coulomb potential G, i.e., $\phi_{\mathrm{RF}} = \phi - G$. Since the potential ϕ_{RF} of NPB cannot be separated into the reaction field produced by all permanent multipoles and all induced dipoles, i.e. $\phi_{\mathrm{RF}} \neq \phi^{\mathbf{M}}_{\mathrm{RF}} + \phi^{\boldsymbol{\mu}}_{\mathrm{RF}}$, because of the nonlinearity like $\sinh(\phi)$ in Eq. (1.5), the polarization is now not only "non-additive", but also "inseparable. Different from Eq. (2.2) in the vacuum phase where only the permanent multipoles are involved, the total singular source ρ in the solvated phase contains not only the permanent multipoles but also the induced dipoles written as:

$$\rho = 4\pi \sum_{n=1}^{N_c} \left(q^n \delta(\mathbf{r} - \mathbf{r}_n) + (\mu^n_\alpha + d^n_\alpha) \partial_\alpha \delta(\mathbf{r} - \mathbf{r}_n) + \Theta^n_{\alpha\beta} \partial_{\alpha\beta} \delta(\mathbf{r} - \mathbf{r}_n) \right). \tag{2.10}$$

Note that the total dipole at $\mathbf{r}_n$ is $\mathbf{p}^n = \mathbf{d}^n + \boldsymbol{\mu}^n$.

2.2.4. Boundary Conditions of the PMPB Model

According to Schnieders [62], the Single Debye-Hückel (SDH) and multiple Debye-Hückel (MDH) boundary conditions represent two common approximations used to delineate the true potential when specifying Dirichlet boundary conditions for nonspherical solutes characterized by a set of atomic multipoles. The SDH approximation simplifies the process by consolidating all atomic multipole sites into a single multipole located at the solute's center, treating it as a "sphere" with radiues a. Conversely, the MDH approximation involves summing the contributions of each atomic multipole independently, considering them in isolation from all other sites that displace solvent. Therefore, to formulate the Dirichlet problem for a solute described by any number of atomic multipole sites, it is necessary to determine the potential outside a solvated multipole positioned at the center of a sphere. The detailed formulations can be found in Appendix A.11.

2.2.5. Analytical Solutions for A Spherical Cavity

Kirkwood's dielectric sphere is widely recognized as a robust benchmark for assessing the effectiveness of Poisson-Boltzmann (PB) solvers in terms of accuracy, convergence speed, and efficiency. Within this framework, we explore the analytical solutions derived from a point monopole, a point dipole, and a point quadrupole, each positioned at the center of a spherical cavity respectively. The detailed derivations for a dipole and a quadrupole can be found in Appendix A.8-A.9.

Consider a dielectric sphere with a radius a and an interior dielectric constant ϵ_1 (ϵ_i), immersed in an infinite dielectric medium characterized by a solvent dielectric constant ϵ_2 (ϵ_s), as shown in Fig. 2.3. An external electric field is induced by a fixed external charge distribution, configured such that in the scenario where $\epsilon_1 = \epsilon_2$, the electric field within the dielectric would adopt a uniform state denoted as E_0.

A conducting sphere within a uniform external field can be addressed by employing the general solution of Laplace's equation (see Appendix A.6) for the potential in regions devoid of charge, assuming axial symmetry. In regions outside and inside the sphere, Laplace's equation $\nabla^2\phi = 0$ holds true. However, on the surface of the sphere, Laplace's equation is not applicable. Therefore, we require two distinct functions, ϕ_{out} and ϕ_{in}, to represent the potential for regions outside and inside the sphere, respectively.

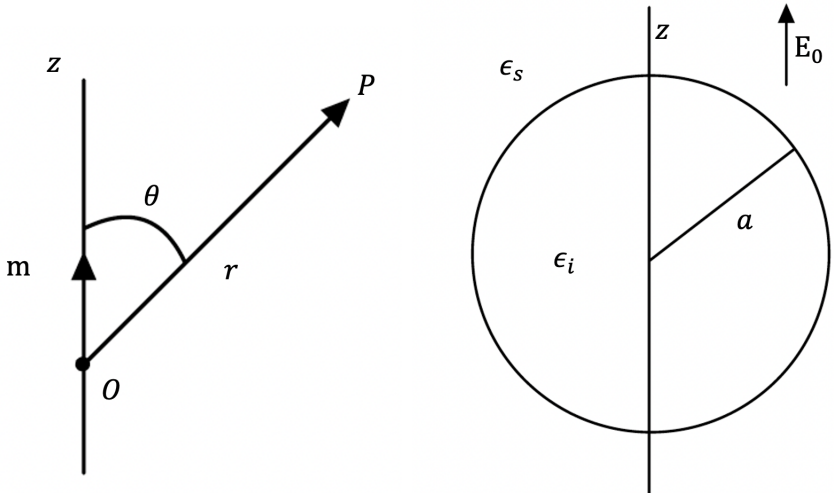


Figure 2.3: Potential measured at P with the dipole moment aligns with the field direction (redraw from Böttcher [63]).

The solution is described using spherical polar coordinates r, θ, and φ (see Appendix A.2), with the center of the spheres serving as the origin of a coordinate system. The z-axis aligns with the direction of the uniform field. In situations where there is symmetry about the z-axis, the general solution of Laplace's equation in terms of Legendre polynomials (see Appendix A.7) takes the following form:

$$\begin{aligned} \phi_{\text{out}} &= \sum_{n=0}^{\infty} \left(A_n r^n + \frac{B_n}{r^{n+1}} \right) \mathrm{P_n}(\cos\theta), \\ \phi_{\text{in}} &= \sum_{n=0}^{\infty} \left(C_n r^n + \frac{D_n}{r^{n+1}} \right) \mathrm{P_n}(\cos\theta), \end{aligned} \tag{2.11}$$

with the boundary conditions:

$$(\phi_{\text{out}})_{r\to\infty} = -E_0 z = -E_0 r\cos\theta, \tag{2.12}$$

$$(\phi_{\text{out}})_{r=a} = (\phi_{\text{in}})_{r=a}\,, \tag{2.13}$$

$$\epsilon_2 \left(\frac{\partial \phi_{\text{out}}}{\partial r}\right)_{r=a} = \epsilon_1 \left(\frac{\partial \phi_{\text{in}}}{\partial r}\right)_{r=a}. \tag{2.14}$$

We derive the following solutions (see Appendix A.8-A.9) neglecting the parameters and units based on the content from Böttcher [63].

Due to A Point Charge: Applying the boundary conditions in Eq. (2.12-2.14) and using the fact that the Legendre functions are linearly independent, the analytical solution satisfying the Laplace's equation for the potential due to a point charge is:

$$\phi_{\text{out}} = \frac{q}{r}, \tag{2.15a}$$

$$\phi_{\text{in}} = \left[\frac{1}{r} - \frac{\epsilon_2 - \epsilon_1}{\epsilon_2}\frac{1}{a}\right] q, \tag{2.15b}$$

where r is the distance between the potential measured and the centered charge q. The potential energy, with subscript m denoted for monopole, is:

$$U_m = -\frac{1}{2}\left(\frac{\epsilon_2 - \epsilon_1}{\epsilon_2}\right)\frac{q^2}{a}. \tag{2.16}$$

Due to A Point Dipole: Similarly, for a point dipole, we derive:

$$\phi_{\text{out}} = \frac{3\epsilon_2}{2\epsilon_2 + \epsilon_1}\frac{1}{r^3}\mathbf{d}\cdot\mathbf{r}, \tag{2.17a}$$

$$\phi_{\text{in}} = \left[\frac{1}{r^3} - \frac{2(\epsilon_2 - \epsilon_1)}{2\epsilon_2 + \epsilon_1}\frac{1}{a^3}\right]\mathbf{d}\cdot\mathbf{r}. \tag{2.17b}$$

where $\mathbf{d}$ stands for the dipole moment vector and $\mathbf{r}$ is the direction vector. The potential energy, with subscript d denoted for dipole, is:

$$U_d = -\frac{1}{2}\left(\frac{2(\epsilon_2-\epsilon_1)}{2\epsilon_2+\epsilon_1}\frac{1}{a^3}\right)\mathbf{d}\cdot\mathbf{d}. \tag{2.18}$$

Due to A Point Quadrupole: Finally, for a point quadrupole, we derive:

$$\phi_{\text{out}} = \frac{5\epsilon_2}{3\epsilon_2+2\epsilon_1}\frac{3}{r^5}\Theta:\mathbf{rr}, \tag{2.19a}$$

$$\phi_{\text{in}} = \left[\frac{1}{r^5} - \frac{3\left(\epsilon_2-\epsilon_1\right)}{3\epsilon_2+2\epsilon_1}\frac{1}{a^5}\right]3\Theta:\mathbf{rr}, \tag{2.19b}$$

where Θ the quadrupole moment and $\mathbf{rr}$ are 3-by-3 tensors. Their product $\Theta:\mathbf{rr}$ is a scaler, performing the same way as a dot product. As the traceless quadrupole is normally used, the potential derived needs to be shifted by a coefficient $\frac{1}{3}$:

$$\phi_{\text{out}} = \frac{5\epsilon_2}{3\epsilon_2+2\epsilon_1}\frac{1}{r^5}\Theta:\mathbf{rr}, \tag{2.20a}$$

$$\phi_{\text{in}} = \left[\frac{1}{r^5} - \frac{3\left(\epsilon_2-\epsilon_1\right)}{3\epsilon_2+2\epsilon_1}\frac{1}{a^5}\right]\Theta:\mathbf{rr}. \tag{2.20b}$$

The potential energy, with subscript q denoted for quadrupole, also shifted by a coefficient $\frac{1}{3}$, is represented as:

$$U_q = -\frac{1}{6}\left(\frac{3\left(\epsilon_2-\epsilon_1\right)}{3\epsilon_2+2\epsilon_1}\frac{1}{a^5}\right)\Theta\Theta. \tag{2.21}$$

2.3. Matched Interface and Boundary Method

In this section, we use the nonlinear PB equation in Eq. (2.22) to elucidate the fundamental principles of the MIB method from Geng's work [26] for tackling the elliptic interface problem characterized by discontinuous coefficients:

$$-\nabla\cdot(\epsilon(\mathbf{r})\nabla\phi(\mathbf{r})) + \bar{\kappa}^2(\mathbf{r})\sinh(\phi(\mathbf{r})) = \rho(\mathbf{r}). \tag{2.22}$$

Note that the solvent domain Ω^+ and the solute domain Ω^- are used interchangeably with Ω_2 and Ω_1, similar for ϵ^+ and ϵ^- with ϵ_2 and ϵ_1 in the following context. The jump conditions across the interface are assumed to be:

$$[\phi]_\Gamma = g_0(\mathbf{r}), \tag{2.23a}$$

$$[\epsilon\phi_{\mathbf{n}}]_\Gamma = g_1(\mathbf{r}), \tag{2.23b}$$

where $\mathbf{n} = (n_x, n_y, n_z)$ is the outer normal direction of the interface $\Gamma, \phi_{\mathbf{n}} = \frac{\partial\phi}{\partial\mathbf{n}}$ is the directional derivative in $\mathbf{n}$, and the notation $[f]_\Gamma = f^+ - f^-$ is the difference of the function f in solvent and in solute respectively. The nonhomogeneous jump data $g_0(\mathbf{r})$ and $g_1(\mathbf{r})$ are either given or computable, and ϵ is the piecewise dielectric constants in solute and solvent. Suitable boundary conditions for the far-field are also defined, and a regular outer boundary, denoted as Ω^+, is selected.

Considering a uniform Cartesian grid partition of the computational domain, it is widely known that the standard finite difference methods encounter challenges in maintaining their intended convergence near the interface. To address this issue, the interface jump conditions are employed to restore the accuracy. For this purpose, all grid points within the domain are categorized into "regular" grid points and "irregular" grid points. The regular points, situated away from the interface, undergo a central difference discretization of Eq. (2.22), involving a grid node (x_i, y_j, z_k) and its six neighboring points. The irregular grid points are defined as nodes where the standard finite difference schemes involve grid points across the interface, indicating that at least one of its six neighboring points belongs to the other side of the interface. At these irregular points, the finite difference approximations are adjusted by incorporating fictitious values from the opposite side of the interface. For example, consider a two-dimensional (2D) cross section $z = z_k$ shown in Fig. 2.4(c), and denote

$\phi_{i,j,k} = \phi(x_i, y_j, z_k)$. The modified finite difference approximation for the x derivative is

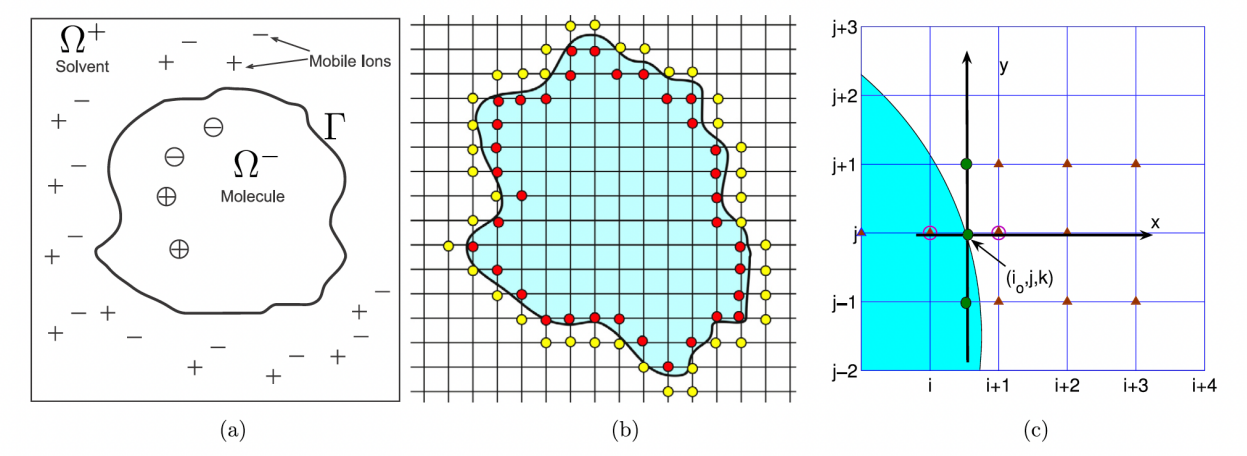
$$\frac{\partial^2}{\partial x^2}\phi_{i+1,j,k} \approx \frac{1}{\Delta x^2}\left(f_{i,j,k} - 2\phi_{i+1,j,k} + \phi_{i+2,j,k}\right), \tag{2.24}$$

where $f_{i,j,k}$ is a fictitious value defined at (x_i, y_j, z_k). With accurately estimated fictitious values, the modified finite difference approximations at irregular points maintain the second order of accuracy.

When considering the derivatives with respect to x, y, and z at all irregular points, it suffices to accurately compute two layers of fictitious values surrounding the interface: one inside and one outside. Figure 2.4(b) illustrates an example in 2D for the MIB scheme, with inside fictitious points depicted in red and outside fictitious points in yellow. From a physical perspective, these fictitious values can be interpreted as smooth extensions of function values from the opposite side of the interface. Numerically, they are determined by discretizing the jump conditions Eq. (2.23a) and (2.23b). The fictitious value $f_{i,j,k}$ at (x_i, y_j, z_k) is expressed as a linear combination of function values on a set of neighboring nodes $\mathbb{S}_{i,j,k}$ and nonhomogeneous jump data (g_0, g_1):

$$f_{i,j,k} = \sum_{(x_I, y_J, z_K) \in \mathbb{S}_{i,j,k}} w_{I,J,K}\phi_{I,J,K} + w_0 g_0 + w_1 g_1. \tag{2.25}$$

The primary objective of a specific MIB approximation is to identify the point set $\mathbb{S}_{i,j,k}$ and the representation weights $w_{I,J,K}$, w_0, and w_1 by discretizing Eq. (2.23a) and (2.23b). Subsequently, Eq. (2.25) is substituted into Eq. (2.24) to adjust the x-derivative approximation. After correcting all necessary x, y, and z derivatives in the discretization of Eq. (2.22), a discretized linear algebraic system of equations is formed, where the nonhomogeneous data (g_0, g_1) only affects the right-hand side of the equation.



(a) (b) (c)

Figure 2.4: (a) The PB model showing the solute environment Ω^- and the solvent environment Ω^+, (b) The MIB scheme illustrated in a 2-D setting with inside fictitious points in red and outside fictitious points in yellow, (c) The MIB scheme in finding the fictitious values at (i, j, k) and $(i + 1, j, k)$ in the cross section of $z = z_k$.

A fundamental concept in the MIB fictitious value determination is to break down the three-dimensional (3D) jump conditions Eq. (2.23a) and (2.23b) so they can be imposed in a one-dimensional (1D) manner. Below, we elucidate the MIB interface treatment by examining two irregular points positioned at (i, j, k) and $(i + 1, j, k)$ in the cross-section of $z = z_k$, as depicted in Fig. 2.4(c).

Assuming the interface Γ intersects the grid line in the x-direction at a point (i_0, j, k), situated between (i, j, k) and $(i + 1, j, k)$, determining the fictitious values $f_{i,j,k}$ and $f_{i+1,j,k}$ necessitates treating a derivative in the normal direction of the interface at point (i_0, j, k) in Eq. (2.23b). Introducing local coordinates (ξ, η, ζ) such that ξ aligns with the normal direction ($\xi = \mathbf{n}$), and η and ζ lie in the tangential plane, the coordinate transformation is given by:

$$\begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = \mathbf{P} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{2.26}$$

where $\mathbf{P}$ is the transformation matrix defined as:

$$\mathbf{P} = \begin{bmatrix} \sin\psi\cos\theta & \sin\psi\sin\theta & \cos\psi \\ -\sin\theta & \cos\theta & 0 \\ -\cos\psi\cos\theta & -\cos\psi\sin\theta & \sin\psi \end{bmatrix}, \tag{2.27}$$

with θ and ψ denoting the azimuth and zenith angles with respect to the normal direction ξ, respectively. By differentiating Eq. (2.23a) along the two tangential directions η and ζ, two additional jump conditions can be derived as:

$$[\phi_\eta]_\Gamma = (\phi_x^+ p_{21} + \phi_y^+ p_{22} + \phi_z^+ p_{23}) - (\phi_x^- p_{21} + \phi_y^- p_{22} + \phi_z^- p_{23}) = \frac{\partial g_0}{\partial \eta}, \tag{2.28}$$

$$[\phi_\zeta]_\Gamma = (\phi_x^+ p_{31} + \phi_y^+ p_{32} + \phi_z^+ p_{33}) - (\phi_x^- p_{31} + \phi_y^- p_{32} + \phi_z^- p_{33}) = \frac{\partial g_0}{\partial \zeta}, \tag{2.29}$$

where p_{ij} denotes the i-th row and j-th column component of the transformation matrix $\mathbf{P}$. The right-hand side terms $\frac{\partial g_0}{\partial \eta}$ and $\frac{\partial g_0}{\partial \zeta}$ can be computed when the analytical form of g_0 is provided. Since Eq. (2.26) implies

$$\begin{bmatrix} \phi_\xi \\ \phi_\eta \\ \phi_\zeta \end{bmatrix} = \mathbf{p} \begin{bmatrix} \phi_x \\ \phi_y \\ \phi_z \end{bmatrix}. \tag{2.30}$$

Eq. (2.23b), (2.28), and (2.29) can be rewritten as follows:

$$\mathbf{C}\begin{bmatrix}\phi_x^+\\ \phi_x^-\\ \phi_y^+\\ \phi_y^-\\ \phi_z^+\\ \phi_z^-\end{bmatrix} = \begin{bmatrix} g_1 \\ \frac{\partial g_0}{\partial \eta} \\ \frac{\partial g_0}{\partial \zeta}\end{bmatrix}, \tag{2.31}$$

where

$$\mathbf{C} = \begin{bmatrix}\mathbf{C}_1\\ \mathbf{C}_2\\ \mathbf{C}_3\end{bmatrix} = \begin{bmatrix} p_{11}\epsilon^+ & -p_{11}\epsilon^- & p_{12}\epsilon^+ & -p_{12}\epsilon^- & p_{13}\epsilon^+ & -p_{13}\epsilon^- \\ p_{21} & -p_{21} & p_{22} & -p_{22} & p_{23} & -p_{23} \\ p_{31} & -p_{31} & p_{32} & -p_{32} & p_{33} & -p_{33}\end{bmatrix}, \tag{2.32}$$

with $\mathbf{C}_i$ representing the i-th row of matrix $\mathbf{C}$. For the three jump conditions given in Eq. (2.31), the matrix C and the right-hand side terms are known. However, none of these conditions is easy to implement due to the coupling of six derivatives $(\phi_x^+, \phi_x^-, \phi_y^+, \phi_y^-, \phi_z^+, \phi_z^-)$. Especially for complex solvent-molecule interfaces of macromolecules, numerically evaluating some of these derivatives can be very challenging. Therefore, in the MIB method, we circumvent calculating two of the most difficult derivatives by excluding them from Eq. (2.31). Symbolically, we illustrate this idea by removing the l-th and m-th elements of the array

$(\phi_x^+, \phi_x^-, \phi_y^+, \phi_y^-, \phi_z^+, \phi_z^-)$. Consequently, Eq. (2.31) is reduced to a single jump condition:

$$(a\mathbf{C}_1 + b\mathbf{C}_2 + c\mathbf{C}_3) \begin{bmatrix} \phi_x^+ \\ \phi_x^- \\ \phi_y^+ \\ \phi_y^- \\ \phi_z^+ \\ \phi_z^- \end{bmatrix} = ag_1 + b\frac{\partial g_0}{\partial \eta} + c\frac{\partial g_0}{\partial \zeta}, \tag{2.33}$$

where

$$\begin{aligned} a &= C_{2l}C_{3m} - C_{3l}C_{2m}, \\ b &= C_{3l}C_{1m} - C_{1l}C_{3m}, \\ c &= C_{1l}C_{2m} - C_{2l}C_{1m}. \end{aligned} \tag{2.34}$$

Note that the jump condition in Eq. (2.33) essentially involves four derivatives out of $(\phi_x^+, \phi_x^-, \phi_y^+, \phi_y^-, \phi_z^+, \phi_z^-)$, even though all six derivatives are shown in the equation.

In this case, it is adequate to discretize two jump conditions Eq. (2.23a) and (2.33) along the x-direction to determine the fictitious values $f_{i,j,k}$ and $f_{i+1,j,k}$. Specifically, ϕ^+ and ϕ_x^+ at (i_0, j, k) can be approximated by the fictitious value $f_{i,j,k}$ and ϕ values at $(i+1, j, k)$ and $(i+2, j, k)$, and ϕ^- and ϕ_x^- are treated similarly. After solving for $f_{i,j,k}$ and $f_{i+1,j,k}$ based on the discretized 1D jump conditions, we can express these fictitious values in a linear combination form as shown in Eq. (2.25), with the point set and representation weights now known. For instance, in the scenario depicted in Fig. 2.4(c), the point set $\mathbb{S}_{i,j,k}$ includes all brown triangles in the plane $z = z_k$, as well as another six points in the planes $z = z_{k-1}$ and $z = z_{k+1}$. The MIB interface treatment is systematically executed to determine all fictitious

values along any mesh lines and at any interface intersection points. Further details of this procedure for handling geometric complexity and singularity can be found in [64].

Finally, it is important to note that in the discretization of jump conditions Eq. (2.23a) and (2.33), determination of fictitious values Eq. (2.25), and finite difference derivative correction Eq. (2.24), it does not matter whether the nonhomogeneous data $g_0(r)$ and $g_1(r)$ are vanishing or not. Allowing $g_0(r)$ to be nonzero provides additional flexibility in devising a new regularization approach for solving the PB equation.

2.4. A Two-Component Regularization of MIB

In this section, we keep presenting the concept from the work by Geng [26].

While implementing second and higher-order numerical representations of delta functions on Cartesian grid points is feasible through interpolation schemes, the overlap between grid points carrying redistributed partial charges and those involved in treating geometric interface singularities can result in a reduction in accuracy. Motivated by the research conducted by Cai et al. [25], we adopt a methodology wherein the electrostatic potential is divided into the "singular" Coulomb component ϕ_{C} and the "regular" reaction field component ϕ_{RF}, such that $\phi = \phi_{\mathrm{C}} + \phi_{\mathrm{RF}}$. Consequently, the nonlinear PB equation in Eq. (2.22) can be rewritten as:

$$-\nabla \cdot (\epsilon \nabla \phi_C(\mathbf{r})) - \nabla \cdot (\epsilon \nabla \phi_{RF}(\mathbf{r})) + \bar{\kappa}^2 \sinh\left(\phi_C(\mathbf{r}) + \phi_{RF}(\mathbf{r})\right) = \rho(\mathbf{r}) \quad \text{in } \Omega. \tag{2.35}$$

The Coulomb potential satisfies the free space Poisson's equation with the singular charges:

$$\begin{cases} -\epsilon^{-} \Delta \phi_C(\mathbf{r}) = \rho(\mathbf{r}) & \text{in } \mathbb{R}^3; \\ \phi_C(\mathbf{r}) = 0 & \text{as } |\mathbf{r}| \to \infty. \end{cases} \tag{2.36}$$

Thus, the analytical solution of ϕ_{C} is the Green's function $G(\mathbf{r})$:

$$\phi_C(\mathbf{r}) = G(\mathbf{r}) := C \sum_{i=1}^{N_c} \frac{q_i}{\epsilon^- |\mathbf{r} - \mathbf{r}_i|}. \tag{2.37}$$

It should be noted that $\nabla \cdot (\epsilon \nabla \phi_C) = 0$ everywhere in solute or solvent domain except at locations with partial charges, given that ϵ is a piecewise constant. By subtracting Eq. (2.36) from Eq. (2.35), we obtain a regularized PB equation for ϕ_{RF}:

$$-\nabla \cdot (\epsilon \nabla \phi_{RF}(\mathbf{r})) + \bar{\kappa}^2 \sinh\left(\phi_C(\mathbf{r}) + \phi_{RF}(\mathbf{r})\right) = 0 \quad \text{in } \Omega^- \cup \Omega^+. \tag{2.38}$$

Thus, ϕ_{RF} satisfies the following:

$$-\nabla \cdot \left(\epsilon^- \nabla \phi_{\mathrm{RF}}\right) = 0 \quad \text{in } \Omega^- \tag{2.39a}$$

$$-\nabla \cdot \left(\epsilon^+ \nabla \phi_{\mathrm{RF}}\right) + \bar{\kappa}^2 \sinh\left(\phi_C + \phi_{\mathrm{RF}}\right) = 0 \quad \text{in } \Omega^+ \tag{2.39b}$$

$$[\phi_{\mathrm{RF}}] = 0 \quad \text{on } \Gamma \tag{2.39c}$$

$$\left[\epsilon \frac{\partial \phi_{\mathrm{RF}}}{\partial n}\right] = \left(\epsilon^- - \epsilon^+\right) \frac{\partial G}{\partial n} \text{ on } \Gamma \tag{2.39d}$$

$$\phi_{\mathrm{RF}} = \phi_b - G \quad \text{on } \partial\Omega \tag{2.39e}$$

However, this two-component regularization faces several numerical challenges. It has been observed in [65] that the magnitudes of ϕ_C and ϕ_{RF} are considerably larger than that of ϕ, and ϕ_C and ϕ_{RF} have opposite signs. Since ϕ_C is analytically determined, even a minor error in ϕ_{RF} can lead to a significant error in ϕ. The amplification factor can be as high as $(\epsilon^+/\epsilon^- - 1)$ [65], which is approximately 79 for our current parameter configuration with $\epsilon^+ = 80$ and $\epsilon^- = 1$. Additionally, this method necessitates the evaluation of ϕ_C or G at all N grid points in Ω^+, incurring a computational cost of $O(N^2)$, which is prohibitively expensive for large N. Although techniques like multipole expansion such as treecode or fast

multipole method can reduce this cost to $O(N \log N)$ or $O(N)$, they come with increased algorithmic complexity.

To address these numerical challenges, Luo et al. [25] proposed solving the original potential ϕ instead of ϕ_{RF} within the domain Ω_2. Without resorting to jump conditions, the finite difference scheme developed in [25] manages the dielectric interface using an integral form of the PDE and discrete Green's functions, proving to be straightforward and effective.

In order to extend the regularization approach of [25] to other finite difference or finite element methods, it is imperative to rigorously formulate the interface jump conditions. Hence, we introduce a new elliptic interface problem with discontinuous function and flux jumps for the two-component regularization. Specifically, we define a regularized potential as:

$$\widetilde{\phi} = \begin{cases} \phi_{RF} & \text{in } \Omega_1 \\ \phi & \text{in } \Omega_2. \end{cases} \tag{2.40}$$

The jump conditions for $\widetilde{\phi}$ can be derived from those for ϕ using the definition $\phi = \phi_C + \phi_{RF}$.

$$\phi^+ = \phi^-_{RF} + \phi^-_C, \quad \epsilon^+ \frac{\partial \phi^+}{\partial n} = \epsilon^- \frac{\partial \phi^-_{RF}}{\partial n} + \epsilon^- \frac{\partial \phi^-_C}{\partial n}, \quad \text{on } \Gamma. \tag{2.41}$$

Thus, the regularized Poisson-Boltzmann (PB) equation for $\widetilde{\phi}$, along with its corresponding interface and boundary conditions, is given as:

$$-\nabla \cdot (\epsilon^- \nabla \widetilde{\phi}) = 0 \qquad \text{in } \Omega^- \tag{2.42a}$$

$$-\nabla \cdot (\epsilon^+ \nabla \widetilde{\phi}) + \bar{\kappa}^2 \sinh(\widetilde{\phi}) = 0 \qquad \text{in } \Omega^+ \tag{2.42b}$$

$$\left[\widetilde{\phi}\right] = G \qquad \text{on } \Gamma \tag{2.42c}$$

$$\left[\epsilon \frac{\partial \widetilde{\phi}}{\partial n}\right] = \epsilon^- \frac{\partial G}{\partial n} \qquad \text{on } \Gamma \tag{2.42d}$$

$$\widetilde{\phi} = \phi_b \qquad \text{on } \partial\Omega \tag{2.42e}$$

It is noteworthy that $\widetilde{\phi}$ actually satisfies the same PB equation without the source term:

$$-\nabla \cdot (\epsilon \nabla \widetilde{\phi}(\mathbf{r})) + \bar{\kappa}^2 \sinh(\widetilde{\phi}(\mathbf{r})) = 0 \quad \text{in } \Omega^- \cup \Omega^+. \tag{2.43}$$

Numerically, only one PB interface problem given in Eq. (2.42a)-(2.42e) needs to be solved. Subsequently, the original potential is recovered as $\phi = \widetilde{\phi}$ in Ω^+ and $\phi = \widetilde{\phi} + G$ in Ω^-, where the Green's function G is analytically provided.

