UNIVERSITY OF CALIFORNIA,
IRVINE

Multi-scaled Modeling of Electrostatics in Biomolecules

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Materials Science and Engineering

by

Haixin Wei

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Ray Luo, for his support and guidance over the past five years.

I would like to thank all the current and previous Luo lab members, including: Ruxi Qi, Shiji Zhao, Vy Duong, Edward King, Erick Aitchison, D'Artagnan Greene, Andrew Schaub, Fanglue Ni, Tianyin Qiu, Zekai Zhao, Luobin Wang and Garik Megerdich. Their friendship has always been an encouragement.

I would like to thank my family and friends all around the world. Thank you all for bringing laugh and happiness to my life.

VITA

Haixin Wei

2017	B.S. in Applied Physics, University of Science and Technology of China
2022	Ph.D. in Materials Science and Engineering, University of California, Irvine

FIELD OF STUDY

Molecular Mechanics, Molecular Dynamics, Multi-scaled Molecular Modeling

PUBLICATIONS

Wei, Haixin, et al. "Stress Tensor and Constant Pressure Simulation for Polarizable Gaussian Multipole Model." *The Journal of Chemical Physics* 156, 114114 (2022)

Wei, Haixin, et al. "Machine-Learned Molecular Surface and Its Application to Implicit Solvent Simulations." *Journal of Chemical Theory and Computation* 17.10 (2021): 6214–6224.

Wei, Haixin, et al. "Efficient Formulation of Polarizable Gaussian Multipole Electrostatics for Biomolecular Simulations." *The Journal of Chemical Physics* 153.11 (2020): 114116.

Wei, Haixin, et al. "Improved Poisson–Boltzmann Methods for High-Performance Computing." *Journal of Chemical Theory and Computation* 15.11 (2019): 6190-6202.

Wei, Haixin, Ray Luo, and Ruxi Qi. "An Efficient Second-order Poisson–Boltzmann Method." *Journal of Computational Chemistry* 40.12 (2019): 1257-1269.

ABSTRACT OF THE DISSERTATION

Multi-scaled Modeling of Electrostatics in Biomolecules

by

Haixin Wei

Doctor of Philosophy in Materials Science and Engineering

University of California, Irvine, 2022

Professor Ray Luo, Chair

Molecular dynamics simulations of biomolecules have been widely adopted in biomedical studies. Efficient simulations require efficient functional forms, i.e. force fields, to model intra- and inter-molecular interactions. As molecular interactions can be separated into two main types: 'long-range' and 'short-range', a reliable force field is often designed to contains different terms model each of these interactions. This dissertation concerns the modeling of long-range interactions, particularly electrostatic and induction interactions. Another long-range interaction, dispersion interaction, would follow widely accepted formulation. Our goal here is to establish an accurate and efficient multi-scaled modeling scheme for biomolecular systems, which will include atomic, coarse-grained and implicit-solvent level of models. Once completed, our multi-scaled models can be combined in different manner for different biomolecular applications, serving for various accuracy and efficiency requirements in molecular dynamics simulations.

In the atomic level, as classical point-charge models continue to be used in routine biomolecular applications, there have been growing demands on developing polarizable force fields for handling more complicated biomolecular processes. Here, we focus on a recently proposed polarizable Gaussian Multipole (pGM) model for biomolecular simulations. We present an efficient formulation for the pGM model, and its implementation under various simulation conditions. It is hoped that the reformulated pGM model will facilitate the development of future force fields.

Since most particles in explicit molecular model are water molecules that solvate the target biomolecules, treating these water molecules implicitly would allow higher computational efficiency without losing any atomic-level resolution of the biomolecules. Poisson–Boltzmann equation (PBE)-based implicit solvent model has been one such attempt and been widely used in biomolecular applications. Thus, in the implicit-solvent level, we developed several new implementations of Poisson–Boltzmann (PB) models. We also present here the GPU implementations of the linear solver needed in those new PB models. Our analysis shows that the new strategies improve the convergence and stability of PB models, and also improves the efficiency of the method.

INTRODUCTION

Many important inter-molecular interactions arise ultimately from the electrostatic interaction between the particles comprising the two molecules. They can be separated into two main types: ‘long-range’, where the energy of interaction behaves as some inverse power of inter-molecular distance, and ‘short-range’, where the energy decreases in magnitude exponentially with the distance.

The long-range effects are of three kinds: electrostatic, induction and dispersion. The electrostatic effects are the simplest to understand in general terms: they arise from the straightforward classical interaction between the static charge distributions of the two molecules. Induction effects arise from the distortion of a particular molecule in the electric field of all its neighbors and are always attractive. Dispersion is an effect that cannot easily be understood in classical terms, but it arises because the charge distributions of the molecules are constantly fluctuating as the electrons move.

Further contributions to the energy arise at short range—that is, at distances where the molecular wavefunctions overlap significantly and electron exchange between the molecules becomes possible. The most important is described as exchange-repulsion. It can be thought of as comprising two effects: an attractive part, arising because the electrons become free to move over both molecules rather than just one, increasing the uncertainty in their positions and so allowing the momentum and energy to decrease; and a repulsive part, arising because the wavefunction has to adapt to maintain the Pauli antisymmetry requirement that electrons of the same spin may not be in the same place, and this costs energy.

Typical simulations usually involve $10^3 - 10^6$ atoms. For systems of this magnitude, the CPU requirements render electronic-structure-based methods impractical. For this reason, force fields have been developed as a set of simplified empirical energy equations capable of handling large molecular systems.

Early force fields were designed using intuitive notions of how atoms and molecules interact. In the short-range part, energy is expanded in a Taylor series about equilibrium bond lengths r_0 and bond angles θ_0, and a truncated Fourier series is employed to describe torsion rotations about bond axes. The force field intramolecular energy, also called the valence energy, is given below:

$$E_{valence} = \sum_{bonds} k_b (r - r_0)^2 + \sum_{angles} k_\theta (\theta - \theta_0)^2 + \sum_{dihedrals} V_n (1 + \cos(n\phi - \gamma))$$

In addition to the valence part, there is a second part of the force field, which models long-range interactions. This non-bonded portion includes an electrostatic term, a short-range exchange/repulsion, and a weakly attractive dispersion term. In traditional force fields, electrostatic interactions are commonly modeled by simple Coulomb interactions among point charges on each atom. The short-range repulsion and long-range dispersion are often modeled by a 12-6 Lennard Jones potential. The resulting non-bond energy is given by:

$$E_{non-bonded} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\varepsilon R_{ij}})$$

While there is a consensus in modeling dispersion-repulsion interactions, accurate and efficient treatment of electrostatic/induction is far from straightforward. However, they are vital in computational analyses of biomolecular structures and dynamics as they play crucial roles in biomolecular structures and functions. Given the simplicity and efficiency, additive models based on the Coulomb law will continue to play important roles in molecular modeling, polarizable models are expected to extend our ability to study more complex biomolecular systems and processes. The importance of non-additive effects is well known. For example, in pH dependent processes, interactions of ions with charged biomolecules, order-disorder transition during folding and binding, enzyme mechanism, and heterogeneous solvent environments. It is also clear that there are multiple limitations with the use of classical point models of electrostatic/induction in all existing approaches. These simple models deviate far from reality, leading to many *ad hoc* attempts to patch its artifacts and numerical issues. These result in lower degree of self-consistency and transferability, which both are important for robust modeling of molecular systems based on classical physics.

Thus, we are exploring a more realistic polarizable Gaussian Multipole (pGM) model for more satisfactory representation of the atomic charge distributions in molecular systems. In the pGM model, all charges and multipoles are represented by Gaussian densities instead of classical points. The more realistic representation of molecular charge distributions allows for higher accuracy, self-consistency, and transferability. However, induction treatments invariably reduce simulation efficiency, leading to a more pronounced efficiency issue.

To address on the efficiency problem, a second major difference from existing approaches is our concurrent focus on efficiency based on a multi-scaled framework with all-atom polarizable, coarse-grained polarizable, and continuum polarizable models, which are consistent with each other. This allows the models to be more easily interfaced in multi-scaled simulation methods. Once completing the multi-scaled framework, we will have all-atom and coarse-grained pGM solutes in one hand and explicit and implicit pGM solvents in the other. Our pGM solute models and solvent models can be combined in different manners for various accuracy and efficiency requirements.

In the following chapters, we will talk about our model development efforts one by one. In chapter 1, we will discuss an efficient second-order Poisson-Boltzmann (PB) model for continuum modeling of solvation. In chapter 2, we will talk about how to improve the harmonic average method in the PB modeling for parallel computing. In chapter 3, we will show how to use the machine-learning technique to reproduce the molecular surface that is needed in the PB modeling. In chapter 4, we will develop an efficient formulation for pGM model for molecular dynamics simulations. In chapter 5, the pGM virial and pressure expression will be presented so the constant pressure MD simulations become feasible. Finally, in chapter 6, we conclude our efforts on the development of multi-scale models for electrostatics.

Chapter 1

1.1 Introduction

Electrostatic interactions play crucial roles in biophysical processes such as protein-protein and protein-ligand interactions. Accurate and efficient treatment of electrostatics is thus vital in computational analyses of biomolecular structures and dynamics. A closely related issue is the modeling of water molecules and their electrostatic interactions with biomolecules that must be considered for any realistic representation of biomolecules at physiological conditions. Since most particles in explicit molecular model are water molecules that solvate the target biomolecules, treating these water molecules implicitly would allow higher computational efficiency without losing any atomic-level resolution of the biomolecules. The main idea of implicit solvation treatment is to model water molecules as continuous medium while still treating the biomolecular solutes in atomic detail. In this model, the solute molecule is treated as a low dielectric constant region with a number of point charges located at atomic centers, and the solvent is treated as a high dielectric constant region. Poisson-Boltzmann equation (PBE)-based implicit solvent model has been one such attempt and been widely used in biomolecular applications.

To solve PBE, numerical solutions are almost always needed for biomolecular applications, since analytic solution of the PBE can be achieved only in a few specific cases with simple solute geometry. Among the numerical solution methods, finite-difference methods (FDM),[1-16] finite-element methods[17-26] and boundary-element methods[27-44] are mostly used. Or alternatively, semi-analytical generalized Born approaches, particularly in MD, were also explored.[7, 9, 11, 12, 45-56] Within numerical methods, FDM has the advantage of being straightforward and physically transparent in its discretization. However, if a direct discretization is used without considering the discontinuity in the dielectric constant, the numerical solutions tend to have large errors, and the errors are particularly obvious near the solute-solvent interface. To overcome this problem, Davis and McCammon proposed a harmonic average (HA) method to approach the approximate dielectric constant near surface in 1991.[57] Efforts have also been reported recently by Wei and co-workers and Li and co-workers to develop higher accuracy interface schemes, the immersed interface method (IIM) and the matched interface and boundary (MIB) method, to improve numerical accuracy of the PBE solution[58-66] Additional higher-order schemes are also developed,[67, 68] some of which are specifically for nonlinear PBE. The idea of IIM is to enforce the interface conditions into the finite-difference schemes at grid points near the interface. On the other hand, the scheme of MIB is enforcing the lowest-order jump condition repeatedly to achieve the high-order jump condition. Alternatively, instead of treating the solute-solvent interface explicitly, Alexov and co-workers proposed an approach that uses Gaussian-based smooth dielectric functions to model the implicit solvation environment in an interface-free manner.[69, 70] Some other interesting approaches have also been proposed to improve implicit solvent models, such as coupling electrostatic and nonelectrostatic interactions within the implicit solvation treatment,[71-75] explicit simulating implicit solvent as a fluid for the purpose of more physical modeling of solvation interactions,[76-78] and using the level set function to better define the solvent and solute interfaces for dielectric assignment.[79, 80] Among those methods, IIM has been a promising high-accuracy numerical scheme, and is able to achieve energy conservation in PB molecular dynamics (MD).[81] This can be attributed, in part, to the fact that a uniform higher accuracy of $O(h^2)$ can be achieved even near the interface.[82]

However, the original IIM has also suffered from numerical instability in complex biomolecular environments.[59] This in part is due to the use of the level set scheme to track the molecular interface and compute associated jump conditions. Here the molecular interface is defined with an analytical density function recently developed by our group.[83] It has been shown that the density function method performs better than geometry-based methods such as van der Waals surface and solvent exclude surface in terms of errors, transferability, conformation dependence, and convergence.[83] The extensive use of the level set scheme in the original IIM, however, was found to introduce noises in interface properties that leads to numerical instability. For example, it uses a least square interpolation approach to obtain all interface parameters and jump conditions.[82] Though this is a general scheme for partial differential equations with interface, it may not always work on the molecular interface formed by densely packed atoms. In addition to the instability issue, the original IIM is also very slow and does not work well even when coarse grids are used in biomolecular applications.[59] This can be attributed to the extensive use of finite-difference interface approximations that may also fail to capture the interface characteristics at coarse grid spacings and lead to inconsistent linear systems that are poorly conditioned and take longer time to solve.

In this study, we present a new implementation of the second-order accuracy IIM to address those issues. Specifically, we show how to calculate geometric properties and jump conditions analytically at the interface. In addition, the necessity and rationality of surface regulation are discussed, along with a simple and efficient regulation scheme. We also present a GPU implementation of the linear solver needed in the IIM, a bottom neck of the IIM. Our analysis shows that the new strategy improves the convergence and stability of

IIM, and also improves the efficiency of the method. The GPU implementation further removes the bottle neck of numerical procedure. These new developments have been implemented into the Amber molecular modeling suite and are freely available for the biomedical community.[84]

1. 2 Method

IIM is a more accurate method for interface treatment than most traditional ones, such as the harmonic average method.[96] In IIM, the interface can be used to separate the problem domain into inside (Ω^{-}), outside (Ω^{+}), and interface (Γ). Next finite-difference grid points are classified as regular grid points that are with all their neighboring grid points (connected with grid edges) in the same region or as irregular points that are not.

Next the jump conditions can be predefined as follows so the PDE is well-posted,

$$[\phi]_{\Gamma} = w$$

$$[\epsilon \phi_{n}]_{\Gamma} = v$$

where w and v are given by users for specific problems at hand. In this study w and v are defined for the singularity-free PBE below.

For regular points, IIM uses the standard 7-point central finite-difference scheme, since those points do not have a neighbor located in a different region. This stencil has an accuracy of ($O(h^{2})$). For irregular points, IIM proposes a new finite-difference stencil that contains at least 10 points instead of 7 (typically 27 points) are used to minimize the magnitude of the local truncation error while satisfying the jump conditions. So the finite-difference scheme becomes,

$$\sum_{m}^{n_s} \gamma_m \phi(i+i_m, j+j_m, k+k_m) = f(i,j,k) + C(i,j,k)$$

where n_s is the number of grid points (27 typically), γ_m are unknown coefficients, C(i,j,k) is the unknown correction term, and f(i,j,k) is the discretized right hand term of the PBE. In this study, it is always zero because the singularity-free PBE is used as discussed later.

The basic idea of IIM is to determine γ_m at the irregular points so that the second-order global accuracy is obtained as in an interface-free problem with the finite-difference/finite-volume discretization scheme. Specifically, IIM uses the interface relations to translate all the outside points into inside (or vice versa) and then forces the truncation error of the inside (or outside) to O(h^3).[119]

The local truncation error T(i,j,k) at grid point (i,j,k) is,

$$T(i,j,k) = \sum_{m}^{n_s} \gamma_m \phi(i+i_m, j+j_m, k+k_m) - C(i,j,k)$$

after setting f(i,j,k)= 0. In the local coordinate frame, the Taylor expansion can be used to expand each ϕ in the neighborhood of the projection point (X^*) of the grid point (i,j,k) from each side of the interface to the second order. Thus there are 20 terms, corresponding to $\phi(X^*)^{\pm}$, $\phi_\xi(X^*)^{\pm}$, $\phi_\eta(X^*)^{\pm}$, $\phi_\tau(X^*)^{\pm}$, $\phi_{\xi\xi}(X^*)^{\pm}$, $\phi_{\xi\eta}(X^*)^{\pm}$, $\phi_{\xi\tau}(X^*)^{\pm}$, $\phi_{\eta\eta}(X^*)^{\pm}$, $\phi_{\eta\tau}(X^*)^{\pm}$, $\phi_{\tau\tau}(X^*)^{\pm}$. Here ξ, η and τ are local coordinates, and "+", "–" denote the outside and inside region, respectively. By using the interface relations shown later,[119] those 20 terms are reduced to 10, all in the inside (or outside) region as shown below.

$$T(i,j,k) = a_1\phi^- + a_2\phi_\xi^- + a_3\phi_\eta^- + a_4\phi_\tau^- + a_5\phi_{\xi\xi}^- + a_6\phi_{\xi\eta}^- + a_7\phi_{\xi\tau}^- + a_8\phi_{\eta\eta}^- + a_9\phi_{\eta\tau}^- \\ + a_{10}\phi_{\tau\tau}^- + T'(i,j,k) - C(i,j,k) + O(\max\{|\gamma_m| h^3\})$$

where $T'(i,j,k)$ is a constant term from imposing the interface relations.