2.5. Preliminary Results

Our numerical results are computed using a 13inch MacBook Pro with intel core-i5 processor and 16 GB of RAM. The dielectric constants of the solvent domain and molecular domain for the Kirkwood's results are set as 80 and 1 respectively. For the test proteins in Table 2.2, the solvent dielectric constant is set as 78.3, and κ is set as 0.125 Å^{-1}. Note that we only use permanent dipole moments discussed in Section 2.2.3.

As shown in Table 2.1, dcel is the grid size for finite difference method, e_{int} is the interface error, Column 3 is the corresponding order of convergence. E_{sol} is the solvation energy (kcal/mol), $e_{E_{\text{sol}}}$ is the corresponding error with Column 6 as the order of convergence. These tables show a 2^{nd} order of convergence, validating the fitness of PM source with our MIB-PB solver. Comparing the values of E_{sol} for monopole and multipole in Column 5, it shows that using the multipole moments significantly enhances the modeling accuracy.

As outlined in Section A.11, the boundary conditions of our PM-PB model rely on selecting the radius a for the approximated "sphere". **Further** investigation is needed to fully understand this concept. In Table 2.2, we use $a = 60\text{Å}$ to compute the results for several test proteins. We compute the solvation energy value at $1/\infty$ by linearly extrapolating it as the grid size approaches infinity. We consider this value as our exact value and compare the

Table 2.1: Kirkwood's spherical cavity results with radius 2Å.

Centered monopole in a spherical cavity					
dcel	e_{int}	order	E_{sol}	$e_{E_{\text{sol}}}$	order
1	4.46E-04		-81.9611	1.90E-02	
0.5	7.31E-05	2.61	-81.9718	8.35E-03	1.19
0.25	2.07E-05	1.82	-81.9784	1.73E-03	2.27
0.125	5.67E-06	1.87	-81.9798	3.38E-04	2.36
0.0625	1.40E-06	2.02	-81.9801	6.30E-05	2.42
Centered dipole in a spherical cavity					
1	2.86E-04		-2.3939	2.27E-03	
0.5	5.85E-05	2.29	-2.3949	1.27E-03	0.84
0.25	1.82E-05	1.68	-2.3959	2.81E-04	2.18
0.125	4.91E-06	1.89	-2.3962	5.41E-05	2.38
0.0625	1.22E-06	2.00	-2.3962	9.95E-06	2.44
Centered quadrupole in a spherical cavity					
1	3.78E-04		-1.7883	4.17E-03	
0.5	1.05E-04	1.85	-1.7901	2.38E-03	0.81
0.25	3.67E-05	1.52	-1.7919	5.37E-04	2.15
0.125	1.07E-05	1.77	-1.7923	1.01E-04	2.41
0.0625	2.65E-06	2.02	-1.7924	1.82E-05	2.47
Centered multipole in a spherical cavity					
1	9.79E-04		-86.1428	2.60E-02	
0.5	1.88E-04	2.38	-86.1565	1.24E-02	1.07
0.25	5.77E-05	1.70	-86.1662	2.62E-03	2.24
0.125	1.56E-05	1.88	-86.1683	5.04E-04	2.38
0.0625	3.67E-06	2.09	-86.1687	9.33E-05	2.43

results with it using the formula:

$$\text{Error} = \frac{|E_{\text{sol}} - E_{\text{sol}}^{ex}|}{|E_{\text{sol}}^{ex}|} \times 100\%. \tag{2.44}$$

The results for these proteins show a convergent pattern validating the performance of our MIB-PB solver coupled with PM source term.

Table 2.2: Results on test proteins with atom number ranging from $504 - 1046$; showing electrostatic solvation energy E_{sol} (kcal/mol) where the value for $\frac{1}{\infty}$ is linearly extrapolated. The error is computed based on this extrapolated value.

	1crn		1enh		1fsv		1pgb		1vii	
h	E_{sol}	Error	E_{sol}	Error	E_{sol}	Error	E_{sol}	Error	E_{sol}	Error
1.00	-230.51	0.68%	-1476.61	1.66%	-777.67	1.64%	-797.66	1.00%	-681.62	0.49%
0.50	-227.98	0.43%	-1450.01	0.17%	-763.14	0.25%	-785.22	0.57%	-676.64	0.24%
0.25	-228.47	0.21%	-1451.28	0.08%	-764.12	0.13%	-787.49	0.29%	-677.47	0.12%
$\frac{1}{\infty}$	-228.96		-1452.50		-765.09		-789.76		-678.29	

2.6. Proposed Polarizable Multipole Nonlinear Poisson-Boltzmann Model

In this section, we propose a novel PM-NPB model for the **first time** in the literature. A PM-NPB model offers a more accurate representation of highly charged biomolecules [59], where electrostatic interactions are intensified, thus amplifying the nonlinear effects on the Boltzmann term. Consequently, the assumption of the LPB model, which assumes weak electrostatic potentials, becomes untenable. Therefore, the nonlinear Poisson-Boltzmann (NPB) model emerges as a superior choice over the LPB model as an implicit solvent theory for interfacing with PM sources. However, implementing the PM-NPB model for real protein systems poses computational challenges due to its highly nonlinear, recursive, and strongly singular nature in sources. To address these challenges, we propose a theoretical framework: utilizing a Green's function-based decomposition method to analytically eliminate the charge singularities arising from the Dirac delta function and its derivatives, plus an iterative algo-

rithm to linearize the self-consistency recursion, thereby reducing the computational burden by solving a linearized Poisson-Boltzmann (LPB) equation in each iteration.

2.6.1. A New Regularization Formulation

A two-component decomposition $\phi = \phi_{\rm RF} + \phi_C$ [26, 27] is proposed here to regularize the singular source in Eq. (2.22) for solving the PM-NPB model. Here, ϕ_C is the Coulomb potential written in terms of Green's function as:

$$\phi_C(\mathbf{r}) = G(\mathbf{r}) := C\sum_{n=1}^{N_c}\frac{1}{\epsilon_1}\left[\frac{1}{|\mathbf{r}-\mathbf{r}_n|}q^n + \frac{r_\alpha - r_{n,\alpha}}{|\mathbf{r}-\mathbf{r}_n|^3}p_\alpha^n + \frac{(r_\alpha - r_{n,\alpha})(r_\beta - r_{n,\beta})}{2|\mathbf{r}-\mathbf{r}_n|^5}\Theta_{\alpha\beta}^n\right]. \quad (2.45)$$

It solves the Gauss's law in the free space:

$$-\epsilon_1\Delta\phi_C(\mathbf{r}) = 4\pi C\sum_{n=1}^{N_c} q^n\delta(\mathbf{r}-\mathbf{r}_n) + p_\alpha^n\partial_\alpha\delta(\mathbf{r}-\mathbf{r}_n) + \Theta_{\alpha\beta}^n\partial_{\alpha\beta}\delta(\mathbf{r}-\mathbf{r}_n). \quad (2.46)$$

By capturing the singularities via ϕ_C, the reaction field potential $\phi_{\rm RF}$ satisfies:

$$-\epsilon_1\Delta\phi_{\rm RF} = 0 \quad \text{in } \Omega_1 \quad (2.47)$$

$$-\epsilon_2\Delta\phi_{\rm RF} + \bar{\kappa}^2\sinh(\phi_{\rm RF}+G) = \epsilon_2\Delta G \quad \text{in } \Omega_2 \quad (2.48)$$

$$[\phi_{\rm RF}] = 0 \quad \left[\epsilon\frac{\partial\phi_{\rm RF}}{\partial n}\right] = (\epsilon_1-\epsilon_2)\frac{\partial G}{\partial n} \quad \text{on } \Gamma \quad (2.49)$$

$$\phi_{\rm RF} = \phi_b - G \quad \text{on } \partial\Omega \quad (2.50)$$

where the derivative of G is known analytically. Note that $\epsilon_2\Delta G = 0$ in Ω_2 is still kept in the formulation because the negligence of this term will significantly reduce the accuracy in numerical discretization near the dielectric interface [27].

2.6.2. Linearized Iterative Algorithm for Self-Consistent Mutual Polarization

Using the superscript (k) to denote the kth iteration, the Eq. (2.47) and (2.48) become:

$$-\Delta\phi_{\mathrm{RF}}^{(k+1)} = 0 \quad \text{in } \Omega_1, \tag{2.51}$$

$$-\Delta\phi_{\mathrm{RF}}^{(k+1)} + \frac{\bar{\kappa}^2}{\epsilon_2}\sinh(\phi_{\mathrm{RF}}^{(k+1)} + G^{(k+1)}) = \Delta G^{(k+1)} \quad \text{in } \Omega_2, \tag{2.52}$$

where $G^{(k+1)}$ is treated as known and $\phi_{\mathrm{RF}}^{(k+1)}$ is to be solved. Note $\bar{\kappa}$ is a constant in $\Omega_1 \cup \Omega_2$. A naive linearization of (2.52) is calculating the nonlinear term based on $\phi_{\mathrm{RF}}^{(k)}$:

$$-\Delta\phi_{\mathrm{RF}}^{(k+1)} = \Delta G^{(k+1)} - \frac{\bar{\kappa}^2}{\epsilon_2}\sinh(\phi_{\mathrm{RF}}^{(k)} + G^{(k+1)}) \text{ in } \Omega_2, \tag{2.53}$$

so that only a LPB equation needs to be solved in each iteration. However, the excess source $\sinh(\phi_{\mathrm{RF}}^{(k)} + G^{(k+1)})$ could be too strong to sufficiently converge [66, 67]. We propose to limit the excess source by adding a LPB term in (2.53):

$$-\Delta\phi_{\mathrm{RF}}^{(k+1)} + \frac{\bar{\kappa}^2}{\epsilon_2}\phi_{\mathrm{RF}}^{(k+1)} = \Delta G^{(k+1)} + \frac{\bar{\kappa}^2}{\epsilon_2}\phi_{\mathrm{RF}}^{(k)} - \frac{\bar{\kappa}^2}{\epsilon_2}\sinh(\phi_{\mathrm{RF}}^{(k)} + G^{(k+1)}) \text{ in } \Omega_2. \tag{2.54}$$

The new excess source in (2.54) becomes weaker, as the hyperbolic sine term cancels its first order Taylor expansion. Then, applying the same addition to (2.51):

$$-\Delta\phi_{\mathrm{RF}}^{(k+1)} + \frac{\bar{\kappa}^2}{\epsilon_2}\phi_{\mathrm{RF}}^{(k+1)} = \frac{\bar{\kappa}^2}{\epsilon_2}\phi_{\mathrm{RF}}^{(k)} \quad \text{in } \Omega_1, \tag{2.55}$$

and combining Eq. (2.54) and (2.55) into a new LPB equation by using the piecewise definition of κ such that $\kappa = 0$ in Ω_1 and $\kappa = \bar{\kappa}$ in Ω_2 will generate:

$$-\Delta\phi_{\mathrm{RF}}^{(k+1)} + \frac{\bar{\kappa}^2}{\epsilon_2}\phi_{\mathrm{RF}}^{(k+1)} = \frac{\kappa^2}{\bar{\kappa}^2}\Delta G^{(k+1)} + \frac{\bar{\kappa}^2}{\epsilon_2}\phi_{\mathrm{RF}}^{(k)} - \frac{\kappa^2}{\epsilon_2}\sinh(\phi_{\mathrm{RF}}^{(k)} + G^{(k+1)}) \quad \text{in } \Omega_1 \cup \Omega_2, \tag{2.56}$$

subject to the jump and boundary conditions (2.49) and (2.50). The proposed linearized iterative algorithm will therefore consist five steps:

1. Step 1 (initialization): $G^{(0)} = G^{\mathbf{M}}$ and $\boldsymbol{\mu}^{(0)} = 0$. Solve the LPB equation with the permanent multipoles for the initial reaction field $\phi_{\mathrm{RF}}^{(0)}$.

2. Step 2 (mutual polarization): Update $\boldsymbol{\mu}^{(k+1)}$ by $\boldsymbol{\mu}^{(k)}$ and $\phi_{\mathrm{RF}}^{(k)}$ according to Eq. (2.9).

3. Step 3 (regularization): Combine the permanent and induced dipoles $\mathbf{p}^{(k+1)} = \mathbf{d}^{(k+1)} + \boldsymbol{\mu}^{(k+1)}$, and update the Green's function $G^{(k+1)}$ according to (2.45).

4. Step 4 (LPB): Solve the LPB equation (2.56) subject to jump and boundary conditions (2.49) - (2.50) and known $G^{(k+1)}$ and $\phi_{\mathrm{RF}}^{(k)}$ for reaction field potential $\phi_{\mathrm{RF}}^{(k+1)}$.

5. Step 5 (while loop): go to Step 2, until convergence.

2.7. Conclusion

In this chapter, we introduce a Polarizable Multipole Poisson-Boltzmann (PM-PB) model that integrates the AMOEBA force field with a linear PB equation. We incorporate our MIB-PB solver with the multipolar PM source term and present initial results focusing on spherical cavities. Our numerical analysis on Kirkwood spheres confirms the suitability of the PM source for the MIB-PB solver by demonstrating second-order accuracy. Additionally, we validate the performance on real test proteins using a free parameter of radius, denoted as $a = 60$Å . While these proteins exhibit consistent and convergent behavior, **further** investigation into the radius of the approximated sphere for simulating the boundary conditions for the PM-PB model is necessary.

Furthermore, we propose a more sophisticated PM nonlinear PB (PM-NPB) model that includes the polarizable PM source term. We address associated challenges such as high

nonlinearity, recursion, and strong singularity in sources by employing a Green's function-based decomposition method to analytically eliminate charge singularities arising from the Dirac delta function and its derivatives. This is combined with an iterative algorithm to linearize the self-consistency recursion. Ultimately, once the solver is fully developed, it can serve as an efficient tool for computing electrostatic solvation energy for biomolecules with significant ionic strength.

Chapter 3

Optimized Parallelization of Boundary Integral Poisson-Boltzmann Solvers

In this chapter, we focus on the parallelization development of Boundary Integral PB (BI-PB) solvers. Our aim is to improve the performance by parallel computing the dense matrix resulting from the BI formulations and to provide guidance for selecting between the Treecode-Accelerated Boundary Integral (TABI) PB solver and the Direct-Sum Boundary Integral (DSBI) PB solver. Leveraging contemporary algorithms and computer hardware capabilities, we concentrate on parallelizing the TABI-PB solver using the Message Passing Interface (MPI) on CPUs and the DSBI-PB solver using KOKKOS on GPUs. Furthermore, we validate the effectiveness of our BI-PB solvers on selected proteins that play pivotal roles in understanding the spread, treatment, and prevention of COVID-19 virus diseases. Throughout our investigation, we explore the factors that influence the choice between employing the MPI-based TABI-PB solver or the GPU-accelerated DSBI-PB solver based on specific computational needs and constraints.

3.1. Introduction

The BI-PB solvers lend themselves well to parallel computing due to their similarity in computing pairwise interactions among charges or induced charges located at the boundary elements, which is in fact an N-body problem. The parallelization of these pairwise interactions with $O(N^2)$ computational cost is straight-forward. However, when fast algorithms are used, such as treecode [42] or fast multipole method (FMM) [68], additional efforts are required to ensure optimal parallel efficiency [10, 69].

In our investigation, we highlight the comparison of parallelization between treecode and direct-sum methods when different computing hardware is available. With MPI implementation using multicore CPUs for moderate numbers of particles, we can build the tree on every CPU/task, thus all particle-tree interactions can be concurrently done with high parallel efficiency [10, 37]. However, in cases where the number of particles is huge, making it impossible to replicate the tree on each task, a domain decomposition approach is used instead [10, 70].

In recent years, Graphic Processing Units (GPUs) have revolutionized computationally intensive tasks across various domains, including high-performance computing, call centers, autopilot systems, artificial intelligence, etc. A single GPU card can consist hundreds to thousands of cores, enabling the swift execution of numerous tasks, especially those involving Single Instruction Multiple Data (SIMD) tasks that leverage large-scale concurrency. The N-body problem, where all particles concurrently interact with all other particles, is highly conducive to GPU computation. Subsequently, with the development of fast algorithms like treecode for addressing the N-body problem, their implementation on GPUs has also become feasible [71–73]. However, these complex algorithms are challenging to implement on GPUs, thereby reducing the achievable level of parallel speedup.

Our contention is that when a faster algorithm, such as treecode, is available to replace the direct summation method, there exists a break-even number n_b where the faster algorithm should only be used when the size of the problem exceeds this threshold. For instance, considering an N-body problem, let us compare the $c_1 N \log N$ complexity of the treecode algorithm with the $c_2 N^2$ complexity of direct summation, where c_1 and c_2 are constants derived from the algorithms. The break-even number n_b satisfies the equation $c_1 N \log N = c_2 N^2$. Because of the algorithmic simplicity of direct summation, the break-even number n_b is considerably larger on GPUs compared to CPUs. Therefore, when implementation efficiency is crucial, such as in repetitive usage within molecular dynamics or Monte Carlo

simulations, direct summation should be used on GPUs when the problem size is smaller than n_b. In Section 3.4, it is demonstrated that the break-even number n_b for the current GPU/CPU hardware conditions used in this project is approximately 250,000. In practice, a molecular surface with 250,000 triangular elements can adequately represent a large group of proteins with less than 2000-3000 atoms. For larger problems, we can use the MPI-based parallel treecode [10] or use the domain decomposition approach [70].

In this study, we concentrate on two strategies for parallelizing boundary integral PB solvers. One approach involves parallelizing the Treecode-Accelerated Boundary Integral (TABI) solver using MPI, where an identical tree is constructed on each task/CPU. Its parallelization involves four stages of the TABI solver: source term computation, treecode for matrix-vector product, preconditioning, and energy computation. Among these stages, we adapt schemes developed for N-body parallelization [10] to the more complex BI-PB problem. Additionally, we develop MPI-based parallelization for the preconditioning scheme designed for BI solvers [44]. Our second parallelization approach centers on GPU-based parallelization of the Direct-Sum Boundary Integral (DSBI) solver, which simultaneously computes the source term, matrix-vector product, and energy computation. At this stage, we have decided not to incorporate the preconditioning scheme into this approach due to its complex and inefficient GPU implementation.

3.2. Boundary Integral Poisson-Boltzmann Solvers

The key numerical algorithms utilized in BI-PB solvers are as follows:

1. A well-posed boundary integral formulation of the PB equation.
2. Triangular discretization using centroids.
3. Parallelization using MPI.
4. Implementation of the treecode algorithm.

5. Preconditioning.

6. Acceleration using GPUs.

There are inherent trade-offs between accuracy and efficiency depending on the selection and combination of these algorithms. As a result, we have developed a treecode-accelerated boundary integral PB (TABI-PB) solver employing algorithms 1 to 5 and a GPU-accelerated direct-sum boundary integral PB (DSBI-PB) solver integrating algorithms 1, 2, and 6 using Kokkos [74]. The parallelization schemes 3 and 6 will be detailed in Section 3.3.

3.2.1. Boundary Integral Formulation of Poisson-Boltzmann Equation

In this context, we provide an overview of the well-conditioned boundary integral formulation [29, 37] of the implicit Poisson-Boltzmann solvent model.

The electrostatic potential equation given in Eq. (1.7) can be discussed separately for the inside and outside domains as:

$$-\epsilon_1 \nabla^2 \phi(\mathbf{x}) = \sum_{k=1}^{N_c} q_k \delta\left(\mathbf{x} - \mathbf{y}_k\right), \quad \mathbf{x} \in \Omega_1, \tag{3.1a}$$

$$-\epsilon_2 \nabla^2 \phi(\mathbf{x}) + \bar{\kappa}^2 \phi(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega_2, \tag{3.1b}$$

with the interface conditions:

$$\phi_1(\mathbf{x}) = \phi_2(\mathbf{x}), \quad \epsilon_1 \frac{\partial \phi_1(\mathbf{x})}{\partial \nu} = \epsilon_2 \frac{\partial \phi_2(\mathbf{x})}{\partial \nu}, \quad \mathbf{x} \in \Gamma, \tag{3.2}$$

where ϵ_1 and ϵ_2 are the dielectric constants in solute domain Ω_1 and solvent domain Ω_2 respectively, $q_k = \frac{e_c Q_k}{k_B T}$ is the partial atomic charge with e_c being the electronic charge, k_B being the Boltzmann's constant, and T being the temperature, δ is the delta function, κ is the Debye–Hückel parameter measuring the ionic concentration with $\kappa^2 = \frac{\bar{\kappa}^2}{\epsilon_2}$, and

$\nu = (n_x, n_y, n_z)$ denotes the outward normal direction of the interface Γ. The far-field boundary condition is:

$$\lim_{|\mathbf{x}|\to\infty} \phi(\mathbf{x}) = 0. \tag{3.3}$$

The fundamental solutions for these domains are the Coulomb potential $G_0(\mathbf{x}, \mathbf{y})$ and the screened Coulomb potential $G_\kappa(\mathbf{x}, \mathbf{y})$, respectively:

$$G_0(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi|\mathbf{x} - \mathbf{y}|}, \quad G_\kappa(\mathbf{x}, \mathbf{y}) = \frac{e^{-\kappa|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x} - \mathbf{y}|}. \tag{3.4}$$

Applying Green's second identity and the fundamental solutions yields the following expressions:

$$\phi(\mathbf{x}) = \int_\Gamma \left[G_0(\mathbf{x}, \mathbf{y}) \frac{\partial \phi(\mathbf{y})}{\partial \nu} - \frac{\partial G_0(\mathbf{x}, \mathbf{y})}{\partial \nu_\mathbf{y}} \phi(\mathbf{y}) \right] \mathrm{d}S_\mathbf{y} + \sum_{k=1}^{N_c} q_k G_0(\mathbf{x}, \mathbf{y}_k), \quad \mathbf{x} \in \Omega_1, \tag{3.5a}$$

$$\phi(\mathbf{x}) = \int_\Gamma \left[-G_\kappa(\mathbf{x}, \mathbf{y}) \frac{\partial \phi(\mathbf{y})}{\partial \nu} + \frac{\partial G_\kappa(\mathbf{x}, \mathbf{y})}{\partial \nu_\mathbf{y}} \phi(\mathbf{y}) \right] \mathrm{d}S_\mathbf{y}, \quad \mathbf{x} \in \Omega_2, \tag{3.5b}$$

In Eq. (3.5a) and (3.5b), the normal derivative with respect to $\mathbf{y}$ is given by

$$\frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \nu_\mathbf{y}} = \nu(\mathbf{y}) \cdot \nabla_\mathbf{y} G(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{3} \nu_n(\mathbf{y}) \partial_{y_n} G(\mathbf{x}, \mathbf{y}), \tag{3.6}$$

where G represents either G_0 or G_κ. Following the steps by Juffer et al. [29], applying the interface conditions in Eq. (3.2) and differentiating the electrostatic potential in each domain yields a set of boundary integral equations relating the surface potential ϕ_1 and its normal derivative $\partial\phi_1/\partial\nu$ on Γ:

$$\frac{1}{2}(1 + \epsilon)\phi_1(\mathbf{x}) = \int_\Gamma \left[K_1(\mathbf{x}, \mathbf{y}) \frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{x}, \mathbf{y}) \phi_1(\mathbf{y}) \right] dS_\mathbf{y} + S_1(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{3.7a}$$

$$\frac{1}{2}\left(1 + \frac{1}{\epsilon}\right) \frac{\partial \phi_1(\mathbf{x})}{\partial \nu} = \int_\Gamma \left[K_3(\mathbf{x}, \mathbf{y}) \frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_4(\mathbf{x}, \mathbf{y}) \phi_1(\mathbf{y}) \right] dS_\mathbf{y} + S_2(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{3.7b}$$

where $\epsilon = \epsilon_2/\epsilon_1$. The kernels $K_{1,2,3,4}$ and the source terms $S_{1,2}$ are linear combinations of G_0, G_k, and their first- and second-order normal derivatives [29, 37] are:

$$K_1(\mathbf{x},\mathbf{y}) = G_0(\mathbf{x},\mathbf{y}) - G_\kappa(\mathbf{x},\mathbf{y}), \quad K_2(\mathbf{x},\mathbf{y}) = \epsilon\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_\mathbf{y}} - \frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \nu_\mathbf{y}}, \tag{3.8}$$

$$K_3(\mathbf{x},\mathbf{y}) = \frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \nu_\mathbf{x}} - \frac{1}{\epsilon}\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_\mathbf{x}}, \quad K_4(\mathbf{x},\mathbf{y}) = \frac{\partial^2 G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_\mathbf{x}\partial \nu_\mathbf{y}} - \frac{\partial^2 G_0(\mathbf{x},\mathbf{y})}{\partial \nu_\mathbf{x}\partial \nu_\mathbf{y}}, \tag{3.9}$$

where the normal derivative with respect to $\mathbf{x}$ is given by:

$$\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \nu_\mathbf{x}} = -\nu(\mathbf{x})\cdot\nabla_\mathbf{x} G(\mathbf{x},\mathbf{y}) = -\sum_{m=1}^{3}\nu_m(\mathbf{x})\partial_{x_m} G(\mathbf{x},\mathbf{y}), \tag{3.10}$$

and the second normal derivative with respect to $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \nu_\mathbf{y}\partial \nu_\mathbf{x}} = -\sum_{m=1}^{3}\sum_{n=1}^{3}\nu_m(\mathbf{x})\nu_n(\mathbf{y})\partial_{x_m}\partial_{y_n} G(\mathbf{x},\mathbf{y}). \tag{3.11}$$

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

$$S_1(\mathbf{x}) = \frac{1}{\epsilon_1}\sum_{k=1}^{N_c} q_k G_0\left(\mathbf{x},\mathbf{y}_k\right), \quad S_2(\mathbf{x}) = \frac{1}{\epsilon_1}\sum_{k=1}^{N_c} q_k \frac{\partial G_0\left(\mathbf{x},\mathbf{y}_k\right)}{\partial \nu_\mathbf{x}}. \tag{3.12}$$

Once the potential and its normal derivative are solved from Eq. (3.7a)-(3.7b), the potential at any point inside the molecule can be determined via Eq. (3.5a). Alternatively, a more precise numerical formulation, as detailed in [29], can be employed for better accuracy:

$$\phi_1(\mathbf{x}) = \int_\Gamma \left[K_1(\mathbf{x},\mathbf{y})\frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{x},\mathbf{y})\phi_1(\mathbf{y})\right] \mathrm{d}S_\mathbf{y} + S_1(\mathbf{x}), \quad \mathbf{x}\in\Omega_1. \tag{3.13}$$

With the potential and its normal derivative on Γ, the electrostatic free energy can be obtained by:

$$E^{\text{free}} = \frac{1}{2}\sum_{k=1}^{N_c} q_k \phi_1(\mathbf{y}_k) = \frac{1}{2}\sum_{k=1}^{N_c} q_k \left(\int_\Gamma \left[K_1(\mathbf{y}_k, \mathbf{y}) \frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{y}_k, \mathbf{y}) \phi_1(\mathbf{y}) \right] \mathrm{d}S_{\mathbf{y}} + S_1(\mathbf{y}_k) \right). \tag{3.14}$$

The electrostatic solvation free energy can also be obtained by:

$$E^{\text{sol}} = \frac{1}{2}\sum_{k=1}^{N_c} q_k \phi^{\text{reac}}(\mathbf{y}_k) = \frac{1}{2}\sum_{k=1}^{N_c} q_k \int_\Gamma \left[K_1(\mathbf{y}_k, \mathbf{y}) \frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{y}_k, \mathbf{y}) \phi_1(\mathbf{y}) \right] \mathrm{d}S_{\mathbf{y}}, \tag{3.15}$$

where $\phi^{\text{reac}}(\mathbf{x}) = \phi(\mathbf{x}) - S_1(\mathbf{x})$ is the reaction field potential [29,37]. In our numerical results in Section 3.4, we focus on solution of the PB equation and calculation of the electrostatic solvation free energy.

3.2.2. Discretization of Boundary Integral Equations

We discretize the molecular surface Γ by employing the MSMS software [46] to perform triangulation. MSMS requires atomic coordinates and radii as input and generates triangles along with normal vectors at their vertices as output. The integrals in Eq. (3.7a)-(3.7b) are discretized using centroid collocation, a popular method known for its simplicity [37].

Denoting the triangle centroids of the N triangular elements as $\mathbf{x}_i, i = 1, \ldots, N$, the discretized equations (3.7a)-(3.7b) take the following form for $i = 1, \ldots, N$:

$$\frac{1}{2}(1+\epsilon)\,\phi_1(\mathbf{x}_i) = \sum_{\substack{j=1 \\ j\neq i}}^{N} \left[K_1(\mathbf{x}_i, \mathbf{x}_j) \frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_2(\mathbf{x}_i, \mathbf{x}_j) \phi_1(\mathbf{x}_j) \right] \Delta s_j + S_1(\mathbf{x}_i), \tag{3.16a}$$

$$\frac{1}{2}\left(1 + \frac{1}{\epsilon}\right) \frac{\partial \phi_1(\mathbf{x}_i)}{\partial \nu} = \sum_{\substack{j=1 \\ j\neq i}}^{N} \left[K_3(\mathbf{x}_i, \mathbf{x}_j) \frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_4(\mathbf{x}_i, \mathbf{x}_j) \phi_1(\mathbf{x}_j) \right] \Delta s_j + S_2(\mathbf{x}_i), \tag{3.16b}$$

where Δs_j represents the area of the jth boundary element. The term $j = i$ is excluded from the summation to prevent singularity issues with the kernel.

Equations (3.16a)-(3.16b) form a linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ contains the surface potentials $\phi_1(\mathbf{x}_i)$ and their normal derivatives $\frac{\partial \phi_1(\mathbf{x}_i)}{\partial \nu}$, and $\mathbf{b}$ contains the source terms $S_1(\mathbf{x}_i)$ and $S_2(\mathbf{x}_i)$. Solving this system involves using the generalized minimal residual (GMRES) iterative method, which requires a matrix-vector product in each step [75]. However, since the matrix is dense, computing the product directly through summation requires $O(N^2)$ operations, which becomes prohibitively expensive for large N. To address this challenge, fast algorithms for N-body computations like the treecode [37, 76] has been applied on the centroid discretization algorithm. In the subsequent section, we outline how the treecode algorithm accelerates the matrix-vector product calculation.