Finally, the expansion is matched against the differential equation to the leading terms to obtain a system of equations for the finite-difference coefficients. For the Poisson equation, we have $a_1 = 0, a_2 = 0, a_3 = 0, a_4 = 0, a_5 = 1, a_6 = 0, a_7 = 0, a_8 = 1, a_9 = 0, a_{10} = 1$. Then the constant term $C(i,j,k)$ can be determined with the accuracy of $O(h^2)$. With 10 equations and 27 unknowns (γ_m), the problem is in principle solvable.[119]
Furthermore, it appears that with 17 extra degrees of freedom, special γ_m can be chosen to fulfill some optimization requirement such as the maximum principle preserving scheme, to achieve even better performance.[119] However, this may not always be the case, which will be discussed latter.

Point charge models are widely used in molecular simulations of biomolecules. However, the representation of point charges by delta functions introduces singularity to the PDE. Several strategies are available to remove the charge singularity,[58, 100, 146-147] and here we adopt the recently developed reaction field potential method.[96]
Briefly this method solves the PDE in two different regions with different potentials, i.e. the reaction field potential inside and the total field potential outside. Here the reaction field potential is the potential caused only by induced charges.[148]
Thus, the unified equation describing the whole region is

$$\nabla \cdot \varepsilon \nabla \phi = 0$$

but with the modified jump conditions as follows,

$$[\phi] = \sum_{charges} \frac{q}{r} = w$$

$$[\varepsilon \phi_n] = \varepsilon_{inside} \left(\sum_{charges} \frac{q}{r}\right)_n = v$$

Here, the subscript n means the gradient is along the interface normal direction n.[96]

To obtain the interface relations and the jump conditions, it is necessary to know the geometry of the interface and the jump condition values on the interface, in addition to the tangential derivatives of the jump conditions. These quantities can be obtained analytically if the density function strategy is used to define the interface.[119]

The main idea of the density function method is to use a smooth function to approximate the solute exclusive surface (SES) of a molecule, and in the meantime adjust the parameters of the density function accordingly so that the reaction field energies obtained from these two models agree the best.[143] Specifically, given the nth atom centered at $\boldsymbol{r}_n$, its density function ρ_n is defined as,

$$\rho_n(x) = \rho_n(\frac{d - r_c}{2r_p})$$

Here, $d=|\boldsymbol{r}-\boldsymbol{r}_n|$ is the distance to the atom center, r_c is the VDW radius of the atom, and r_p is the solvent probe radius.

In addition, the density function satisfies the following constraints to be physical and reasonable,[143]

$$\rho_n(x) > 1, when\ x < 0$$

$$\rho_n(x) = 1, when\ x = 0$$

$$\rho_n(x) < 1, when\ x > 0$$

Here points with x=0 are on the surface of an atom. To guarantee smoothness and good numerical behaviors, a cubic-spline interpolation function that fulfills the above constraints is used. Next, under the requirement of achieving the best agreement of reaction field energies with the given benchmark for a diversified set of training molecules, all the coefficients in the interpolation are optimized and the following formula is obtained.[143]

$$\rho_n(x) = \begin{cases} 1.00 - 4.527143x - 3.640532x^2 + 32.631235x^3 & 0.0 < \mathrm{x} < 0.2 \\ 0.21 - 2.067608x + 15.938209x^2 - 35.500854x^3 & 0.2 < \mathrm{x} < 0.4 \\ 0.15 + 0.047573x - 5.362303x^2 + 13.12218x^3 & 0.4 < \mathrm{x} < 0.6 \\ 0.05 - 0.522686x + 2.511005x^2 - 4.487867x^3 & 0.6 < \mathrm{x} < 0.8 \\ 0.01 - 0.056828x - 0.181716x^2 + 1.079289x^3 & 0.8 < \mathrm{x} < 1.0 \end{cases}$$

Given the atomic density function, the molecular density function is a summation of all the atomic density functions as

$$\rho(x) = \sum_{n}^{atom} \rho_n(x_n)$$

More details can be found in the literature.[143]

Note that in the original IIM, all the geometric and physical interface properties are obtained via the level-set numerical toolkit.[119] However, with the use of the density function, we can obtain all these quantities analytically, including the interface normal direction $\boldsymbol{\xi}$, interface curvatures $\xi_{\eta\eta}$, $\xi_{\tau\tau}$ and $\xi_{\eta\tau}$, jump condition values w and v, and their tangential derivatives on the interface. More importantly, by using this method, the numerical instability arising from applying hard spheres in classical molecular surface representation can also be minimized due to the use of a smoothly varying function.[143]

A key improvement in the new analytical IIM is the complete elimination of the level set functions used to interpolate all interface properties. This is a more natural strategy because the underlining density function to represent the interface is a continuous cubic-spline function. In addition, the jump conditions, i.e. Coulomb potential and field, are also analytically available.

There are roughly three steps to determine all the analytical interface conditions on the interface.[119] The first step is to find the projection point of an irregular grid point. The projection point here means a point right on the interface that is close to the irregular point. This is necessary for the Taylor expansion used to compute the local truncation error. The best choice of projection point is the interface point that is closest to the irregular point. However, it is almost impossible to locate it, and even if we can, it is definitely not worth all the effort.[119] Instead, we used its normal projection point on the interface. The normal projection point can be easily obtained by finding the intersection point of the interface and the steepest descent vector starting from the irregular point (i.e. along the gradient of the density function). Because we are looking along the steepest descent direction, the projection point may also approximate the closest projection point reasonably well. Specifically the Newton's method was used to search along the normal direction for the projection point. One of the advantages of this method is that we can locate the projection point as precise as possible and also relatively efficiently. Specifically, the precision was set to 10^{-6} in our method so that its location would never be an issue in most finite-difference setups, which means,

$$|\rho(projection\ point) - 1| \leq 10^{-6}$$

The second step is to calculate the local coordinate transformation matrix and the interface curvature. This is straightforward given the analytical density function. To calculate the transformation matrix, we need the normal direction of the interface, which is the gradient direction of the density function of the projection point. Next, the other two tangential directions can be determined as follows,[119]

$$\begin{cases} \boldsymbol{\xi} = \left(\rho_x, \rho_y, \rho_z\right)^T \\ \boldsymbol{\eta} = (\rho_y, -\rho_x, 0)^T \\ \boldsymbol{\tau} = \left(\rho_x\rho_z, \rho_y\rho_z, -\rho_x^2 - \rho_y^2\right)^T \quad , \quad if\ \rho_y^2 \geq \rho_z^2 \\ \\ \boldsymbol{\xi} = \left(\rho_x, \rho_y, \rho_z\right)^T \\ \boldsymbol{\eta} = (\rho_z, 0, -\rho_x)^T \\ \boldsymbol{\tau} = (-\rho_x\rho_y, \rho_x^2 + \rho_y^2, -\rho_y\rho_z)^T \quad , \quad otherwise \end{cases}$$

The three principal curvatures can then be obtained as follows. Under the local coordinates of the interface, the following relations hold,

$$\rho_{\boldsymbol{\eta}}(0,0,0) = \rho_{\boldsymbol{\tau}}(0,0,0) = 0$$

Based on that, the three curvatures are,

$$\xi_{\eta\eta} = -\frac{\rho_{\boldsymbol{\eta\eta}}}{\rho_{\boldsymbol{\xi}}}$$

$$\xi_{\tau\tau} = -\frac{\rho_{\boldsymbol{\tau\tau}}}{\rho_{\boldsymbol{\xi}}}$$

$$\xi_{\eta\tau} = -\frac{\rho_{\boldsymbol{\eta\tau}}}{\rho_{\boldsymbol{\xi}}}$$

The last step is to calculate all those jump conditions. Given above equations, we obtained the following equations of derivatives.

$$w_\eta = \sum_i \frac{q_i}{{r_i}^3}\eta_i$$

$$w_\tau = \sum_i \frac{q_i}{{r_i}^3}\tau_i$$

$$w_{\eta\eta} = \sum_i 3\frac{q_i}{{r_i}^5}{\eta_i}^2 - \frac{q_i}{{r_i}^3} + \frac{q_i}{{r_i}^3}\xi_i \cdot (\xi_{\eta\eta})_i$$

$$w_{\tau\tau} = \sum_i 3\frac{q_i}{{r_i}^5}{\tau_i}^2 - \frac{q_i}{{r_i}^3} + \frac{q_i}{{r_i}^3}\xi_i \cdot (\xi_{\tau\tau})_i$$

$$w_{\eta\tau} = \sum_i 3\frac{q_i}{{r_i}^5}\eta_i\tau_i + \frac{q_i}{{r_i}^3}\xi_i \cdot (\xi_{\eta\tau})_i$$

$$v_\eta = \sum_i 3\frac{q_i}{{r_i}^5}\xi_i\eta_i$$

$$v_\tau = \sum_i 3\frac{q_i}{{r_i}^5}\xi_i\tau_i$$

The summation here is over all the charges or atoms. Now all the quantities needed are obtained, the interface relations can be constructed as,

$$\phi^+ = \phi^- + w$$

$$\phi_\xi^+ = \frac{\varepsilon^-}{\varepsilon^+}\phi_\xi^- + \frac{v}{\varepsilon^+}$$

$$\phi_\eta^+ = \phi_\eta^- + w_\eta$$

$$\phi_\tau^+ = \phi_\tau^- + w_\tau$$

$$\phi_{\xi\xi}^+ = \frac{\varepsilon^-}{\varepsilon^+}\phi_{\xi\xi}^- + \left(\frac{\varepsilon^-}{\varepsilon^+} - 1\right)\phi_{\eta\eta}^- + \left(\frac{\varepsilon^-}{\varepsilon^+} - 1\right)\phi_{\tau\tau}^- + \left(\phi_\xi^+ - \phi_\xi^-\right)\left(\xi_{\eta\eta} + \xi_{\tau\tau}\right) - w_{\eta\eta} - w_{\tau\tau}$$

$$\phi_{\xi\eta}^+ = \frac{\varepsilon^-}{\varepsilon^+}\phi_{\xi\eta}^- + \left(\phi_\eta^+ - \frac{\varepsilon^-}{\varepsilon^+}\phi_\eta^-\right)\xi_{\eta\eta} + \left(\phi_\tau^+ - \frac{\varepsilon^-}{\varepsilon^+}\phi_\tau^-\right)\xi_{\eta\tau} + \frac{v_\eta}{\varepsilon^+}$$

$$\phi_{\xi\tau}^+ = \frac{\varepsilon^-}{\varepsilon^+}\phi_{\xi\tau}^- + \left(\phi_\eta^+ - \frac{\varepsilon^-}{\varepsilon^+}\phi_\eta^-\right)\xi_{\eta\tau} + \left(\phi_\tau^+ - \frac{\varepsilon^-}{\varepsilon^+}\phi_\tau^-\right)\xi_{\tau\tau} + \frac{v_\tau}{\varepsilon^+}$$

$$\phi_{\eta\eta}^+ = \phi_{\eta\eta}^- + \left(\phi_\xi^- - \phi_\xi^+\right)\xi_{\eta\eta} + w_{\eta\eta}$$

$$\phi_{\tau\tau}^+ = \phi_{\tau\tau}^- + \left(\phi_\xi^- - \phi_\xi^+\right)\xi_{\tau\tau} + w_{\tau\tau}$$

$$\phi_{\eta\tau}^+ = \phi_{\eta\tau}^- + \left(\phi_\xi^- - \phi_\xi^+\right)\xi_{\eta\tau} + w_{\eta\tau}$$

Even if the density function surface was designed to handle the deeply buried atoms as the standard SES approach, it is still limited in approximating the solvent reentry interface formed among solvent accessible atoms.[143] For example, the reentry interface is often flatter than what the density function predicts, as shown in Figure 1. In addition, it is almost impossible to model the reentry surface as being formed by a spherical probe as in SES. We thus resort to a surface regulation scheme to adjust the interface geometric parameters to partially alleviate the limitations.

Our regulation scheme follows two principles. First, we exploited the original SES idea that the molecular interface is simply a surface formed by probing the molecules of hard spheres of different radii. Thus, the interface is a union of solvent-exposed spheres of different radii. Second, the inward-facing portion (aka solvent reentry portion) of the interface cannot have a surface curvature higher than that of solvent probe. Thus, a regulation scheme was developed with the following two steps for the solvent reentry portion, i.e. when the surface curvature is negative. First, if the curvature of the surface is larger than $1/r_p$, it is reset to $1/r_p$, where r_p is the solvent probe radius (Figure 1.1). Second, the mixing curvature $\xi_{\eta\tau}$ is set to be zero, and the other two curvatures is set to equal to whichever has a smaller absolute value. This step implies the use of a sphere to represent the reentry surface. These changes were found to improve the numerical stability and convergence of the IIM calculations, especially for coarse grid situations, as discussed in Results and Discussion.

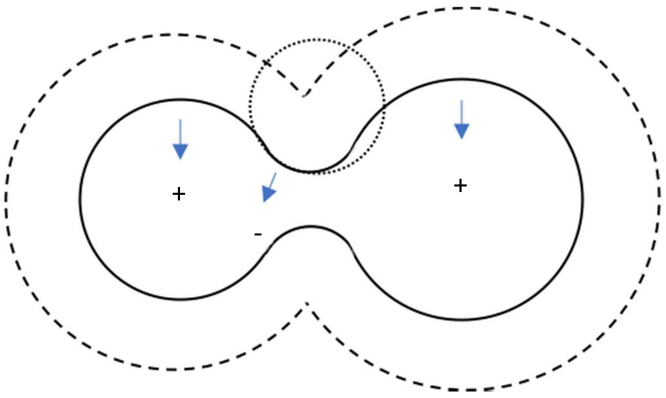


Figure 1.1. Treatment of the reentry surface. The solid line represents the solvent excluded surface. The dash line represents the solvent accessible surface. The dot circle represents the solvent probe. "+" signs represent surface elements with positive curvature and "-" signs represent surface elements with negative curvature.

Apparently, the whole process of solving PBE with IIM involves many procedures. Here we outline the main numerical procedures step by step.

1) Use the molecular density function to classify grid points into regular and irregular points;

2) For regular points, set up the discretized equation with the standard seven-point stencil;

3) For irregular points, calculate all the needed interface geometry parameters (e.g. curvatures) using the density function;

4) Apply the regulation scheme to revise the obtained interface geometry parameters;

5) For irregular points, continue calculating the needed jump conditions, i.e. w and v, analytically using the density function and the geometry parameters just obtained;

6) Set up the discretized equation for irregular points using the obtained geometry parameters and jump conditions;

7) Combine the equations of both regular and irregular points and solve the full linear system to obtain the potentials;

In the above procedures, steps 3), 4), and 5) were completely rewritten in our analytical implementation of IIM. Step 7 was also rewritten for both CPU and GPU calculations.

To validate the accuracy and precision of the proposed method, we first used a well-studied analytical model, i.e. a single dielectric sphere imbedded with point charges. The analytical potential in the inside region is,

$$\phi_{RF}^{-}(r,\theta,\varphi) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{4\pi}{2l+1}\frac{1}{R^{2l+1}}\frac{(l+1)(\varepsilon^{-}-\varepsilon^{+})}{\varepsilon^{-}(l\varepsilon^{-}+(l+1)\varepsilon^{+})}Q_{lm}r^{l}Y_{lm}(\theta,\varphi)$$

$$Q_{lm} = \sum_{k=1}^{N_q} q_k r_k^l Y_{lm}^{*}(\theta_k,\varphi_k)$$

where R is the radius of the sphere, and the center is set to the origin. N_q is the number of charges, Y_{lm} are the spherical harmonics as spherical coordinates are used.