3.2.3. Treecode

In this context, we offer a brief overview of the treecode algorithm, with more comprehensive details available in previous works [42, 77–79].

The particles are partitioned into a hierarchy of clusters, as shown in Fig. 3.1(a). This partitioning process involves subdividing each cluster into four (or eight for 3D) sub-clusters until the predefined treecode parameter N_0, which represents the maximum number of particles per leaf is reached. Here, we illustrate the 2D scenario using $N_0 = 3$; the 3D case follows a similar approach. The treecode evaluates the potential as a sum of particle-cluster interactions:

$$V_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} q_j G(\mathbf{x}_i, \mathbf{x}_j), \quad i = 1, \ldots, N, \tag{3.17}$$

where q_j denotes a charge associated with $\mathbf{x}_j$, and G denotes the interaction between each individual target particle $\mathbf{x}_i$ interacts with a cluster of $\mathbf{x}_j$ particles.

The matrix-vector product $\mathbf{Ax}$ for Eq. (3.16a)-(3.16b) takes the form of N-body potentials same as Eq. (3.17), where $\mathbf{x}_i$ and $\mathbf{x}_j$ denote centroids (also referred to as particle locations in this context). To this end, the q_j term in Eq. (3.17) corresponds to $\Delta s_j \phi_1(\mathbf{x}_j)$ or $\Delta s_j \frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu}$ in Eq.(3.16a)-(3.16b), and G represents one of the kernels K_{1-4}.

Then, V_i can be fastly computed as a summation of particle-particle interactions and particle-cluster interactions, as illustrated in Fig. 3.1(b):

$$V_i \approx \sum_{c \in N_i} \sum_{\mathbf{x}_j \in c} q_j G(\mathbf{x}_i, \mathbf{x}_j) + \sum_{c \in F_i} \sum_{\|\mathbf{k}\|=0}^{p} a^{\mathbf{k}}(\mathbf{x}_i, \mathbf{x}_c)\, m_c^{\mathbf{k}}, \tag{3.18}$$

where c denotes a cluster, and N_i, F_i denote the near-field and far-field clusters of particle $\mathbf{x}_i$. The first term on the right represents a direct summation for particles $\mathbf{x}_j$ located near $\mathbf{x}_i$, while the second term denotes a pth order Cartesian Taylor approximation around the cluster center $\mathbf{x}_c$ for clusters that are well-separated from $\mathbf{x}_i$ [42]. The Taylor coefficients are given by:

$$a^{\mathbf{k}}(\mathbf{x}_i, \mathbf{x}_c) = \frac{1}{\mathbf{k}!} \partial_{\mathbf{y}}^{\mathbf{k}} G(\mathbf{x}_i, \mathbf{x}_c), \tag{3.19}$$

and the cluster moments are given by:

$$m_c^{\mathbf{k}} = \sum_{\mathbf{x}_j \in c} q_j (\mathbf{x}_j - \mathbf{x}_c)^{\mathbf{k}}. \tag{3.20}$$

where $\mathbf{k} = (k_1, k_2, k_3), k_i \in \mathbb{N}, \|\mathbf{k}\| = k_1 + k_2 + k_3, \mathbf{k}! = k_1!\, k_2!\, k_3!$ [37]. A particle $\mathbf{x}_i$ and a cluster c are defined to be well-separated if the multipole acceptance criterion (MAC) is satisfied,

$$r_c / R \leq \theta, \tag{3.21}$$

where r_c is the cluster radius, $R = |\mathbf{x}_i - \mathbf{x}_c|$ is the particle-cluster distance and θ is a user-specified parameter [77]. If the criterion is not satisfied, the code examines the children of the

cluster recursively until the leaves of the tree are reached, at which point direct summation is used. The Taylor coefficients are computed using recurrence relations [42].

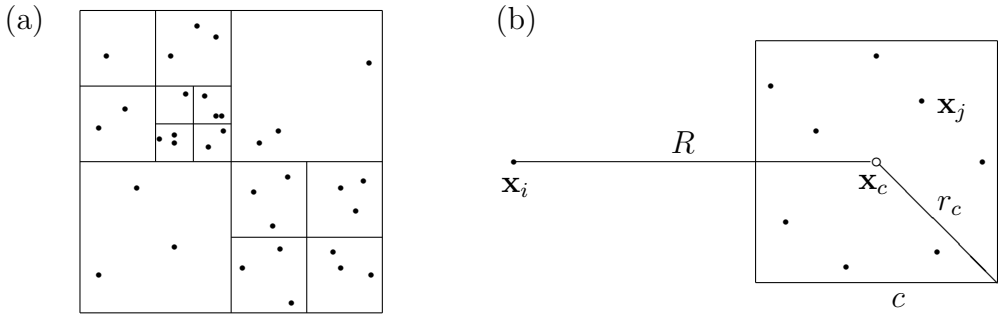


Figure 3.1: Details of treecode in 2D. (a) tree structure of particle clusters. (b) particle-cluster interaction between the particle $\mathbf{x}_i$ and the cluster $c = \{\mathbf{x}_j\}$. $\mathbf{x}_c$: cluster center; R: particle-cluster distance; and r_c: cluster radius.

The accuracy and efficiency of the treecode depend on the chosen combination of the order p, the MAC parameter θ, and the maximum particles per leaf N_0. Employing the treecode, the operation count for the matrix-vector product scales as $O(N \log N)$, where N represents the number of particles $\mathbf{x}_i$, and $\log N$ is the number of levels in the tree.

3.2.4. Preconditioning

In this section, we summarize the work by Chen [44].

In order to precondition Krylov subspace methods for solving $\mathbf{Ax} = \mathbf{b}$, we employ a left-preconditioning scheme. With a preconditioning matrix $\mathbf{M}$, we consider the modified linear system $\mathbf{M^{-1}Ax} = \mathbf{M^{-1}b}$. We aim to find a preconditioning matrix or a preconditioner $\mathbf{M}$ that satisfies two conditions:

1. $\mathbf{M}$ should be similar to $\mathbf{A}$, resulting in $\mathbf{M^{-1}A}$ having improved conditioning compared to $\mathbf{A}$, thus requiring fewer GMRES iterations.

2. The equation $\mathbf{M^{-1}z} = \mathbf{y}$ can be efficiently computed, which is equivalent to solving $\mathbf{y}$ from $\mathbf{My} = \mathbf{z}$.

It is important to note that these two conditions cannot be optimized simultaneously, and a trade-off must be made. For example, setting $\mathbf{M} = \mathbf{A}$ maximizes the first condition, while setting $\mathbf{M} = \mathbf{I}$ maximizes the second condition. The ideal preconditioner lies between these two extremes.

We base our preconditioner design on the observation that in electrostatic interactions, particularly between boundary elements in integral equation solutions, short-range interactions are fewer in number but more significant in strength compared to long-range interactions, which are numerous and computationally more expensive. Hence, long-range interactions are typically computed using multipole expansions. With this insight, we construct the preconditioner $\mathbf{M}$ to contain only short-range interactions and disregard long-range interactions. Specifically, we consider interactions between elements within the same leaf only. This choice of $\mathbf{M}$ offers significant advantages in efficiency and accuracy when solving $\mathbf{My} = \mathbf{z}$.

To be more precise, we give the explicit definition of $\mathbf{A}$ and $\mathbf{M}$ as from the discretized system (3.16a)-(3.16b). Let $\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \in \mathbb{R}^{2N\times 2N}$, where $\mathbf{A}_{mn} \in \mathbb{R}^{N\times N}$ for $m, n = 1, 2$. Then the entries of these block matrices are given as:

$$\mathbf{A}_{11}(i,j) = \frac{1}{2}(1+\epsilon)\,\delta_{ij} + K_2(\mathbf{x}_i, \mathbf{x}_j)\;\Delta s_j(1-\delta_{ij}), \tag{3.22}$$

$$\mathbf{A}_{12}(i,j) = K_1(\mathbf{x}_i, \mathbf{x}_j)\;\Delta s_j(1-\delta_{ij}), \tag{3.23}$$

$$\mathbf{A}_{21}(i,j) = K_4(\mathbf{x}_i, \mathbf{x}_j)\;\Delta s_j(1-\delta_{ij}), \tag{3.24}$$

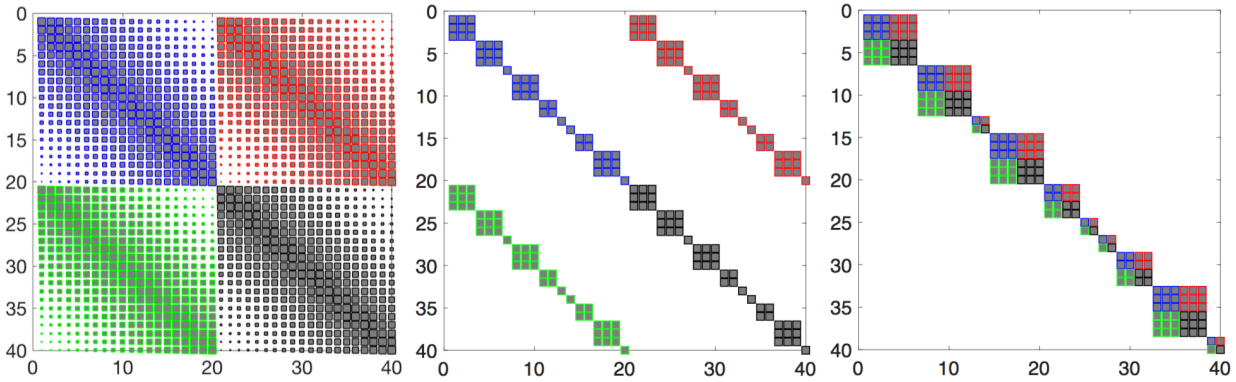
$$\mathbf{A}_{22}(i,j) = \frac{1}{2}\left(1+\frac{1}{\epsilon}\right)\delta_{ij} + K_3(\mathbf{x}_i, \mathbf{x}_j)\;\Delta s_j(1-\delta_{ij}), \tag{3.25}$$

for $i, j = 1, \cdots, N$, where $\delta_{i,j}$ is the Kronecker delta function and Δs_j is the area of the jth element. The definition of $\mathbf{M}$ will be essentially similar to $\mathbf{A}$ except that the entries of $\mathbf{M}$

are zero if i and j are not on the same leaf of the tree, i.e.

$$\mathbf{M}_{m,n}(i,j) = \begin{cases} \mathbf{A}_{m,n}(i,j) & \text{if } i,j \text{ are on the same leaf,} \\ 0 & \text{otherwise,} \end{cases} \tag{3.26}$$

for $i,j = 1,\cdots,N$ and $m,n = 1,2$.



(a) (b) (c)

Figure 3.2: A schematic illustration of the boundary element dense matrix $\mathbf{A}$ and its preconditioning matrix $\mathbf{M}$: (a) matrix $\mathbf{A}$ for the case of $N = 20$ elements (the size of the matrix entry shows the strength of the interaction; the four different color-coded region relates to K_{1-4} in Eq. (3.16a)-(3.16b)); (b) the "block diagonal block" preconditioning matrix $\mathbf{M}$ ($N_0 = 3$ in this schematic illustration and there are 10 leaves with 1-3 particles/elements each); (c) the "block diagonal" preconditioning matrix $\mathbf{M}$, which is a permuted matrix from $\mathbf{M}$ in (b) after switching the order of the unknowns.

Here, we utilize Fig. 3.2 to elucidate the design and benefits of our preconditioning approach. Figure 3.2(a) depicts the dense boundary element matrix $\mathbf{A}$ for the discretized system (3.16a)-(3.16b) with 20 boundary elements. The four distinct colors represent the entries related to the four kernels K_{1-4} of the linear algebraic matrix $\mathbf{A}$ in Eq. (3.16a)-(3.16b). It is note worthy that the unknowns are arranged according to the potentials ϕ_1 on all elements, followed by the normal derivative of the potential $\frac{\partial \phi_1}{\partial \nu}$. The size of the matrix entry in Fig. 3.2 indicates the magnitude of the interaction between a target element and a source element, which diminishes from the main diagonal to its two wings. By solely incorporating the interactions between elements in the same leaf, we derive our

tailored preconditioning matrix $\mathbf{M}$ as depicted in Fig. 3.2(b). This preconditioning matrix $\mathbf{M}$ comprises four blocks, with each block being a diagonal block matrix. Strictly speaking, $\mathbf{M}$ should be termed a "block diagonal block matrix," but for simplicity, we use the term "block diagonal matrix," as explained next.

The primary advantage of the designed preconditioning matrix $\mathbf{M}$ lies in its rapid computation of $\mathbf{y} = \mathbf{M^{-1}z}$, or equivalently, solving $\mathbf{y}$ from $\mathbf{My} = \mathbf{z}$. To elucidate this, we reorganize the unknowns in vector $\mathbf{y}$. Initially, $\mathbf{y}$ comprises the first segment of potential on all elements followed by the second segment of normal derivatives of the potential on all elements. Upon rearrangement, $\mathbf{y}$ is segmented into N_l parts, where N_l denotes the total number of leaves in the tree structure. Each segment of the rearranged $\mathbf{y}$, progressing leaf by leaf, encapsulates the potentials on elements belonging to the leaf, succeeded by the normal derivatives of the potential on elements of the same leaf. This reordering results in a block diagonal matrix $\mathbf{M}$, depicted in Fig. 3.2(c). This rearranged matrix $\mathbf{M}$ exhibits equivalence with the original matrix $\mathbf{M}$ concerning properties such as condition number, eigenvalues, and singular values. Besides, it offers significantly improved efficiency and convenience in solving $\mathbf{My} = \mathbf{z}$ and computing characteristic quantities of $\mathbf{M}$. For simplicity and coherence with the aforementioned equivalence, we do not differentiate between the original $\mathbf{M}$ and the rearranged $\mathbf{M}$ in this context.

Since $\mathbf{M} = \text{diag}\{\mathbf{M_1}, \mathbf{M_2}, \cdots, \mathbf{M_{N_l}}\}$, as shown in Fig. 3.2(c), is a block diagonal matrix, $\mathbf{My} = \mathbf{z}$ can be efficiently solved using direct methods such as LU factorization, by solving each individual $\mathbf{M_i y_i} = \mathbf{z_i}$. Here, each $\mathbf{M_i}$ is a square nonsingular matrix, representing the interaction between particles/elements on the ith leaf of the tree. It is noteworthy that the efficiency is not compromised even when $\mathbf{M_i}$ has a large condition number, since a direct solver is used for solving $\mathbf{My} = \mathbf{z}$. Additionally, we can easily compute the condition number, eigenvalues, and singular values of $\mathbf{M}$, which are similar or close to corresponding values of matrix $\mathbf{A}$ and are very useful for studying the properties of $\mathbf{A}$.

We also provide an estimate of the computational cost for the preconditioning scheme, which essentially corresponds to the cost of solving $\mathbf{My} = \mathbf{z}$. Let N_0 be the number of particles per leaf, a user-specified treecode parameter as explained in the previous section. Since we only consider interactions between particles on the same leaf for preconditioning, the dimension of the matrix $\mathbf{M_i}$ is less than or equal to N_0, and we use the upper limit N_0 to represent it. The total number of blocks is N/N_0. Hence, the total cost of solving $\mathbf{My} = \mathbf{z}$ by solving all $\mathbf{M_i y_i} = \mathbf{z_i}$ is $O(N_0^3 N/N_0) = O(NN_0^2)$, which is essentially $O(N)$ if a small N_0 is used. However, N_0 cannot be too small, or else $\mathbf{M}$ will not include the singular element interactions to cancel with the corresponding interaction in $\mathbf{A}$. These considerations suggest using a small N_0, e.g., 100, in our preconditioning scheme.

3.3. Parallelization Schemes

In this section, we delve into the parallelization procedures, encompassing a MPI-based TABI solver developed by other collaborators and a GPU-accelerated DSBI solver.

3.3.1. MPI-based Treecode Accelerated Boundary Integral Solver

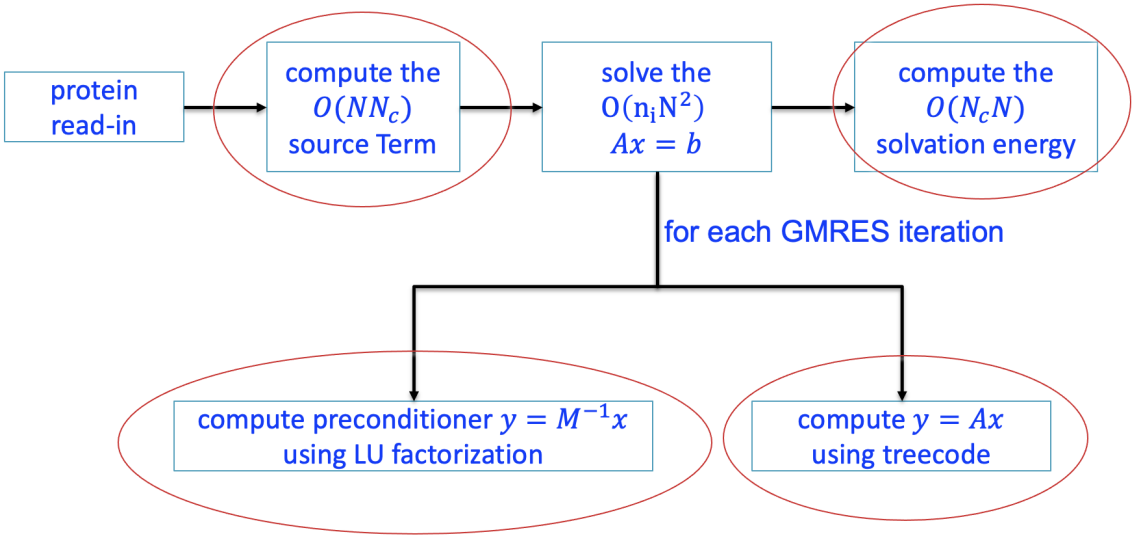


Figure 3.3: Pipeline for parallelized TABI solver.

As depicted by the red circled items in Fig. 3.3, our parallelization of the TABI solver focuses on four key stages of the pipeline: the $O(NN_c)$ computation of the source term, the $O(n_i N \log N)$ matrix-vector product using treecode, the $O(n_i N n_0^2)$ preconditioning stage, and the $O(N_c N)$ computation of the solvation energy. Here, N represents the number of surface triangles, N_c signifies the number of partial charges, n_i indicates the number of GMRES iterations, and n_0 denotes the maximum number of particles per leaf. Among these stages, the most time-consuming and challenging component is the matrix-vector product using treecode. To address this, we explore two potential strategies for solving N-body problems, as outlined in [80], which are also summarized below.

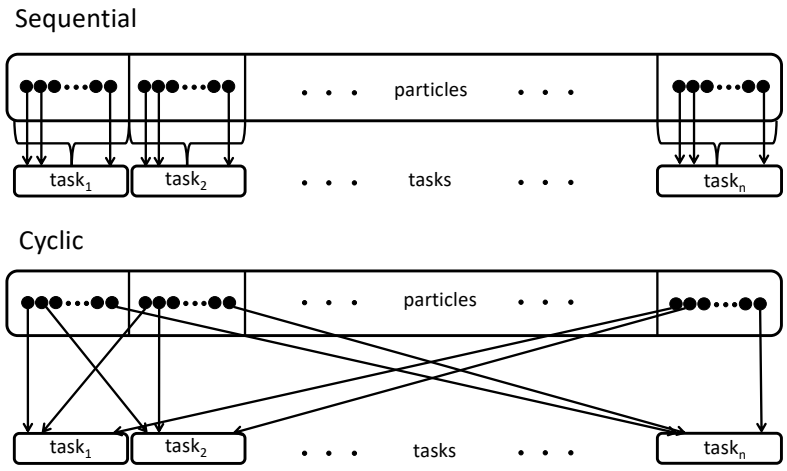


Figure 3.4: Methods for assigning target particles to tasks: sequential order (top) *vs* cyclic order (bottom).

The initial and intuitive approach to allocate target particles to tasks is through "sequential ordering". In this scheme, for a system consists of n_p processors, the 1st task manages the first N/n_p particles in a consecutive segment, the 2nd task handles the subsequent N/n_p particles, and so on. This job assignment is illustrated at the top of Fig. 3.4. However, upon examining the resulting CPU times for each task, we observed significantly different times on each task, indicating a notable load imbalance. This discrepancy can be attributed to the fact that particles at different locations may exhibit varying types of interactions with other

particles through the treecode. For instance, a particle with only a few close neighbors tends to engage in more particle-cluster interactions than particle-particle interactions, resulting in shorter CPU time requirements compared to a particle with numerous close neighbors. Additionally, we observed that particles in close proximity to each other exhibit similar interactions with other particles, whether through particle-particle or particle-cluster interactions. Consequently, certain consecutive segments ended up computing significantly more particle-particle interactions than others, which were instead dominated by particle-cluster interactions.

In light of these observations, we design a "cyclic ordering" scheme aimed at enhancing load balancing, as depicted at the bottom of Fig. 3.4. In this approach, particles located near each other are evenly allocated to different tasks. For instance, in a cluster of particles in close proximity, the first task handles the first particle, the second task handles the second particle, and so forth. This cycle continues from the $(n_p + 1)$-th particle onwards. The numerical findings presented in Section 3.4 illustrate the substantial enhancement in load balancing achieved through this simple strategy.

The pseudocode for our MPI-based parallel TABI solver using the replicated data algorithm is presented in Table 3.1. Each task builds identical trees, as depicted in line 6. The four-stage MPI-based parallelization, encompassing the source term, matrix-vector product with treecode, preconditioning, and solvation energy, is executed in lines 4, 10, 12, and 17, respectively, followed by MPI communications.

Table 3.1: Pseudocode for MPI-based parallel TABI solver using replicated data algorithm.

```
1  on main processor
2        read protein data
3        call MSMS to generate triangulation
4        copy protein data and triangulation to all other processors
5  on each processor
6        build local copy of tree
7        compute assigned segment of source terms by direct sum
8              copy result to all other processors
9        set initial guess for GMRES iteration
10       compute assigned segment of matrix-vector product Ax by treecode
11             copy result to all other processors
12       compute assigned segment of solving Mx = y for x by LU factorization
13             copy result to all other processors
14       test for GMRES convergence
15             if no, go to step 10 for next iteration
16             if yes, go to step 15
17       compute assigned segment of electrostatic solvation free energy by direct sum
18             copy result to main processor
19 on main processor
20       add segments of electrostatic solvation free energy and output result
```

3.3.2. GPU-accelerated Direct-Sum Boundary Integral Solver

For the GPU parallelization, we use Kokkos [74], which is a programming model and C++ library designed to facilitate performance portability of applications across diverse high-performance computing (HPC) architectures, including CPUs, GPUs, and other accelerators.

The pseudocode for the DSBI-PB solver utilizing GPUs is provided in Table 3.2. In this pseudocode, we categorize all operations into those executed on the host by the CPUs and those executed on the device by the GPUs. The three compute-intensive stages involve computing the source term, matrix-vector product, and solvation energy, each performed on GPUs as shown in lines 6, 11, and 20, followed by data transfer from the device to the host. The host CPU manages all complex and non-concurrent tasks.

We acknowledge that lines 13 and 14 are currently under investigation due to concerns about parallel efficiency, and thus, we have deactivated these two lines in our current numerical implementation. One of the challenges we encounter is the varying sizes of the block matrices $\mathbf{M_i}$ for $i = 1, \ldots, n$ comprising the preconditioner $\mathbf{M}$, resulting in significant load imbalance on the GPU. Additionally, the use of LU factorization for solving the $\mathbf{M_i y_i} = \mathbf{z_i}$ equations is highly inefficient on GPUs in our present configuration. However, it is crucial to note that disabling the preconditioner in the GPU-based parallelization could substantially increase computation time, particularly when matrix $\mathbf{A}$ is ill-conditioned.

Table 3.2: Pseudocode for DSBI-PB solver using GPU.

1	On host (CPU)
2	read biomolecule data (charge and structure)
3	call MSMS to generate triangulation
4	copy biomolecule data and triangulation to device
5	On device (GPU)
6	each thread concurrently computes and stores source terms for assigned triangles
7	copy source terms on device to host
8	On host
9	set initial guess $\mathbf{x}_0$ for GMRES iteration and copy it to device
10	On device
11	each thread concurrently computes assigned segment of matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{x}$
12	copy the computed matrix-vector $\mathbf{y}$ to host memory
13*	each thread concurrently solves its assigned portion of $\mathbf{M}\mathbf{x} = \mathbf{y}$
14*	copy the solution $\mathbf{x}$ to host memory
15	On host
16	test for GMRES convergence
17	if no, generate new $\mathbf{x}$ and copy it to device, go to step 10 for the next iteration
18	if yes, generate and copy the final solution to device and go to step 19
19	On device
20	compute assigned segment of electrostatic solvation free energy
21	copy computed electrostatic solvation free energy contributions to host
22	On host
23	add segments of electrostatic solvation free energy and output result

* currently disabled

3.4. Numerical Results

In this study, we select several proteins relevant to COVID-19 and utilized our parallel PB solvers to compute their electrostatic properties, including both global solvation energies and local surface potentials. Our numerical results are conducted using supercomputers supported by the O'Donnell Data Science and Research Computing Institute at Southern Methodist University. The MPI-based computations are performed on M3 (https://www.smu.edu/oit/services/m3), while the GPU-accelerated computations are executed on SuperPOD (https://www.smu.edu/oit/services/superpod).

3.4.1. Parallel Efficiency of MPI-based Computing

We initially evaluate the parallel efficiency of our MPI-based algorithm using both sequential and cyclic schemes by calculating the solvation energy for protein 7n3c with an MSMS density of 12, resulting in 529,911 boundary elements. We employ up to 256 MPI tasks, and the results are presented in Table 3.3. Column 1 indicates the increasing number of MPI tasks, while Column 2 reports the total CPU time when employing the DSBI scheme to compute the electrostatic solvation free energy.

Due to its $O(N^2)$ computational cost, the CPU time for the DSBI solver is excessively long, even with 256 tasks utilized. Columns 4 and 5 in Table 3.3 present the total CPU time and parallel efficiency for the TABI solver using the sequential and cyclic schemes, both of them significantly outperform DSBI. Columns 8 and 9 provide a closer examination of the time required for a single matrix-vector product $\mathbf{Ax}$, denoted as $\bar{t}_{\mathbf{Ax}}$. This metric represents the average of the iteration's maximum CPU time across all tasks:

$$\bar{t}_{Ax} = \frac{1}{n_i} \sum_{k=1}^{n_i} \max_j t_{\mathbf{Ax}}^{j,k}, \tag{3.27}$$

where $t_{\mathbf{Ax}}^{j,k}$ is the CPU time to compute $\mathbf{Ax}$ from the jth task in the kth GMRES iteration.

Table 3.3: CPU time and parallel efficiency (P.E.) for parallelized direct sum, sequentially parallelized treecode (seq.) and cyclically parallelized treecode (cyc.) for computing electrostatic solvation energy (-6020.52 kcal/mol from TABI solver and -6013.68 kcal/mol from DSBI solver) for protein 7n3c with 529,955 boundary elements. The treecode parameters are $\theta = 0.8$, $N_0 = 100$, and $p = 3$; The number of tasks n_p ranges over $1, \ldots, 256$. The time for one $\mathbf{Ax}$ ($\bar{t}_{\mathbf{Ax}}$) is the average iteration's maximum CPU time over all tasks.

n_p	DSBI Solver		TABI solver							
	Total Time		Total Time				Time for one $\mathbf{Ax}$ ($\bar{t}_{\mathbf{Ax}}$,)			
	CPU (s)	P.E. (%)	CPU (s)		P.E. (%)		CPU (s)		P.E. (%)	
			seq.	cyc.	seq.	cyc.	seq.	cyc.	seq.	cyc.
1	106063.17	100.00	1874.88	1873.60	100.00	100.00	89.75	89.60	100.00	100.00
2	53132.86	99.81	971.25	967.12	96.52	96.87	45.49	45.22	98.63	99.07
4	26549.87	99.87	561.25	502.60	83.51	93.20	25.69	22.57	87.34	99.26
8	13291.47	99.75	321.42	285.25	72.91	82.10	13.94	12.02	80.46	93.22
16	6710.06	98.79	171.41	158.04	68.36	74.09	6.43	5.77	87.30	97.08
32	3928.71	84.37	128.13	114.73	45.73	51.03	3.99	3.26	70.22	85.84
64	2022.84	81.93	99.75	90.81	29.37	32.24	2.14	1.66	65.55	84.24
128	1042.49	79.48	79.82	76.83	18.35	19.05	1.08	0.85	64.65	82.77
256	554.50	74.72	71.70	71.16	10.21	10.28	0.56	0.45	62.27	78.61

Parallel efficiencies are presented in Columns 3, 6, 7, 10, and 11. The parallelization of the DSBI solver exhibits high efficiency, as indicated in Column 3. This is attributed to the simplicity of the algorithm. Apart from the four stages identified in Fig. 3.3, there is minimal serial computation or communication required. However, the parallel efficiency of the TABI solver is not as high as that of the DSBI solver, as illustrated in Columns 6 and 7. This is mainly attributed to the utilization of treecode, which incurs some serial time for tree construction and moment computation. Although the serial time is relatively short when n_p is small, it becomes more significant as n_p increases, while the time spent within parallelized stages decreases.

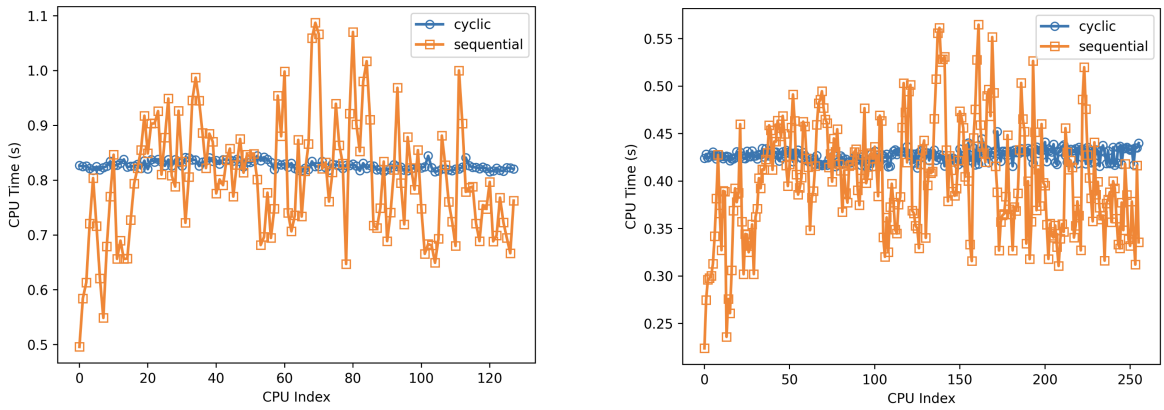


Figure 3.5: MPI-based parallelization with sequential and cyclic schemes: left: 128 tasks, right: 256 tasks. The CPU time reported is $\bar{t}_{\mathbf{Ax}}$, the averages GMRES iteration's maximum CPU times among all tasks.

If we specifically analyze the parallelization of the treecode in computing $\mathbf{Ax}$, Columns 10 and 11 demonstrate a high level of parallel efficiency. It is also evident that the cyclic scheme notably enhances the parallel efficiency compared to the sequential scheme. However, as the fraction of runtime spent on computing matrix-vector products diminishes with increasing n_p, the overall parallel efficiencies in Columns 6 and 7 do not exhibit a significant disparity between the sequential and cyclic schemes. To further scrutinize the performance distinctions between the sequential and cyclic decomposition schemes, we plot $\bar{t}_{\mathbf{Ax}}$ from Eq. (3.27) in Fig. 3.5 when 128 and 256 MPI tasks are utilized. The reduced variance in the cyclic scheme compared to the sequential scheme is evident, attributed to its superior load balancing.

3.4.2. MPI-based TABI solver *vs* GPU-accelerated DSBI solver

Next, we compute the solvation energy for the six COVID-19 proteins using MSMS with a density of 12 to provide sufficient detail of the molecular surface. We employ both the MPI-based TABI solver and the GPU-accelerated DSBI solver. For a fair comparison of computing power, we utilize 64 CPU cores for the MPI-related computing and 1 GPU card for the GPU-related computing. Table 3.4 presents the simulation results.

Table 3.4: Computing electrostatic solvation energies in (kcal/mol) for the involved proteins: ionic strength = 0.15M; $\epsilon_1 = 1$, $\epsilon_2 = 80$; MSMS [46] density=12; N_c is the number of atoms/charges, N is the number of boundary elements, n_i is the number of GMRES iterations, S_{ses} is the solvent excluded surface area, and E_{sol} is the electrostatic solvation energy.

PDB	N_c	N	S_{SES}	n_i^{MPI}	n_i^{GPU}	$E_{\text{sol}}^{\text{MPI}}$	$E_{\text{sol}}^{\text{GPU}}$	t^{MPI} (s)	t^{GPU} (s)
6yi3	2083	169,968	7516.44	10	10	-1941.81	-1945.18	14.76	8.96
7act	2352	188,054	8286.70	14	14	-1893.88	-1934.49	21.35	17.39
7cr5	8133	513,226	22524.23	16	100+	-5713.52	-5786.69	89.52	695.17
7n3c	8459	530,084	23244.85	19	17	-6020.52	-6013.68	99.13	132.20
6wji	10182	641,266	28116.88	13	14	-14009.55	-14016.02	112.82	152.71
7sts	15797	993,572	43457.63	26	100+	-11622.63	-11583.26	422.67	2544.70

Column 1 displays the PDB ID for the proteins in ascending order of their size, followed by the number of atoms in Column 2, the number of boundary elements in Column 3, and the areas of the solvent-excluded surface in Column 4. Columns 5 and 6 represent the number of GMRES iterations. It is noteworthy that the TABI solver exhibits a significantly improved condition number compared to the DSBI solver, attributed to the TABI preconditioner discussed in Section 3.2.4. The solvation energies are reported in Columns 7 and 8, showing a close agreement between the values. Any differences observed can be attributed to factors such as the treecode approximation, the application of the preconditioning scheme, and the error tolerance achieved when the iteration is stopped.

When computing the electrostatic solvation energy of proteins, it is important to note that there is not a precise reference value for comparison. However, if the DSBI solver achieves convergence before reaching the maximum allowed GMRES iterations, its outcome is expected to be more precise compared to that of the TABI solver. This is because the treecode and preconditioning used in TABI may introduce additional approximations. For instance, the solvation energy results ($E_{\text{sol}}^{\text{GPU}}$) for proteins 6yi3, 7act, 7n3c, and 6wji are likely to be more accurate than the corresponding results ($E_{\text{sol}}^{\text{MPI}}$) obtained using the TABI

solver. Indeed, when the GMRES solver reaches its maximum iteration limit 100 without achieving the desired accuracy threshold 10^{-4}, as observed for proteins 7cr5 and 7sts, it becomes difficult to ascertain which result, either $E_{\text{sol}}^{\text{GPU}}$ or $E_{\text{sol}}^{\text{MPI}}$, is more precise.

The computation times are presented in Columns 9 and 10, revealing comparable computing power between 64 CPUs and 1 GPU. However, the choice of algorithms, such as preconditioning versus no preconditioning, and direct summation versus treecode, can lead to significant variations, especially for ill-conditioned or larger systems. For instance, in the case of protein 7sts, which involves nearly one million boundary elements, the MPI-based TABI solver outperforms the GPU-based DSBI solver significantly due to the ill-conditioned nature of the system and the substantial size of the problem.

We subsequently delve deeper into determining the conditions favoring the use of the GPU-accelerated DSBI solver or the MPI-based TABI solver. An illustrative example, detailed in Table 3.5, offers crucial insights. In this scenario, we compute the solvation energy for protein 6yi3 while progressively increasing the MSMS density (d), leading to larger problem sizes, as depicted in Columns 1 and 2. Columns 3 and 4 display the corresponding solvation energy computed using these two approaches. Columns 5 and 6 provide the number of GMRES iterations. The similarity in these results suggests that the discretized system for this protein is well-conditioned, thereby limiting the impact of the preconditioning scheme. We solve the problem using a single CPU core and present the time in Column 7 for reference. Subsequently, we report the time for solving the problem using 64 MPI tasks and one A100 GPU card in Columns 8 and 9, respectively.

Table 3.5: Computing electrostatic solvation energies in (kcal/mol) for the protein 6yi3 at different MSMS densities: ionic strength = 0.15M; $\epsilon_1 = 1$, $\epsilon_2 = 80$; d is the MSMS density, N is the number of boundary elements, n_i is the number of GMRES iterations, E_{sol} is the electrostatic solvation energy. Results are generated using KOKKOS and MPI on ManeFrame III; MPI results are from using 64 tasks; GPU results are from using one A100 GPU.

d	N	$E_{\text{sol}}^{\text{MPI}}$	$E_{\text{sol}}^{\text{GPU}}$	n_i^{CPU}	n_i^{GPU}	t_{CPU} (s)	t_{MPI} (s)	t_{GPU} (s)
2	28,767	-2057.61	-2056.26	10	10	29.34	2.76	0.83
4	56,127	-1999.01	-1997.03	10	10	66.40	4.46	1.52
6	84,903	-1968.00	-1966.87	10	16	108.52	7.05	4.98
8	110,307	-1954.62	-1952.23	10	10	145.13	9.35	4.32
12	169,955	-1945.18	-1941.81	10	10	240.54	14.76	8.91
16	229,901	-1940.96	-1936.87	10	11	340.82	19.86	17.66
18	257,236	-1938.27	-1933.67	10	11	385.96	**21.69**	**23.73**
20	287,202	-1937.18	-1931.77	10	12	438.86	24.54	28.73
24	343,806	-1933.63	-1928.65	10	11	534.31	35.13	38.62
28	407,196	-1933.04	-1927.56	10	12	653.06	41.84	55.18
32.5	471,307	-1931.76	-1926.04	10	12	760.12	51.23	77.83
64	946,335	-1928.51	-1921.81	10	13	1701.06	145.65	311.07

The findings indicate that for a protein with a well-conditioned discretized system and a number of boundary elements less than 250,000, opting for the GPU-accelerated DSBI solver is preferable. This preference arises because smaller systems demonstrate the superior performance of the GPU-accelerated DSBI solver compared to the MPI-based TABI solver. If the condition of matrix **A** highlights a significant requirement for preconditioning, the threshold number will be lower for the GPU-accelerated DSBI solver. The rapid performance of GPUs, at the very least, offers the possibility of conducting molecular dynamics or Monte Carlo simulations for small and medium-sized proteins. For instance, if a protein can be adequately described with 50,000 boundary elements, a single PB equation solution would take approximately one second using one GPU card, whereas it would take about 4 seconds on a 64-core cluster.

It is worth mentioning that solving the boundary integral PB equation provides both the electrostatic potential and its normal derivative on the molecular surface. By plotting the potential on the surface elements and color-coding them accordingly, we can gain insights into various aspects such as identifying docking sites for ligands or understanding protein-protein interactions. Fig. 3.6 illustrates some examples of such visualizations.

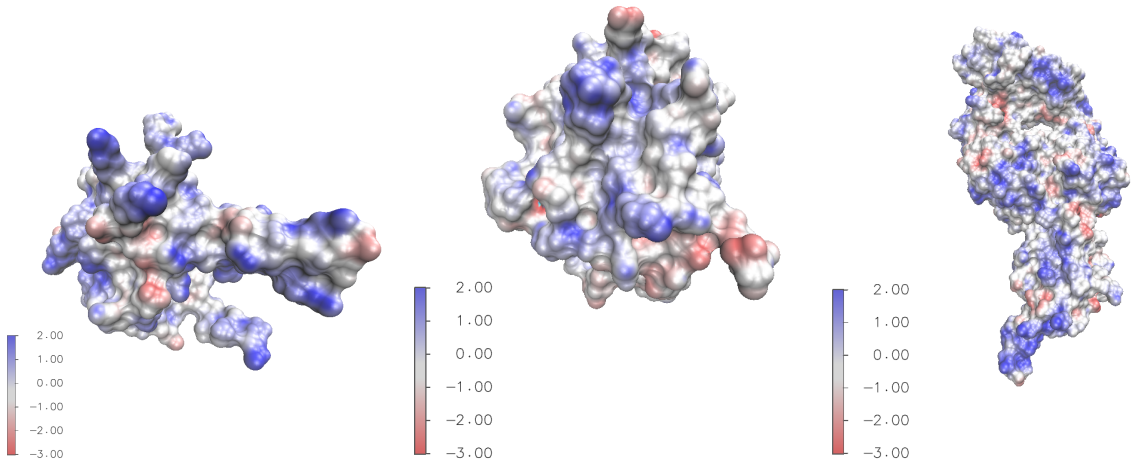


Figure 3.6: Color coded electrostatic surface potential in kcal/mol/e_c on the molecular surface of proteins 6yi3 (left), 7act (middle), and 7n3c (right); plot is drawn with VMD [81].

3.5. Conclusion

In this chapter, we investigate on the practical application of the PB model to examine selected proteins that hold crucial roles in the transmission, treatment, and prevention of COVID-19 viral diseases. For this purpose, we apply the boundary integral form of the PB equation to the molecular surfaces of these proteins. Through these computations, we obtain both the electrostatic solvation energy, which serves as a global measurement, and the electrostatic surface potential, offering insights into the local characteristics of the selected proteins.

We examine the parallel performance of two competing solvers for solving the BI-PB equations on these selected proteins. Taking into account the strengths of current algorithms and computational hardware, our focus is on parallelizing TABI-PB solver using MPI on

CPUs and DSBI-PB solver using KOKKOS on GPUs. Our numerical investigations reveal that the DSBI solver running on a single A-100 GPU outperforms the TABI solver with MPI on 64 CPUs when the number of elements is below 250,000. Thus, when suitable hardware is available, employing the DSBI solver on GPUs for PB model-based molecular dynamics or Monte Carlo simulations is advisable, particularly when dealing with smaller numbers of boundary elements to achieve rapid calculations.

When both GPU and MPI resources are available, and the quality of the triangulation is satisfactory enough to obviate the need for the TABI preconditioner in achieving GMRES convergence, we propose the use of the GPU-accelerated DSBI solver when the number of boundary elements is below 250,000. In cases where the number of elements exceeds this threshold, the MPI-based TABI solver is recommended. Moreover, if the number of elements grows to a point where the memory capacity of a CPU task cannot accommodate an entire tree, it is advisable to consider employing a domain-decomposition MPI scheme [10,70,76,82]. It is worth mentioning that the memory utilization of the TABI solver scales linearly with the size of the problem. For instance, when employing one million boundary elements, the memory consumption slightly exceeds 1GB. Consequently, for computing tasks on clusters equipped with at least 64GB of memory per MPI rank, we can efficiently handle problems of up to approximately 64 million boundary elements. This capacity is adequate for simulating medium to large-sized proteins comprising tens of thousands of atoms. However, for even larger biomolecules, such as the viral capsids of Zika or H1N1 viruses, which may contain tens of millions of atoms [83], a domain decomposition approach should be considered [76].

Chapter 4

Bridging Eulerian and Lagrangian Poisson-Boltzmann Solvers by ESES

In this chapter, we investigate the capabilities of the Eulerian Solvent Excluded Surface (ESES) software [84] in generating conjugated Eulerian and Lagrangian representations of molecular surfaces. Our motivation is to identify a tool that can accurately and efficiently simulate the molecular surface prior to conducting the PB modeling. We evaluate the quality of surface discretization under both frameworks through numerical validation using two recently developed PB solvers: a Cartesian-based MIB-PB solver (discussed in Chapter 2) and a Lagrangian TABI-PB solver (discussed in Chapter 3).

4.1. Background

Gauss's law, one of the four fundamental equations of electromagnetism, establishes a relationship between the distribution of electric charges and the resulting electric field. It can be formulated in both integral and differential forms. The integral form describes the electric flux passing through a closed surface, while the differential form yields the Poisson equation. The connection between finite difference PB solvers and boundary integral PB solvers lies in the relationship between the differential and integral forms of Gauss's law. When a finite difference method is used, a rigorous treatment of interface conditions requires the location where the mesh line intersects the molecular surface and the normal direction of the surface at the intersection. This is called a "Eulerian" representation of the molecular surface as seen in Fig. 4.1(a). Conversely, the boundary integral method requires a body-fitted triangulation of the molecular surface, referred to as the "Lagrangian" representation in Fig. 4.1(b).

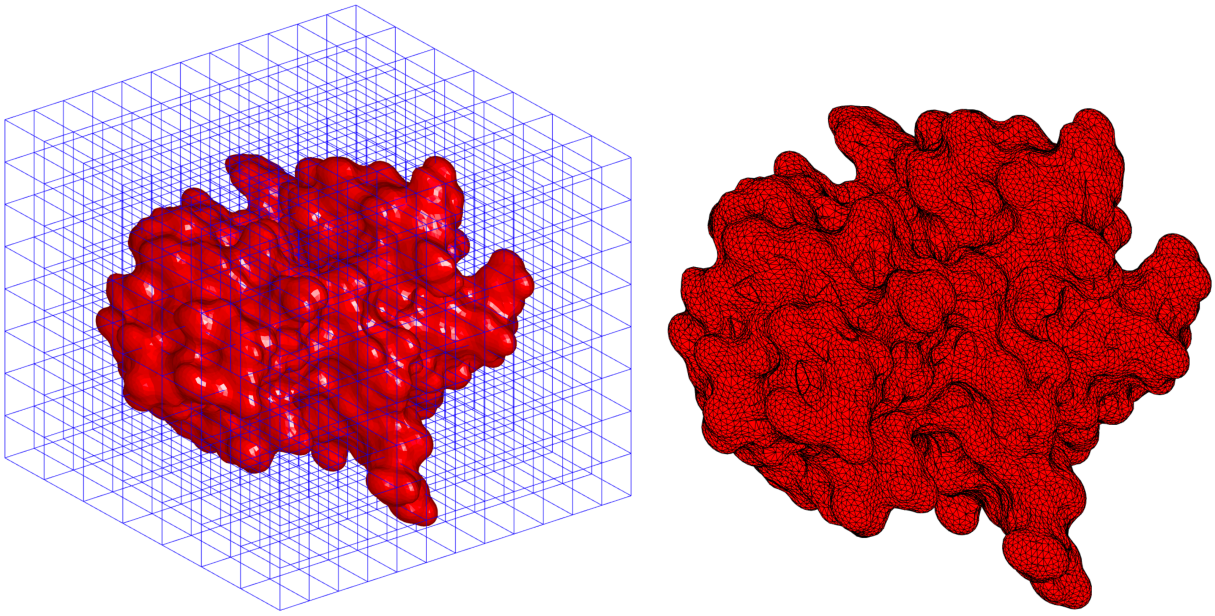


Figure 4.1: The two SES representation of protein 1a63: (a) Eulerian representation with location and surface normal direction of intersection between mesh lines and the SES surface; (b) Lagrangian representation with triangles and surface normal direction at the vertices.

Previously, Liu et al. developed a software package known as Eulerian Solvent Excluded Surface (ESES) [84] for accurately generating Solvent Excluded Surfaces (SESs) on Cartesian grids. Besides, for a given biomolecule, ESES not only calculates the total surface area but also partitions the surface area based on atomic types.

In this chapter, we focus on ESES's body-fitted or Lagrangian representation of the molecular surface. While previous studies primarily used the Lagrangian representation of ESES for visualization [84], we aim to quantitatively measure the performance of it in triangulation and investigate its usage for Lagrangian PB solvers like the TABI solver [37], compared with Eulerian PB solvers like the MIB solver [24, 26, 85]. This work presents the **first exploration** in the literature of the conjugated Eulerian and Lagrangian surfaces of ESES, along with an examination of the associated differential and integral forms of Gauss's law.

4.2. Molecular Surface Definitions and Generators

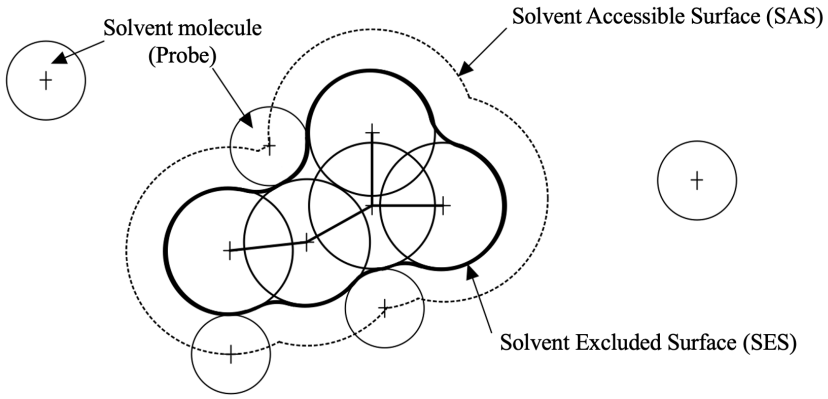


Figure 4.2: Molecular surface desciptions.

The molecular surface typically has three popular definitions. The van der Waals (VDW) surface, formed by the union of exposed atomic surfaces, provides a simple definition that is useful for both molecular modeling and visualization. However, the VDW surface introduces numerous geometric singularities, which can cause numerical instability when used as an interface for implicit solvation modeling.

The other two surfaces involve a spherical solvent probe rotating upon contacting the VDW surface. The trace of the probe's center yields the Solvent Accessible Surface (SAS), while the contact, toroidal, and reentrant surfaces of the probe sphere constitute the Solvent Excluded Surface (SES), as originally described by Lee and Richards [86]. The SAS boasts a relatively simple definition and is easier to describe numerically. Nonetheless, despite being smoother than the VDW surface, the SAS still exhibits many sharp corners when transitioning between atoms.

The SES emerges as the smoothest among the three surfaces, displaying an essentially C^1-continuous surface with only occasional cusps under extreme conditions [22]. Consequently, it has gained prominence as the predominant surface definition in the fields of biophysics

and molecular biology. In addition to its effectiveness in visualizing biomolecules, it is widely employed in implicit solvent models owing to its smoothness akin to C^1. Connolly formulated the mathematical representation of the SES for arbitrary biomolecules in terms of convex patches, saddle patches, and concave patches [87].

Several software tools are available for generating the SES in Lagrangian representations, such as MSMS [46] and NanoShaper [88].

- MSMS, based on the use of the reduced surface introduced by Sanner [46], generates a triangulation of specified density by fitting predefined triangulated patches to the surface. The density of the mesh is controlled by a user-specified parameter d that sets the number of triangles in units of vertices per Å^2.
- NanoShaper, developed by Decherchi and Rocchia [88], utilizes a ray-casting algorithm, where rays parallel to the coordinate axes are projected, and intersections with the surface are determined, and then utilize the resulting vertex positions of intersection to generate the triangulation via Marching Cubes Algorithm [89]. The density of the mesh is controlled by a scaling parameter s, which dictates the inverse side length of a cubic grid cell in units of Å.

These representations are particularly suitable for boundary integral methods. In a previous study [90], a comparison between the popular MSMS and NanoShaper has been conducted. However, when the Eulerian representation is necessary, additional efforts are required to determine the intersection of the mesh line and its normal direction, which poses a significant challenge and often involves complex interpolation schemes [20, 91].

ESES excels in generating accurate SESs for solving the PB equation on Cartesian grids. It provides surface information by identifying intersection points between grid lines and the interface, offering coordinates for each intersection point and their corresponding surface

normals. Moreover, each grid point can be categorized as inside or outside the interface, facilitating the initialization of the PB equation and enforcement of interface conditions.

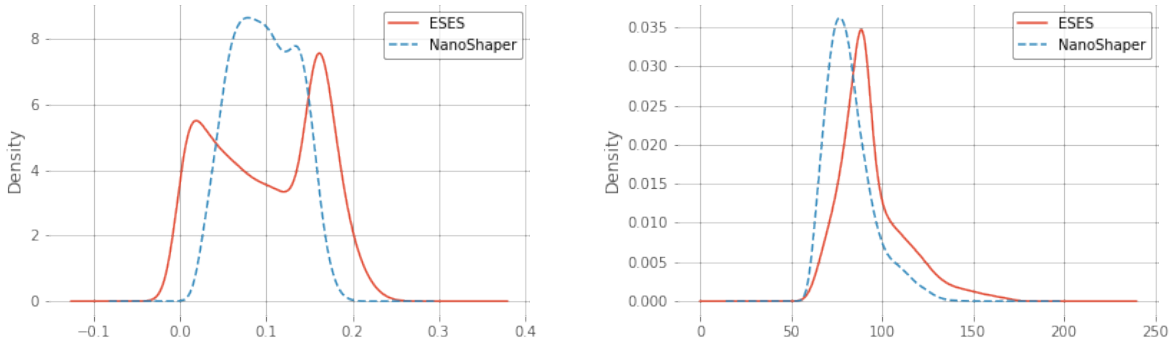
The analytical approach used by ESES ensures high-order PB solvers achieve their intended convergence or accuracy in the L_1 norm, crucial for assessing the performance of these methods on complex biomolecular surfaces. ESES computes the analytical representation of potential SES patches based on Connolly's work [87], where accurate convex, saddle, and concave patches are generated and stored for efficient access. Each Cartesian grid point is then classified as inside or outside the SES based on its relationship with these patch types in its neighborhood. For Cartesian edges spanning from inside to outside the SES, their analytical intersection positions and corresponding normal directions with respect to the outermost surface patch are computed. The SES comprises three types of patches: convex patches, saddle patches, and concave patches. These patches are determined by single atoms, pairs of atoms, or triplets of atoms, respectively.

Upon completing the Cartesian representation of the SES, the intersection points between the grid points and the SES can be connected to form a triangulated surface using the Marching Cubes Algorithm [89]. Initially intended for visualization purposes, this triangulation unexpectedly yields a compatible Lagrangian representation of the SES, suitable for solving boundary integral PB equations. Concerns have been raised regarding the non-equilateral nature of the triangles generated by ESES, potentially impacting the convergence of the GMRES method for solving the discretized integral equations. However, the preconditioning schemes outlined in Chapter 3 effectively address this issue, rendering the ESES Lagrangian representation another viable option as a body-fitted surface for PB solvers.

4.3. Results and Discussions of ESES Performance

In this section, we numerically demonstrate that ESES can generate high-fidelity Solvent Excluded Surfaces (SESs) with both Eulerian and Lagrangian representations. In all calcu-

lations, the solute dielectric constant is set to $\epsilon_1 = 1$ and the solvent dielectric constant is set to $\epsilon_2 = 80$. For test cases on spherical cavities with analytical solutions, the tolerance for the bi-conjugate gradient solver used by the MIB solver and the tolerance for the GMRES solver used by the TABI solver are all set to 10^{-10}. Other than this, the tolerance for the MIB solver is set to 10^{-6}, and the tolerance for the TABI solver is set to 10^{-4} as default values. In the TABI solver [37], the treecode parameters are specified as follows: the Maximum Acceptance Criterion (MAC) parameter $\theta = 0.8$, Taylor expansion order $p = 3$, and maximum number of particles per leaf $N_0 = 100$. These results are computed using a 13inch MacBook Pro with intel core-i5 processor and 16 GB of RAM.



(a) (b)

Figure 4.3: Triangulation quality comparison between ESES and NanoShaper using SES of protein 1AJJ: (a) distribution in terms of kernel density estimation (KDE) of triangles' areas in Å^2; (b) distribution of triangles' maximum angles in degrees.

First of all, the "triangulation quality", which is crucial for numerical simulation methods such as the finite element and the boundary element methods, needs to be validated between ESES and NanoShaper. Ideally, the triangles should be quasi-uniform, meaning that they are essentially the same size, and they should resemble equilateral triangles. As shown in Figure 4.3, the triangles generated by ESES [84] and NanoShaper [88] are compared using their areas and maximum angles. When a similar number of triangles are generated by ESES and NanoShaper, the distribution of areas of triangles from NanoShaper is more centered

around 0.1 Å^2 compared to that from ESES, and the distribution of maximum angles of triangles from NanoShaper is closer to 60° than that from ESES, meaning that NanoShaper provides a slightly better triangulation quality than ESES.

4.3.1. TABI using Lagrangian ESES *vs* TABI using NanoShaper

Table 4.1: Computing electrostatic solvation energy of a protein (PDB 1AJJ) with TABI solver using two molecular surface generators ESES and NanoShaper (NS): Densities are used as a parameter to control the number of triangular faces N; Dimension (for ESES only) is the number of grid points in each direction; ΔE_{sol} is the difference of solvation energy calculated using TABI solver with ESES and NanoShaper surface generators.

Densities		Dimension	N		E_{sol} (kcal/mol)			CPU time (s)	
ESES	NS	ESES	ESES	NS	ESES	NS	ΔE	ESES	NS
1.12	0.90	28*30*33	5084	5096	-1222.20	-1261.92	39.72	2.18	2.14
0.80	1.26	39*42*46	10124	10168	-1180.16	-1188.74	8.58	5.70	5.55
0.56	1.80	56*60*65	20640	20884	-1157.24	-1160.53	3.29	15.16	15.17
0.40	2.50	78*84*91	40752	40772	-1146.93	-1148.53	1.60	34.89	34.05
0.28	3.58	111*119*130	83232	83640	-1141.76	-1142.54	0.77	76.33	76.84
0.20	5.00	155*167*182	163276	163292	-1138.94	-1139.26	0.32	196.14	193.70

We present an example here for comparing the ESES performance in solving PB model with NanoShaper [88] by using the TABI solver, as depicted in Table 4.1. When refining the mesh, we adjust the parameters in ESES (Column 1) and NanoShaper (Column 2) to generate a similar number of triangles N as seen in Columns 4-5. Concurrently, ESES reports the number of Cartesian grid points as shown in Column 3. From this example, we observe that as the mesh is refined, the electrostatic solvation energy computed by the TABI solver using ESES in Column 6 approaches that computed using NanoShaper in Column 7, resulting in a smaller energy difference ΔE_{sol} as seen in Column 8. The total CPU time is similar when different surfaces are used, as shown in Columns 9 and 10. This is owed to the

preconditioning scheme [44]; similar numbers of iterations are used when different surfaces from ESES and NanoShaper are utilized.

There are extensive studies on the Eulerian SES representation from ESES as opposed to the Lagrangian representation from MSMS [46] in [84]. Both comparison demonstrate that when ESES's Lagrangian surface representation is used for the BI-PB solver, ESES performs as well as popular surface generators such as NanoShaper and MSMS.

4.3.2. MIB using Eulerian ESES *vs* MIB using Eulerian MSMS

Following the validation of the Lagrangian representation of ESES using the TABI solver, we proceed to validate the Eulerian representation of ESES using the MIB solver. Similar to the need for the well-established NanoShaper for comparison, we use MSMS instead [46]. It is worth noting that MSMS and NanoShaper have direct generation of a Lagrangian SES, and Zhou developed a Fortran wrapper of MSMS in his thesis [92] to convert the Lagrangian representation to the Eulerian representation, so that we can use MSMS for the Eulerian comparison here.

Table 4.2: Computing electrostatic solvation energy of a spherical cavity with a centered charge using MIB solver on the Lagrangian representation from ESES and from MSMS.

h	MSMS(den=10)	MSMS(den=20)	MSMS(den=100)	ESES
1	-82.48219907	-82.27048894	-82.16175924	-79.58699949
0.5	-82.52123400	-82.32504614	-82.19843970	-82.01643791
0.25	-82.53823124	-82.33111065	-82.21280699	-82.28426183
0.125	-82.53534892	-82.34891499	-82.21596355	-82.26497101

We calculate the solvation energies for three different molecules: a spherical cavity of radius 2Å with a centered unit charge, and proteins 2pde and 1aho. The results for the spherical cavity are reported in Table 4.2. From this table, we observe that when MSMS surface with a fixed density is used, the solvation energy converges as the mesh is refined

(from top to bottom with h changing from 1 to 0.125). Similarly, when ESES surface is used, the same convergence pattern can be observed. The most significant observation from this table is that the results obtained with MSMS using refined density converge to those obtained with ESES using a sufficiently fine mesh ($h = 0.125$).