After solving the finite-difference equations, only potentials at grid points are known. To obtain potential at any position (x,y,z), we utilized the one-side least-square interpolation method.[96] Next the reaction field energy is the summation of the products of the reaction field potential and the charges.

$$\Delta G = \frac{1}{2}\sum_{charges} q_i\phi_{RF}$$

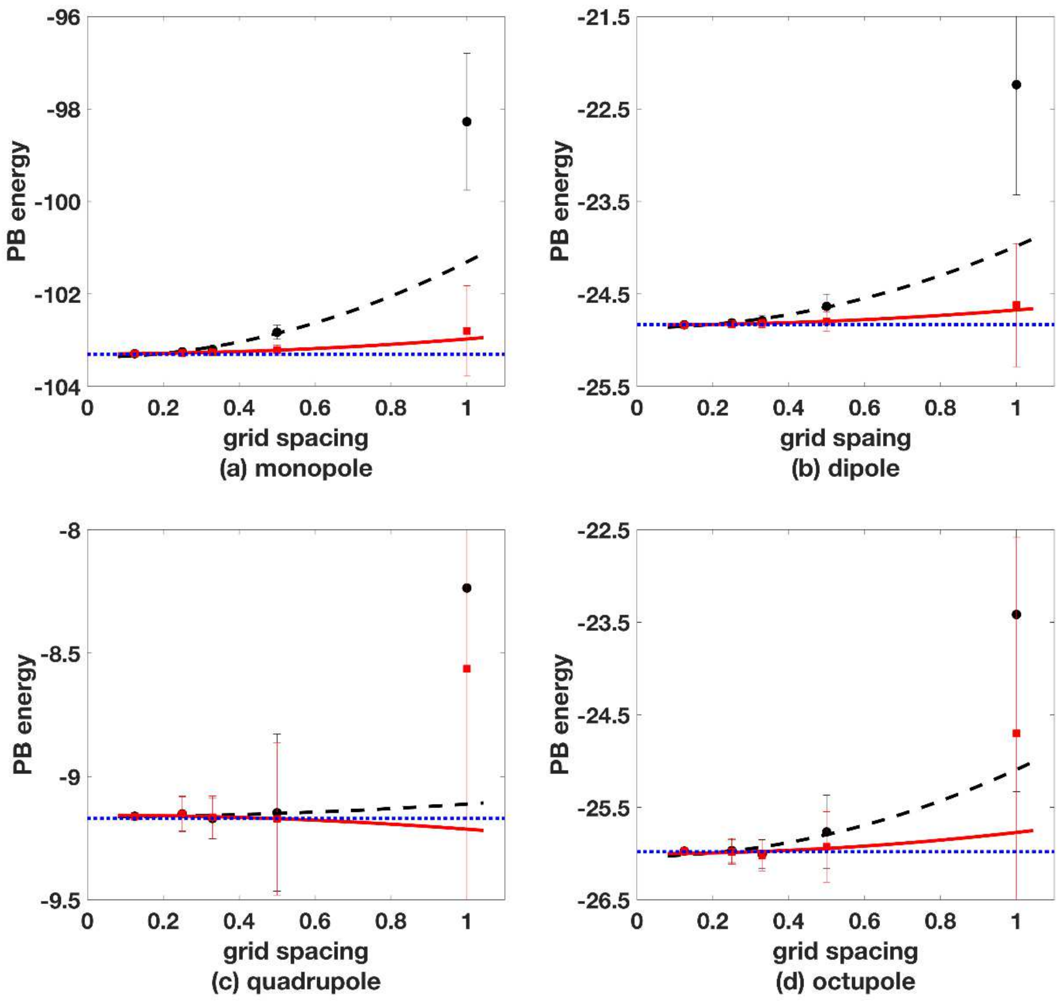
For the analytical test, four off-centered charged models were used, which are monopole, dipole, quadrupole, and octupole. The radius of sphere was set to 2 Å for convenience, which is about the size of a united carbon atom. The Cartesian coordinates of each charges are listed in Table S1. The inside dielectric constant was set to 1.0, with that of outside set to 80.0. The truncation order l in ϕ_{RF} was chosen to be 120, leading to a precision of 10^{-6}.

A total of 573 biomolecular structures from the Amber PBSA benchmark suite were used to study the accuracy and efficiency of the new method.[50] We further chose eight small proteins about 1000 atoms from the benchmark suite to analyze the convergence of the new method, so that the jobs can be handled on our local compute nodes at the finest grid spacing tested. These biomolecules were assigned charges of Cornell *et al*[149] and the modified bond radii. The probe radius was set to 1.4 Å. All testing runs were performed with the following conditions unless specified otherwise. The convergence criterion of 10^{-4} was used for the BiCG linear system solver. The default grid spacing was 0.5 Å for most calculations, except that they were set as 1.0 Å, 0.5 Å, 0.33 Å, 0.25 Å, and 0.125 Å in the convergence analysis. The ratio of the grid dimension over the solute dimension (the *fillratio* keyword in Amber) was set to 1.5 and 4.0 for biomolecular tests and analytical tests, respectively. No electrostatic focusing was applied. The potential values on all grid points were initialized to zero. The conductor boundary, i.e. zero potential, was used in all non-periodic PBSA calculations. The dielectric constants were set to 1 and 80 for solute and solvent, respectively. All other parameters were set as default in the PBSA module in Amber 18 package.[150]

A BiCG linear system solver for GPUs was also implemented using the Nvidia CUDA Sparse Matrix (cuSPARSE) library, which provides basic linear algebra procedures for sparse matrix operations.[151] The CSR matrix format was used for the non-symmetric coefficient matrix as in our previous publication.[152] All GPU and CPU tests were conducted on a dedicated compute node with two NVIDIA TITAN Xp GPU cards, one Intel Xeon E5-1620 v3 CPU, and 16GB main memory. Our time measurements for both solvers include all execution time of the solver routines, i.e. time elapsed on device (GPU) and on host (CPU) and also for transferring data between the device and the host.

1.2 Results and Discussion

The accuracy and precision of reaction field energies were investigated with five different grid spacings, 1 Å, 0.5 Å, 0.33 Å, 0.25 Å and 0.125 Å. The detailed results of different programs with and without analytical setup are shown in Figure 1.2, Table 1.1.



— analytical setup --- numerical setup

⋯ analytical value

Figure 1.2. Convergence of reaction field energies (kcal/mol) versus grid spacing (Å) for analytical sphere models with both numerical and analytical setups for IIM. The energy

results are obtained by averaging 30 grid orientations/offsets for each test case. All the curves are obtained by fitting data to parabolas. Note that the data points at grid spacing 1 Å are not included in the fitting.

Clearly, the analytical setup obtains better accuracy than the numerical setup in all the cases regardless of the grid spacing. Interestingly, the analytical setup version can obtain a good result even with coarse grid, for example in 1 Å spacing. In Figure 2 (a) and (b), though not included in the fitting process, data points at grid spacing of 1 Å are close to the fitting curves in the analytical setup, showing that those data have already entered the convergence region. However, for numerical setup, there is no sign of entering convergence region for data at 1 Å for any test cases. Table 1 further shows that the relative errors are already less than 0.1% for all the cases at the grid spacing of 0.5 Å when the analytical setup was used, much less than those of the numerical setup with the medium value of about 0.7%. This indicates that the analytical setup leads to faster converged results than the numerical setup.

Table 1.1. Reaction field energies (kcal/mol) for the analytical sphere models with the analytical setup. Each energy is an average of 30 random placements of the finite difference grid over a tested model. Analytical reaction field energies were calculated using Mathematica 7.0.

Spacing	1.000Å	0.500Å	0.330Å	0.250Å	0.125Å	Analytical
Monopole	-102.80	-103.22	-103.26	-103.28	-103.29	-103.30
Dipole	-24.62	-24.80	-24.81	-24.83	-24.83	-24.83
Quadrupole	-8.56	-9.17	-9.17	-9.15	-9.16	-9.17
Octupole	-24.70	-25.93	-26.02	-25.98	-25.98	-25.98

Surface properties are often important for calculation of electrostatic forces, so we used the electric field at interface to show how well the interface properties can be reproduced in both setups. The results from different setups are presented in Figure 1.3. The root mean square deviation (RMSD) values were calculated by comparing values from different interpolation schemes and the theoretical values, where the three interpolation methods used are those studied in detail in the literature.[153]

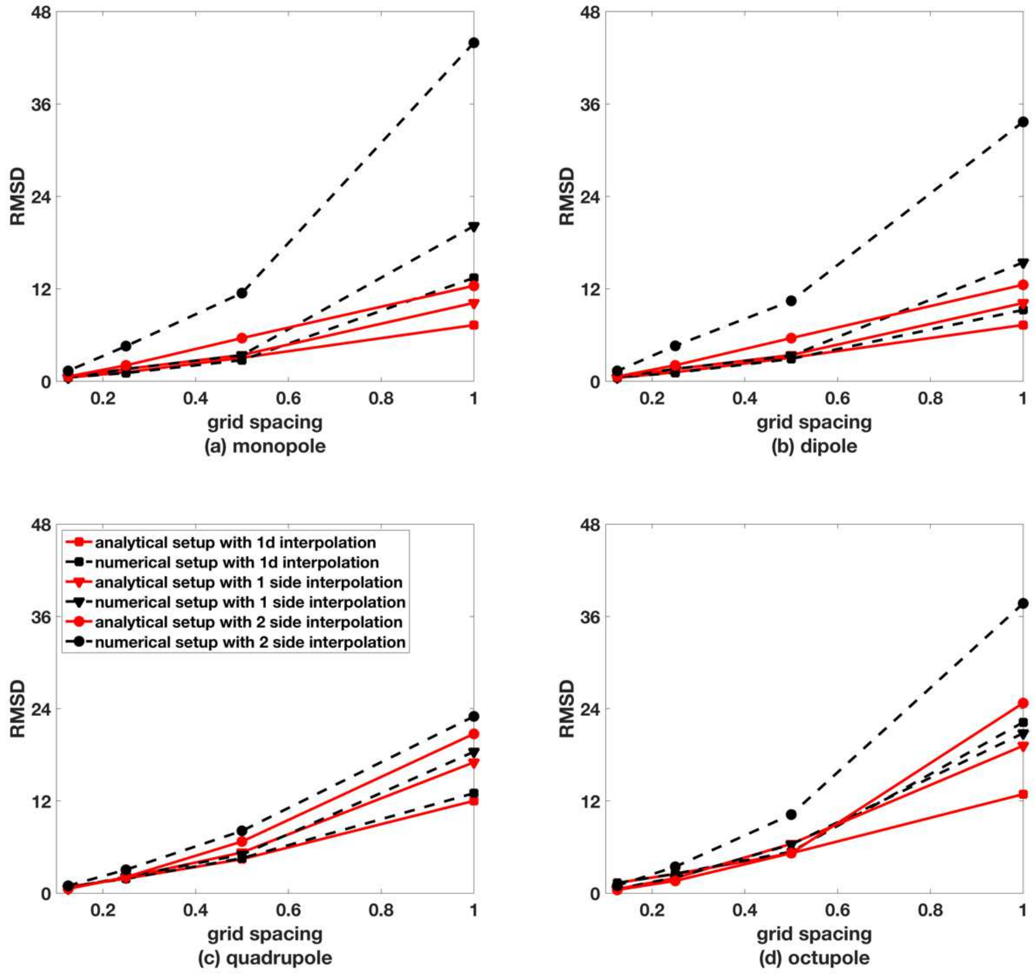
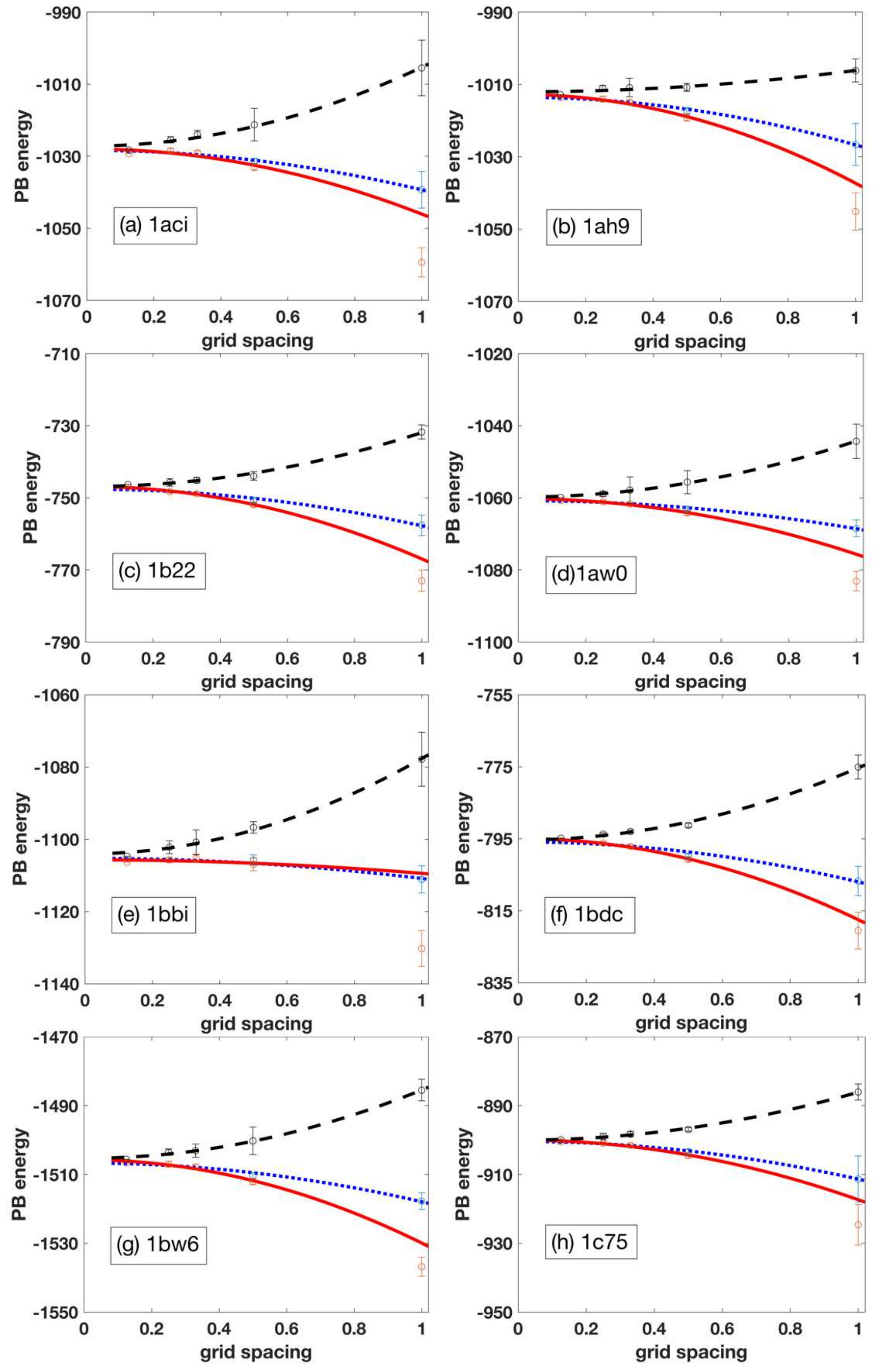


Figure 1.3. RMSDs between computed normal component of the surface electric field with analytical values (kcal/mol-e-Å) versus grid spacing (Å) for analytical sphere models with both numerical and analytical setups for IIM.

It is clear that for all the three interpolation schemes, the analytical setup always gives better agreement with theory. Of all the interpolation schemes used, the 1-d interpolation shows the best performance, which is consistent with the literature observation.[153] Interestingly for some cases, the worst results for the analytical setup can even beat the best results for the numerical setup.

We conducted convergence analysis of the PB reaction field energy versus grid spacing for the IIM with different setups. The results for the eight chosen proteins are shown in Figure 1.4. IIM with both analytical and numerical setups converge at very fine grids in a unified quadratic manner, as predicted by the theory. The converged values are consistent with our previously implemented solvers, and also consistent with other existing solvers as in DelPhi and APBS, and higher order solver such as MIBPB.[154] However, the convergence curves do not show a significant improvement after we analytically computed all interface parameters and jump conditions. Indeed, some of the test cases were found to lead to even worse results (as in case (b) of Figure 1.4).

The reason is that we do not have proper interface parameters. When numerically solving PDEs for systems with interfaces, interface parameters must converge in a quadratic form, so we can guarantee the whole numerical scheme converges in a quadratic manner. However, this does not hold for the analytical setup. Even if the analytical setup can obtain extremely accurate interface parameters (with relative error of 10^{-6}) on each projection point, the projection point may not necessarily be a good representation of its neighborhood on the interface. Therefore, when we resort to a finer grid spacing, a new projection point in this neighborhood may give a set of totally different interface parameters (even the sign may be different for curvatures due to the crowded atomic packing in biomolecules), causing the method fail to converge in a quadratic manner. Indeed, if we include the data at the grid spacing of 1.0 Å in the fitting curves, the asymptotic energy would converge to a wrong value. This phenomenon shows that the data at the space 1.0 Å have not entered the convergence region, thus leading to poor performance with the analytical setup. We believe this is because the interface parameters do not converge in a quadratic form, which motivated us to develop a surface regulation scheme.



— analytical setup --- numerical setup

⋯ analytical setup with surface regulation

Figure 1.4. Convergence trends for reaction field energies of nontrivial biomolecules (kcal/mol) versus grid spacing (Å). The energy results are obtained by averaging of systematic 30 rotations/offsets of the tested molecules. All the curves are obtained by fitting data at discrete grid point to a parabola ($y=a+bx^2$). (Note that for the analytical setup without surface regulation, data at grid spacing equal to 1 Å are not included in the fitting.)

That better interface parameters are required to unleash the full potential of the analytical setup can also be illustrated in the following rationale. Nontrivial biomolecules in our test cases contain atoms with radii as small as 0.6 Å. Thus, a sampling grid spacing of 1.0 Å or 0.5 Å is inadequate to capture the surface curvatures ($\xi_{\eta\eta}$, etc). In the numerical IIM scheme, the interface properties (such as curvatures) are not accurate as they would be in the analytical IIM, because certain fuzziness from the use of the coarse grid is introduced within the neighborhood of the projection point.[119] Coincidently, however, these inaccurate properties lead to a better representation of the surface for the numerical IIM when the coarse grid is used (Figure 1.4). Therefore, for a coarse grid such as of 1.0 Å, the accuracy of the IIM scheme cannot achieve what the theory best predicts and thus the data deviate from the fitting curves. Thus, a better way to deal with coarse grids is to adopt some fuzziness for the surface description. We implemented the surface regulation to introduce some “inaccuracy” but more “representativeness” into the analytical IIM for coarse grid spacings such as 1.0 Å.

As shown in Figure 1.4, the convergence performance of the analytical setup with surface regulation is much better than the results without regulation and the data follow the quadratic pattern more closely. Worth noting is that the PB energies at the grid spacing of

1.0 Å enter the convergence region when the surface regulation is used. In addition, the analytical setup makes it converge faster (flatter convergence curve) than the numerical setup for every test case. The detailed discussion is shown below.

For the widely used coarse grid spacing of 1.0 Å, the analytical setup without regulation does not show an improvement. The maximum, medium and minimum relative errors of reaction field energy at 1.0 Å for the eight tested proteins are 3.51%, 2.90% and 2.05%, respectively (Table 1.2). Those for the numerical setup are 2.54%, 1.79%, and 0.58%, respectively. It can be seen that the analytical setup alone does not improve the convergence for complex interfaces. The reason is that those projection points do not have enough representativeness as we discussed above. However, for analytical setup with regulation, those three errors are 1.22%, 1.03%, and 0.49%, respectively (Table 1.3), with the accuracy almost doubled for all chosen cases. This shows that by introducing surface regulation, we successfully made the interface more representative and thus improved the convergence properties. As for the widely used coarse grid spacing of 0.5 Å, the convergence improvement is similar to that of 1.0 Å, with the convergence of the analytical setup with regulations is also roughly twice better than the other two schemes.
Since our method is developed to balance accuracy and efficiency as it is mostly used to process tens of thousands of structures to post-process MD trajectories, such as in MMPBSA calculations, it is essential to make sure the PB energy error is negligible compared to the intrinsic error of MMPBSA, which is usually about 10% as discussed in one of our recent papers.[155] For the recommended grid spacing of 0.5 Å, the relative error of PB energies with respect to the limiting values at 0 Å is about 0.5% (Table 1.3), which is

twenty times smaller than the intrinsic error and clearly negligible. It is also interesting to compare with some existing higher-order solvers, such as MIBPB, which showed a relative error of about 0.2% at their recommended grid spacing of 1 Å with respect to the finest tested grid of 0.2 Å.[154] This is basically the same if we also compare the values with those at 0.2 Å instead of 0 Å for the analytical IIM. In summary the accuracy of both our second-order solver and higher-order solvers is sufficient for post-processing methods that are based on intensive PBSA calculations. While on the other hand, the trade-off in accuracy and efficiency implies that our second-order solver shall benefit more in CPU intensive applications.

Table 1.2. Reaction field energies (kcal/mol) of eight selected proteins with analytical setup.

spacing	1aci	1ah9	1b22	1aw0	1bbi	1bdc	1bw6	1c75
1.000Å	-1059.46	-1045.14	-772.98	-1083.10	-1130.23	-820.50	-1536.82	-924.65
0.500Å	-1032.95	-1019.10	-751.89	-1064.22	-1106.95	-800.71	-1512.05	-904.49
0.333Å	-1029.06	-1014.96	-748.88	-1061.70	-1105.35	-797.09	-1507.84	-901.55
0.250Å	-1028.47	-1014.02	-748.30	-1061.11	-1105.72	-796.15	-1507.01	-900.93
0.125Å	-1029.26	-1013.55	-747.08	-1060.72	-1106.34	-795.56	-1506.63	-900.68
lim values	-1028	-1013	-746.8	-1060	-1106	-794.9	-1506	-900.0

Table 1.3. Reaction field energies (kcal/mol) of eight selected proteins with analytical setup and the regulation treatment.

spacing	1aci	1ah9	1b22	1aw0	1bbi	1bdc	1bw6	1c75

1.000Å	-1035.44	-1023.20	-754.93	-1065.92	-1108.35	-804.33	-1514.27	-908.97
0.500Å	-1028.61	-1014.24	-748.51	-1061.08	-1103.45	-797.99	-1507.33	-901.28
0.333Å	-1025.94	-1011.44	-746.64	-1059.50	-1102.77	-795.28	-1504.58	-899.36
0.250Å	-1026.06	-1011.24	-746.53	-1059.39	-1103.41	-794.77	-1504.37	-899.20
0.125Å	-1027.52	-1011.6	-745.98	-1059.60	-1104.72	-794.70	-1504.93	-899.54
lim values	-1026	-1011	-745.9	-1059	-1103	-794.6	-1504	-898.7

Another point worth addressing is the numerical uncertainty of numerical reaction field energies. As stated previously, the final PB energy is an average over PB energies of 30 different grid orientations/offsets of our test molecule. The standard deviations of collected data can be used to show the numerical uncertainty of the algorithm. Table 1.4 shows the standard deviations of the 30 energies computed at grid spacings of 0.5 Å and 1.0 Å. For the numerical setup at grid spacing of 1.0 Å, the maximum, medium and minimum fluctuations among the 8 tested proteins are 7.70, 3.27, and 1.93 kcal/mol, respectively. Those for the analytical setup at the same grid spacing are 5.87, 4.51 and 2.68 kcal/mol, respectively. And for the analytical setup with regulation, they are 3.25, 2.62 and 1.86 kcal/mol, respectively. In general, the energy of the numerical setup may fluctuate about 2~3 times larger than the analytical setup, which clearly shows the analytical setup is much more stable and reliable than the numerical setup. Similar behaviors were also observed with the grid spacing of 0.5 Å. The regulation scheme almost always shows a smaller fluctuation than the other two, and the performance is roughly 2~3 times better as shown in Table 1.4. Thus it is clear that both the convergence accuracy and the numerical fluctuation of our new analytical IIM solver are in the order of a few tenths of a percentage point at the recommended grid spacing of 0.5 Å, and also with higher efficiency.

Table 1.4. Reaction field energies standard deviation (kcal/mol) of eight selected proteins with three setups at grid spacing of 0.5 Å and 1.0 Å.

	1aci	1ah9	1b22	1aw0	1bbi	1bdc	1bw6	1c75
0.5 Å								
numerical	4.529	1.070	1.209	3.214	1.571	0.509	4.015	0.532
analytical	1.017	1.037	0.660	0.721	1.782	0.628	0.944	0.821
regulation	0.838	1.046	0.707	0.570	1.332	0.591	0.971	0.722
1.0 Å								
numerical	7.704	3.202	1.934	4.743	7.476	3.332	3.128	2.333
analytical	4.024	5.191	2.938	2.678	4.997	5.154	2.762	5.873
regulation	3.250	2.937	2.752	2.292	2.768	1.885	2.495	1.856

Figure 1.5 shows the timing comparison between two different setups for IIM. It is clear that the analytical setup is faster in all the cases, and the improvement in the solver time is significant for grid spacings of widely used 1.0 Å and 0.5 Å, for which the performance of analytical setup is five-times/double of that of the numerical setup on average. We believe this is because the analytical setup gives much more precise coefficients for the final discretized equations, leading to better-conditioned linear systems. Thus, the numerical solver takes fewer iterations to reach the convergence. Because the solver phase is the most time-consuming part in these programs as shown in Table 1.5, this leads to a dramatic

reduction in the overall running time. For finer grid spacing of 0.25 Å, the speedup is not that significant any more, which is understandable, as the difference between analytical setup and numerical setup becomes negligible.

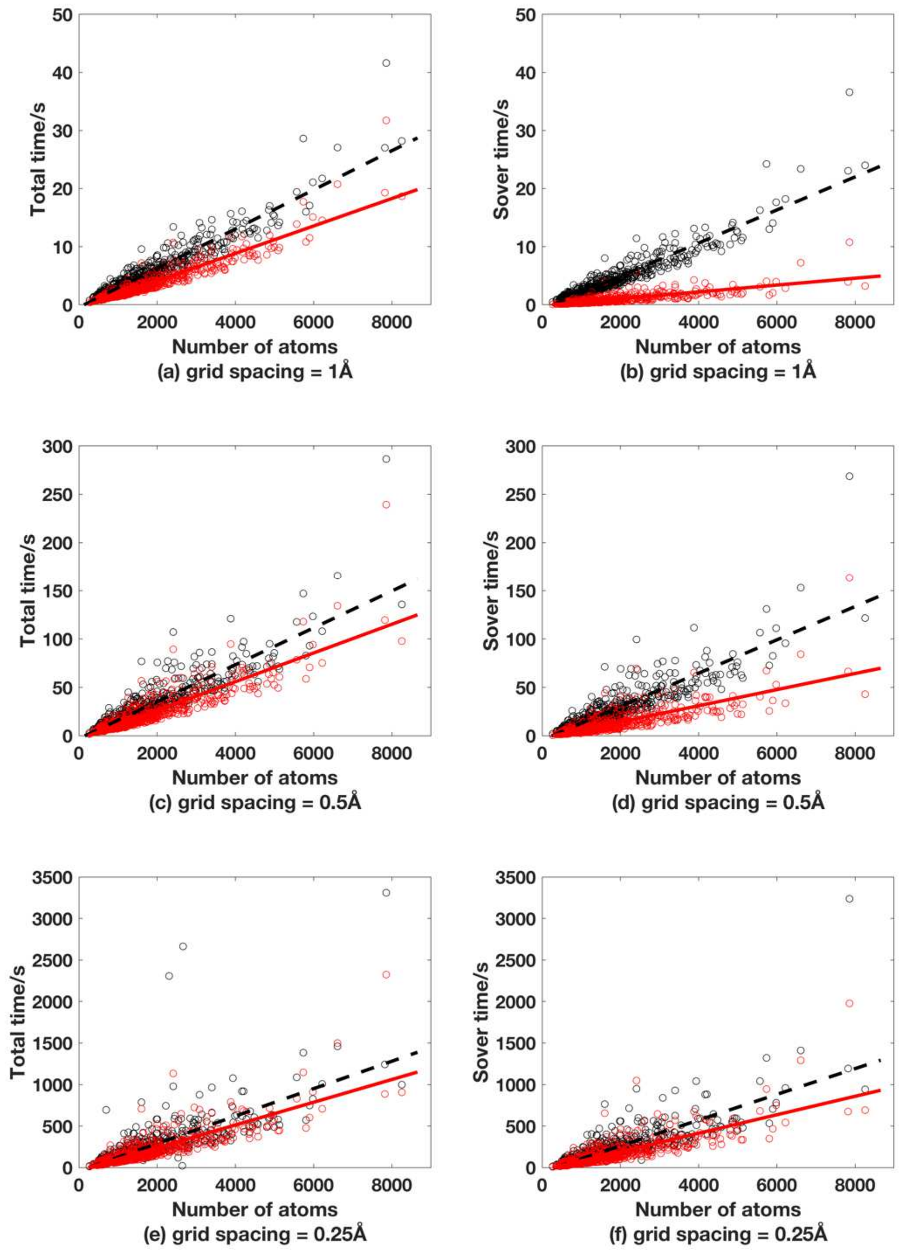
Table 1.5. Percentages of the average solver time in IIM runs for the protein test set.

grid spacing	1Å	0.5Å	0.25Å
numerical setup	74%	82%	90%
analytical setup	23%	51%	78%

Figure 1.5 also shows that the analytical setup can be more stable. For spacing of both 1Å and 0.5Å cases, the timing data are more concentrated near the fitting curves, which shows a more unified scaling behavior. Besides, for the protein test set, 3 (out of 573) cases failed for the numerical setup (bad data or NaN for PB energies), but none for the analytical setup, as showed in Table 1.6. We further confirmed this result with the analytical setup tests in which no case failed. This is consistent with the observations in other analyses of the methods.

Table 1.6. Failed test cases for the IIM runs with numerical setup.

grid spacing	1Å	0.5Å	0.25Å
Failures	1lyp	1pih	2jhb
	1dip	1mut	1hyk
		1e6q	1gnc



— analytical setup --- numerical setup

Figure 1.5. Timing comparison between two different setups (without regulation).

Finally, we implemented the BiCG linear system solver for the IIM, which is the most time-consuming portion of the algorithm as shown in Table 5. The results with the protein test set are showed in Figure 1.6. The PB energies agree excellently between GPU and CPU programs, with the medium relative error of 0.07% for the test set. The timing of the GPU solver also shows an impressive performance boost here, with the speedup ratio of about 20. The implementation on GPU makes the analytical setup IIM program more feasible for practical biomolecular applications.

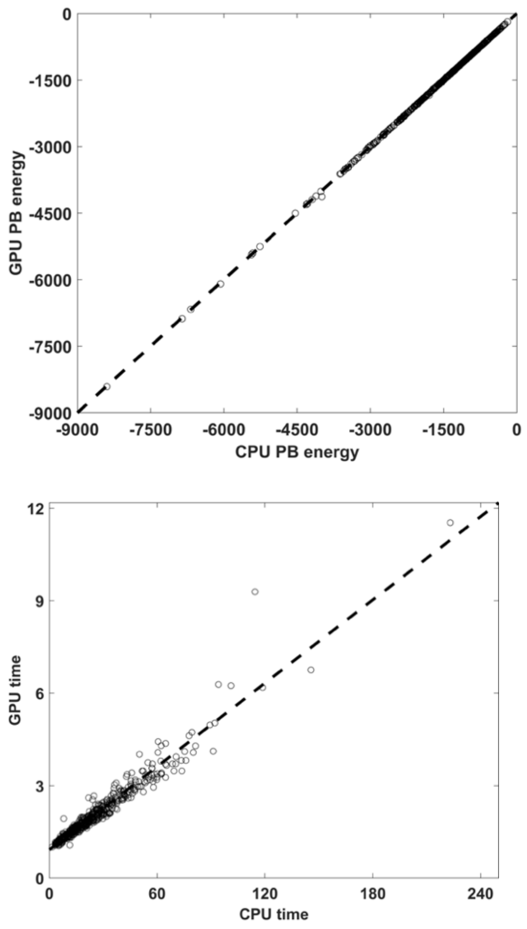


Figure 1.6. Comparison of PB energies (kcal/mol) and solver timing (seconds) between GPU and CPU runs. The energy trend line is 1.001x+1.088, with the medium relative deviation between the two sets 0.066%. The timing trend line is 0.045x+0.921, with the GPU program about 20 times faster than the CPU program.

1.4 Conclusion

In this study, we introduced both analytical setup of interface conditions and surface regulation into the IIM to address its instability and slow convergence. The analytical setup was found to obtain more accurate solutions than the numerical setup in the analytical test cases. The method was found to be able to obtain a good result even with coarse grid spacings, like 1 Å. At the widely used grid spacing of 0.5 Å, the analytical setup can also lead to much smaller relative errors (less than 0.1%) than the numerical setup. Overall the analytical setup converges in a quadratic manner, as predicted by theory.

The surface regulation scheme further speeds up the convergence for nontrivial biomolecules. For the widely used coarse grid spacings of 0.5 and 1.0 Å, the analytical setup with the surface regulation was found to roughly halve the relative errors from the numerical setup. Specifically, the relative error of PB energies at 0.5 Å is about 0.5% with respect to the limiting values at 0 Å or about 0.2% with respect to the values computed at 0.2 Å. These are similar to those of the higher-order solvers such as MIBPB at their recommended grid spacing. In addition, the numerical uncertainty of the PB energies is reduced to a few tenth of a percentage point of the mean. In summary both numerical error and uncertainty of the new analytical IIM method are much smaller than the thermal noise of typical room-temperature MD simulations, for which the PB method is often used as a post-processing tool.

More interestingly the analytical setup significantly improves the solver efficiency at the tested coarser grid spacings. For grid spacings of 1 Å and 0.5 Å, the solver time was decreased by 5x and 2x, respectively. This is because the analytical setup generates more precise coefficients for the final discretized equations, leading to better-conditioned linear systems. This also leads to a more stable algorithm. In our protein test set over 500 proteins, none of them failed with the analytical setup. Finally, we implemented the GPU BiCG solver for the IIM. The PB energies agree excellently between GPU and CPU programs, and the timing analysis shows that the GPU solver is about 20 faster than the CPU solver, which makes the analytical setup IIM program more feasible for practical biophysical studies.

Chapter 2

2.1. Introduction

Electrostatic interactions are of great significance in biophysical processes, such as protein-protein and protein-ligand associations. Thus, accurate and efficient treatment of electrostatics is crucial to computational studies of biomolecular structures, dynamics, and functions. Since water molecules are always involved in biophysical processes, effectively modeling their electrostatic interactions with biomolecules has been under active exploration. Implicit solvation modeling scheme has been such an attempt, which models solvent molecules in a structureless dielectric medium but biomolecular solute in atomic details. In this scheme, the solvent and solute molecules are often modeled as high and low dielectric regions, respectively, or as Gaussian-based smooth dielectrics in an interface-free manner[1-3]. Because most atoms are used to represent solvent molecules in explicit all-atom simulations of biomolecular systems under physiological conditions, modeling solvent molecules implicitly would allow higher computational efficiency without sacrificing the atomic resolution for the biomolecules. Among all the attempts, Poisson-Boltzmann equation (PBE) based implicit solvent models have proven to be among the most successful and are widely used in computational studies of biomolecules.

For biomolecular applications, numerically solving PBE is always inevitable because its analytical solution is only possible in a few specific cases with extremely simple solute geometries. Beside semi-analytical generalized Born approaches,[4-15] most common numerical strategies include the following: finite-difference methods (FDM),[16-31] finite-element methods[32-41] and boundary-element methods.[42-59] FDM, out of its simplicity and physical transparency in the discretization process, is one of the most popular schemes.