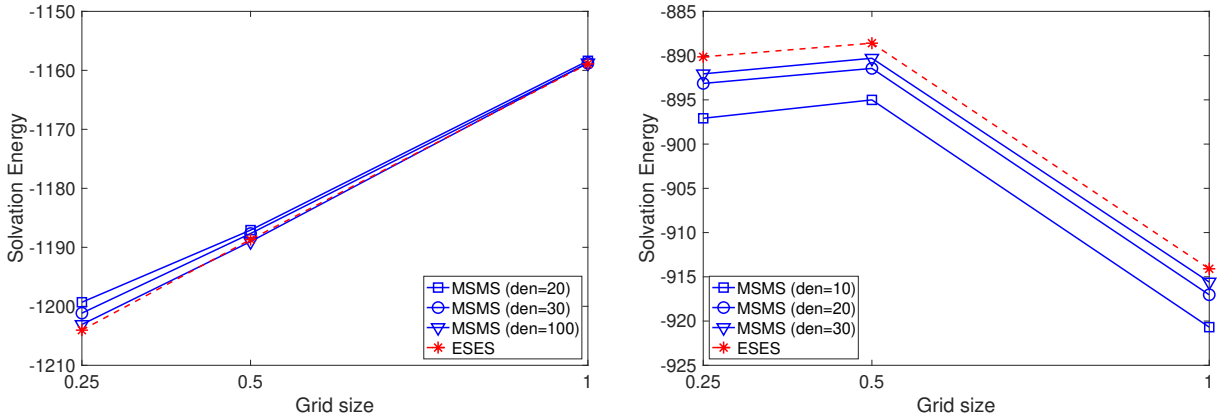


Figure 4.4: Computing electrostatic solvation energy for the proteins 2pde (left) and 1aho (right) using MIB solver on the Lagrangian representation from ESES and from MSMS.

For the two proteins, we present the results in Fig. 4.4. Depending on the size of the protein, we use MSMS density 20, 30, 100 for protein 1pde, and MSMS density 10, 20, 30 for protein 1aho. The figure illustrates that increasing MSMS density leads to a solvation energy approaching that obtained with ESES.

4.3.3. MIB using Eulerian ESES *vs* TABI using Lagrangian ESES

We then provide an example demonstrating the connection between the Eulerian representation and the Lagrangian representation of ESES. These representations are employed to solve the PB model using finite difference methods with interface treatment, i.e., the MIB solver [26, 85], and using the boundary integral method, i.e., the TABI solver [37].

The results for Protein 1ajj are displayed in Table 4.3. In this scenario, we have no control over the number of triangles with the Lagrangian representation. Once the mesh size h in Column 1 is determined for the Eulerian representation, the number of grid points for

MIB in Column 2 and the number of triangles in Column 3 are automatically determined as well. We observe that the solvation energy from MIB in Column 4 and that from TABI in Column 5 approach each other as the mesh is refined. Regarding the CPU time in Columns 7-8, the TABI solver exhibits slower performance at the initial stage but demonstrates an advantage in convergence speed as the mesh is refined, owing to the $O(N \log N)$ treecode acceleration.

Table 4.3: Computing electrostatic solvation energy of protein 1ajj using TABI solver [37] on the Lagrangian representation and using MIB solver [26] on the Eulerian representation of the molecular surface, both generated by ESES; zero ionic strength.

h	Dimension	N	E_{sol} (kcal/mol)			CPU time (s)	
			MIB	TABI	ΔE	MIB	TABI
1	31*34*37	6464	-1137.35	-1208.98	71.63	0.95	2.97
0.5	62*67*73	25956	-1138.04	-1153.46	15.42	8.80	20.45
0.25	124*134*146	104412	-1139.25	-1140.60	1.35	107.04	116.53
0.125	248*267*292	418544	-1139.37	-1137.48	-1.89	1312.65	522.24

4.4. Conclusion

In this study, we explore the unique capabilities of ESES software [84] for generating Eulerian and Lagrangian surfaces. We conduct a numerical assessment of surface discretization quality under both frameworks using two PB solvers: a Cartesian-based MIB-PB solver and a Lagrangian TABI-PB solver. Our numerical findings demonstrate that, owing to ESES, both solvers achieve the desired convergence when employing the Eulerian and Lagrangian representations of the molecular surface generated by ESES. This study marks the initial investigation in the literature of the combined Eulerian and Lagrangian surfaces provided by ESES, accompanied by an analysis of the differential and integral forms of Gauss's law. Consequently, ESES proves to be a valuable asset for producing two distinct mesh types, thereby enhancing our ability to tackle various types of PB solvers with greater flexibility.

Chapter 5

Poisson-Boltzmann based Machine Learning Model

In this chapter, we introduce a Poisson-Boltzmann based Machine Learning (PB-ML) model designed to predict electrostatic solvation free energies of biomolecules. Our aim is to provide an efficient ML-based tool that achieves a similar level of accuracy in solving the PB equation but requires significantly less time for computation. We start from evaluating various PB solvers to identify the most accurate one for generating ML labels. Additionally, we adopt the Multiscale Weighted Colored Subgraph (MWCS) technique, which provides intrinsically low-dimensional representations of biomolecular structures, in conjuration with the fast GB models, to generate our ML features. To guarantee computational efficiency, we evaluate various ML algorithms, such as Linear Regression (LR), Random Forest (RF), Gradient Boosting Decision Tree (GBDT), and Deep Neural Network (DNN), to identify the most effective ML techniques.

5.1. Introduction

Acquiring highly accurate electrostatic potentials for large biomolecules often comes with substantial expenses. For instance, solving the PB model for a protein containing around 50,000 atoms with a mesh size of 0.2 Å can take several days on a single CPU. Additionally, the insights gained from electrostatic analysis of one biomolecule cannot be directly applied to others. Consequently, separate electrostatic analysis are required for different proteins or for the same protein with different protonation states or conformations, placing significant demands on computational resources. These challenges underscore the importance of innovative approaches, such as machine learning and dynamic programming, in biomolecular electrostatic analysis.

To address this, we develop a machine learning solution for the PB equation to analyze the electrostatic properties of biomolecules. We begin by constructing a precise and efficient mathematical representation of the electrostatic potential to effectively characterize its probability distribution within the space of protein structures and electric charge configurations. Although the exact form of this distribution is theoretically unknown, it can be sampled using a PB solver, which generates training labels for machine learning. Our approach is guided by two main hypotheses: the "representability" hypothesis and the "learning" hypothesis.

The "representability" hypothesis suggests that the electrostatic potential of a biomolecule can be represented by a set of partial charges and their geometric relationships with the solvent. This hypothesis informs the construction of the feature vector used to characterize the probability distribution of biomolecular electrostatics. The "learning" hypothesis suggests that biomolecular electrostatics can be effectively represented by a feature vector, as defined by the representability hypothesis. With a sufficiently sampled probability distribution obtained from a training set, we can establish a machine learning model based on training labels and associated feature vectors. This model can accurately predict the electrostatic potential of unseen datasets which share the same probability distribution as the training set.

The outlined protocol requires the utilization of a precise Poisson-Boltzmann (PB) solver to calculate machine learning labels and subsequently determine the probability distribution of molecular and biomolecular electrostatics. In this context, we utilize the reliable MIB-PB solver [24] with a fine mesh size of 0.2 Å to generate solvation energy labels, thus mitigating numerical inaccuracies.

In addition, the representability hypothesis lacks specific methodologies for developing an accurate and efficient representation. Typically, a biomolecule within the human body consists of approximately 6,000 atoms, leading to a Euclidean space of 18,000 dimensions ($\mathbb{R}^{18,000}$). The high dimensionality makes calculations based on first principles infeasible for

predicting data on a large scale. Therefore, it becomes crucial to devise representations of biomolecular structures that are scalable and inherently of lower dimensionality. Our hypothesis suggests that the essential physics of these structures can be found within low-dimensional areas or manifolds within the vast dimensional space. We choose to utilize graph theory for its simplicity, complemented by a set of features derived from the rapid Generalized Born (GB) models, to predict electrostatics based on the Poisson-Boltzmann (PB) model.

In the following sections, we delineate our data preparation procedure for our Poisson-Boltzmann Machine Learning (PBML) model, which involves employing graph theory to generate features. We then conduct several convergence tests to verify that our MIB-PB solver is the most accurate and suitable for generating labels for our PBML model. Subsequently, we utilize these features and labels to train our PBML model and compare its performance with other PB solvers.

5.2. Data Preparation

In this study, we employ a dataset consisting of 4294 protein structures sourced from the "PDBbind v2015 refined set" and "PDBbind v2018 refined set" for training purposes [93]. Additionally, we utilize the "PDBbind v2015 core set", consisting of 195 proteins, as our test dataset. The protein structures in the training set range in size from 997 to 27,713 atoms, while those in the test set vary from 1,702 to 26,236 atoms. The original dataset comprises protein-ligand complexes, some of which contains missing atoms and side chains. To rectify this, we pre-process the data using the protein preparation wizard utility from the "Schrodinger 2015-2 Suite" with default parameters to fill the missing atoms and side-chains. Furthermore, we assign atomic van der Waals radii and partial charges to these structures using the "Amber ff14SB" general force field.

We formulate the prediction of PB electrostatic solvation free energy as a standard supervised learning task. The training dataset $\mathcal{D}$ is represented as:

$$\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}) | \mathbf{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \mathbb{R}, i = 1, \cdots, M\},$$

where $\mathbf{x}^{(i)}$ denotes the feature vector for the ith sample in the training set, $y^{(i)} = \Delta G_i^{\mathrm{PB}} - \Delta G_i^{\mathrm{GB}}$ represents the difference between the electrostatic solvation free energy computed by the PB and GB model for the ith sample, and n and M denote the sizes of the feature vector and the training set, respectively. The term G_i^{PB} is obtained from the accurate MIB-PB solver, as verified in Section 5.4.2. The other term ΔG^{GB} serves as a core feature outside the network, offering a global estimate, while other features are fed into the network to provide local details. The feature vector is constructed utilizing graph theory and the GB model.

5.2.1. Feature Description

We currently employ 367 features considering protein structures, force field, graph theory representation, etc. These features can be categorized into two groups: 15 paired elements among {C,N,S,O,H} and 6 types of kernels (Exponential and Lorentz kernels). Among these features, 240 are GB model related features based on effective Born Radii and element-specific relationships; 51 are protein features based on protein structure and charges distribution; 31 are environment features based on properties of residual groups; and finally 45 features from graph theory representation:

- GB model related features (240): As previously described in Chapter 1, the GB approximation can be formulated as:

$$\Delta G^{\mathrm{GB}} \approx \sum_{ij} \Delta G_{ij}^{\mathrm{GB}} \tag{5.1}$$

$$= \frac{1}{2}\Big(\frac{1}{\epsilon_2} - \frac{1}{\epsilon_1}\Big)\frac{1}{1+\alpha\beta}\sum_{ij} q_i q_j \Big(\frac{1}{f_{ij}(r_{ij}, R_i, R_j)} + \frac{\alpha\beta}{A}\Big). \tag{5.2}$$

  For each of the 15 paired elements among {C,N,S,O,H}, there are 16 features, encompassing the summand terms and absolute summand terms of Eq. (5.1), the first term of the summand terms of Eq. (5.1) and its absolute value, the second term of the summand terms of Eq. (5.1) and its absolute value, the multiplication of paired charges with dielectric and its absolute value, the multiplication of paired charges and its absolute value, and 6 rigidity indices involving both Lorentz and exponential functions with $\nu = 5$ and $\kappa = 5$ respectively, along with their products affecting change values and magnitude. We use the BornRadius code [94] developed by Onufriev's group, integrated with Python scripts, to generate these features.

- Protein features (51): area (7), charge (7), absolute charge (7), van der Waals force (15), Coulomb force (15). These features are categorized based on atom groups, comprising the following 7 types: {C, N, O, S, H, CNOS, CNOSH}. Each feature represents the sum of atoms belonging to the respective atom group.

- Environment features (31): volume (6), hydropathy (6), area (6), weight (6), pharma (6), sum of charge (1). These features are computed based on associated residue groups: hydro, polarall, polaruncharged, polarposcharged, polarnegcharged, and specialcase. For each type of environmental feature, we calculate the sum of atoms belonging to the respective residue group.

- Features from graph theory representation (45): Exponential and Lorentz kernels (15 × 3 kernels). In this work, we use MWCS centrality $\mu^{k,\sigma,\tau,\nu,w}$ to construct a mathematical

representation of biomolecular electrostatics. In our pursuit of an intrinsically low-dimensional representation of electrostatic solvation free energies for a large set of proteins, we examine both the exponential kernel (σ = E) and the Lorentz kernel (σ = L). Each kernel is parameterized with rigidity ($w = 1$) and charge ($w = q$) weight selections. We set the powers of the exponential kernel to $\nu = 2$ and the Lorentz kernel to $\nu = 3$. We consider atomic features with 15 types of element partitions ($\mathcal{P}_k$, $k = 1, 2, \ldots, 15$).

5.2.2. Graph Theory Representation

In conjunction with machine learning algorithms, the multiscale weighted colored subgraph (MWCS) method has shown superior performance compared to many other approaches in representing complex biomolecular structures [95,96]. Initially, we adopt the weighted colored subgraph (WCS) to elucidate electrostatic interactions within a protein composed of N atoms. This method integrates kernels to delineate pairwise distance-weighted atomic correlations, categorizing interactions based on element types and generating colored subgraphs. To apply WCS for analyzing protein electrostatic interactions, we represent all atoms and their pairwise interactions as a weighted graph $G(V, E)$, where vertices V are defined as:

$$V = \{(\mathbf{r}_i, \alpha_i) | \mathbf{r}_i \in \mathbb{R}^3, \alpha_i \in \mathcal{C}, i = 1, 2, \ldots, N\}, \tag{5.3}$$

where $\mathbf{r}_i$ represents the position of the ith atom, and $\mathcal{C} = \{\mathrm{C}, \mathrm{N}, \mathrm{O}, \mathrm{S}, \mathrm{H}\}$ denotes commonly occurring element types in proteins, which may be adjusted for different biomolecular systems.

To represent pairwise interactions between atoms in a protein, we define a colored set $\mathcal{P} = \{\alpha\beta\}$ where $\alpha, \beta \in \mathcal{C}$. For each subset of element pairs $\mathcal{P}_k$, $k = 1, 2, \ldots, 15$, a set of involved vertices $V_{\mathcal{P}_k}$ is a subset of V that includes all atoms belonging to the pair in $\mathcal{P}_k$. For

instance, a partition $\mathcal{P}_2 = \{\text{CN}\}$ contains all pairs of atoms in the protein where one atom is carbon and the other is nitrogen. Based on this setting, all edges in such WCS describing pairwise atomic interactions are defined as follows:

$$E_{\mathcal{P}_k}^{\sigma,\tau,\zeta} = \{\Phi_{\tau,\zeta}^{\sigma}(\|\mathbf{r}_i - \mathbf{r}_j\|) | \alpha_i\beta_j \in \mathcal{P}_k; i, j = 1, \ldots, N\}, \tag{5.4}$$

where $\|\mathbf{r}_i - \mathbf{r}_j\|$ represents the Euclidean distance between the i-th and j-th atoms, σ denotes the type of radial basis functions (e.g., $\sigma = \text{L}$ for Lorentz kernel, $\sigma = \text{E}$ for exponential kernel), τ is a scale distance factor between two atoms, and ζ is a parameter of power in the kernel (i.e., $\zeta = \kappa$ when $\sigma = \text{E}$, $\zeta = \nu$ when $\sigma = \text{L}$). The kernel $\Phi_{\tau,\zeta}^{\sigma}$ characterizes a pairwise correlation satisfying the following conditions:

$$\Phi_{\tau,\zeta}^{\sigma}(\|\mathbf{r}_i - \mathbf{r}_j\|) = \begin{cases} 0 & \text{as } \|\mathbf{r}_i - \mathbf{r}_j\| \to 0, \\ 1 & \text{as } \|\mathbf{r}_i - \mathbf{r}_j\| \to \infty. \end{cases} \tag{5.5}$$

Commonly used radial basis functions include generalized exponential functions:

$$\Phi_{\tau,\kappa}^{\text{E}}(\|\mathbf{r}_i - \mathbf{r}_j\|) = e^{-(\|\mathbf{r}_i - \mathbf{r}_j\|/\tau(r_i + r_j))^{\kappa}}, \quad \kappa > 0, \tag{5.6}$$

and generalized Lorentz functions:

$$\Phi_{\tau,\nu}^{\text{L}}(\|\mathbf{r}_i - \mathbf{r}_j\|) = \frac{1}{1 + (\|\mathbf{r}_i - \mathbf{r}_j\|/\tau(r_i + r_j))^{\nu}}, \quad \nu > 0, \tag{5.7}$$

where r_i and r_j are the van der Waals radii of the i^{th} and j^{th} atoms, respectively.

Centrality is a fundamental concept in graph theory and network analysis, providing insights into the significance of nodes within a network [97]. It encompasses measures like closeness and harmonic centralities, which are defined as $1/\sum_j \|\mathbf{r}_i - \mathbf{r}_j\|$ and $\sum_j 1/\|\mathbf{r}_i - \mathbf{r}_j\|$, respectively. The degree of centrality simply counts the number of edges upon a node. Our

atomic centrality for the i^{th} atom can be viewed as an extension of the harmonic formulation:

$$\mu_i^{k,\sigma,\tau,\nu,w} = \sum_{j=1}^{|V_{\mathcal{P}_k}|} w_{ij}\Phi_{\tau,\nu}^{\sigma}(\|\mathbf{r}_i - \mathbf{r}_j\|),$$
$$\alpha_i\beta_j \in \mathcal{P}_k, \quad \forall i = 1, 2, \ldots, |V_{\mathcal{P}_k}|, \tag{5.8}$$

where w_{ij} denotes a weight function assigned to each atomic pair, with $w_{ij} = 1$ for atomic rigidity or $w_{ij} = q_j$ for atomic charge.

To quantify centrality for the entire MWCS $G(V_{\mathcal{P}_k}, E_{\mathcal{P}_k}^{\sigma,\tau,\zeta})$, we aggregate the atomic centralities as follows:

$$\mu^{k,\sigma,\tau,\nu,w} = \sum_{i=1}^{|V_{\mathcal{P}_k}|} \mu_i^{k,\sigma,\tau,\nu,w}. \tag{5.9}$$

This subgraph centrality ensures that partitions like {CN} are equivalent to {NC}. Given the 15 options for the set of weighted colored edges $\mathcal{P}_k$, we derive 15 corresponding subgraph centralities $\mu^{k,\sigma,\tau,\nu,w}$. By adjusting kernel parameters (σ, τ, ν, w), we can attain multiscale centralities for multiscale weighted colored subgraphs (MWCS) [95]. For a two-scale WCS, this yields a total of 60 descriptors for a protein. The collection of all edges $E = \{E_{\mathcal{P}_k}^{\sigma,\tau,\zeta} | k = 1, 2, \ldots, 15\}$, together with vertices V, forms a weighted graph $G(V, E)$. However, $G(V, E)$ alone has limited descriptive power in machine learning prediction. In this work, we utilize MWCSs $G(V_{\mathcal{P}_k}, E_{\mathcal{P}_k}^{\sigma,\tau,\zeta})$ and their centralities $\mu^{k,\sigma,\tau,\nu,w}$ to characterize protein electrostatics.

5.3. Machine Learning Algorithms

Various machine learning algorithms, such as linear regression (LR), random forest (RF), gradient boosting decision tree (GBDT), and deep neural network (DNN), can be employed to predict the electrostatic free energy of the PB model. LR offers a straightforward approach for approximating the mapping linearly. RF and GBDT are ensemble methods based on decision trees. RF constructs numerous uncorrelated trees and employs bootstrapping and

aggregation (i.e., bagging). GBDT utilizes gradient descent along with boosting, sequentially introducing weak learners to compensate for the errors of existing learners. DNN methods are powerful as they correct neural weights by backprapagation. However, DNN methods typically involve many weights and are prone to overfitting. They may not provide improved predictions unless the training data size is sufficiently large.

5.3.1. Generalized-Born based Gradient Boosting Decision Tree

The core principle behind the GBDT model involves leveraging initial data features and labels to construct a primary decision tree that generates initial label predictions. Following this, the residual between the actual labels and these initial predictions is used in conjunction with the original data features to create a subsequent decision tree. This tree then provides a new set of label predictions. This iterative process, which employs the residuals from previous predictions as the new labels for subsequent trees, continues recursively. The aggregate of the predictions from all the trees constitutes the final predicted labels. The model is optimized by minimizing the cost function, which measures the disparity between the initial labels and their corresponding predicted values, utilizing the gradient descent method.

In general, the predicted model for the data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$ based on K consecutive decision trees is

$$\hat{y}_K^{(i)} = \sum_{k=1}^{K} p_k(\mathbf{x}^{(i)}), i = 1, 2, \cdots, M. \tag{5.10}$$

This is the so called boosting tree procedure. We setup a loss function

$$L_k = \sum_{i=1}^{M} l_k \left(y^{(i)} - \hat{y}_k^{(i)} \right) = \sum_{i=1}^{M} \frac{1}{2} \left(y^{(i)} - \hat{y}_k^{(i)} \right)^2, \tag{5.11}$$

to minimize the loss via the gradient descent optimization of decision trees.

In our GB-based GBDT framework, the initial decision tree is the GB model, with its solvation energy serving as the first predicted labels corresponding to the PB-model based solvation energy. Subsequent decision trees are constructed using MWCS features. The loss function is influenced by various factors including the number and structure of trees and MWCS features.

5.3.2. Generalized-Born based Deep Neural Network

For our DNN model, we utilize 367 features for each protein as inputs during training, testing, and prediction phases. During prediction, we have $\Delta\hat{G}_i^{\mathrm{PB}} = \Delta G_i^{\mathrm{GB}} + \hat{y}^{(i)}$, where $\hat{y}^{(i)}$ represents the predicted value from the DNN. The value of ΔG_i^{PB} is obtained as training or testing data by solving the PB equation with MIB-PB at a refined mesh (e.g., $h = 0.2$). The DNN comprises multiple layers, and the network's weights are determined via back-propagation. We optimize the network parameters by sampling the parameter space to obtain an optimized combination of parameters for achieving the best prediction accuracy.

5.4. Results

In this section, we initially validate the selection of the MIB-PB solver [85] for label generation by comparing it with well-known PB solvers like Amber [98] and DelPhi [99]. We then demonstrate the accuracy and efficiency of the proposed PBML model in predicting electrostatic solvation free energies. The numerical computations using MIB-PB, Amber, and DelPhi are performed on an "Intel Xeon E5-2670v2" processor provided by the HPCC at Michigan State University (MSU). The machine learning computations using the "scikit-learn" Python package are performed on an "AMD EPYC 7763" processor provided by the HPC at Southern Methodist University (SMU). The electrostatic solvation free energies are computed at room temperature ($T = 298.15$ K) with dielectric constants $\epsilon_1 = 1$ and $\epsilon_2 = 80$.

5.4.1. Evaluation Metrics

Throughout this paper, we utilize the mean absolute percentage error (MAPE) and absolute relative error (ARE) to assess prediction accuracy. These metrics are defined as follows:

$$E_{\text{MAPE}} = \frac{100\%}{M} \sum_{i=1}^{M} \left| \frac{y^{(i)} - \hat{y}^{(i)}}{y^{(i)}} \right|, \quad E_{\text{ARE}} = \left| \frac{y^{(i)} - \hat{y}^{(i)}}{y^{(i)}} \right|, \tag{5.12}$$

where $y^{(i)}$ represents the ith label, i.e., the PB electrostatic solvation free energy of the ith molecule, and $\hat{y}^{(i)}$ is the corresponding predicted value. These metrics provide insights into the accuracy of the predictions.

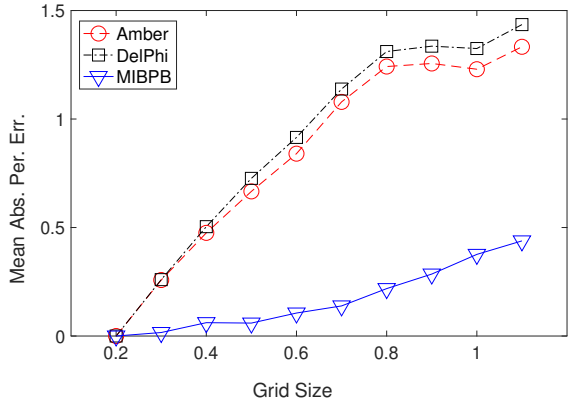
5.4.2. Convergence Comparison of the PB Solvers

We begin by conducting a convergence analysis of three PB solvers using a test set comprising 195 proteins. The aim is to determine the suitability of the MIB-PB solver for generating accurate electrostatic solvation energy labels. Note this comparison is under the assumption of a linear PB model with infinitely sharp dielectric boundary and point charges. For each protein, we compute their electrostatic solvation free energies at ten different mesh sizes, ranging from 0.2 Å to 1.1 Å. The results obtained at the finest mesh size of 0.2 Å for each PB solver serve as references to evaluate the relative errors for other mesh sizes. As shown in Fig. 5.1(a), the MAPEs using Amber and DelPhi are less than 1.5 %, but that from MIB-PB is less than 0.5% at all mesh sizes.

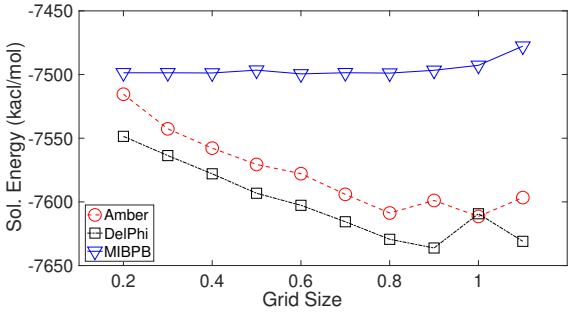
We next examine the electrostatic solvation free energies computed by the three PB solvers on two sampled proteins, 3gnw and 3owj, as illustrated in Fig. 5.1(b-c). It is observed that the energies obtained by MIB-PB do not vary significantly over the mesh refinement, whereas those computed by Amber and DelPhi exhibit more pronounced variations. Additionally, we observed that energies obtained by Amber and DelPhi approach those of

MIB-PB as the mesh is refined. These tests confirm MIB-PB as the most accurate method among these three PB solvers for computing labels for the ML models.

(a) 195 proteins



(b) protein 3gnw



(c) protein 3owj

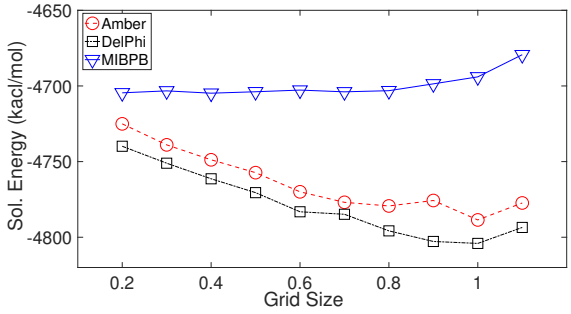


Figure 5.1: Convergence comparison among Amber, DelPhi, and MIB-PB; (a) MAPEs at ten grid sizes for Amber, DelPhi, and MIB-PB in computing the electrostatic solvation free energies of 195 test proteins. For a protein in each method, the reference value is computed at the mesh size of 0.2 Å; (b)-(c) Illustration of the electrostatic solvation free energies obtained by Amber, DelPhi, and MIB-PB at ten different mesh sizes from 0.2 to 1.1Å for proteins 3gnw and 3owj respectively.

To further investigate the convergence of the three PB solvers, we randomly select 30 proteins from the core set and plot their AREs at two mesh sizes, 0.4 Å (circles) and 0.8 Å (squares), compared with the results at 0.2 Å . These results are illustrated in Fig. 5.2.

It is evident that Amber and DelPhi exhibit a similar level of convergence, while MIB-PB returns significantly smaller errors across the selected proteins. The list of these 30 proteins is provided below: 2qmj, 3ebp, 2x8z, 3bkk, 1gpk, 1f8b, 3ge7, 3huc, 3pe2, 3mfv, 2qbr, 3imc, 1saq, 3ivg, 3b3w, 3mss, 2v7a, 2zcq, 3utu, 3u9q, 3kwa, 3gbb, 1uto, 2yge, 2iwx, 3bpc, 2pcp, 2gss, 4dew, 3nox.

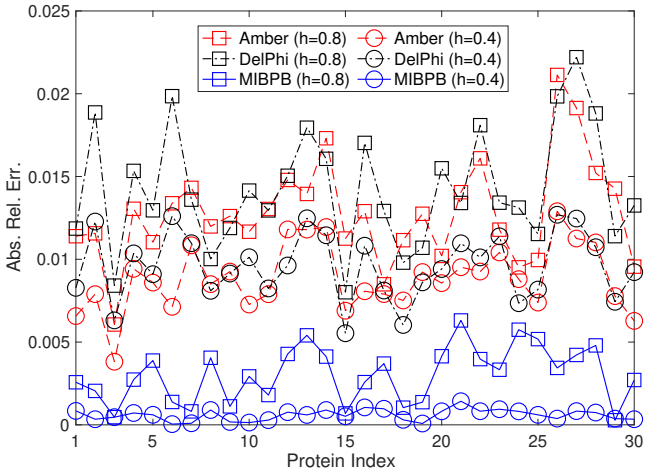


Figure 5.2: Convergence comparison among Amber, DelPhi, and MIB-PB. The graph shows Absolute relative errors of Amber (dashed lines), DelPhi (dash-dot lines), and MIB-PB (solid lines) at mesh sizes 0.4 Å (cycles) and 0.8 Å (squares) for 30 proteins. For a protein in each method, the reference value is calculated at the mesh size of 0.2 Å.

5.4.3. Comparison between Different ML models

Table 5.1: MAPEs of LR, RF, GBDT, and DNN for the test set of 195 proteins. For LR and RF, default parameters were used. For GBDT, parameters were set as follows: learning rate 0.05, number of estimators 1500, maximum depth 5. The DNN was trained with about 500 different combinations of parameters, and the final optimized choice uses a batch size of 400, an adjustable learning rate starting at 0.01, and a training duration of 3300 epochs on an architecture with 127 neurons/features in the input layer, (200, 500, 500, 500) neurons in the four hidden layers, and one neuron in the output layer.