However, if a direct discretization is used without considering the discontinuity of the dielectric constant, the numerical solutions tend to have large errors, especially at locations near the solute-solvent interface. In order to overcome this difficulty, Davis and McCammon proposed a harmonic average (HA) method to alleviate the dielectric discontinuity near the interface in 1991.[60] Since then further efforts have also been reported to develop better interface schemes, such as the immersed interface method (IIM) by Li and co-workers,[61-64] and the matched interface and boundary (MIB) method by Wei and co-workers.[65-68] The idea of IIM is to enforce the interface conditions into the finite-difference schemes at grid points near the interface; while that of MIB is to enforce the lowest-order jump condition repeatedly to achieve an high-order jump condition. In addition to these higher-order numerical methods, some other approaches have also been proposed to improve the implicit solvent models, such as using level set functions to define the interfaces for dielectric assignment,[69-71] and coupling electrostatic and nonelectrostatic interactions within the implicit solvation treatment.[72-79]

Apparently higher order interface schemes such as IIM and MIB in principle would lead to higher accuracy for PBE applications.[80-81] However, they are expensive to use. For example, the IIM is quite involving for complex biomolecular environment due to its extensive use of the level set method to compute interface geometric properties.[61-64] This is because the second-order solvers (e.g. IIM and MIB) are usually very liberal in the use of grid points nearby the interface to improve the numerical accuracy.[61-68] For example IIM uses 27 grid points for interface grid points, leading to 27-banded matrices. This is different from the first-order HA method, which only uses six nearest grid points to set up discretized linear equations with 7-banded matrices. Interestingly, the HA method shows surprisingly good global convergence property, particularly at typically used coarse grids (e.g. 0.5 Å). However, even with the advantages of simplicity and effectiveness, the HA method is clearly limited due to its lower accuracy. Thus, it would be of great interest if the accuracy of simpler 7-point-stencil methods such as HA can be further improved. Furthermore, the 7-banded matrix feature is also advantageous for high-performance computing platforms, such as graphics processing units (GPUs).[82] GPUs have a parallel architecture that is suited for high-performance computation with dense data parallelism and have been used to accelerate linear PBE solutions for biomolecular systems with impressive speedup.[82-84] The compact 7-banded matrix is more favorable in GPU implementation than the complex 27-banded matrix due to its denser data parallelism. In addition, in the latter case 20 bands out of the 27-banded matrix have non-zero entries only for grid points nearby the interface, which leads to huge waste of memory if implemented in the band-optimal diagonal matrix format on GPUs. Therefore, improving the accuracy of 7-point stencil discretization schemes such as HA while still maintaining the simplicity and effectiveness will greatly promote applications of PBE implicit modeling on modern GPU computing platforms.

In this study, we have developed two new algorithms in the 7-point stencil discretization scheme by introducing better physical interface relations and imposing the discretized Poisson's equation to the second order, respectively. To achieve this goal, we first re-derived the HA relations in a more general scheme and then developed further extensions from the original idea. Our testing data show that the new methods significantly improve the accuracy to that comparable to the second-order IIM. Meanwhile, the excellent convergence property of the HA method is still maintained. Finally, we present our implementation of the new method onto GPU platforms, which shows impressive efficiency enhancement, making the new methods capable of high-performance biomolecular computational studies.

2.1 Method

As widely used for numerically solving partial differential equations (PDE), the FDM uses a uniform Cartesian grid to discretize the PDE. The grid points are numbered as (i, j, k), where $i = 1, \dots, x_m, j = 1, \dots, y_m, k = 1, \dots, z_m$, and x_m, y_m and z_m are the numbers of points along the three axes. The grid spacing between neighboring points can be uniformly set to h.

To discretize Poisson's equation,

$$\nabla \cdot \varepsilon \nabla \phi = -4\pi\rho$$

the charge density $\rho(i, j, k)$ can be expressed as $q(i, j, k)/h^3$, where $q(i, j, k)$ is the total charge within the cubic volume centered at (i, j, k). The final discretized Poisson's equation is,

$$\begin{aligned}\{\, & \varepsilon\left(i-\frac{1}{2}, j, k\right)[\phi(i-1, j, k)-\phi(i, j, k)]+\varepsilon\left(i+\frac{1}{2}, j, k\right)[\phi(i+1, j, k)-\phi(i, j, k)] \\ & +\varepsilon\left(i, j-\frac{1}{2}, k\right)[\phi(i, j-1, k)-\phi(i, j, k)] \\ & +\varepsilon\left(i, j+\frac{1}{2}, k\right)[\phi(i, j+1, k)-\phi(i, j, k)] \\ & +\varepsilon\left(i, j, k-\frac{1}{2}\right)[\phi(i, j, k-1)-\phi(i, j, k)] \\ & +\varepsilon\left(i, j, k+\frac{1}{2}\right)[\phi(i, j, k+1)-\phi(i, j, k)]\}/h^2=-4\pi q(i, j, k)/h^3\end{aligned}$$

where, $\phi(i,j,k)$ is the potential at grid (i,j,k), and $\varepsilon\left(i-\frac{1}{2},j,k\right)$ is the dielectric constant at the mid-point of the grids (i,j,k) and $(i-1,j,k)$, and all other ϕ and ε are defined similarly here.

A key feature of the discretized PDE is that the coefficient matrix of the linear equation system is in a 7-banded structure, and thus can be naturally wrapped into the diagonal (DIA) matrix format. The advantage lies in that for a banded matrix, the DIA format is the most efficient sparse matrix form for matrix-related linear operations in terms of both memory and CPU time. For example, comparing to the widely used compressed sparse row (CSR) format, the DIA format uses only about half of the memory since it does not store the coefficient indices for the matrix. It is also advantageous for matrix operations, such as matrix-vector multiplications, as there is no need to retrieve index information from memory. The simplicity and orderliness in matrix operations in the banded format further promote the performance on GPUs as our recent analysis of multiple PB solvers has shown.[152] Thus an important motivation of the current study is to explore how to preserve the banded structure in the coefficient matrix while achieving higher-order accuracy numerical solutions.

In above equation, the dielectric constants are defined at the mid-points of all grid edges. As each edge is flanked by two grid points, it is natural to assign the dielectric constant to that of solute (or solvent) if both grid points belong to solute (or solvent). The complication arises when the two flanking grid points belong to different regions, i.e. the grid edge is across the solute/solvent interface.

A pioneering idea was the Harmonic Average (HA) method as proposed by Davis and McCammon in 1991. The HA method was derived from an infinite parallel plate capacitor model.[94] Although it is an excellent simple model to capture the basic physics of the PBE, it is limited in generality for further development. Thus, the HA relation is first re-derived in a more general scheme for discretized PDEs below.

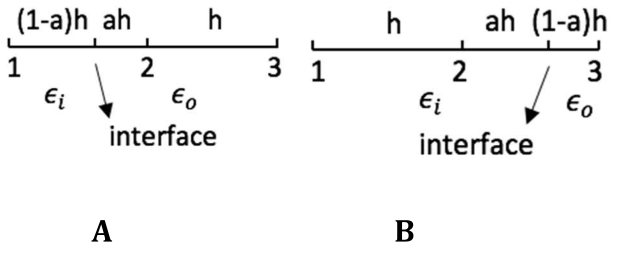


Figure 2.1. Two types of interface geometries. The left side of the interface is considered the inside region with the dielectric constant ϵ_i, and the right side the outside region with the dielectric constant ϵ_o. h is the grid spacing and 0 < a < 1.

As PBE is a 3-d (dimensional) second-order PDE, the 7-point stencil is the simplest way to discretize the equation to achieve the second-order accuracy in both solvent and solute regions, but not at the interface. The goal of HA is to achieve an accuracy level as high as possible while still using the simplest 7-stencil scheme. To illustrate its idea, consider a 1-d interface illustrated in Figure 2.1. The general form of a discretized PDE in 1-d can be written as

$$\gamma_1\phi_1 + \gamma_2\phi_2 + \gamma_3\phi_3 + C = 0$$

where γ_i are the coefficients to be determined, and C is a constant.

First, consider the 1-d (or the *x*-direction) interface geometry in Figure 1A. Expressing φ_i in the Taylor expansion at the interface point yields

$$\begin{cases} \phi_1 = \phi - (1-a)h \cdot \phi_x^- + \frac{1}{2}(1-a)^2h^2 \cdot \phi_{xx}^- \\ \phi_2 = \phi + ah \cdot \phi_x^+ + \frac{1}{2}a^2h^2 \cdot \phi_{xx}^+ \\ \phi_3 = \phi + (1+a)h \cdot \phi_x^+ + \frac{1}{2}(1+a)^2h^2 \cdot \phi_{xx}^+ \end{cases}$$

where "–" and "+" denote the solute and solvent regions, respectively. Similar expressions in *y* and *z* directions can also be obtained simply by changing subscripts. In the HA method, the following interface relations are assumed

$$\phi_x^+ = \frac{\epsilon_i}{\epsilon_o}\phi_x^-$$

$$\phi_{xx}^+ = \frac{\epsilon_i}{\epsilon_o}\phi_{xx}^-$$

which actually imply a conductor-like approximation, i.e. with all tangential components assumed to be zero. Of course, this is different from a conductor-like solvent model with an infinite dielectric constant. Combine above equations we obtain

$$(\gamma_1+\gamma_2+\gamma_3)\cdot\phi + \left[-(1-a)h\gamma_1 + \frac{\epsilon_i}{\epsilon_o}ah\gamma_2 + \frac{\epsilon_i}{\epsilon_o}(1+a)h\gamma_3\right]\cdot\phi_x^-$$
$$+\left[\frac{1}{2}(1-a)^2h^2\gamma_1 + \frac{1}{2}\frac{\epsilon_i}{\epsilon_o}a^2h^2\gamma_2 + \frac{1}{2}\frac{\epsilon_i}{\epsilon_o}(1+a)^2h^2\gamma_3\right]\cdot\phi_{xx}^- + C = 0$$

It should be consistent with the PBE in 1-d so that the coefficients for the 0th- and 1st-order terms are all zero. These conditions lead to

$$\begin{cases} C = 0 \\ \gamma_1+\gamma_2+\gamma_3 = 0 \\ -(1-a)h\gamma_1 + \frac{\epsilon_i}{\epsilon_o}ah\gamma_2 + \frac{\epsilon_i}{\epsilon_o}(1+a)h\gamma_3 = 0 \end{cases}$$

It is obvious that the coefficients used in the HA method, shown in below, are a set of special solutions of it,

$$\begin{cases} \gamma_1 = \dfrac{1}{\dfrac{a}{\epsilon_o} + \dfrac{1-a}{\epsilon_i}} \\ \gamma_2 = -\dfrac{1}{\dfrac{a}{\epsilon_o} + \dfrac{1-a}{\epsilon_i}} - \epsilon_o \\ \gamma_3 = \epsilon_o \end{cases}$$

For the interface geometry in Figure 2.1B, a similar result can also be obtained by the same argument as

$$\begin{cases} \gamma_1 = \epsilon_i \\ \gamma_2 = -\dfrac{1}{\dfrac{a}{\epsilon_i} + \dfrac{1-a}{\epsilon_o}} - \epsilon_i \\ \gamma_3 = \dfrac{1}{\dfrac{a}{\epsilon_i} + \dfrac{1-a}{\epsilon_o}} \end{cases}$$

It is clear from the above derivations that the HA method makes two vital approximations: utilizing the conductor-like interface conditions and imposing the discretization conditions only on the 0th- and 1st-order terms. In this study, we aim to develop new methods by modifying these two approximations, while maintaining the simplicity of a 7-point stencil as in the HA method for GPU implementations.

A straightforward modification can be proposed by imposing the discretization conditions to the second order, which yields

$$\begin{cases} C = 0 \\ \gamma_1 + \gamma_2 + \gamma_3 = 0 \\ -(1-a)h\gamma_1 + \dfrac{\epsilon_i}{\epsilon_o} ah\gamma_2 + \dfrac{\epsilon_i}{\epsilon_o}(1+a)h\gamma_3 = 0 \\ \dfrac{1}{2}(1-a)^2h^2\gamma_1 + \dfrac{1}{2}\dfrac{\epsilon_i}{\epsilon_o} a^2h^2\gamma_2 + \dfrac{1}{2}\dfrac{\epsilon_i}{\epsilon_o}(1+a)^2h^2\gamma_3 = h^2\epsilon_i \end{cases}$$

This replaces as the new coefficient relations for the interface in Figure 2.1A. Here the right-hand side of the fourth equation above can be arbitrarily set to any nonzero constant. The choice of h^2 is for the simplicity of canceling the same factor on the left-hand side, and the use of ϵ_i is to make sure that the coefficients of the new discretized linear equation reduce to those of the standard 7-point stencil when there is no interface. It should be noted that though the PBE discretization is imposed to the second order, this method is rather different from the second order IIM, as IIM considers more detailed interface relations which utilize a 27-point stencil.[156]

Solving above equation yields,

$$\begin{cases} \gamma_1 = \dfrac{2\epsilon_i}{\frac{\epsilon_i}{\epsilon_o}(1+a)a + (2+a)(1-a)} \\ \gamma_2 = -2\epsilon_i \dfrac{(1+a) + \frac{\epsilon_o}{\epsilon_i}(1-a)}{\frac{\epsilon_i}{\epsilon_o}(1+a)a + (2+a)(1-a)} \\ \gamma_2 = 2\epsilon_i \dfrac{a + \frac{\epsilon_o}{\epsilon_i}(1-a)}{\frac{\epsilon_i}{\epsilon_o}(1+a)a + (2+a)(1-a)} \end{cases}$$

Similarly, for the interface in Figure 2.1B, the following solutions can be obtained,

$$\begin{cases} \gamma_1 = 2\epsilon_i \dfrac{a + \frac{\epsilon_i}{\epsilon_o}(1-a)}{(1+a)a + \frac{\epsilon_i}{\epsilon_o}(2+a)(1-a)} \\ \gamma_2 = -2\epsilon_i \dfrac{(1+a) + \frac{\epsilon_i}{\epsilon_o}(1-a)}{(1+a)a + \frac{\epsilon_i}{\epsilon_o}(2+a)(1-a)} \\ \gamma_2 = \dfrac{2\epsilon_i}{(1+a)a + \frac{\epsilon_i}{\epsilon_o}(2+a)(1-a)} \end{cases}$$

Obviously, the modification only scales the coefficients of the classical HA method. Thus, it is reasonable to assume this modification would improve the accuracy without jeopardizing its other properties, such as the stability and convergence behaviors. In addition, the proposed change retains the compact 7-banded structure of the final coefficient matrix, which is favorable for GPU implementations. For simplicity, we term this revised method as the second-order HA method and will refer to it throughout the manuscript.

The interface relations here are similar to those on a conductor surface,

$$\phi_\tau^- = 0$$

where τ denotes the tangential direction of the electric field on the interface. However, the outside region obviously does not have an infinite dielectric constant (ϵ_o = 80 for water). As a result, the tangential electric field cannot be exactly zero. The physical interface relations for the first derivatives of electric potential are,

$$\phi_x^- = \phi_\xi^- \cos(\hat{\xi}, \hat{x}) + \phi_\tau^- \cos(\hat{\tau}, \hat{x})$$

$$\phi_x^+ = \frac{\epsilon_i}{\epsilon_o} \phi_\xi^- \cos(\hat{\xi}, \hat{x}) + \phi_\tau^- \cos(\hat{\tau}, \hat{x})$$

where ξ denotes the normal direction of the electric field on the interface. The above equations show that the factor $\frac{\epsilon_i}{\epsilon_o}$ should be replaced by a more physical term, χ, because $\frac{\phi_x^+}{\phi_x^-}$ is no longer equal to $\frac{\epsilon_i}{\epsilon_o}$ as in the conductor-like interface,

$$\chi = \frac{\phi_x^+}{\phi_x^-} = \frac{\frac{\epsilon_i}{\epsilon_o} \phi_\xi^- \cos(\hat{\xi}, \hat{x}) + \phi_\tau^- \cos(\hat{\tau}, \hat{x})}{\phi_\xi^- \cos(\hat{\xi}, \hat{x}) + \phi_\tau^- \cos(\hat{\tau}, \hat{x})}$$

Now the challenge is to evaluate ϕ_ξ^- and ϕ_τ^- in Eqn (15). This requires a careful look at the conductor approximation. For a perfect conductor (ϵ_o goes to infinite), the Coulomb field ϕ_C^- and the induced field (reaction field) ϕ_{RF}^- have the following relations,

$$(\phi_\xi^-)_C = (\phi_\xi^-)_{RF}$$

$$(\phi_\tau^-)_C = -(\phi_\tau^-)_{RF}$$

The first equation shows that the normal field is zero inside the conductor (or water) and is twice the magnitude of the Coulomb field outside the conductor (or molecular interior). The second equation shows that the tangential field is always zero as the two terms cancel out. However, for a dielectric medium such as water (ϵ_o = 80), the above relations in the normal direction may still be similar for the two fields, while in the tangential direction the induced field cannot fully cancel the Coulomb field and thus a non-zero residual component exists. The tangential residual component may be small, but definitely not zero. Therefore, we use the following approximation relations to evaluate the normal and tangential components ϕ_{ξ}^{-} and ϕ_{τ}^{-},

$$\phi_{\xi}^{-} \approx 2(\phi_{\xi}^{-})_C$$

$$\phi_{\tau}^{-} \approx \frac{2\epsilon_i}{\epsilon_o}(\phi_{\tau}^{-})_C$$

It is obvious that the above relations reduce to the conductor-like interface relations when ϵ_o goes to infinite. Note that while the first relation has a reasonable physical basis, the second relation is an approximation proposed after comparing with the simple sphere geometries that can be solved analytically, as many proteins are shaped (approximately) as spheres (globular proteins).