	LR	RF	GBDT	DNN
MAPE (Training)	3.0549	0.4929	0.1553	0.1491
MAPE (Test)	1.7652	0.7040	0.4342	0.4300

After justifying the use of the MIB-PB solver to generate the labels, we proceed to apply LR, RF, GBDT, and DNN to produce corresponding learned models using the training dataset. These learned models are then utilized to predict the solvation energy for the 195 proteins in the test set. The MAPE for each learned model is summarized in Table 5.1. The results indicate that DNN outperforms the other three methods. Therefore, we choose DNN as our ML algorithm for further comprehensive training and testing of the PBML model.

5.4.4. Performance of the PBML Model

Our final PBML model is essentially the GB-based DNN model, which uses the GB core feature with an additional 367 features as outlined earlier. To understand the advantages of the model and its prediction, we conducted a comparison of its MAPE with those of Amber and DelPhi at ten mesh sizes in Fig. 5.3. Notably, while the MAPEs of Amber and DelPhi decrease significantly with mesh refinement, even at the finest mesh resolution of 0.2 Å, these methods fall short of matching the accuracy achieved by PBML, which remains unaffected by grid size once the model is trained.

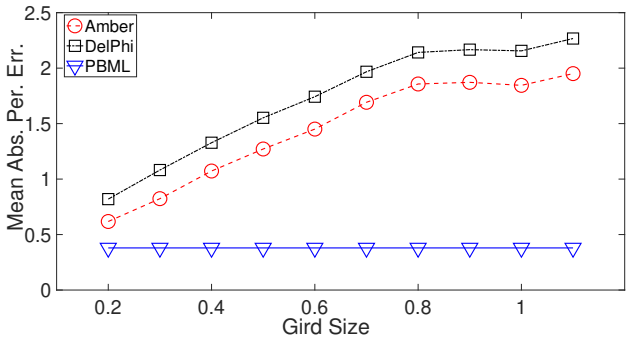


Figure 5.3: Comparison of the MAPEs of Amber, DelPhi and PBML (use result in Table 5.1 from DNN model) of the electrostatic solvation free energies of the test set at ten mesh sizes. The reference values are the results of MIB-PB at the grid size of 0.2 Å. The DNN is trained with 448 different combinations of parameters and the final optimized choice uses a batch size of 400, a learning rate of 0.005, and a training duration of 900 epochs on an architecture with 367 neurons in input layer, (500, 500, 500) neurons in the three hidden layers respectively, and one neuron in the output layer.

To further validate the accuracy and efficiency of our PBML model, we evaluate the solvation energy of 195 test proteins using both the MIB-PB solver with a mesh size of $h = 0.5$ and the PBML model trained on the dataset comprising over 4000 proteins as described before. It is important to note that the PBML model is trained using solvation energy computed using MIB-PB with a mesh size of $h = 0.2$. In this test, we consider the solvation energy results obtained from MIB-PB with $h = 0.2$ as benchmark values, comparing them with the results obtained from MIB-PB with $h = 0.5$ and those from the PBML model for the 195 proteins in the test set.

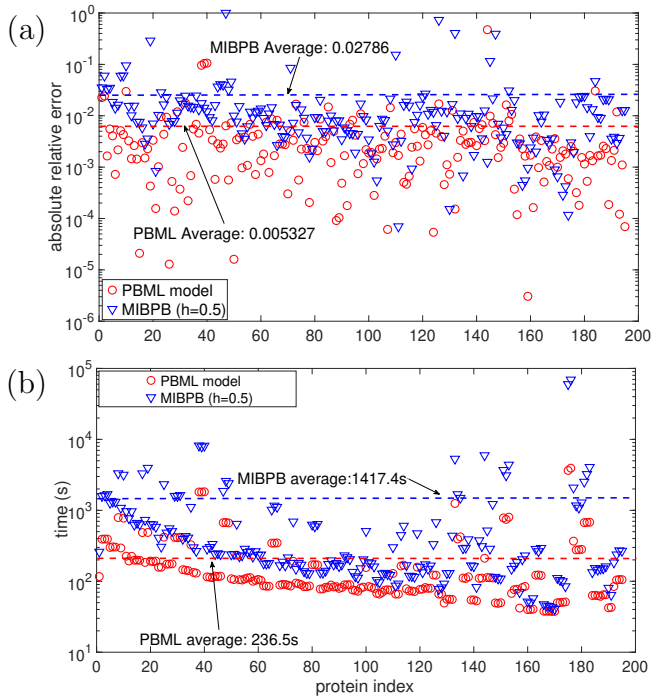


Figure 5.4: Accuracy and efficiency comparison on computing solvation energy on 195 proteins whose indices are labeled along horizontal axis using MIB-PB at $h = 0.5$ and DNN based PBML model:(a): relative error in solvation energy; (b): time. The average relative errors for PBML and MIB-PB are 0.005327 and 0.02786. The average time for PBML and MIB-PB is 236.5s and 1417.4s respectively. Note the time for the PBML includes the time to generate features but not the training time.

Figure 5.4(a) illustrates the relative error in solvation energy, where both individual samples and their average demonstrate that the PBML model outperforms the MIB-PB model at $h = 0.5$. In Figure 5.4(b), depicting elapsed time, we observe that both at the

level of individual samples and in terms of average, the PBML model exhibits significantly higher efficiency compared to the MIB-PB model at $h = 0.5$. All figures are plotted on a logarithmic scale for error and time, considering the substantial variation in results across different proteins.

5.5. Conclusion

In this chapter, we present the Poisson-Boltzmann based machine learning (PBML) model designed for predicting electrostatic solvation free energies of biomolecules, offering an efficient approach for electrostatic analysis of new molecules or conformations from molecular dynamics simulations. Our investigation begins with the selection of the most accurate PB solver for generating ML labels, identifying the second-order accurate MIB-PB solver as superior in convergence compared to DelPhi and Amber. We then employ multiscale weighted colored subgraph (MWCS) techniques for ML feature generation, facilitating the creation of highly effective low-dimensional intrinsic representations of biomolecules. Additionally, we incorporate a global core feature derived from the generalized Born (GB) model. To ensure computational efficiency, we evaluate various ML algorithms, ultimately finding that the PBML model utilizing DNN outperforms traditional grid-based PB solvers in producing electrostatics. Our numerical analysis confirms that the PBML model achieves comparable accuracy to traditional PB solvers while significantly reducing computational time.

Chapter 6

Software Dissemination

In this chapter, we demonstrate the usage of the PM MIB-PB solver as discussed in Chapter 2, the CPU-accelerated TABI-PB solver and the GPU-accelerated DSBI-PB solver as described in Chapter 3, plus the PB-ML solver discussed in Chapter 5.

- PM MIB-PB

  The Matched Interface and Boundary Poisson-Boltzmann (MIB-PB) solver coupled with the Polarizable Multipole source term from the AMOEBA force field is available on the GitHub main branch at https://github.com/gengwh/rMIB-PB. Users must utilize the MSMS software [46], or the Eulerian Solvent Excluded Surface (ESES) software maintained by Dr. Wei's group at Michigan State University, accessible at https://github.com/WeilabMSU/ESES, for molecular surface generation.

  Additionally, Tinker from https://github.com/TinkerTools/tinker is required to convert a `pdb` file obtained from the Protein Data Bank (https://www.rcsb.org/) to a multipolar `xyz` file using the `pdbxyz` package in Tinker. The users need to prepare a `.key` file, which contains or redirects to the force field information. For instance, it can be as simple as one line file:

  ```
  parameters amoebapro04.prm
  ```

  The AMOEBA force field files, i.e., `amoebapro04.prm`, can be found under the subdirectory `params` on Tinker GitHub. Once files are ready, users can call the program from Tinker by:

```
pdbxyz name_of_pdb.pdb -key name_of_pdb.key
```

Then we need to convert the `xyz` files into files containing the charge positions, charge radius, and multipole moments. Please note that the code for transferring between the `xyz` file and `pqr` file is still under maintenance, temporarily written in Python scripts `readData.py` and `tinker_to_xyzr.py`, under the subdirectory of `src/test_proteins/` on MIB-PB GitHub. For the test proteins analyzed in Chapter 2, users can test them by first adjusting the first line in `usrdata.in` file and then directly run the program, as all files mentioned above has already fully generated.

- DSBI-PB

  The GPU-accelerated Direct-Sum Boundary Integral Poisson-Boltzmann solver is a standalone application written in C and C++, compatible with platforms such as MacOS and Linux/Unix. To use the solver, the user must have the MSMS software [46] for molecular surface generation and build the Kokkos library [74] https://github.com/kokkos/kokkos for parallelism.

  To compute the solvation free energy, the user needs to prepare a `pqr` file, which can be generated using `pdb2pqr` with a `pdb` file obtained from the Protein Data Bank https://www.rcsb.org/. The solver is on GitHub https://github.com/yangxinsharon/bimpb-parallelization. User can build the solver using the following command within the subdirectory of `kokkos` on the GitHub repository:

  ```
  cmake . -DKokkosKernels_SOURCE_DIR=$HOME/repos/kokkos-kernels
  ```

  Once built, the program can be executed using the following command:

  ```
  ./bimpb_kokkos.exe PDBID.pqr den
  ```

  Here, `PDBID` represents a four-digit protein ID from the Protein Data Bank, and `den` is a parameter used to control the mesh size for boundary integral formulations. More

details on the usage of the OpenMP and MPI versions of the DSBI-PB solver can be found in the `README.md` file on GitHub.

- TABI-PB

  The Treecode Accelerated Boundary Integral solver is originated from Dr. Robert Krasny's group at University of Michigan. The Fortran version is maintained by Dr. Geng's group on GitHub https://github.com/gengwh/TABI-PB. The CPU TABI solver is compatible with platforms such as MacOS and Linux/Unix. The MPI code for the TABI-PB solver can be found on https://github.com/elyssasliheet/tabi_mpi_code, maintained by SMU graduate student Elyssa Sliheet. Similar to the DSBI-PB solver, users still require the MSMS software and need to build the MPI library for computation.

- PBML

  The Poisson-Boltzmann based Machine Learning Model can be found on GibHub github.com/yangxinsharon/PB-ML under subdirectory `saved_model`. The coefficients of the DNN are stored in a standard file and a python script, which prepares the features, assembles the DNN, and returns the electrostatic solvation energy. The user also needs to install corresponding packages for generating geometric features and GB features based on the `README.md` file. The entire training data can also be shared upon request.

Chapter 7

Summary of Contributions and Future Work

7.1. Dissertation Contributions

The main contributions of this dissertation are outlined as follows.

- In Chapter 2, we introduce a Polarizable Multipole Poisson-Boltzmann (PM-PB) model, which merges the AMOEBA force field with a linear PB equation. To handle interface conditions effectively, we integrate this PM-PB model with a regularized Matched Interface and Boundary PB (MIB-PB) solver. We implement the boundary conditions, numerical approaches incorporating surface potential gradients, and Kirkwood analytical solutions within the MIB-PB solver framework. This combined approach yields improved PB simulation outcomes with a more precise and realistic source term. Additionally, we propose a sophisticated PM polarizable multipole nonlinear PB (PM-NPB) model for a future study, given that PM source models enhance electrostatic interactions, leading to significant nonlinear effects in the Boltzmann term. Overall, these methodologies serve as a valuable tool for simulating biomolecules accurately, especially given their capability to capture the potent dielectric effects arising from the multipolar and polarizable charge density distribution.

- In Chapter 3, we introduce two parallelized solvers for solving the Boundary Integral Poisson-Boltzmann (BI-PB) equations: the Direct-Sum Boundary Integral (DSBI) PB solver, developed using KOKKOS on GPUs, and the Treecode-Accelerated Boundary Integral (TABI) PB solver, developed by our collaborators using the Message Passing Interface (MPI) on CPUs. These solvers are evaluated for their parallel performance in

solving the BI-PB equations on selected proteins critical in the context of COVID-19 transmission, treatment, and prevention. The parallelization schemes implemented in this chapter substantially enhance computational efficiency for macromolecular simulations. Additionally, we determine a threshold value, denoted as $n_b = 250,000$, under current hardware conditions on ManeFrameIII, provided by O'Donnell Data Science and Research Computing Institute at Southern Methodist University, which serves as a guideline for selecting the appropriate solver. Based on this threshold, we provide optimization recommendations: the GPU-accelerated DSBI solver is preferable when the number of boundary elements is below 250,000, while the MPI-based TABI solver is more suitable for problems exceeding this threshold. These solvers offer the opportunity to handle various types of biomolecules more efficiently.

- In Chapter 4, we delve into the distinctive features of the Eulerian Solvent Excluded Surface (ESES) software for generating both Eulerian and Lagrangian surfaces. We conduct a comprehensive numerical evaluation of the surface discretization quality under both frameworks using two recently developed Poisson-Boltzmann (PB) solvers: a Cartesian-based MIB-PB solver and a Lagrangian TABI-PB solver. Our numerical analysis reveals that, facilitated by ESES, both solvers achieve the desired convergence when utilizing the Eulerian and Lagrangian representations of the molecular surface generated by ESES. As a valuable tool for generating two types of meshes, ESES enhances the effectiveness of both finite difference based and boundary element based PB solvers.

- In Chapter 5, we introduce a Poisson-Boltzmann based Machine Learning (PB-ML) model aimed at predicting electrostatic solvation free energies of biomolecules. We meticulously evaluate various PB solvers to pinpoint the most accurate one, e.g., MIB-PB for generating ML labels. Leveraging the Multiscale Weighted Colored Subgraph (MWCS) technique and the Generalized Born (GB) model, we generate ML

features. We test several ML algorithms, including Linear Regression (LR), Random Forest (RF), Gradient Boosting Decision Tree (GBDT), and Deep Neural Network (DNN). The culmination of our efforts results in a PBML model, trained on over 4000 biomolecules using the DNN architecture, which exhibits superior efficiency and accuracy in electrostatics prediction compared to traditional grid-based PB solvers. This approach provides an efficient ML-based tool that attains a comparable level of accuracy in solving the PB equation while significantly reducing computation time.

The contents of this dissertation are primarily drawn from the following publications:

- **Yang, X.**, Sliheet, E., Iriye, R., Reynolds, D., and Geng, W. Optimized parallelization of boundary integral Poisson-Boltzmann solvers. *Computer Physics Communications* (2024), 299, 109125.
- Chen, J., Xu, Y., **Yang, X.**, Cang, Z., Geng, W. and Wei, G. W. Poisson-Boltzmann based machine learning (PBML) model for electrostatic analysis. *Biophysical Journal* (2024), S0006—3495(24)00107—3.
- Ullah, S. A., **Yang, X.**, Jones, B., Zhao, S., Geng, W. and Wei, G. W. Bridging Eulerian and Lagrangian Poisson–Boltzmann solvers by ESES. *Journal of Computational Chemistry* (2024), 45(6), 306-320.
- **Yang, X.**, Zhao, S. and Geng, W. A regularized matched interface and boundary method (MIB) for solving polarizable multipole Poisson-Boltzmann model. In preparation (2024).

7.2. Future Work

- As explored in Chapter 2, even though our PM-PB model is promising, it is not yet fully established. Our near-future endeavors involve determining the radius parameter a for

setting boundary conditions. Looking further ahead, our long-term plans encompass the development of the PM-NPB model. Once one of these solvers reaches full fruition, we can leverage this updated MIB-PB solver, coupled with the AMOEBA force field PM source term, to generate more accurate labels for training our new PBML model.

- As detailed in Chapter 3, regarding the DSBI solver, we have effectively incorporated the preconditioning scheme. Nonetheless, the recurring LU decomposition on GPU presents a challenge in parallel computing applications. Our ongoing endeavors revolve around exploring this aspect using Kokkos within our implementation.

- As confirmed in Chapter 4, ESES offers a bridge between finite difference based PB solvers and boundary element based PB solvers. Given the complexity of addressing the NPB equation within the boundary element framework, the only existing nonlinear boundary element approach is a hybrid method that tackles the nonlinear term through finite difference discretization on 3D meshes [100]. Leveraging our validated ESES, we can proceed to develop such a hybrid PB solver.

- As presented in Chapter 5, we did not initially prioritize feature engineering, which could impact the efficiency of our trained model due to the wide range of values these features can assume, ranging from less than 1 to over a million. Our future work involve studying and using the features more thoroughly to optimize model performance. One of near-future approaches is to apply the Kernel-Independent Treecode (KITC) based on barycentric Lagrange interpolation [101] to generate electrostatic features.

Appendix A

Appendix

A.1. Differentiation with respect to Cartesian coordinates

Scalar fields and the components of vector fields are functions that rely on spatial coordinates and are liable to be differentiated concerning these coordinates. A practical means of consolidating the results of such differentiation processes is symbolized by the vector operator ∇ (nabla), which is formally treated as a vector. Within a Cartesian coordinate framework, the definition of the nabla operator is as follows:

$$\nabla = \boldsymbol{i}\frac{\partial}{\partial x} + \boldsymbol{j}\frac{\partial}{\partial y} + \boldsymbol{k}\frac{\partial}{\partial z}, \tag{1.1}$$

where $\boldsymbol{i}$,$\boldsymbol{j}$, and $\boldsymbol{k}$ are the unit vectors along the x-, y-, and z-axes, respectively. When restricting to scalar and vector fields, we encounter three potential types of differentiations utilizing this nabla operator:

$$\nabla\phi = \nabla\phi, \tag{1.2}$$

$$\nabla \cdot \boldsymbol{A} = \operatorname{div} \boldsymbol{A}, \tag{1.3}$$

$$\nabla \times \boldsymbol{A} = \operatorname{curl} \boldsymbol{A}, \tag{1.4}$$

where ϕ is a scalar field and $\boldsymbol{A}$ is a vector field. The gradient and the curl or rotation are vector fields, the divergence is a scalar field. These equations can be written explicitly as:

$$\nabla\phi = \boldsymbol{i}\frac{\partial\phi}{\partial x} + \boldsymbol{j}\frac{\partial\phi}{\partial y} + \boldsymbol{k}\frac{\partial\phi}{\partial z}, \tag{1.5}$$

$$\nabla \cdot \boldsymbol{A} = \frac{\partial}{\partial x}(\boldsymbol{A} \cdot \boldsymbol{i}) + \frac{\partial}{\partial y}(\boldsymbol{A} \cdot \boldsymbol{j}) + \frac{\partial}{\partial z}(\boldsymbol{A} \cdot \boldsymbol{k}) = \frac{\partial}{\partial x}A_x + \frac{\partial}{\partial y}A_y + \frac{\partial}{\partial z}A_z, \tag{1.6}$$

$$\nabla \times \boldsymbol{A} = \boldsymbol{i}\left(\frac{\partial}{\partial y}A_z - \frac{\partial}{\partial z}A_y\right) + \boldsymbol{j}\left(\frac{\partial}{\partial z}A_x - \frac{\partial}{\partial x}A_z\right) + \boldsymbol{k}\left(\frac{\partial}{\partial x}A_y - \frac{\partial}{\partial y}A_x\right). \tag{1.7}$$

The operator $\nabla \cdot \nabla$ is commonly referred to as the del operator or the Laplace operator. In a Cartesian coordinate system, it is expressed as:

$$\Delta\phi = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\phi. \tag{1.8}$$

When applying the gradient operation to a function dependent on the distance between two points or the radius vector $\boldsymbol{r}$, it's important to specify which endpoint of the vector the differentiation occurs at. This distinction can be illustrated by considering r as the distance between two points P and Q with radius vectors $\boldsymbol{r}_P$ and $\boldsymbol{r}_Q$, where $\boldsymbol{r}$ points from P to Q. In this scenario, we have:

$$\boldsymbol{r} = \boldsymbol{r}_Q - \boldsymbol{r}_P. \tag{1.9}$$

Differentiating a function $f(r)$, where $f(r)$ is any differentiable function of r, leads to:

$$\nabla f(r) = \nabla f(|\boldsymbol{r}|) = \nabla f\left(|\boldsymbol{r}_Q - \boldsymbol{r}_P|\right), \tag{1.10}$$

or:

$$\nabla f(r) = \nabla_Q f(r), \tag{1.11}$$

where the subscript Q indicates that the differentiation occurs at the endpoint Q of the vector $\boldsymbol{r}$. However, in certain situations, differentiation must be performed at the point P. Since we can express:

$$\frac{\partial}{\partial x_Q}\left|\boldsymbol{r}_Q - \boldsymbol{r}_P\right| = -\frac{\partial}{\partial x_P}\left|\boldsymbol{r}_Q - \boldsymbol{r}_P\right|, \tag{1.12}$$

and analogous equations for the derivatives with respect to y and z, we have:

$$\nabla_Q f(r) = -\nabla_P f(r). \tag{1.13}$$

This formula is applicable when transforming the solution of Poisson's equation, which provides the potential at the origin, to the potential at an arbitrary point. Additionally, applying the gradient to the function $\phi = r^n$, where $r^2 = x^2 + y^2 + z^2$, yields:

$$\nabla r^n = n r^{n-1} \nabla r = n r^{n-2} \boldsymbol{r}. \tag{1.14}$$

A specific application of this general equation for $n = -1$ is:

$$\nabla \frac{1}{r} = -\frac{1}{r^3} \boldsymbol{r}. \tag{1.15}$$

When the nabla operator is applied to a product $\phi \boldsymbol{A} \cdot \boldsymbol{B}$, we utilize Leibniz' rule for differentiating products to obtain:

$$\nabla \left(\phi \boldsymbol{A} \cdot \boldsymbol{B} \right) = (\boldsymbol{A} \cdot \boldsymbol{B}) \nabla \phi + \phi \nabla (\boldsymbol{A} \cdot \boldsymbol{B}). \tag{1.16}$$

For the specific scenario where $\boldsymbol{B} = \boldsymbol{r}$, the radius vector of each point, and $\boldsymbol{A} = \boldsymbol{m}$, a constant vector, the last term of Eq. (1.16) can be further simplified:

$$\nabla \boldsymbol{m} \cdot \boldsymbol{r} = \boldsymbol{i} \frac{\partial}{\partial x} \left(m_x x \right) + \boldsymbol{j} \frac{\partial}{\partial y} \left(m_y y \right) + \boldsymbol{k} \frac{\partial}{\partial z} \left(m_z z \right), \tag{1.17}$$

where m_x, m_y, and m_z represent the three components of $\boldsymbol{m}$ in the x, y, and z directions. As the constant factors remain unaffected by differentiation, we have:

$$\nabla \boldsymbol{m} \cdot \boldsymbol{r} = \boldsymbol{i} m_x + \boldsymbol{j} m_y + \boldsymbol{k} m_z = \boldsymbol{m}. \tag{1.18}$$

Another differentiation of a product is utilized:

$$\operatorname{div}(\phi \boldsymbol{A}) = \frac{\partial}{\partial x}(\phi A_x) + \frac{\partial}{\partial y}(\phi A_y) + \frac{\partial}{\partial z}(\phi A_z), \tag{1.19}$$

and consequently:

$$\operatorname{div}(\phi \boldsymbol{A}) = \phi \left\{ \frac{\partial}{\partial x} A_x + \frac{\partial}{\partial y} A_y + \frac{\partial}{\partial z} A_z \right\} + A_x \frac{\partial \phi}{\partial x} + A_y \frac{\partial \phi}{\partial y} + A_z \frac{\partial \phi}{\partial z}, \tag{1.20}$$

or:

$$\operatorname{div}(\phi \boldsymbol{A}) = \phi \operatorname{div} \boldsymbol{A} + \boldsymbol{A} \cdot \nabla \phi. \tag{1.21}$$

A.2. Differentiation with respect to spherical coordinates

In many problems of electrostatics it is advantageous to use spherical coordinates r, θ, φ instead of the Cartesian coordinates x, y, z used so far. The transformation rules from spherical to Cartesian coordinates are derived easily from geometrical considerations:

$$\left.\begin{aligned} x &= r \sin\theta \cos\varphi \\ y &= r \sin\theta \sin\varphi \\ z &= r \cos\theta \end{aligned}\right\}, \tag{1.22}$$

with $0 \leqslant r < \infty, 0 \leqslant \theta \leqslant \pi, 0 \leqslant \varphi < 2\pi$. The inverse transformation is:

$$\left.\begin{aligned} r^2 &= x^2 + y^2 + z^2 \\ \operatorname{tg}\theta &= \tfrac{1}{z}\sqrt{x^2 + y^2} \\ \operatorname{tg}\varphi &= y/x \end{aligned}\right\}. \tag{1.23}$$

The unit vectors $\boldsymbol{e}_r, \boldsymbol{e}_\theta$, and $\boldsymbol{e}_\varphi$ in the spherical coordinate system are found by considering the planes of constant r, of constant θ, and of constant φ. They are represented as:

$$\left.\begin{aligned} \boldsymbol{e}_r &= \sin\theta\cos\varphi\boldsymbol{i} + \sin\theta\sin\varphi\boldsymbol{j} + \cos\theta\boldsymbol{k} \\ \boldsymbol{e}_\theta &= \cos\theta\cos\varphi\boldsymbol{i} + \cos\theta\sin\varphi\boldsymbol{j} - \sin\theta\boldsymbol{k} \\ \boldsymbol{e}_\varphi &= \quad -\sin\varphi\boldsymbol{i} \quad + \cos\varphi\boldsymbol{j} \end{aligned}\right\}. \tag{1.24}$$

and for the inverse transformation:

$$\left.\begin{aligned} \boldsymbol{i} &= \sin\theta\cos\varphi\boldsymbol{e_r} + \cos\theta\cos\varphi\boldsymbol{e}_\theta - \sin\varphi\boldsymbol{e}_\varphi \\ \boldsymbol{j} &= \sin\theta\sin\varphi\boldsymbol{e_r} + \cos\theta\sin\varphi\boldsymbol{e}_\theta + \cos\varphi\boldsymbol{e}_\varphi \\ \boldsymbol{k} &= \quad \cos\theta\boldsymbol{e_r} \quad - \sin\theta\boldsymbol{e}_\theta \end{aligned}\right\}. \tag{1.25}$$

An example of the use of spherical coordinates is the following. Given that:

$$E = \frac{e}{r^3}\boldsymbol{r},$$

we need to calculate $\oiint \boldsymbol{E}\cdot\boldsymbol{dS}$ for an arbitrary surface enclosing the charge e. Considering the surface of a sphere with radius R, we have:

$$\oiint \boldsymbol{E}\cdot\boldsymbol{dS} = \int_0^{2\pi}\int_0^{\pi} \frac{e}{R^3}\boldsymbol{R}\cdot(R\ \mathrm{d}\theta\boldsymbol{e}_\theta)\times(R\sin\theta\mathrm{d}\varphi\boldsymbol{e}_\varphi) = \int_0^{2\pi}\int_0^{\pi} e\sin\theta\boldsymbol{e}_r\cdot\boldsymbol{e}_\theta\times\boldsymbol{e}_\varphi\mathrm{d}\theta\mathrm{d}\varphi, \tag{1.26}$$

or:

$$\oiint \boldsymbol{E}\cdot\boldsymbol{dS} = e\int_0^{2\pi\pi}\int_0^{\pi}\sin\theta\mathrm{d}\theta\mathrm{d}\varphi = 2\pi e\int_0^{\pi}\sin\theta\mathrm{d}\theta = 4\pi e. \tag{1.27}$$

For the gradient of ϕ in the direction of $\boldsymbol{e}_r, \boldsymbol{e}_\theta$, and $\boldsymbol{e}_\varphi$, respectively, we derive:

$$\left.\begin{aligned}(\nabla\phi)_r &= (\nabla\phi)\cdot\boldsymbol{e}_r = \tfrac{\partial\phi}{\partial r}\\ (\nabla\phi)_\theta &= (\nabla\phi)\cdot\boldsymbol{e}_\theta = \tfrac{1}{r}\tfrac{\partial\phi}{\partial\theta}\\ (\nabla\phi)_\varphi &= (\nabla\phi)\cdot\boldsymbol{e}_\varphi = \tfrac{1}{r\sin\theta}\tfrac{\partial\phi}{\partial\varphi}\end{aligned}\right\}. \tag{1.28}$$

Then, the Laplacian of ϕ is:

$$\Delta\phi = \left(\frac{\partial}{\partial r}+\frac{2}{r}\right)\frac{\partial\phi}{\partial r}+\frac{1}{r^2}\left(\frac{\partial}{\partial\theta}+\cot\theta\right)\frac{\partial\phi}{\partial\theta}+\frac{1}{r^2\sin^2\theta}\frac{\partial^2\phi}{\partial\varphi^2}, \tag{1.29}$$

The first term of this Laplacian expression can also be written as:

$$\left(\frac{\partial}{\partial r}+\frac{2}{r}\right)\frac{\partial\phi}{\partial r} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\phi}{\partial r}\right) = \frac{1}{r}\frac{\partial^2}{\partial r^2}(r\phi), \tag{1.30}$$

An analogous form of the second term is:

$$\frac{1}{r^2}\left(\frac{\partial}{\partial\theta}+\operatorname{cotg}\theta\right)\frac{\partial\phi}{\partial\theta} = \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\phi}{\partial\theta}\right). \tag{1.31}$$

A.3. Tensor multiplications

The summation of elements along the principal diagonal corresponds to the inner product of two vectors $\boldsymbol{w}$ and $\boldsymbol{v}$. This operation is also known as the contraction of the tensor product. Generally, contraction is represented by a dot between the involved symbols, resulting in a summation over the adjacent indices. For two vectors, this is expressed as:

$$\boldsymbol{w}\cdot\boldsymbol{v} = \sum_i w_i v_i, \tag{1.32}$$

and for a tensor and a vector:

$$(\boldsymbol{T} \cdot \boldsymbol{v})_i = \sum_j t_{ij} v_j. \tag{1.33}$$

Similarly, for two tensors $\boldsymbol{T}$ and $\boldsymbol{S}$:

$$(\boldsymbol{T} \cdot \boldsymbol{S})_{ik} = \sum_j t_{ij} s_{jk}, \tag{1.34}$$

and:

$$\boldsymbol{T} : \boldsymbol{S} = \sum_{ij} t_{ij} s_{ji}, \tag{1.35}$$

as after the initial contraction, i and k become adjacent indices.

A.4. Electric dipoles and multipoles

In this context, we briefly introduce the dipole moment and the quadrupole moment as detailed in the study by Böttcher [63].