Once factor χ is obtained, substituting it yields

$$\begin{cases} \gamma_1 = \epsilon_o \dfrac{1}{a + \dfrac{1-a}{\chi}} \\ \gamma_2 = -\epsilon_o \dfrac{1}{a + \dfrac{1-a}{\chi}} - \epsilon_o \\ \gamma_3 = \epsilon_o \end{cases}$$

$$\begin{cases} \gamma_1 = \epsilon_i \\ \gamma_2 = -\epsilon_i \dfrac{1}{a + \chi(1-a)} - \epsilon_i \\ \gamma_3 = \epsilon_i \dfrac{1}{a + \chi(1-a)} \end{cases}$$

This modification would also keep the simple 7-banded coefficient matrix from a standard 7-point stencil, suitable for GPU implementations. Also, because this modification involves only a minor change (replacing $\frac{\epsilon_i}{\epsilon_o}$ with χ) to the classical HA method, we assume it would not compromise other properties, such as fast convergence, good stability etc., while improving its accuracy. In the following, we term this modification as the χ-factor HA method and will refer to it throughout the manuscript.

An analytical model of a single dielectric sphere embedded with point charges was used to validate the accuracy and precision of the proposed methods. The analytical reaction and total fields of the inside region in spherical coordinates are,

$$\frac{\partial \phi_{RF}^{-}(r,\theta,\varphi)}{\partial r} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{4\pi}{2l+1} \frac{l}{R^{2l+1}} \frac{(l+1)(\varepsilon^- - \varepsilon^+)}{\varepsilon^-(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} r^{l-1} Y_{lm}(\theta,\varphi)$$

$$\frac{\partial \phi_{RF}^{-}(r,\theta,\varphi)}{\partial \theta} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{4\pi}{2l+1} \frac{1}{R^{2l+1}} \frac{(l+1)(\varepsilon^- - \varepsilon^+)}{\varepsilon^-(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} r^{l} \frac{\partial Y_{lm}(\theta,\varphi)}{\partial \theta}$$

$$\frac{\partial \phi_{RF}^{-}(r,\theta,\varphi)}{\partial \varphi} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{4\pi}{2l+1} \frac{1}{R^{2l+1}} \frac{(l+1)(\varepsilon^- - \varepsilon^+)}{\varepsilon^-(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} r^{l} \frac{\partial Y_{lm}(\theta,\varphi)}{\partial \varphi}$$

$$\frac{\partial \phi^{-}(r,\theta,\varphi)}{\partial r} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{\varepsilon^+}{\varepsilon^-} \frac{-4\pi(l+1)}{(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{1}{R^{l+2}} Y_{lm}(\theta,\varphi)$$

$$\frac{\partial \phi^{-}(r,\theta,\varphi)}{\partial \theta} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{4\pi}{(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{1}{R^{l+1}} \frac{\partial Y_{lm}(\theta,\varphi)}{\partial \theta}$$

$$\frac{\partial\phi^-(r,\theta,\varphi)}{\partial\varphi} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \frac{4\pi}{(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{1}{R^{l+1}} \frac{\partial Y_{lm}(\theta,\varphi)}{\partial\varphi}$$

$$with, Q_{lm} = \sum_{k=1}^{N_q} q_k r_k^l Y_{lm}^*(\theta_k,\varphi_k)$$

where R is the radius of the sphere with the center set to the origin, N_q is the number of charges, and Y_{lm} is the spherical harmonic function.

With the above analytical results, we are able to estimate the new jump condition, i.e. the χ factor. From above, we can obtain the electric fields of the inside region on the interface as,

$$\frac{\partial\phi_{RF}^-(r,\theta,\varphi)}{\partial r} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \frac{4\pi}{2l+1} \frac{(l+1)(\varepsilon^- - \varepsilon^+)}{\varepsilon^-(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{l}{R^{l+2}} Y_{lm}(\theta,\varphi)$$

$$\frac{\partial\phi_{RF}^-(r,\theta,\varphi)}{\partial\theta} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \frac{4\pi}{2l+1} \frac{(l+1)(\varepsilon^- - \varepsilon^+)}{\varepsilon^-(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{1}{R^{l+1}} \frac{\partial Y_{lm}(\theta,\varphi)}{\partial\theta}$$

$$\frac{\partial\phi_{RF}^-(r,\theta,\varphi)}{\partial\varphi} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \frac{4\pi}{2l+1} \frac{(l+1)(\varepsilon^- - \varepsilon^+)}{\varepsilon^-(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{1}{R^{l+1}} \frac{\partial Y_{lm}(\theta,\varphi)}{\partial\varphi}$$

$$\frac{\partial\phi^-(r,\theta,\varphi)}{\partial r} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \frac{\varepsilon^+}{\varepsilon^-} \frac{-4\pi(l+1)}{(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{1}{R^{l+2}} Y_{lm}(\theta,\varphi)$$

$$\frac{\partial\phi^-(r,\theta,\varphi)}{\partial\theta} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \frac{4\pi}{(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{1}{R^{l+1}} \frac{\partial Y_{lm}(\theta,\varphi)}{\partial\theta}$$

$$\frac{\partial\phi^-(r,\theta,\varphi)}{\partial\varphi} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \frac{4\pi}{(l\varepsilon^- + (l+1)\varepsilon^+)} Q_{lm} \frac{1}{R^{l+1}} \frac{\partial Y_{lm}(\theta,\varphi)}{\partial\varphi}$$

From the analysis of above equations, we have,

$$\frac{\partial\phi^-(r,\theta,\varphi)}{\partial r} \approx 2\frac{\partial\phi_{RF}^-(r,\theta,\varphi)}{\partial r} \approx 2\frac{\partial\phi_{C}^-(r,\theta,\varphi)}{\partial r}$$

Comparing above results, we obtain,

$$\frac{\partial \phi^-(r,\theta,\varphi)}{\partial \theta} \approx -\frac{2\varepsilon^-}{\varepsilon^+}\frac{\partial \phi_{RF}^-(r,\theta,\varphi)}{\partial \theta} \approx \frac{2\varepsilon^-}{\varepsilon^+}\frac{\partial \phi_C^-(r,\theta,\varphi)}{\partial \theta}$$

$$\frac{\partial \phi^-(r,\theta,\varphi)}{\partial \varphi} \approx -\frac{2\varepsilon^-}{\varepsilon^+}\frac{\partial \phi_{RF}^-(r,\theta,\varphi)}{\partial \varphi} \approx \frac{2\varepsilon^-}{\varepsilon^+}\frac{\partial \phi_C^-(r,\theta,\varphi)}{\partial \varphi}$$

Note that although above equation is a good approximation for spherical cases, its correctness and effectiveness for complex biomolecular systems can only be validated by actual numerical experiments. A good approximation should have at least the following two features. First, the estimated χ values follow the trend of true χ values relatively well, so they can capture more accurate interface geometries and field distributions. Second, the systematical error should be as small as possible.

2.2. Computation Details

The potentials at grid points are obtained by solving PBE numerically. Then the potential at any position (x,y,z) is obtained with a one-sided least-squares interpolation method.[96] The linear Poisson-Boltzmann energy (PB energy) can then be obtained as the summation of the charges multiplying their respective reaction field potentials as

$$\Delta G = \frac{1}{2}\sum_{charges} q_i \phi_{RF}$$

For the analytical test cases, four off-center charge models were used, which are monopole, dipole, quadrupole, and octupole, respectively. The radius of the sphere was set to 1.6 Å, which is about the size of a carbon atom. The Cartesian coordinates of each charge model are listed in Table 1. The inside and outside dielectric constants were set to 1.0 and 80.0, respectively. The truncation order in Eqn (20) was chosen to be 120, which gives a precision of 10^{-6}.

Table 2.1. Cartesian coordinates of charges for the analytical monopole, dipole, quadrupole, and octupole models.

Model	Positive Charge Coordinates	Negative Charge Coordinates
Monopole	(-0.10, -0.10, 0.90)	
Dipole	(-0.10, -0.10, 0.90)	(-0.10, -0.10, -0.90)
Quadrupole	(0.90, -0.90, -0.10) (-0.10, 0.90, -0.10)	(0.90, 0.90, -0.10) (-0.10, -0.90, -0.10)
Octupole	(0.90, -0.10, 0.90) (-0.10, 0.90, 0.90) (0.90, -0.10, -0.10) (-0.10, 0.90, -0.10)	(0.90, 0.90, 0.90) (-0.10, -0.10, 0.90) (0.90, 0.90, -0.10) (-0.10, -0.10, -0.10)

In addition, a total of 573 biomolecular structures from the Amber PBSA benchmark suite were used to study the overall accuracy and efficiency of the new methods.[50] Of these molecules, eight small proteins of about 1000 atoms from the suite were selected to analyze the properties of the new method in more details, so that computational jobs could be handled on our local compute nodes at the finest grid spacings tested. These biomolecules were assigned charges of Cornell *et al*[149] and the modified Bondi radii. The probe radius was set to 1.4 Å. All testing jobs were performed with the following conditions unless specified otherwise. The convergence criterion of 10^{-4} was used for the biconjugate gradient (BiCG) linear solver. The default grid spacing was set to 0.5 Å for most calculations, except that in the convergence analysis it was set to a range of values of from

1.0 Å to 0.1 Å. The ratio of the grid dimension to the solute dimension (the *fillratio* keyword in Amber) was set to 1.5 and 4.0 for biomolecules and analytical test cases, respectively. No electrostatic focusing was applied. The free boundary condition was used to assign the finite-difference grid boundary potentials and the charge singularity was removed. The dielectric constants were set to 1.0 and 80.0 for solute and solvent, respectively. All other parameters were set as default in the PBSA module in Amber 18 package.[46, 48, 50-51, 116-118, 143, 150, 152, 157-165]

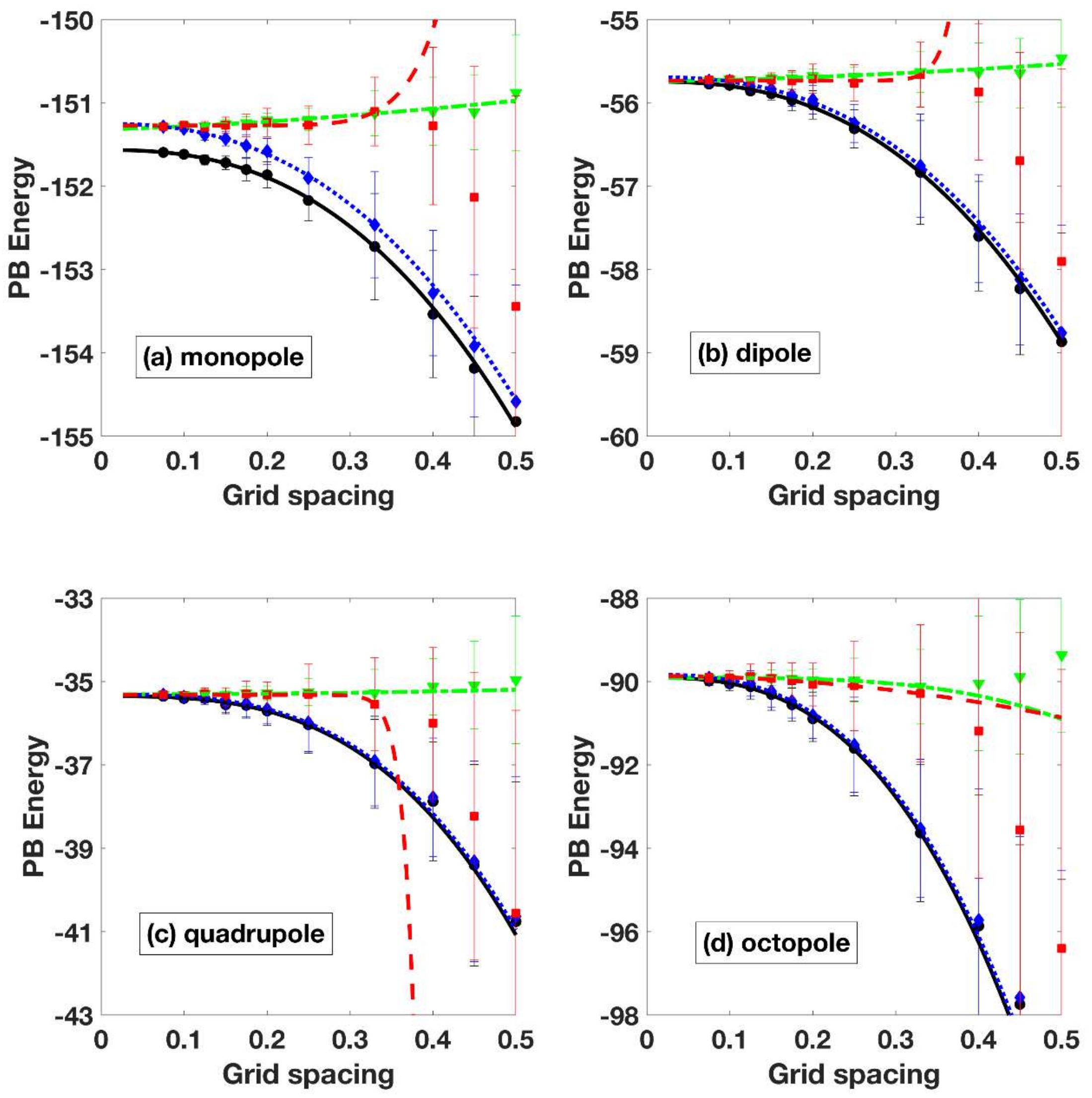
A BiCG linear system solver for GPUs was also implemented using the Nvidia CUDA Sparse Matrix (cuSPARSE) and CUSP libraries, which provide basic linear algebra procedures for sparse matrix operations.[151] The DIA matrix format was used for the non-symmetric coefficient matrix as in our previous publication.[152] All GPU and CPU tests were conducted on a dedicated compute node with two Nvidia RTX 2080Ti GPU cards, one Intel Xeon E5-1620 v3 CPU, and 16GB main memory. Our time measurements for all solvers include all execution time of the solver routines, i.e. time elapsed on the device (GPU) and on the host (CPU) and also for transferring data between the device and the host.

2.3. Results and Discussion

We extended the classical HA method by two improvements, i.e. imposing the discretization conditions to the second order and relaxing the conductor-like interface relation, respectively. These efforts resulted in the second-order and the χ-factor HA methods. In the following, we report the numerical experiments to study the quality of the approximations, and also compare their accuracy and performance with the classical HA method and the analytical IIM. Specifically, analytical models were first used to evaluate the quality of the approximations. Then eight small proteins were studied to evaluate their convergence behaviors. Next the robustness and correctness of these methods were investigated using a large set of biomolecular structures at the widely used numerical conditions. Furthermore, the BiCG linear solver was implemented for the two new methods utilizing the compact 7-banded structure of the coefficient matrices to boost their efficiency on GPUs. Finally, both the improvement and limitation of the new methods are also discussed.

We first tested the new methods using analytical models from monopole to octupole and compared them with the classical HA method and the analytical IIM. The convergence trends of PB energies over the tested grid spacing for all the methods are shown in Figure 2.2. Clearly, all tests converge well except for the classical HA method on the monopole model. This indicates that the classical HA method may give inaccurate solvation energies even for simple spherical cases, yet the two new HA methods were found to overcome the limitation.

More detailed convergence properties are shown in Table 2.2, where the extrapolated limiting PB energies at the grid spacing of zero are listed for all models by the four tested methods. The analysis confirms that the PB energies of all the four methods converge to the analytical values reasonably well except the monopole case with the classical HA method.