To begin with, the electric moment of a system of point charges e_i relative to a fixed origin is defined as:

$$\mathbf{m} = \sum_i e_i \mathbf{r}_i, \tag{1.36}$$

where each $\mathbf{r}_i$ is the radius vector from the origin to each e_i. When the origin is displaced over a distance $\mathbf{r}_0$, the change in $\mathbf{m}$ is given by:

$$\Delta\mathbf{m} = -\sum_i e_i \mathbf{r}_0 = -\mathbf{r}_0 \sum_i e_i. \tag{1.37}$$

In this scenario, Eq. (1.36) can be expressed differently. Introducing the electric centers of gravity for the positive and negative charges, the centers are defined by:

$$\begin{aligned} \sum_{\text{positive}} e_i \mathbf{r}_i &= \mathbf{r}_\text{p} \sum_{\text{positive}} e_i = \mathbf{r}_\text{p} e, \\ \sum_{\text{negative}} e_i \mathbf{r}_i &= \mathbf{r}_\text{n} \sum_{\text{negative}} e_i = -\mathbf{r}_\text{n} e, \end{aligned} \tag{1.38}$$

where $\mathbf{r}_\text{p}$ and $\mathbf{r}_\text{n}$ are the radius vectors from the origin to these centers of gravity respectively, and e is the total positive charge. Thus, for a net zero charge system, Eq. (1.36) can be written as:

$$\mathbf{m} = (\mathbf{r}_p - \mathbf{r}_n)e = \mathbf{l}e, \tag{1.39}$$

where the difference $\mathbf{r}_p - \mathbf{r}_n$ is the distance between centers of gravity, represented by a vector $\mathbf{l}$ pointing from the negative center to the positive center, as shown in Fig. 1.1.

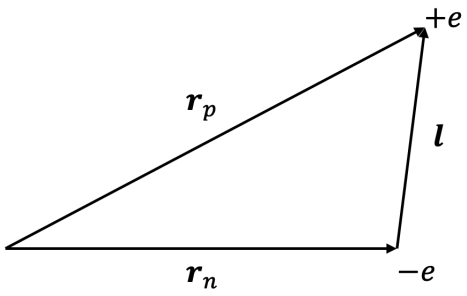


Figure 1.1: A simple electric dipole consisting of one positive charge and one negative charge at a distance $\mathbf{l}$.

In pyhsical sense, a simple system consisting of only one positive charge $+e$ and one negative charge $-e$ at a distance $\mathbf{l}$, is called an electric dipole, and its moment is $\mathbf{l}e$. Hence, the "electric moment" of a system of zero net charge is generally referred to as the "electric dipole moment" of the system. In mathematical abstraction, the "ideal dipole", often referred to as the "point dipole", can be derived by substituting $\mathbf{l}$ with $\mathbf{l}/n$ and e with en, and letting n approach to infinity.

Many neutral molecules exhibit non-ideal electric dipole moments because the centers of positive and negative charge distributions do not align. In addition to these permanent dipole moments, an induced dipole moment arises when a particle is exposed to an external electric field. This external field causes the positive and negative charges within the particle to separate, resulting in "polarization" of the particle. Generally, induced dipoles can be treated as ideal; however, permanent dipoles may not be treated as ideal, especially when calculating the field at molecular distances.

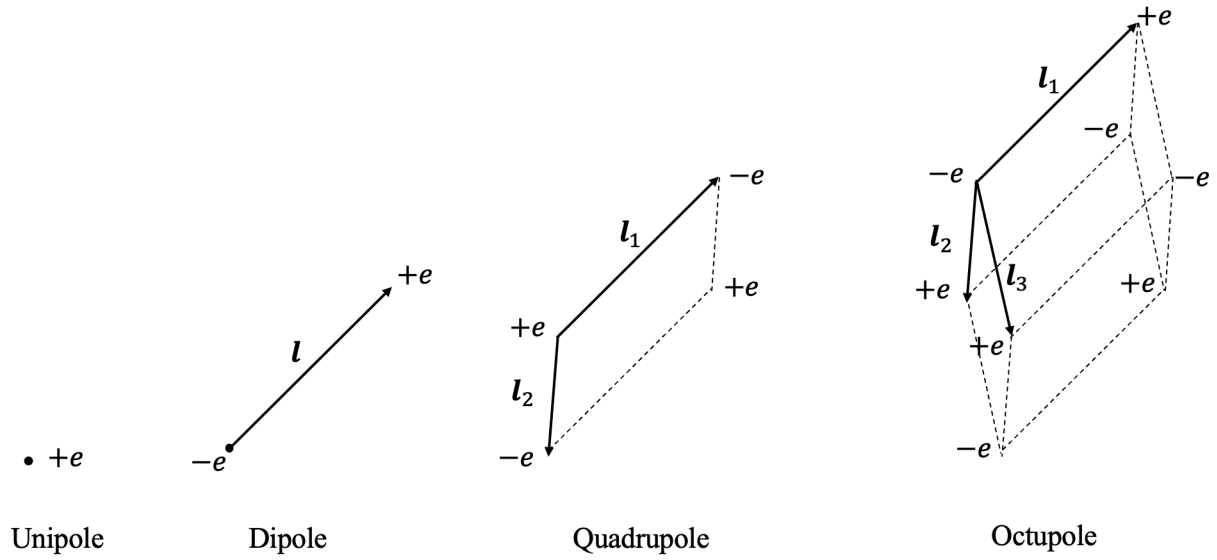


Figure 1.2: Electric multipoles.

To characterize the potential resulting from a charge distribution, it is beneficial to expand upon the notion of a dipole to encompass the broader concept of a multipole. As illustrated in Fig. 1.2, the configuration of two equal dipoles, aligned along the same axis but oriented oppositely, is called an axial quadrupole. Likewise, an axial octupole is formed as a combination of two quadrupoles etc. The quadrupole strength is determined by the product of the dipole moment $\mathbf{m}$ along a specific axis, denoted as a scaler m, and the distance between the dipoles. Therefore, similar to the definition of an ideal dipole, an ideal axial quadrupole is obtained by replacing the dipole moment m by mn and the distance $s = |\mathbf{l}_2|$ by s/n as n approaches to infinity.

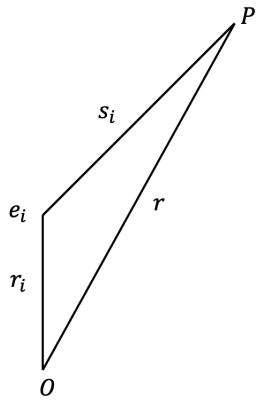


Figure 1.3: Potential at P outside a sphere encompassing all the charges, i.e., $r \gg r_i$.

Our analysis extends beyond charge distributions with axial symmetry. Consider a system of point charges e_i at distances $r_i(x_i, y_i, z_i)$ from an origin O. We compute the potential generated by this system at a point P with coordinates x, y, z, as shown in Fig. 1.3. It is assumed that the position P is outside a sphere encompassing all the charges. Let s_i represent the distance of a charge e_i to P. Then, the potential at P due to this charge system is expressed as:

$$\phi(P) = \sum_i \frac{e_i}{s_i}. \tag{1.40}$$

We expand $1/s_i$ in a Taylor series where the differentiations at the origin O:

$$\begin{aligned} \frac{1}{s_i} = \frac{1}{r} + x_i \frac{\partial}{\partial x}\left(\frac{1}{r}\right) + y_i \frac{\partial}{\partial y}\left(\frac{1}{r}\right) + z_i \frac{\partial}{\partial z}\left(\frac{1}{r}\right) + \\ + \frac{1}{2}\left[x_i^2 \frac{\partial^2}{\partial x^2}\left(\frac{1}{r}\right) + 2x_i y_i \frac{\partial^2}{\partial x \partial y}\left(\frac{1}{r}\right) + \cdots\right] + \cdots. \end{aligned} \tag{1.41}$$

Thus, when the differentiations are made at P (see Appendix A.1):

$$\begin{aligned} \phi(P) = \sum_i \frac{e_i}{r} - \sum_i \left[e_i x_i \frac{\partial}{\partial x}\left(\frac{1}{r}\right) + e_i y_i \frac{\partial}{\partial y}\left(\frac{1}{r}\right) + e_i z_i \frac{\partial}{\partial z}\left(\frac{1}{r}\right)\right] + \\ + \frac{1}{2}\sum_i \left[e_i x_i^2 \frac{\partial^2}{\partial x^2}\left(\frac{1}{r}\right) + 2e_i x_i y_i \frac{\partial^2}{\partial x \partial y}\left(\frac{1}{r}\right) + \cdots\right] - \cdots. \end{aligned} \tag{1.42}$$

The first term of this series represents the potential induced at point P by a single charge $e = \sum_i e_i$ located at the origin, and the second term represents the potential at P caused by a point dipole $\mathbf{m} = \sum_i e_i \mathbf{r}_i$. Similar to how the vectorial dipole moment $\mathbf{m}$ is defined, the quadrupole moment $\mathbf{Q}$ is defined as a tensor:

$$\boldsymbol{Q} = \frac{1}{2!} \sum_i e_i \mathbf{r}_i \mathbf{r}_i, \tag{1.43}$$

and the octupole moment $\boldsymbol{U}$ is defined as a tensor of the third degree:

$$\boldsymbol{U} = \frac{1}{3!} \sum_i e_i \mathbf{r}_i \mathbf{r}_i \mathbf{r}_i. \tag{1.44}$$

Hence, Eq. (1.42) can be written in terms of the multipole moments:

$$\phi(P) = e\frac{1}{r} - \boldsymbol{m} \cdot \nabla \frac{1}{r} + \boldsymbol{Q} : \nabla\nabla \frac{1}{r} - \boldsymbol{U} \vdots \nabla\nabla\nabla \frac{1}{r} + \cdots. \tag{1.45}$$

It has been demonstrated that the potential resulting from an ideal multipole exhibits the same structure as the corresponding term in the general multipole expansion for the potential expressed by Eq. (1.45) [63].

A.5. Electric potential and energy for multipoles

In this context, we present the derivation of the potential and energy associated with a multipole up to quadrupole moments [63] [102].

In order to compare the quadrupole moments of diverse charge distributions, such as molecules, it is imperative to standardize them in advance. This standardization process guarantees a uniform basis for comparison across different systems. The common normal-

ization defines a traceless quadrupole moment as:

$$\boldsymbol{Q}_{\text{norm}} = \frac{1}{2}\sum_i e_i \left(3\mathbf{r}_i\mathbf{r}_i - r_i^2\boldsymbol{I}\right), \tag{1.46}$$

where $\boldsymbol{I}$ is the identity tensor and $r_i^2 = x_i^2 + y_i^2 + z_i^2$, such that $\text{Tr}\left(\boldsymbol{Q}_{\text{norm}}\right) = 0$. This quadrupole moment tensor comprises nine components, yet it exhibits symmetry, and, along with the traceless property, this implies that there are only five independent non-zero quadrupole components. When this traceless quadrupole is used, denoted as $\boldsymbol{Q} = \boldsymbol{Q}_{\text{norm}}$, the quadrupole contribution to the potential in Eq. (1.45) is updated to $\frac{1}{3}\mathbf{Q} : \nabla\nabla\frac{1}{r}$. Thus, our potential becomes:

$$\phi(P) = e\frac{1}{r} - \boldsymbol{m}\cdot\nabla\frac{1}{r} + \frac{1}{3}\boldsymbol{Q} : \nabla\nabla\frac{1}{r}. \tag{1.47}$$

To establish an expression for the potential energy of an arbitrary charge distribution within an external field in terms of the multipole moments of this charge distribution, we can designate a volume enclosing the charge distribution and denote the potential due to the charges outside this volume by ϕ. Inside the volume, the potential ϕ satisfies Laplace's equation, given that its sources reside exclusively outside the designated volume. As the potential is defined as the potential energy of a unit charge in the field, the potential energy of the system of n charges e_i is then given by:

$$V = \sum_{i=1}^{n} e_i\phi\left(\mathbf{r}_i\right), \tag{1.48}$$

where $\mathbf{r}_i$ is the point at which the i-th charge e_i is situated. Similarly, all potentials $\phi\left(\mathbf{r}_i\right)$ can now be developed in a Taylor series around the origin O. This leads to:

$$V = \sum_{i=1}^{n} e_i\left[\phi(O) + \mathbf{r}_i\cdot\nabla\phi(O) + \frac{1}{2}\mathbf{r}_i\mathbf{r}_i : \nabla\nabla\phi(O) + \cdots\right], \tag{1.49}$$

where the gradients are taken at the origin. The coefficients associated with $\phi(O)$ and its subsequent gradients now correspond to the multipole moments of the charge distribution, delineated in Eq. (1.36) and (1.43). Combined with the fact using the traceless quadrupole moment in Eq. (1.46), the potential energy corresponding to Eq. (1.47) at point P (see Appendix A.1) is thus:

$$V = e\phi(P) - \boldsymbol{m} \cdot \nabla\phi(P) + \frac{1}{3}\boldsymbol{Q} : \nabla\nabla\phi(P). \tag{1.50}$$

So far, it becomes evident that the same set of multipole moments, initially introduced to represent the potential due to a particular arrangement of charges, can be effectively utilized to depict the potential energy of these charges when influenced by an external field.

A.6. The general solution of Laplace's equation in spherical coordinates for the case of axial symmetry

Laplace's equation for the potential ϕ, $\nabla^2\phi = 0$, holds true in regions devoid of any true or apparent charge. Generally, within a charge-free region, the potential is determined by its values (or its normal derivative) at the region's boundary. Consequently, the solution of Laplace's equation for a specific region can be derived from the general solution and the potential values at the boundary, dictated by the problem's conditions. If the region extends to infinity, the behavior for large distances should be specified.

For many problems in the theory of electric polarization, a spherical coordinate system (r, θ, ϕ) proves to be appropriate. This system is particularly advantageous when the problem exhibits axial symmetry (also known as cylindrical symmetry) or spherical symmetry. In the case of axial symmetry, $\frac{\partial}{\partial\phi} = 0$, and in the case of spherical symmetry, both $\frac{\partial}{\partial\theta} = 0$ and $\frac{\partial}{\partial\phi} = 0$.

The Laplace's equation in spherical coordinates for the case of at least axial symmetry simplifies to:

$$\Delta\phi = \left(\frac{\partial}{\partial r} + \frac{2}{r}\right)\frac{\partial\phi}{\partial r} + \frac{1}{r^2}\left(\frac{\partial}{\partial\theta} + \cot\theta\right)\frac{\partial\phi}{\partial\theta} = 0. \tag{1.51}$$

This partial differential equation can be reduced to two ordinary differential equations using the method of separation of variables. We assume ϕ as:

$$\phi(r,\theta) = U(r)V(\theta). \tag{1.52}$$

Substituting Eq. (1.52) into Eq. (1.51) and multiplying by r^2/ϕ yields:

$$\frac{r^2}{U(r)}\left(\frac{\partial}{\partial r} + \frac{2}{r}\right)\frac{\partial U(r)}{\partial r} + \frac{1}{V(\theta)}\left(\frac{\partial}{\partial\theta} + \cot\theta\right)\frac{\partial V(\theta)}{\partial\theta} = 0. \tag{1.53}$$

The left-hand side comprises an r-dependent and an r-independent term, equating to a constant value of 0. Therefore, the first term's value remains constant with r. By setting this term equal to C, the separation constant, we can express it as:

$$r^2\frac{d^2U}{dr^2} + 2r\frac{dU}{dr} - CU = 0, \tag{1.54}$$

and:

$$\frac{d^2V}{d\theta^2} + \cot(\theta)\frac{dV}{d\theta} + CV = 0. \tag{1.55}$$

Both differential equations can be straightforwardly solved. Assuming U to follow the form:

$$U(r) = r^\alpha, \tag{1.56}$$

we substitute it into Eq. (1.54) to obtain:

$$\alpha(\alpha - 1) + 2\alpha - C = 0, \tag{1.57}$$

provided r does not reach the values 0 or ∞. Eq. (1.57) becomes a quadratic equation in α with solutions α_1 and α_2. Thus, we find:

$$U(r) = ar^{\alpha_1} + br^{\alpha_2}, \tag{1.58}$$

where a and b are arbitrary constants. Eq. (1.58) represents the general solution of Eq. (1.54), as it encompasses two adjustable constants.

Eq. (1.55) can be transformed into Legendre's differential equation using the substitutions $\cos(\theta) = x$ and $V(\theta) = y(x)$. Utilizing the relations:

$$\frac{d}{d\theta} = -\sin(\theta)\frac{d}{dx}, \quad \text{and} \quad \frac{d^2}{d\theta^2} = -\cos(\theta)\frac{d}{dx} + \sin^2(\theta)\frac{d^2}{dx^2},$$

we derive:

$$(1-x^2)\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + Cy = 0. \tag{1.59}$$

The general solution of this equation is presented as:

$$y(x) = AP_v(x) + BQ_v(x), \tag{1.60}$$

where P_v and Q_v represent the Legendre functions of the first and second kind, respectively. The index v is defined by $C = v(v+1)$, and A and B denote arbitrary constants. The general solution provided in Eq. (1.60) is applicable in scenarios where the z-axis ($\cos(\theta) = x = \pm 1$) is excluded from the region where the potential needs evaluation. Given that singularities are permissible at $x = \pm 1$, as neither $P_v(x)$ nor $Q_v(x)$ for all v values are finite and continuous at these points, we can utilize the general solution:

$$y(x) = AP_n(x), \tag{1.61}$$

where n represents an integer and P_n denotes a Legendre polynomial. Consequently, we also have:

$$C = n(n+1), \tag{1.62}$$

leading to the subsequent solution of Eq. (1.57):

$$\alpha_1 = n, \quad \alpha_2 = -(n+1). \tag{1.63}$$

Combining our findings from Eq. (1.58), (1.61), and (1.63), we can express:

$$\phi_n(r,\theta) = UV = (a_n r^n + b_n r^{-(n+1)}) P_n(\cos(\theta)). \tag{1.64}$$

Here, $a_n = Aa$ and $b_n = Ab$; the index n is appended to underscore that this solution corresponds to a specific choice for the integer n.

Given the linearity and homogeneity of Eq. (1.51), the solutions are linearly independent. Consequently, any linear combination of solutions akin to Eq. (1.64) will likewise serve as a solution, adhering to the continuity requisites.

Therefore, the general solution may be established as a linear combination of solutions in the form Eq. (1.64) with undetermined coefficients a_n and b_n:

$$\phi(r,\theta) = \sum_{n=0}^{\infty} (a_n r^n + b_n r^{-(n+1)}) P_n(\cos(\theta)). \tag{1.65}$$

The determination of coefficients a_n and b_n arises from the potential values at the boundary, or as applicable, from the asymptotic behavior at infinity.

A.7. Legendre polynomials

The Legendre polynomials $P_n(\cos\theta)$ are defined by the equation:

$$F(x,z) = \frac{1}{\sqrt{1-2xz+z^2}} = \sum_{n=0}^{\infty} P_n(x) z^n, \tag{1.66}$$

where $x = \cos\theta$ and $0 < z < 1$. The function $F(x,z)$ is referred to as the generating function. When a polynomial of high order n is required, it is more convenient to use the Rodrigues' equation:

$$P_n(x) = \frac{1}{2^n n!} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \left(x^2-1\right)^n. \tag{1.67}$$

The results of the computations for the first eight polynomials are presented in the following table:

$$\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
P_2(x) &= \frac{1}{2}\left(3x^2-1\right) \\
P_3(x) &= \frac{1}{2}\left(5x^3-3x\right) \\
P_4(x) &= \frac{1}{8}\left(35x^4-30x^2+3\right) \\
P_5(x) &= \frac{1}{8}\left(63x^5-70x^3+15x\right) \\
P_6(x) &= \frac{1}{16}\left(231x^6-315x^4+105x^2-5\right) \\
P_7(x) &= \frac{1}{16}\left(429x^7-693x^5+315x^3-35x\right)
\end{aligned} \tag{1.68}$$

A.8. The solution of Laplace's equation due to a point dipole

In many instances, it is recommended to choose a value for a that is approximately equivalent to what is commonly defined as the "molecular radius". Onsager [103] used the relation

$$\frac{4}{3}\pi N a^3 = 1. \tag{1.69}$$

In order to compute the reaction field, it is essential to determine the potential within the cavity resulting from both the dipole itself and the interaction of the dipole with the surrounding dielectric. Here, the spherical polar coordinates r, θ, and φ is used here by taking the center of the dipole as the origin of a coordinate system, and aligning the z-axis in the direction of the dipole vector. In the case of symmetry about the z-axis, the general solution of Laplace's equation is:

$$\phi_{\text{outside}} = \sum_{n=0}^{\infty} \left(A_n r^n + \frac{B_n}{r^{n+1}} \right) \mathrm{P_n}(\cos\theta), \tag{1.70a}$$

$$\phi_{\text{inside}} = \sum_{n=0}^{\infty} \left(C_n r^n + \frac{D_n}{r^{n+1}} \right) \mathrm{P_n}(\cos\theta). \tag{1.70b}$$

The boundary conditions are:

$$\left(\phi_{\text{outside}}\right)_{r\to\infty} = 0, \tag{1.71}$$

$$\left(\phi_{\text{outside}}\right)_{r=a} = \left(\phi_{\text{inside}}\right)_{r=a}, \tag{1.72}$$

$$\epsilon_1 \left(\frac{\partial \phi_{\text{outside}}}{\partial r} \right)_{r=a} = \epsilon_2 \left(\frac{\partial \phi_{\text{inside}}}{\partial r} \right)_{r=a}. \tag{1.73}$$

The terms $\frac{D_n}{r^{n+1}} P_n(\cos\theta)$ in ϕ_{inside} are due to charges within the cavity. In this case, the only source of field lines within the cavity is the permanent dipole $\boldsymbol{\mu}$. The potential of it along the z-axis is:

$$\phi = \frac{\mu}{r^2} \cos\theta. \tag{1.74}$$

Thus all coefficients D_n are zero except D_1, which has the value $D_1 = \mu$, so we have:

$$\phi_{\text{inside}} = \sum_{n=0}^{\infty} C_n r^n P_n(\cos\theta) + \frac{\mu}{r^2} \cos\theta. \tag{1.75}$$

Because of Eq. (1.71), and the fact that the Legendre functions are linearly independent, all coefficients A_n are zero. Thus,

$$\phi_{\text{outside}} = \sum_{n=0}^{\infty} \frac{B_n}{r^{n+1}} P_n(\cos\theta). \tag{1.76}$$

Applying Eq. (1.72), and Eq. (1.73), for all values except $n \neq 1$:

$$\frac{B_n}{a^{n+1}} = C_n a^n, \tag{1.77a}$$

$$-\epsilon_1(n+1)\frac{B_n}{a^{n+2}} = \epsilon_2 n C_n a^{n-1}, \tag{1.77b}$$

which is only possible if for $n \neq 1$:

$$B_n = 0, \tag{1.78a}$$

$$C_n = 0. \tag{1.78b}$$

For $n = 1$:

$$\frac{B_1}{a^2} = C_1 a + \frac{\mu}{a^2}, \tag{1.79a}$$

$$-2\epsilon_1\frac{B_1}{a^3} = \epsilon_2 C_1 - 2\epsilon_2\frac{\mu}{a^3}. \tag{1.79b}$$

By solving these two equations, we get:

$$B_1 = \frac{3\epsilon_2}{2\epsilon_1 + \epsilon_2}\mu, \tag{1.80a}$$

$$C_1 = -\frac{2(\epsilon_1 - \epsilon_2)}{2\epsilon_1 + \epsilon_2}\frac{\mu}{a^3}, \tag{1.80b}$$

Therefore,

$$\phi_{\text{outside}} = \frac{3\epsilon_2}{2\epsilon_1 + \epsilon_2} \frac{\mu}{r^2} \cos\theta, \tag{1.81a}$$

$$\phi_{\text{inside}} = \frac{\mu}{r^2} \cos\theta - \frac{2(\epsilon_1 - \epsilon_2)}{2\epsilon_1 + \epsilon_2} \frac{\mu}{a^3} r \cos\theta, \tag{1.81b}$$

along the z-axis. Considering the 3D space:

$$\begin{aligned} \phi_{\text{outside}} &= \frac{3\epsilon_2}{2\epsilon_1 + \epsilon_2} \frac{1}{r^3} \boldsymbol{\mu} \cdot \mathbf{r}, \\ \phi_{\text{inside}} &= \left[\frac{1}{r^3} - \frac{2(\epsilon_1 - \epsilon_2)}{2\epsilon_1 + \epsilon_2} \frac{1}{a^3} \right] \boldsymbol{\mu} \cdot \mathbf{r}. \end{aligned} \tag{1.82}$$

A.9. The solution of Laplace's equation due to a point quadrupole

Similar to the above derivation, as the ideal quadrupole potential along the z-axis is:

$$\phi = \frac{q}{r^3} \left(3\cos^2\theta - 1 \right), \tag{1.83}$$

and all $A_n = 0$ plus all $D_n = 0$ except $D_2 = 2q$. Thus,

$$\phi_{\text{out}} = \sum_{n=0}^{\infty} \frac{B_n}{r^{n+1}} P_n(\cos\theta), \tag{1.84a}$$

$$\phi_{\text{in}} = \sum_{n=0}^{\infty} C_n r^n \, \mathrm{P_n}(\cos\theta) + \frac{q}{r^3} \left(3\cos^2\theta - 1 \right). \tag{1.84b}$$

For the case when $n = 2$,

$$\frac{B_2}{a^3} = C_2 a^2 + \frac{2q}{a^3}, \tag{1.85a}$$

$$-3\epsilon_1 \cdot \frac{B_2}{a^4} = 2\epsilon_2 C_2 a - 3\epsilon_2 \frac{2q}{a^4}. \tag{1.85b}$$

By solving the equation we get:

$$B_2 = \frac{10q\epsilon_2}{3\epsilon_1 + 2\epsilon_2}, \tag{1.86a}$$

$$C_2 = -2q\frac{3(\epsilon_1 - \epsilon_2)}{(3\epsilon_1 + 2\epsilon_2)\, a^5}. \tag{1.86b}$$

Therefore, we have:

$$\phi_{\text{out}} = \frac{10q\epsilon_2}{(3\epsilon_1 + 2\epsilon_2)\, 2 \cdot r^3}\left(3\cos^2\theta - 1\right) = \frac{5\epsilon_2}{3\epsilon_1 + 2\epsilon_2}\frac{q}{r^3}\left(3\cos^2\theta - 1\right), \tag{1.87a}$$

$$\phi_{\text{in}} = -2q\frac{3(\epsilon_1 - \epsilon_2)}{(3\epsilon_1 + 2\epsilon_2)\, a^5}r^2 \cdot \frac{1}{2}\left(3\cos^2\theta - 1\right) + \frac{q}{r^3}\left(3\cos^2\theta - 1\right) \tag{1.87b}$$

$$= \left[-\frac{3\left(\epsilon_1 - \epsilon_2\right)}{(3\epsilon_1 + 2\epsilon_2)}\frac{r^5}{a^5} + 1\right]\frac{q}{r^3}\left(3\cos^2\theta - 1\right). \tag{1.87c}$$

Considering the 3D domain:

$$\phi_{\text{out}} = \frac{5\epsilon_2}{3\epsilon_1 + 2\epsilon_2}\frac{3}{r^5}\Theta : \mathbf{rr}, \tag{1.88a}$$

$$\phi_{\text{in}} = -2q\frac{3(\epsilon_1 - \epsilon_2)}{(3\epsilon_1 + 2\epsilon_2)\, a^5}r^2 \cdot \frac{1}{2}\left(3\cos^2\theta - 1\right) + \frac{q}{r^3}\left(3\cos^2\theta - 1\right) \tag{1.88b}$$

$$= \left[-\frac{3\left(\epsilon_1 - \epsilon_2\right)}{(3\epsilon_1 + 2\epsilon_2)}\frac{r^5}{a^5} + 1\right]\frac{3}{r^5}\Theta : \mathbf{rr}, \tag{1.88c}$$

where Θ denotes the quadrupole moment.

A.10. The dipole theory of dielectrics

In this context, we discuss the concepts of polarization due to the electric moment, the cavity field and the reaction field, as identified by Onsager [103].

According to Debye's dipole theory [104], the "internal field" that polarizes a molecule in the dielectric is equal to the external field, augmented by $\frac{4\pi r}{3}$ times the electric moment induced in a unit volume of the dielectric. Debye's implicit assumption is that the force-couple, which tends to orient an electrically asymmetric molecule in a polarized dielectric,

is proportional to the same internal field. This assumption leads to Debye's well-known formula for the dielectric constant ϵ:

$$\frac{\epsilon - 1}{\epsilon + 2} = \frac{4\pi}{3} \sum N \left(\alpha + \frac{d^2}{3kT} \right), \tag{1.89}$$

where α denotes the polarizability of the molecule, d its permanent electric moment, kT the energy of thermal agitation, N the concentration (molecules/cm^3), and the summation is extended over all species of molecules present.

The molecular shape influences the outcome, but we limit our discussion to "spheres", denoted by the radius a, which raises another open question regarding the appropriate choice of a in Section A.11. Other relevant properties of a molecule include its polarizability α, which is associated with an "internal refractive index" n as follows:

$$\alpha = \frac{n^2 - 1}{n^2 + 2} a^3. \tag{1.90}$$

and a permanent dipole moment d (in vacuum). In an electric field $\mathbf{F}$, the total electric moment $\mathbf{p}$ is the vector sum of the permanent and the induced dipole moments such that

$$\mathbf{p} = d\mathbf{d} + \alpha\mathbf{F}, \tag{1.91}$$

where $\mathbf{d}$ denotes a unit vector in the direction of the dipole axis.

The field exerted on a molecule within a polarized dielectric can be separated into two components: a "cavity field", denoted as G, determined by the molecular shape and proportional to the external field intensity, derived from a homogeneous field E due to an empty spherical cavity; and a "reaction field", denoted as R, which is proportional to the total electric moment and influenced by the moment's instantaneous orientation, derived from a dipole with moment p placed at the center of the spherical cavity and immersed into an

unpolarized medium with a dielectric constant ϵ. They are formulated as:

$$G = \frac{3\epsilon}{2\epsilon + 1} E, \tag{1.92}$$

$$R = \frac{2(\epsilon - 1)}{2\epsilon + 1} \frac{p}{a^3}. \tag{1.93}$$

For a neutral, spherical molecule with an arbitrary charge distribution, these relationships remain valid, therefore the total field F exerted on a spherical molecule in a polarized dielectric is

$$\mathbf{F} = \mathbf{G} + \mathbf{R} = \frac{3\epsilon}{2\epsilon + 1} \mathbf{E} + \frac{2(\epsilon - 1)}{(2\epsilon + 1)a^3} \mathbf{p}. \tag{1.94}$$

The average orientation of a molecule is governed by the orienting force-couple exerted by the cavity field G on the electric moment of the molecule, with R never exerting torque to the molecule. As all real molecules possess electrically deformable properties, the reaction field R will amplify the electric moment of any molecule submerged in a dielectric. Similarly, the induced moment resulting from the cavity field G will also be augmented by the corresponding component of R.