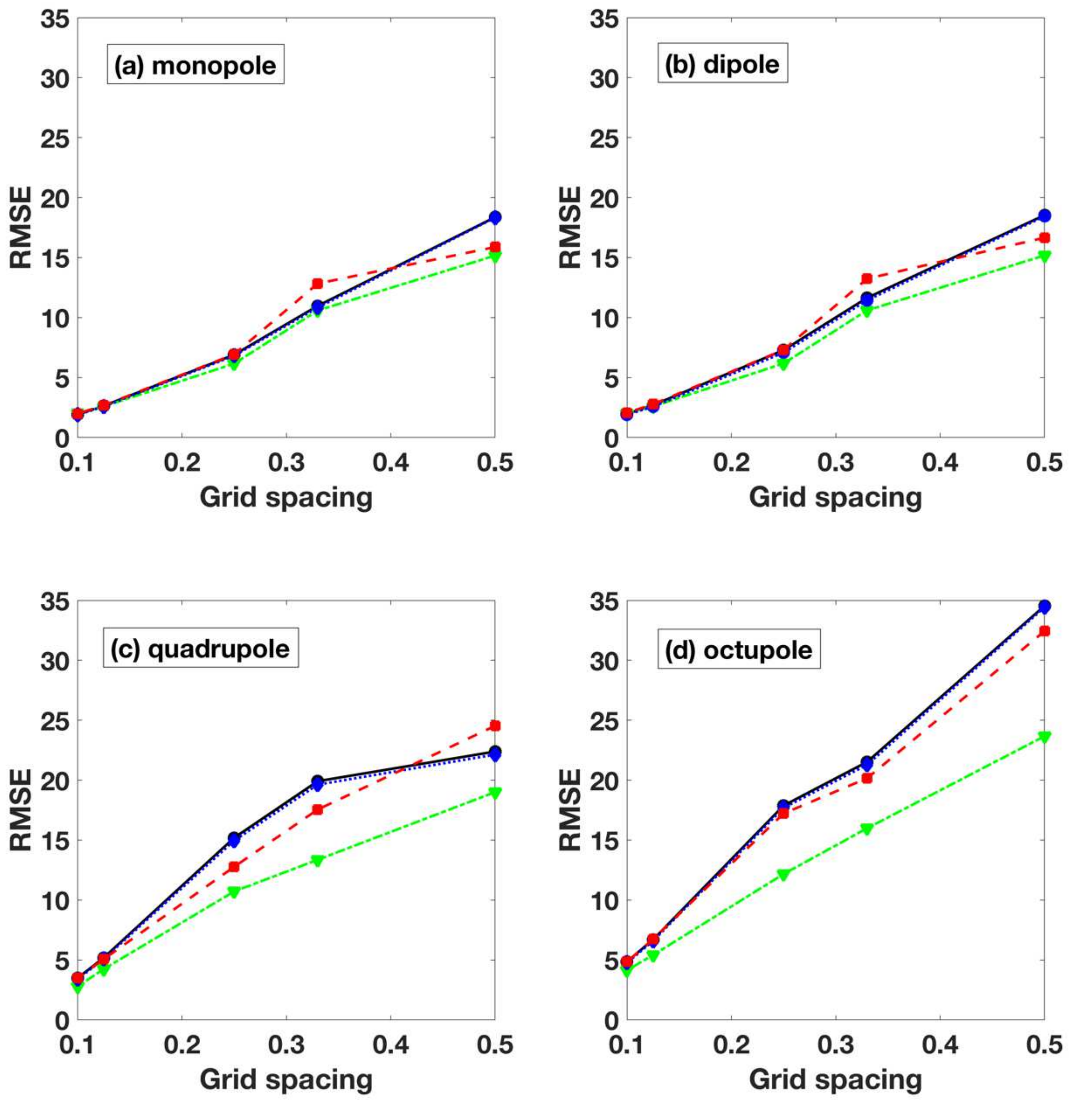
— HA method --- 2nd-order HA method • • • χ-factor HA method -·- IIM method

Figure 2.2. Tests of PB energies (kcal/mol) versus grid spacing (Å) for analytical sphere models. The energy results were obtained by averaging 30 grid orientations/offsets of the molecules. All the curves were obtained by fitting data to trend lines of y=a*x^b+c. Note that only data points at grid spacing ≤ 0.33 Å were included in the fitting.

Table 2.2. PB energies (kcal/mol) for the analytical sphere models at limiting grid spacing of 0 Å. (Analytical PB energies were calculated using Mathematica 7.0.)

Methods	HA	2nd order HA	χ factor HA	IIM	Analytical
Monopole	-151.6	-151.3	-151.2	-151.3	-151.29
Dipole	-55.74	-55.74	-55.69	-55.73	-55.73
Quadrupole	-35.35	-35.31	-35.32	-35.32	-35.31
Octupole	-89.91	-89.91	-89.84	-89.87	-89.91

As interface plays a critical role in the PBE modeling, we further examined the abilities of these methods to capture the interface property by calculating the electric field on the molecular surface. The root mean squared error (RMSE) values were calculated by comparing the numerical electric fields with the analytical results obtained before. Here the surface electric field was interpolated with the one-dimensional method, shown to perform the best for these analytical systems.[153] Figure 2.3 indicates that all 7-stencil methods agree well with analytical values, though not as accurate as the analytical IIM as expected. It is also worth noting that the χ-factor HA method is better than the classical HA method on every test point, though not by much. In summary, it is clear that the two approximations work well for all tested analytical systems with no systematic biases.



— HA method --- 2nd order HA method • • • χ factor HA method -·- IIM method

Figure 2.3. RMSEs of normal component of the surface electric field between the computed and the analytical values (kcal/mol-e-Å) versus grid spacing (Å). The surface field values were obtained by one-dimensional interpolation.

To assess the quality of the two new methods on biomolecules, we studied their numerical behaviors with eight small proteins. The convergence trends of PB energies for all the methods are shown in Figure 2.4 and the detailed extrapolation analysis is presented in Table 2.3. Figure 2.4 shows that all tested methods converge smoothly but to two different limiting values: the classical HA method and the second-order HA method converge to more negative values; and the χ-factor HA method and the analytical IIM converge to more positive values. More specifically, the difference between the limiting PB energies for the latter two methods are mostly within 1 kcal/mol as shown in Table 2.3. As the analytical IIM is the most accurate method of the four since there is no approximation in its discretization scheme,[119] this means the second-order and the classical HA methods tend to give more negative PB energies than the true values for molecular systems. Note that this behavior has been observed for the classical HA method in the analytical monopole test already (Figure 2.2). However, the behavior is a new observation for the second-order HA method since it performs well in the analytical tests. This shows that the second-order HA method tends to degenerate to the classical HA method when it is applied to complicated molecular geometries such as proteins. In contrast, the χ-factor HA method still maintains its excellent numerical behaviors even on proteins, and almost produces the same accuracy as the high-order IIM does with its 27-point stencil.

Another observation from Figure 2.4 is that the classical and the χ-factor HA methods have much flatter convergence curves than the second-order HA method. A closer look of the PB energies at the grid spacing of 0.5 Å in Table 2.4 shows that the relative errors with respective to methods' own limiting values are 0.1%, 0.14%, and 0.21%, respectively, for the classical, the χ-factor, and the second-order HA methods. This behavior is rather different from what is observed in the analytical spherical tests, where the second-order HA method converges the fastest among the three HA methods.

Taken all the observations together, we can conclude that when dealing with realistic molecular systems such as proteins, the χ-factor HA method preserves the overall good convergence behavior of the classical HA method while it converges to the correct limiting values. Worth noting is that the χ-factor HA method performs even better than the IIM with overall flatter convergence curves. It is worth pointing out that the slower convergence of the second-order methods may result from their overly precise description of the local complex interface geometries in typical proteins.[156] It is likely that fuzzy handling of complex geometries as done in simpler methods may lead to a good apparent convergence trend simply because the methods are not designed to detect complex geometries. However, what the χ-factor HA method gained over the classical HA method is that it converges to the correct limiting values as those from the analytical IIM, while still preserving the good convergence trend of the classical HA method. As a result, its simplicity, good convergence property, and numerical stability make the χ-factor HA method an outstanding scheme among the four investigated methods in practices.

Table 2.3**.** PB energies (kcal/mol) for the 8 selected protein molecules at the limiting grid spacing of 0 Å.

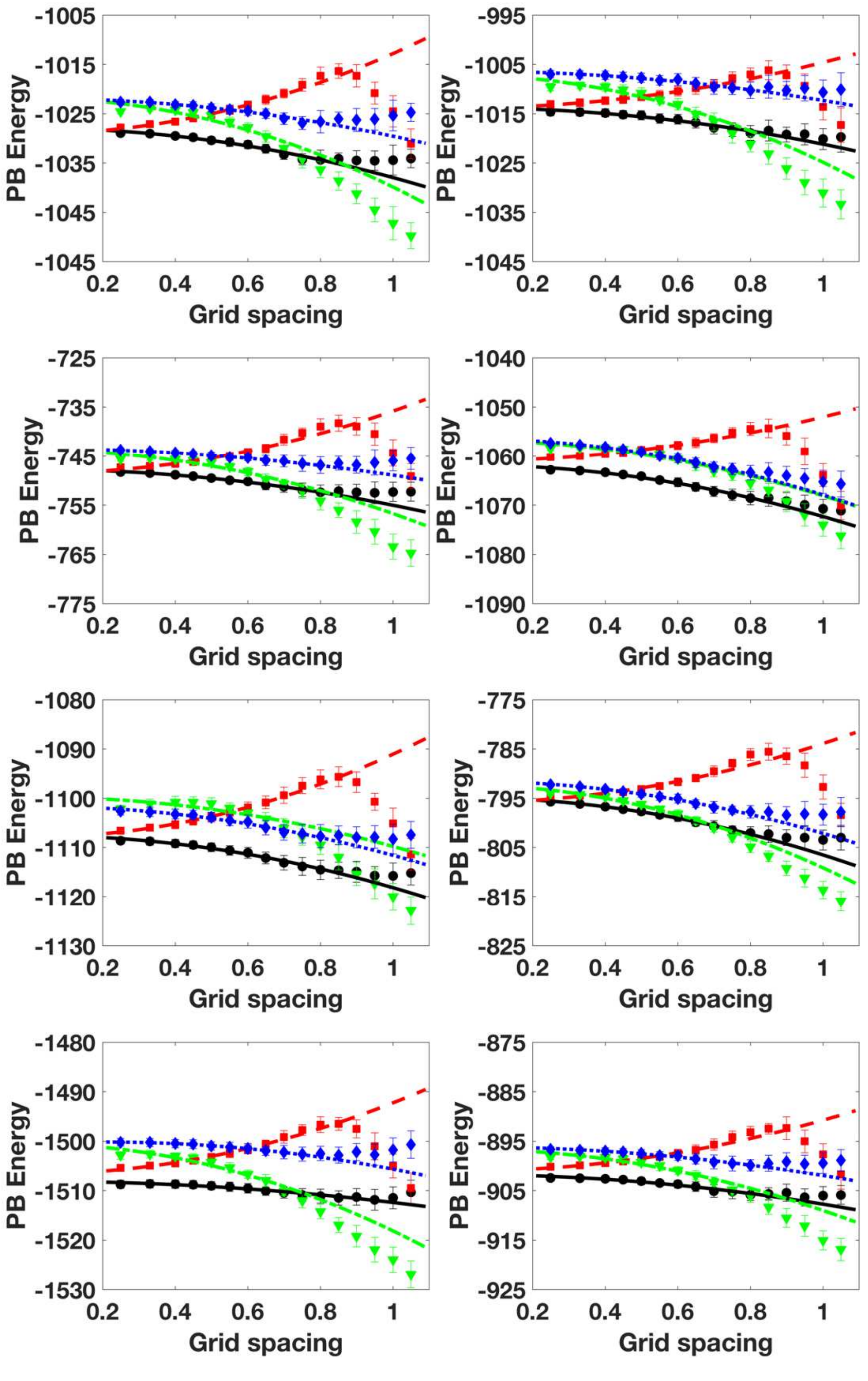
Methods	HA	2nd order HA	χ factor HA	IIM
1aci	-1029	-1028	-1025	-1024

1ah9	-1015	-1013	-1010	-1009
1b22	-747.9	-747.5	-745.5	-745.4
1aw0	-1063	-1060	-1059	-1058
1bbi	-1108	-1107	-1105	-1101
1bdc	-795.2	-794.9	-793.6	-793.8
1bw6	-1509	-1506	-1503	-1503
1c75	-902.4	-900.4	-898.4	-898.2

Table 2.4. PB energies (kcal/mol) and standard deviations (kcal/mol in parenthesis) for the 8 selected protein molecules at the grid spacing of 0.5 Å. The energy results were obtained by averaging 30 grid orientations/offsets of the molecules.

Methods	HA	2nd order HA	χ factor HA	IIM
1aci	-1030.38(0.73)	-1025.26(0.69)	-1026.63(0.76)	-1025.37(0.88)
1ah9	-1015.32(0.94)	-1011.52(0.89)	-1010.66(0.93)	-1010.84(1.1)
1b22	-749.49(0.61)	-745.64(0.64)	-746.93(0.65)	-746.32(0.71)
1aw0	-1064.12(0.59)	-1058.73(0.60)	-1060.93(0.59)	-1059.01(0.55)
1bbi	-1109.95(0.90)	-1103.83(0.85)	-1106.20(0.98)	-1101.10(1.2)
1bdc	-797.71(0.71)	-793.18(0.67)	-795.81(0.70)	-796.21(0.60)

1bw6	- 1509.01(0.83)	- 1503.28(0.81)	- 1503.83(0.78)	- 1504.29(0.90)
1c75	-903.08(0.70)	-898.59(0.72)	-899.27(0.69)	-899.27(0.73)



— HA method --- 2nd-order HA method • • • χ-factor HA method -·- IIM method

Figure 2.4. Tests of PB energies (kcal/mol) versus grid spacing (Å) for biomolecules. The energy results are obtained by averaging 30 grid orientations/offsets of the molecules. All trend curves were obtained by fitting data to y=a*x^b+c form. Only the data points within grid spacing 0.5 Å were included in the fitting.

To examine the robustness of these methods with large protein molecules under typical numerical conditions, we computed the PB energies with 573 biomolecular structures from the Amber PBSA benchmark suite at the commonly used grid spacing of 0.5 Å. No failure was observed for all the 573 test cases, which confirms that our new methods are robust enough for various kinds of biomolecules.

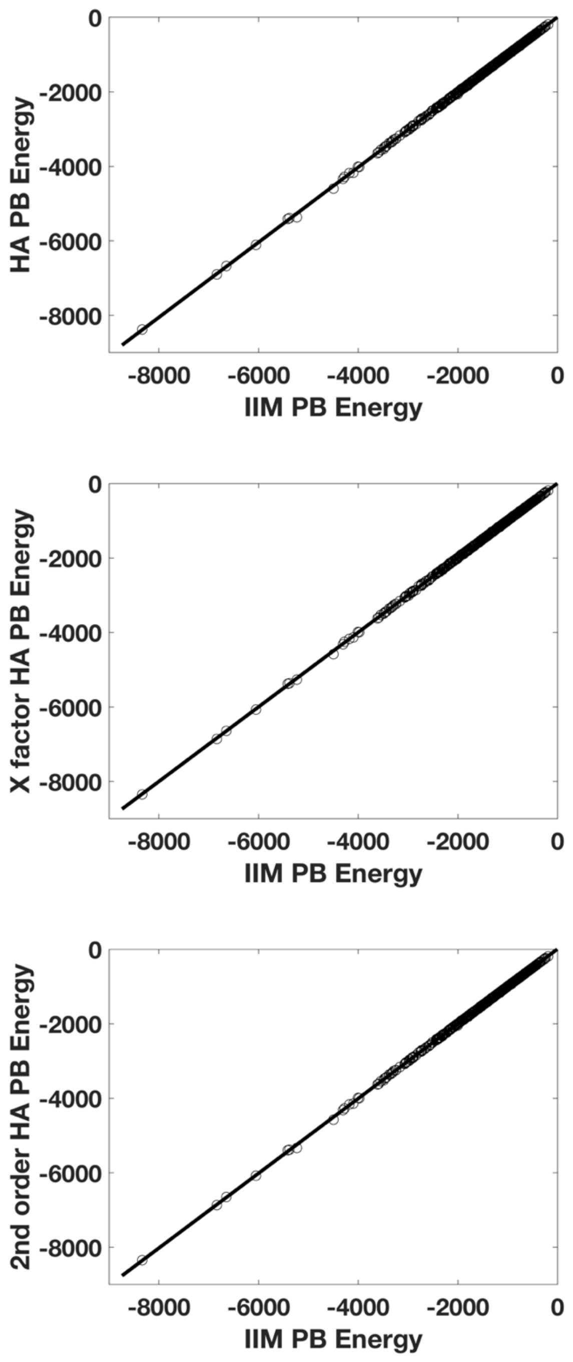


Figure 2.5. Tests of PB energies (kcal/mol) at 0.5 Å for 573 biomolecules. The data were fitted linearly and the intercepts were set to zero.

To confirm the correctness of the new methods, we analyzed the correlations between the PB energies from the three HA methods with those from the analytical IIM. As shown in Figure 2.5, the linear fitting slopes are 1.004, 0.998, and 1.000 for the classical, the χ-factor, and the second-order HA methods, respectively. This indicates that compared to the IIM, the classical HA method has a systematic tendency to give more negative results and the χ-factor HA method reduces the error to about a half, and the second-order HA method has almost no systematic tendency at the tested condition. Interestingly, the second-order HA method is the closest to the IIM at the tested condition, which may seem to be more accurate than the other two HA methods. However, note that the calculations were carried out at the grid spacing of 0.5 Å, where these methods may deviate from the limiting values (i.e. at the grid spacing of 0.0 Å). While it is difficult and impractical to obtain extrapolated limiting PB energies for all 573 biomolecular structures, the results from the eight representative proteins as discussed in section 4.2 can serve as a guidance. As shown in Figure 4, the convergence trendlines of the IIM and the second-order HA method cross over near the grid spacing of 0.5 Å and thus their PB energies are closer to each other.

To quantitively measure the accuracy of each method at the tested grid spacing, we calculated the relative error of the PB energy from each method (see Table 2.4) with respective to the IIM's limiting values (see Table 2.3). The medium relative errors are 0.6%, 0.2%, 0.1% and 0.13% for the classical, the χ-factor HA, the second-order HA, and the IIM, respectively. This means that at the grid spacing of 0.5 Å, a common choice for practical biomolecular applications, the χ-factor and the second-order HA methods are with very similar errors as the IIM. This further illustrates the impressive improvements achieved in the two new HA methods. It is interesting that the second-order HA method coincidently behaves well at the widely used grid spacing even if its limiting convergence is closer to the classical HA method. However, we expect that in a wider range of conditions, the χ-factor HA method would be more accurate and stable as it has better convergence behavior with various grid spacings as discussed above.