A.11. Boundary conditions of the PMPB model

In this context, we adopt notation introduced by Schnieders [62] to discuss the boundary conditions for the PMPB model.

For a solvent described by a simplified Linearized Poisson-Boltzmann Equation (LPBE),

$$\nabla^2 \phi(\mathbf{r}) = \bar{\kappa}^2 \phi(\mathbf{r}), \tag{1.95}$$

the potential at $\mathbf{r}$ due to a symmetric, traceless multipole in a homogeneous dielectric ϵ_1 is

$$\begin{aligned}\phi\left(\mathbf{r}_{ij}\right) &= \left(\mathbf{T}\right)^t \mathbf{M}_j \\ &= \left(\begin{bmatrix} 1 \\ -\partial/\partial x \\ -\partial/\partial y \\ -\partial/\partial z \\ (1/3)\left(\partial^2/\partial x \partial x\right) \\ \vdots \end{bmatrix} 1/\epsilon_1 r_{ij} \right)^t \begin{bmatrix} q_j \\ \mu_{x,j} \\ \mu_{y,j} \\ \mu_{z,j} \\ \Theta_{xx,j} \\ \vdots \end{bmatrix},\end{aligned} \tag{1.96}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{s}_j$ represents the difference between a grid location and a multipole site, the superscript t denotes the transpose, and $\mathbf{M}_j$ represents the multipole moments of j-th charge. The potential inside the spherical cavity is the superposition of the homogeneous potential and the reaction potential:

$$\phi_{\text{in}}\left(\mathbf{r}_{ij}\right) = \left[\left(\mathbf{I} + \mathbf{R}_{\text{in}}\right)\mathbf{T}\right]^t \mathbf{M}_j, \tag{1.97}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{R}_{\text{in}}$ is a diagonal matrix with diagonal elements:

$$[c_{\text{in}}(0), c_{\text{in}}(1), c_{\text{in}}(1), c_{\text{in}}(1), c_{\text{in}}(2), \ldots], \tag{1.98}$$

based on coefficients for multipoles of order n, determined by the boundary conditions:

$$c_{\text{in}}(n) = \beta_n \left(\frac{r_{ij}}{a}\right)^{2n+1}. \tag{1.99}$$

Similarly, the potential outside the cavity is given by:

$$\phi_{\text{out}}(\mathbf{r}_{ij}) = (\mathbf{R}_{\text{out}}^T \mathbf{T})^t \mathbf{M}_j, \tag{1.100}$$

where $\mathbf{R}_{\text{out}}$ is a diagonal matrix with diagonal elements:

$$[c_{\text{out}}(0), c_{\text{out}}(1), c_{\text{out}}(1), c_{\text{out}}(1), c_{\text{out}}(2), \ldots], \tag{1.101}$$

based on a second set of coefficients for multipoles of order n also determined by the boundary conditions:

$$c_{\text{out}}(n) = \frac{\epsilon_1}{\epsilon_2} \kappa r_{ij} \alpha_n k_n(\kappa r_{ij}) \left(\frac{r_{ij}}{a}\right)^n, \tag{1.102}$$

where $k_n(x)$ is the modified spherical Bessel function of the third kind given by:

$$k_n(x) = \frac{\pi e^{-x}}{2x} \sum_{i=0}^{n} \frac{(n+i)!}{i!\,(n-i)!\,(2x)^i}. \tag{1.103}$$

Kirkwood [105] determined α_n and β_n as:

$$\alpha_n = \frac{(2n+1)/(\kappa a)}{n k_n(\kappa a)/\hat{\epsilon} - \kappa a k_n'(\kappa a)}, \tag{1.104}$$

and

$$\beta_n = \frac{(n+1) k_n(\kappa a)/\hat{\epsilon} + \kappa a k_n'(\kappa a)}{k_n(\kappa a)/\hat{\epsilon} - \kappa a k_n'(\kappa a)}, \tag{1.105}$$

where $k_n'(x)$ is the derivative of $k_n(x)$ and $\hat{\epsilon}$ is the ratio of the permittivity in solvent to that inside the sphere, ϵ_2/ϵ_1. We only require the potential outside the cavity to construct SDH and MDH boundary conditions, and therefore, we provide specific values of n and k_n through quadrupole order, as shown in Table 1.1. As the ionic strength goes to zero, the Laplace equation is obeyed in solvent. For multipoles through quadrupole order, the difference between the LPBE and Laplace potentials outside the cavity is summarized in Table 1.2.

Table 1.1: Explicit values for the functions $\alpha_n(x)$ and $k_n(x)$ up to quadrupole order.

n	$\alpha_n(x)/(2\exp(x)/\pi)$	$k_n(x)/(\pi/(2\exp(x)))$
0	$1/(1+x)$	$1/x$
1	$3\hat{\epsilon}x/[1+x+\hat{\epsilon}\,(2+2x+x^2)]$	$(1+x)/x^2$
2	$5\hat{\epsilon}x^2/[2\,(3+3x+x^2)+\hat{\epsilon}\,(9+9x+4x^2+x^3)]$	$(3+3x+x^2)\,/x^3$

Table 1.2: Explicit values of the coefficients used to calculate the potential at the grid boundary of LPBE and Poisson equation calculations, respectively, under the SDH or MDH approximation. The LPBE coefficients reduce to the Poisson equation coefficients as salt concentration goes to zero.

n	$\kappa r\alpha_n(\kappa a)k_n(\kappa r)(r/a)^n$	$\lim_{\kappa\to 0}\left[\kappa r\alpha_n(\kappa a)k_n(\kappa r)(r/a)^n\right]$
0	$\exp(\kappa(a-r))/[1+\kappa a]$	1
1	$3\hat{\epsilon}\exp(\kappa(a-r))(1+\kappa r)/[1+\kappa a+\hat{\epsilon}\,(2+2\kappa a+(\kappa a)^2)]$	$3\hat{\epsilon}/(1+2\hat{\epsilon})$
2	$5\hat{\epsilon}\exp(\kappa(a-r))\,(3+3\kappa r+(\kappa r)^2)\,/[2\,(3+3\kappa a+(\kappa a)^2)+\hat{\epsilon}(9$ $+9\kappa a+4(\kappa a)^2+(\kappa a)^3)]$	$5\hat{\epsilon}/(2+3\hat{\epsilon})$

A.12. Units of PB equation

This section describes the units of PB equation. The Debye-Hückel parameter κ is defined $\kappa^2 = \frac{8\pi N_A e_C^2}{1000\epsilon_2 k_B T} I_S$ where $N_A = 6.022045 \times 10^{23}$ is the Avagadro's number, $e_c = 4.8032424 \times 10^{-10}$esu $= 1.60217646 \times 10^{-19}$C is the Fundamental charge, $k_B = 1.380662 \times 10^{-16}$erg/K $= 1.380662 \times 10^{-23}$ J/K is Boltzmann's constant, T is the absolution temperature in K, $I_s = \frac{1}{2}\sum_{i=1}^{N_i} c_i z_i^2$ is the ionic strength with N_i the number of different types of ions, and c_i the molar concentration of ion type i with charge $q_i = z_i e_c$. In simulation, we have $\kappa^2 = 8.430325455 * I_S/\varepsilon_2$ for $T = 300K$.

The solution ϕ of the PB model is the electrostatic potential, whose units could be various depending on which units system is applied. In our numerical implementation, unit for the length is Å and the unit of the potential ϕ is e_c/Å. After computing ϕ, one needs to multiply it with a factor 332.0716 to convert it to the unit of kcal/mol/e_c for free energy calculations. The reader can refer to [106, 107] for more details about definition and units of these coefficients.

BIBLIOGRAPHY

[1] J. Warwicker and H. C. Watson, *Calculation of the electric potential in the active site cleft due to alpha-helix dipoles*, *Journal of Molecular Biology* **157** (1982) 671–9. 1

[2] C. Holm, P. Kékicheff and R. Podgornik, *Electrostatic effects in soft matter and biophysics*, vol. 46. Springer Science & Business Media, 2001. 1

[3] B. Honig and A. Nicholls, *Classical electrostatics in biology and chemistry*, *Science* **268** (1995) 1144–9. 1, 3, 13

[4] M. E. Davis and J. A. McCammon, *Electrostatics in biomolecular structure and dynamics*, *Chem. Rev.* **94** (1990) 509–21. 1

[5] J. Gu and P. E. Bourne, *Structural bioinformatics*, vol. 44. John Wiley & Sons, 2009. 1

[6] N. Huang, Y. Chelliah, Y. Shan, C. A. Taylor, S.-H. Yoo, C. Partch et al., *Crystal structure of the heterodimeric CLOCK:BMAL1 transcriptional activator complex*, *Science* **337** (2012) 189–194, [http://www.sciencemag.org/content/337/6091/189.full.pdf]. 2

[7] D. A. Beard and T. Schlick, *Modeling salt-mediated electrostatics of macromolecules: the discrete surface charge optimization algorithm and its application to the nucleosome*, *Biopolymers* **58** (2001) 106–115. 2

[8] E. Alexov, E. L. Mehler, N. Baker, A. M. Baptista, Y. Huang, F. Milletti et al., *Progress in the prediction of pka values in proteins.*, *Proteins* **79** (Dec, 2011) 3260–3275. 2

[9] J. Hu, S. Zhao and W. Geng, *Accurate pka computation using matched interface and boundary (MIB) method based Poisson-Boltzmann solver*, *Communication in Computational Physics* **2** (2018) 520–539. 2

[10] J. Chen, J. Hu, Y. Xu, R. Krasny and W. Geng, *Computing protein pKas using the TABI Poisson-Boltzmann solver*, *J. Comput. Biophys. Chem.* **20** (2021) 175–187. 2, 40, 41, 42, 66

[11] Y. C. Zhou, B. Lu and A. A. Gorfe, *Continuum electromechanical modeling of protein-membrane interactions*, *Phys. Rev. E* **82** (Oct, 2010) 041923. 2

[12] D. D. Nguyen, B. Wang and G.-W. Wei, *Accurate, robust, and reliable calculations of Poisson-Boltzmann binding energies*, *Journal of Computational Chemistry* **38** (2017) 941–948. 2

[13] J. A. Wagoner and N. A. Baker, *Assessing implicit models for nonpolar mean solvation forces: The importance of dispersion and volume terms*, Proceedings of the National Academy of Sciences **103** (2006) 8331–8336, [http://www.pnas.org/content/103/22/8331.full.pdf]. 2

[14] N. Unwin, *Refined structure of the nicotinic acetylcholine receptor at 4Å resolution*, Journal of Molecular Biology **346** (2005) 967 – 989. 2

[15] W. C. Still, A. Tempczyk, R. C. Hawley and T. Hendrickson, *Semianalytical treatment of solvation for molecular mechanics and dynamics*, Journal of the American Chemical Society **112** (1990) 6127–6129. 2

[16] N. A. Baker, *Poisson-Boltzmann methods for biomolecular electrostatics*, Methods Enzymol. **383** (2004) 94–118. 3, 6, 9

[17] Q. Lu and R. Luo, *A Poisson-Boltzmann dynamics method with nonperiodic boundary condition*, Journal of Chemical Physics **119** (2003) 11035–11047. 3

[18] R. Luo, L. David and M. K. Gilson, *Accelerated Poisson-Boltzmann calculations for static and dynamic systems*, Journal of Computational Chemistry **23** (2002) 1244–53. 3

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

[20] W. Rocchia, S. Sridharan, A. Nicholls, E. Alexov, A. Chiabrera and B. Honig, *Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: Applications to the molecular systems and geometric objects*, Journal of Computational Chemistry **23** (2002) 128–137. 3, 70

[21] Z. Qiao, Z. Li and T. Tang, *Finite difference scheme for solving the nonlinear poisson-boltzmann equation modeling charged spheres*, Journal of Computational Mathematics **24** (04, 2006) . 3

[22] S. Yu, W. Geng and G. W. Wei, *Treatment of geometric singularities in implicit solvent models*, Journal of Chemical Physics **126** (2007) 244108. 3, 69

[23] R. Egan and F. Gibou, *Geometric discretization of the multidimensional Dirac delta distribution – Application to the Poisson equation with singular source terms*, Journal of Computational Physics **346** (2017) 71 – 90. 3

[24] W. Geng, S. Yu and G. W. Wei, *Treatment of charge singularities in implicit solvent models*, J. Chem. Phys. **127** (2007) 114106. 3, 4, 68, 78

[25] Q. Cai, J. Wang, H.-K. Zhao and R. Luo, *On removal of charge singularity in Poisson-Boltzmann equation*, The Journal of Chemical Physics **130** (2009) . 3, 30, 32

[26] W. Geng and S. Zhao, *A two-component matched interface and boundary (mib) regularization for charge singularity in implicit solvation*, J. Comput. Phys. **351** (2017) 25–39. xvii, 3, 12, 23, 30, 36, 68, 75, 76

[27] A. Lee, W. Geng and S. Zhao, *Regularization methods for the Poisson-Boltzmann equation: comparison and accuracy recovery*, J. Comput. Phys. **426** (2020) 109958. 3, 36

[28] R. J. Zauhar and R. S. Morgan, *A new method for computing the macromolecular electric potential*, *Journal of Molecular Biology* **186** (1985) 815–20. 3

[29] A. Juffer, B. E., B. van Keulen, A. van der Ploeg and H. Berendsen, *The electric potential of a macromolecule in a solvent: a fundamental approach*, *J. Comput. Phys.* **97** (1991) 144–171. 3, 43, 44, 45, 46

[30] J. Liang and S. Subranmaniam, *Computation of molecular electrostatics with boundary element methods*, *Biophys. J.* **73** (1997) 1830–1841. 3

[31] A. H. Boschitsch, M. O. Fenley and H.-X. Zhou, *Fast boundary element method for the linear Poisson-Boltzmann equation*, *The Journal of Physical Chemistry B* **106** (2002) 2741–2754, [http://dx.doi.org/10.1021/jp013607q]. 3

[32] B. Lu, X. Cheng and J. A. McCammon, *A new-version-fast-multipole-method-accelerated electrostatic calculations in biomolecular systems*, *Journal of Computational Physics* **226** (2007) 1348 – 1366. 3

[33] M. D. Altman, J. P. Bardhan, J. K. White and B. Tidor, *Accurate solution of multi-region continuum biomolecule electrostatic problems using the linearized Poisson–Boltzmann equation with curved boundary elements*, *J. Comput. Chem.* **30** (2009) 132–153. 3

[34] L. Greengard, D. Gueyffier, P.-G. Martinsson and V. Rokhlin, *Fast direct solvers for integral equations in complex three-dimensional domains*, *Acta Numerica* **18** (005, 2009) 243–275. 3

[35] C. Bajaj, S.-C. Chen and A. Rand, *An efficient higher-order fast multipole boundary element solution for Poisson-Boltzmann-based molecular electrostatics*, *SIAM Journal on Scientific Computing* **33** (2011) 826–848, [http://dx.doi.org/10.1137/090764645]. 3

[36] B. Zhang, B. Lu, X. Cheng, J. Huang, N. P. Pitsianis, X. Sun et al., *Mathematical and numerical aspects of the adaptive fast multipole Poisson-Boltzmann solver*, *Communications in Computational Physics* **13** (001, 2013) 107–128. 3

[37] W. Geng and R. Krasny, *A treecode-accelerated boundary integral Poisson-Boltzmann solver for electrostatics of solvated biomolecules*, *Journal of Computational Physics* **247** (2013) 62 – 78. xvii, 3, 41, 43, 45, 46, 47, 48, 68, 72, 75, 76

[38] Q. Sun, E. Klaseboer and D. Y. C. Chan, *A robust and accurate formulation of molecular and colloidal electrostatics*, *The Journal of Chemical Physics* **145** (2016) 054106, [https://doi.org/10.1063/1.4960033]. 3

[39] Y. Zhong, K. Ren and R. Tsai, *An implicit boundary integral method for computing electric potential of macromolecules in solvent*, *Journal of Computational Physics* **359** (2018) 199 – 215. 3

[40] B. J. Yoon and A. M. Lenhoff, *A boundary element method for molecular electrostatics with electrolyte effects*, *Journal of Computational Chemistry* **11** (1990) 1080–1086. 3

[41] L. F. Greengard and J. Huang, *A new version of the fast multipole method for screened coulomb interactions in three dimensions*, *J. Comput. Phys.* **180** (2002) 642 – 658. 3

[42] P. Li, H. Johnston and R. Krasny, *A Cartesian treecode for screened Coulomb interactions*, *J. Comput. Phys.* **228** (2009) 3858–3868. 3, 40, 47, 48, 49

[43] S. Wang, A. Lee, E. Alexov and S. Zhao, *A regularization approach for solving poisson's equation with singular charge sources and diffuse interfaces*, *Applied Mathematics Letters* **102** (2020) 106144. 3

[44] J. Chen and W. Geng, *On preconditioning the treecode-accelerated boundary integral (TABI) Poisson-Boltzmann solver*, *J Comput Phys* **373** (2018) 750–762. 4, 42, 49, 74

[45] M. Born, *Volumen und Hydratationswärme der Ionen*, *Zeitschrift fur Physik* **1** (Feb., 1920) 45–48. 8

[46] M. F. Sanner, A. J. Olson and J. C. Spehner, *Reduced surface: An efficient way to compute molecular surfaces*, *Biopolymers* **38** (1996) 305–320. xvi, 10, 46, 62, 70, 74, 93, 94

[47] K. A. Sharp and B. Honig, *Electrostatic interactions in macromolecules - theory and applications*, *Annual Review of Biophysics and Biophysical Chemistry* **19** (1990) 301–332. 13

[48] J. W. Ponder, C. J. Wu, P. Y. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders et al., *Current status of the AMOEBA polarizable force field*, *J. Phys. Chem. B* **114** (2010) 2549. 13

[49] A. J. Stone, *The Theory of Intermolecular Forces*. Oxford University Press, 2013. 13

[50] C. Sagui, L. G. Pedersen and T. A. Darden, *Towards an accurate representation of electrostatics in classical force fields: Efficient implementation of multipolar interactions in biomolecular simulations*, *The Journal of Chemical Physics* **120** (2004) 73–87. 13, 15

[51] P. Ren and J. W. Ponder, *Polarizable atomic multipole water model for molecular mechanics simulation*, *The Journal of Physical Chemistry B* **107** (2003) 5933–5947, [http://dx.doi.org/10.1021/jp027815+]. 13, 14, 15

[52] P. Ren, C. Wu and J. W. Ponder, *Polarizable atomic multipole-based molecular mechanics for organic molecules*, *Journal of Chemical Theory and Computation* **7** (2011) 3143–3161, [http://dx.doi.org/10.1021/ct200304d]. 13, 15

[53] Y. Shi, Z. Xia, J. Zhang, R. Best, C. Wu, J. W. Ponder et al., *Polarizable atomic multipole-based amoeba force field for proteins*, *Journal of Chemical Theory and Computation* **9** (2013) 4046–4063, [http://dx.doi.org/10.1021/ct4003702]. 13, 14, 15

[54] J. W. Ponder and F. M. Richards, *An efficient newton-like method for molecular mechanics energy minimization of large molecules*, *Journal of computational chemistry* **8** (1987) 1016–1024. 13

[55] C. E. Kundrot, J. W. Ponder and F. M. Richards, *Algorithms for calculating excluded volume and its derivatives as a function of molecular conformation and their use in energy minimization*, *Journal of computational chemistry* **12** (1991) 402–409. 13

[56] R. V. Pappu, R. K. Hart and J. W. Ponder, *Analysis and application of potential energy smoothing and search methods for global optimization*, *The Journal of Physical Chemistry B* **102** (1998) 9725–9742. 13

[57] M. S. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A. L. Beberg et al., *Accelerating molecular dynamic simulation on graphics processing units*, *Journal of computational chemistry* **30** (2009) 864–872. 13

[58] D. A. Case, T. E. Cheatham III, T. Darden, H. Gohlke, R. Luo, K. M. Merz Jr et al., *The amber biomolecular simulation programs*, *Journal of computational chemistry* **26** (2005) 1668–1688. 13

[59] C. Zhang, C. Lu, Z. Jing, C. Wu, J.-P. Piquemal, J. W. Ponder et al., *Amoeba polarizable atomic multipole force field for nucleic acids*, *Journal of Chemical Theory and Computation* **14** (04, 2018) 2084–2108. 14, 35

[60] D. J. Tannor, B. Marten, R. Murphy, R. A. Friesner, D. Sitkoff, A. Nicholls et al., *Accurate first principles calculation of molecular charge distributions and solvation energies from ab initio quantum mechanics and continuum dielectric theory*, *Journal of the American Chemical Society* **116** (12, 1994) 11875–11882. 14

[61] Y. Mei, C. Ji and J. Z. H. Zhang, *A new quantum method for electrostatic solvation energy of protein*, *The Journal of Chemical Physics* **125** (2006) 094906, [https://doi.org/10.1063/1.2345201]. 14

[62] M. J. Schnieders, N. A. Baker, P. Ren and J. W. Ponder, *Polarizable atomic multipole solutes in a Poisson–Boltzmann continuum*, *The Journal of Chemical Physics* **126** (2007) . 17, 20, 122

[63] C. J. F. BOTTCHER, *Chapter i electric dipoles and multipoles*, *Theroy of Electric Polarization (Second Edition)* (1973) . xiii, 21, 22, 106, 110

[64] W. Geng, S. Yu and G. W. Wei, *Treatment of charge singularities in implicit solvent models*, *J. Chem. Phys.* **127** (2007) 114106. 30

[65] M. Holst, J. McCammon, Z. Yu, Y. Zhou and Y. Zhu, *Adaptive finite element modeling techniques for the Poisson-Boltzmann equation*, *Communication in Computational Physics* **11** (2012) 179–214. 31

[66] A. Nicholls and B. Honig, *A rapid finite difference algorithm, utilizing successive over-relaxation to solve the poissonâboltzmann equation*, *Journal of Computational Chemistry* **12** (1991) 435–445, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.540120405]. 37

[67] C. Li, M. McGowan, E. Alexov and S. Zhao, *A newton-like iterative method implemented in the delphi for solving the nonlinear poisson-boltzmann equation*, *Mathematical Biosciences and Engineering* **17** (2020) 6259. 37

[68] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, *Journal of Computational Physics* **73** (1987) 325 – 348. 40

[69] B. Zhang, B. Peng, J. Huang, N. P. Pitsianis, X. Sun and B. Lu, *Parallel {AFMPB} solver with automatic surface meshing for calculation of molecular solvation free energy*, *Computer Physics Communications* **190** (2015) 173 – 181. 40

[70] J. K. Salmon, M. S. Warren and G. S. Winckelmans, *Fast parallel tree codes for gravitational and fluid dynamical n-body problems*, *Int. J. Supercomputer Appl* **8** (1986) 129–142. 41, 42, 66

[71] T. Hamada, K. Nitadori, K. Benkrid, Y. Ohno, G. Morimoto, T. Masada et al., *A novel multiple-walk parallel algorithm for the barnes–hut treecode on gpus – towards cost effective, high performance n-body simulation*, *Computer Science - Research and Development* **24** (Sep, 2009) 21–31. 41

[72] J. Bédorf, E. Gaburov and S. Portegies Zwart, *A sparse octree gravitational N-body code that runs entirely on the GPU processor*, *Journal of Computational Physics* **231** (2012) 2825–2839. 41

[73] M. Burtscher and K. Pingali, *An efficient CUDA implementation of the tree-based Barnes-Hut N-body algorithm*, pp. 75–92. Elsevier Inc., 12, 2011. 10.1016/B978-0-12-384988-5.00006-1. 41

[74] H. C. Edwards, C. R. Trott and D. Sunderland, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, *Journal of Parallel and Distributed Computing* **74** (2014) 3202 – 3216. 43, 57, 94

[75] Y. Saad and M. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, *SIAM Journal on Scientific and Statistical Computing* **7** (1986) 856–869, [http://dx.doi.org/10.1137/0907058]. 47

[76] L. Wilson and R. Krasny, *Comparison of the MSMS and NanoShaper molecular surface triangulation codes in the TABI Poisson–Boltzmann solver*, *Journal of Computational Chemistry* **42** (2021) 1552–1560. 47, 66

[77] J. Barnes and P. Hut, *A hierarchical O(N log N) force-calculation algorithm*, *Nature* **324** (12, 1986) 446–449. 47, 48

[78] Z.-H. Duan and R. Krasny, *An adaptive treecode for computing nonbonded potential energy in classical molecular systems*, *J. Comput. Chem.* **22** (2001) 184–195. 47

[79] K. Lindsay and R. Krasny, *A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow*, *Journal of Computational Physics* **172** (2001) 879 – 907. 47

[80] J. Chen, W. Geng and D. Reynolds, *Cyclically paralleled treecode for fast computing electrostatic interactions on molecular surfaces*, *Comput. Phys. Commun.* **260** (2021) 107742. 54

[81] W. Humphrey, A. Dalke and K. Schulten, *VMD – visual molecular dynamics*, *Journal of Molecular Graphics* **14** (1996) 33–38. xiv, 65

[82] N. Vaughn, L. Wilson and R. Krasny, *A GPU-accelerated barycentric Lagrange treecode*, in *2020 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*, pp. 701–710, 2020. 66

[83] L. Wilson, W. Geng and R. Krasny, *TABI-PB 2.0: An Improved Version of the Treecode-Accelerated Boundary Integral Poisson-Boltzmann Solver*, *The Journal of Physical Chemistry B* **126** (09, 2022) 7104–7113. 66

[84] B. Liu, B. Wang, R. Zhao, Y. Tong and G.-W. Wei, *ESES: Software for Eulerian solvent excluded surface*, *Journal of Computational Chemistry* **38** (2017) 446–466. 67, 68, 72, 74, 76

[85] D. Chen, Z. Chen, C. Chen, W. Geng and G. W. Wei, *MIBPB: A software package for electrostatic analysis*, *J. Comput. Chem.* **32** (2011) 657 – 670. 68, 75, 86

[86] B. Lee and F. M. Richards, *The interpretation of protein structures: estimation of static accessibility*, *J Mol Biol* **55** (1971) 379–400. 69

[87] M. L. Connolly, *Depth buffer algorithms for molecular modeling*, *J. Mol. Graphics* **3** (1985) 19–24. 70, 71

[88] S. Decherchi and W. Rocchia, *A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale*, *PLOS ONE* **8** (04, 2013) 1–15. 70, 72, 73

[89] L. WE, *Marching cubes: A high resolution 3d surface construction algorithm*, *Computer graphics* **21** (1987) 7–12. 70, 71

[90] L. Wilson and R. Krasny, *Comparison of the msms and nanoshaper molecular surface triangulation codes in the tabi poisson-boltzmann solver.*, *J Comput Chem* **42** (Aug, 2021) 1552–1560. 70

[91] Y. Zhou, *Matched interface and boundary (MIB) method and its applications to implicit solvent modeling of biomolecules*. PhD thesis, Michigan State University, 2006. 70

[92] Y. C. Zhou and G. W. Wei, *On the fictitious-domain and interpolation formulations of the matched interface and boundary (MIB) method*, *J. Comput. Phys.* **219** (2006) 228–246. 74

[93] Z. Liu, Y. Li, L. Han, J. Liu, Z. Zhao, W. Nie et al., *PDB-wide collection of binding data: current status of the PDBbind database*, *Bioinformatics* **31** (2015) 405–412. 79

[94] A. Onufriev, D. Bashford and D. A. Case, *Modification of the generalized Born model suitable for macromolecules*, *Journal of Physical Chemistry B* **104** (2000) 3712–3720. 81

[95] D. Bramer and G.-W. Wei, *Multiscale weighted colored graphs for protein flexibility and rigidity analysis*, *The Journal of chemical physics* **148** (2018) 054103. 82, 84

[96] D. D. Nguyen, Z. Cang, K. Wu, M. Wang, Y. Cao and G.-W. Wei, *Mathematical deep learning for pose and binding affinity prediction and ranking in d3r grand challenges*, *Journal of Computer Aided Molecular Design* **33** (2019) 71–82. 82

[97] S. P. Borgatti, *Centrality and network flow*, *Social networks* **27** (2005) 55–71. 83

[98] Q. Cai, M. J. Hsieh, J. Wang and R. Luo, *Performance of nonlinear finite-difference Poisson-Boltzmann solvers*, *Journal of Chemical Theory and Computation* **6(1)** (2009) 203–211. 86

[99] L. Li, C. Li, S. Sarkar, J. Zhang, S. Witham, Z. Zhang et al., *Delphi: a comprehensive suite for delphi software and associated resources*, *BMC Biophysics* **5:9** (2012) 2046–1682. 86

[100] A. H. Boschitsch and M. O. Fenley, *Hybrid boundary element and finite difference method for solving the nonlinear Poisson-Boltzmann equation*, *J. Comput. Chem.* **25** (2004) 935–955. 99

[101] L. Wang, R. Krasny and S. Tlupova, *A kernel-independent treecode based on barycentric lagrange interpolation*, *Commun. Comput. Phys.* **28** (2020) 1415–1436. 99

[102] A. J. Stone, *The Theory of Intermolecular Forces*. Clarendon, Oxford, 1996. 110

[103] L. Onsager, *Electric moments of molecules in liquids*, *Journal of the American Chemical Society* **58** (1936) 1486–1493, [https://doi.org/10.1021/ja01299a050]. 116, 120

[104] P. Debye, *Einige resultate einer kinetischen theorie der isolatoren*, *Physik Z.* **13** (1912) 97. 120

[105] J. G. Kirkwood, *Theory of solution of molecules containing widely separated charges with special application to zwitterions*, *J. Comput. Phys.* **7** (1934) 351 – 361. 124

[106] W. Geng, *A boundary integral Poisson–Boltzmann solvers package for solvated bimolecular simulations*, *Computational and Mathematical Biophysics* **3** (2015) 43–58. 125

[107] M. J. Holst, *The Poisson-Boltzmann Equation: Analysis and Multilevel Numerical Solution*. PhD thesis, UIUC, 1994. 125