An important motivation in developing the new HA methods is to utilize their compact 7-banded structure in the coefficient matrices to improve computational efficiency on high-performance platforms such as GPUs. We implemented the BiCG linear solver on GPUs for the two new HA methods, as the BiCG algorithm is suitable for asymmetric matrices. The DIA matrix format was used to store the matrices in the new methods. In contrast the CSR matrix format was used in the analytical IIM. The timing results for 573 biomolecules are shown in Figure 2.6, with the timings of the BiCG solver implemented with the IIM on GPUs as the benchmark. The results show that the average solver time (Figure 2.6 top two) for the χ-factor and the second-order HA methods is about 35% and 39% of that of the IIM, respectively. This means that by using a more compact matrix structure, the solver time can be reduced by about two thirds. As simpler interface schemes, the new methods also reduce the setup time significantly (Figure 2.6 bottom two). This is because the IIM uses a multistep, complicated setup procedure,[156] which requires the interface geometry parameters up to the second order (curvatures) and the Coulomb field parameters of the second order (gradients). While for the two new methods, all these parameters (interface normal directions and Coulomb field) are required only to the first order. As a result, the intrinsically simpler HA methods leads to significant efficiency gain in PBE modeling on GPU platforms.

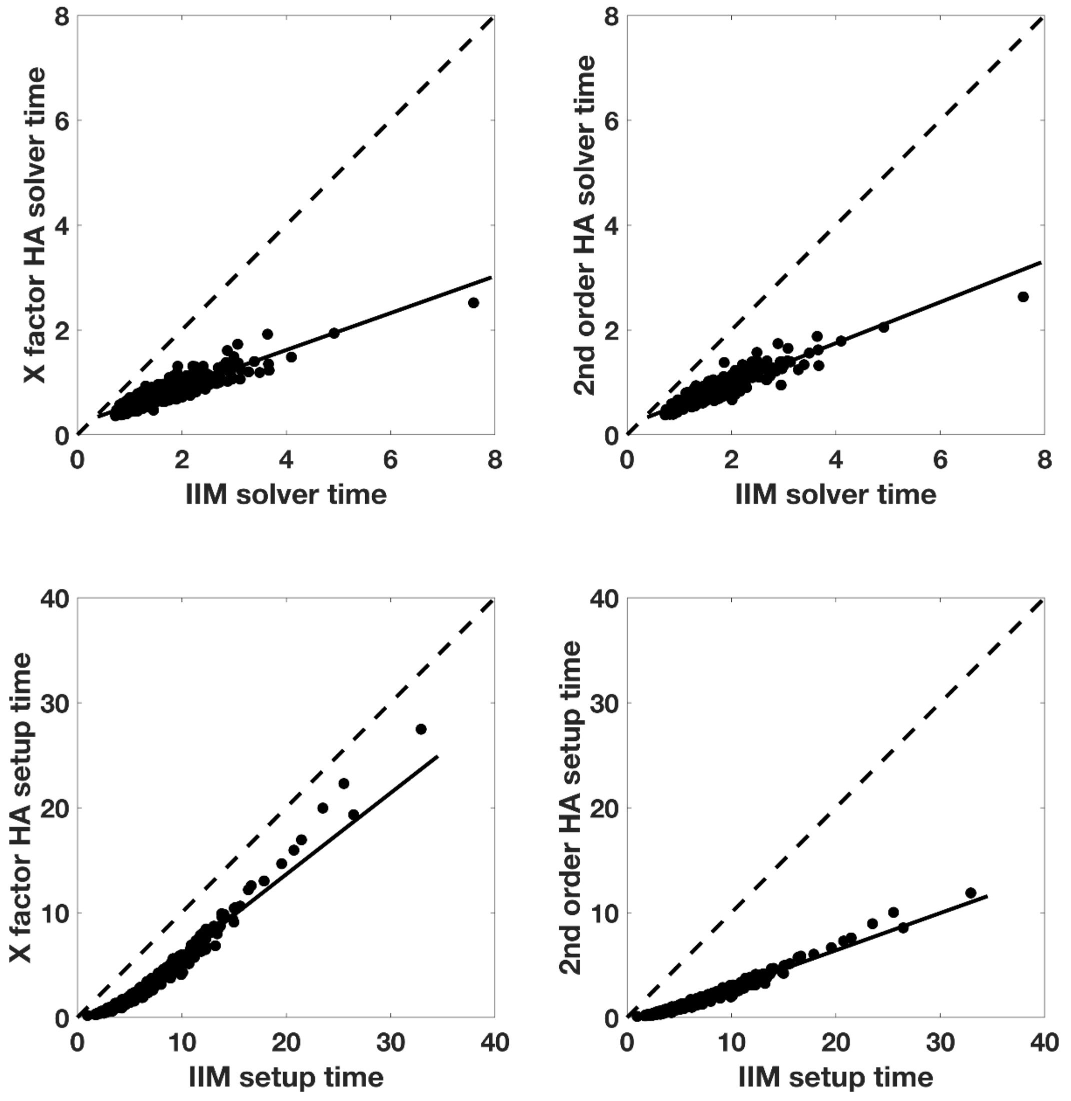


Figure 2.6. Timing (seconds) tests of GPU implementation at 0.5 Å for 573 biomolecules. The data were fitted linearly. The fitting line of the χ factor HA method solver time is y=0.3526x+0.2069, and that of the second-order HA method is y=0.3917x+0.1819. The fitting line of the χ factor HA method setup time is y=0.7397x-2.405, and that of the second-order HA method is y=0.3355x+0.1703. The dashed lines are diagonal.

2.5 Conclusion

In this study, we explored new 7-stencil algorithms based on the spirit of the classic HA method for PBE modeling, namely, the χ-factor HA method and the second-order HA method by introducing better physical interface relations and imposing the discretized Poisson's equation to the second order. The new methods improve on the original one and still maintain the compact 7-banded structure for the discretized equations. As a result, the advantages of fast matrix operation and reduced memory requirement are obtained for numerical implementation, especially on GPUs for parallel computation.

We first tested our new methods on multiple analytical test cases to serve as a primary check for the quality of the approximations used in the new methods. The results showed that the new methods overcome the limitation of the classical HA method in the monopole case by giving more accurate limiting PB energy. In addition, the new methods also give correct electric field on the interface, showing that the new methods are overall consistent with theory for the tested analytical systems.

We further analyzed the convergence of the new methods on small proteins using the analytical IIM as reference, to study their advantages on biomolecules. The results show that the χ-factor HA method performs very well in both accuracy and convergence with its limiting PB energies mostly within 1 kcal/mol difference from the reference values. Therefore, its simplicity, good convergence property, and stability on complex protein molecules make χ-factor HA an outstanding scheme among the four investigated methods.

We next examined the robustness and correctness of the new methods with 573 biomolecular structures at the commonly used grid spacing of 0.5 Å. The results show that all tests passed with no failure, confirming that our new methods are robust enough for various complex interface geometries in biomolecules. Interestingly, the second-order HA method agrees the best with the IIM in terms of PB energies. Our further analysis of the relative error of the PB energies with respective to the IIM limiting values shows that two new HA methods are with very similar errors as the analytical IIM at the widely used numerical conditions. We further pointed out that though the second-order HA method performs well at the grid spacing of 0.5 Å, it is expected that the χ-factor HA method will be more accurate and stable in a wider range of application conditions.

Finally, we implemented the BiCG linear solver for the new HA methods and tested their performance on GPUs. The results show that by using a more compact 7-banded matrix structure, the solver time can be reduced by about two thirds, and the setup time are also significantly reduced over the IIM method. As a result, the intrinsic simple design of the HA methods leads to significant efficiency gain of PBE modeling on GPU platforms, which makes the new methods more suitable for high-performance biomolecular computational studies.

Chapter 3

3.1. Introduction

Electrostatic interactions play crucial roles in biophysical processes such as protein and RNA folding, enzyme catalysis, and molecular recognition. Thus, accurate and efficient treatment of electrostatics is vital to computational studies of biomolecular structures, dynamics, and functions. A closely related issue is the modeling of water molecules and their electrostatic interactions with biomolecules that must be considered for any realistic representation of biomolecules at physiological conditions. Implicit solvent model has been such an attempt, in which, the solute molecule is treated as a low dielectric constant region with a number of point charges located at atomic centers, and the solvent is treated as a high dielectric constant region. Among all the attempts, Poisson-Boltzmann (PB) equation based implicit solvent models have proven to be among the most successful ones and are widely used in computational studies of biomolecules.

A crucial component of all implicit solvent models within the PB framework is the dielectric model, *i.e.* the dielectric constant distribution of a given solution system. Typically, a solution system is divided into the low dielectric interior and the high dielectric exterior by a molecular surface. That is to say that the molecular surface is used as the dielectric interface between the two piece-wise dielectric constants. The solvent excluded surface (SES)[1-3] is the most used surface definition.[4, 5] Indeed, previous comparative analyses of PB-based solvent models and TIP3P solvent models show that the SES definition is reasonable in calculation of reaction field energies and electrostatic potentials of mean force of hydrogen-bonded and salt-bridged dimers with respect to the TIP3P explicit solvent.[6-8] Given the complexity of the

SES, one possible approach in adapting the SES in numerical solutions is to build the molecular surface analytically and then to map it onto a grid.[9-11] Since analytical procedures can be time consuming, they are mostly used in the visualization of SES.[3, 12-21] Rocchia *et al.* subsequently converted an analytical algorithm to a semi-analytical version with numerical representation of solvent accessible arcs at a very high resolution (i.e. much higher than the finite-difference resolution) to facilitate the mapping of the SES to the grid so the mapping error is negligible in numerical PB calculations.[5] This method is much faster and accurate enough for numerical solution of the PB equation, though it is without an analytical expression.

The van der Waals (VDW) surface, or the hard sphere surface, represents the low-dielectric molecular interior as a union of atomic van der Waals spheres. This is a very efficient algorithm, though there exist many nonphysical high (solvent) dielectric pockets inside the solute interior when the VDW definition is used. Considering the limitation, the modified VDW definition was proposed. The basic idea of the modified VDW definition is to use the solvent accessible surface (SAS) definition for fully buried atoms and the VDW definition for fully exposed atoms.[22] However, the method is difficult to be optimized to reproduce the more physical SES definition.

The density approaches have recently been developed and can be used for numerical PB solutions. Either a Gaussian-like function or a smoothed step function has been explored in previous developments.[23, 24] It has been shown that if the functional form is allowed to change, the density function can be explicitly optimized to reproduce the classical SES definition at least for certain “small” solvent probe radii, though this cannot be generalized to arbitrary probe radius values.[25] In these types of approaches, a distance-dependent density/volume exclusion function is used to define each atomic volume or the dielectric constant directly.[26-28] This is in contrast to the hard-sphere definition of atomic volume as in the VDW or the SES definition. Therefore, the surface cusps are removed by the use of smooth density functions.[23, 24]

In this study, we intend to address the difficulties of computing SES of complex molecular structures. The difficulties arise because SES is formed by different patches thus its analytical formulation must be piecewise and localized.[29] There are at least three consequences from the difficulties. First, the algorithms are very costly. For example, the optimized SES still takes about a third of total CPU time of the numerical PB solvers in Amber.[30, 31] Second, they are hard to be ported to modern parallel platforms, such as GPUs. Third, it is difficult to obtain surface curvatures or other higher-order surface parameters that are often used for implicit solvent simulations. To overcome the difficulties, our solution is to use a smooth level set function to approximate SES. As will be shown below, the single-thread CPU cost can be reduced noticeably when the level set function is used. Given the level set function, the computation of SES can be carried for all grid points independently and simultaneously, and thus achieve parallel computation. Finally, surface curvatures or other higher-order surface parameters are readily available from the smooth level set function.[32]

Given the complexity of the SES algorithms, we explored a machine learning strategy to look for a smooth level set function. Indeed, in recent years, deep learning techniques have been widely adapted in handling multi-variable and highly non-linear functions similarly challenging such as SES. Deep learning/neural network is inspired by and resembles the human brain. The input variables are multiplied by the respective weights and then undergo a transformation based on an activation function to obtain the outputs.[33] It has been mathematically proven that a single hidden layer was able to solve any continuous problem.[34] With all its advantages, deep learning has been applied in various biological fields, including gene expression[35-42], protein secondary and tertiary structures[43-55], protein-protein interactions[56-62], and others.[63]

In the following, we first overview the necessary physical and mathematical concepts needed to set up a machine learning SES model, along with the computational details in collecting the training data and conducting the training process. This is followed with accuracy analyses of the machine-learned SES, robustness analyses, timing analysis, and finally applications to implicit solvent simulations. We end the presentation with a discussion of potential future developments.

3.2. Method

As widely adopted for numerically solving partial differential equations, the finite difference method uses a uniform Cartesian grid to discretize the PB equation. The grid points are numbered as (i, j, k), where $i = 1, \dots, x_m$, $j = 1, \dots, y_m$, $k = 1, \dots, z_m$, and x_m, y_m and z_m are the numbers of points along the three axes. The grid spacing between neighboring points can be uniformly set to h. To discretize the linearized PB equation,

$$\nabla \cdot \varepsilon \nabla \phi - \lambda \sum_i n_i q_i^2 \phi / kT = -4\pi\rho$$

where the charge density ρ can be expressed as $q(i, j, k)/h^3$, $q(i, j, k)$ is the total charge within the cubic volume centered at (i, j, k), λ is a masking function for the Stern layer. In the salt related term, n_i is the number density of the ion of type i in the bulk solution, q_i is the charge of the ion of type i, k is the Boltzmann constant, and T is the temperature. The final discretized PB equation can be expressed as follows, if we ignore the salt related term as it is often modeled to be away from the molecular surface,

$$
\begin{aligned}
\{\, \varepsilon\left(i-\frac{1}{2},j,k\right)[\phi(i-1,j,k)-\phi(i,j,k)] &+ \varepsilon\left(i+\frac{1}{2},j,k\right)[\phi(i+1,j,k)-\phi(i,j,k)] \\
&+ \varepsilon\left(i,j-\frac{1}{2},k\right)[\phi(i,j-1,k)-\phi(i,j,k)] \\
&+ \varepsilon\left(i,j+\frac{1}{2},k\right)[\phi(i,j+1,k)-\phi(i,j,k)] \\
&+ \varepsilon\left(i,j,k-\frac{1}{2}\right)[\phi(i,j,k-1)-\phi(i,j,k)] \\
&+ \varepsilon\left(i,j,k+\frac{1}{2}\right)[\phi(i,j,k+1)-\phi(i,j,k)]\}/h^2 = -4\pi q(i,j,k)/h^3
\end{aligned}
$$

where $\phi(i,j,k)$ is the potential at grid (i,j,k), and $\varepsilon\left(i-\frac{1}{2},j,k\right)$ is the dielectric constant at the mid-point of the grids (i,j,k) and $(i-1,j,k)$. All other ϕ and ε are defined similarly here.

If a grid point and all its six neighbors are in the same region, either in solvent or in solute, the point is termed a regular grid point. Clearly, the simple discretization scheme above can be applied only to regular grid points, because only then the mid-point dielectric constant can be determined (equal to either solvent or solute dielectric constant). However, for an irregular grid point, which means that it has at least one neighboring grid point belongs to the different region, complication arises. Thus, how to define and determine the interface parameters, and then to determine the dielectric constant based on the interface information, is a key for setting up the linearized PB equation. Without question, SES is the mostly adopted interface definition, as it was found to reasonably reproduce both energetic and dynamic properties of solvated molecules in explicit solvent. [125-127, 143] After choosing an interface definition, the discretization can then be handled with many different schemes. For example, the harmonic averaging method is a class of such strategies widely used in biomolecular simulations.[94, 116, 166]

It is often more convenient to use a level set function to represent an interface.[167] The level set function is often in the form of a second-order continuous distribution function φ, satisfying,

$$\varphi(x, y, z) < 0, when\ (x, y, z) \in \Omega^-$$

$$\varphi(x, y, z) = 0, when\ (x, y, z) \in \Gamma$$

$$\varphi(x, y, z) > 0, when\ (x, y, z) \in \Omega^+$$

Here, Γ represents the interface, Ω^- and Ω^+ denote the solute and solvent regions, respectively. For example, a signed distance function representing a sphere interface, centered at (x_0, y_0, z_0) with a radius of R can be expressed as,

$$\varphi(x, y, z) = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} - R$$

Level set functions can be used to determine the interface by setting $\varphi = 0$ conveniently, but also to compute various interface parameters, such as the normal and tangential directions on the interface as[167]

$$\boldsymbol{\xi} = (\varphi_x, \varphi_y, \varphi_z)$$

$$\boldsymbol{\eta} = (\varphi_y, -\varphi_x, 0)$$

$$\boldsymbol{\tau} = \left(\varphi_x\varphi_z, \varphi_y\varphi_z, -\varphi_x^2 - \varphi_y^2\right)$$

Here, $\boldsymbol{\xi}$, $\boldsymbol{\eta}$ and $\boldsymbol{\tau}$ are the normal and two tangential directions, respectively. In addition, some advanced discretization schemes, like IIM,[96, 98, 119, 156] also require higher-order interface parameters, such as surface curvatures that can also be computed as follows,

$$\xi_{\eta\eta} = -\frac{\varphi_{\boldsymbol{\eta\eta}}}{\varphi_{\boldsymbol{\xi}}}$$

$$\xi_{\tau\tau} = -\frac{\varphi_{\boldsymbol{\tau\tau}}}{\varphi_{\boldsymbol{\xi}}}$$

$$\xi_{\eta\tau} = -\frac{\varphi_{\eta\tau}}{\varphi_{\xi}}$$

In summary, level set functions are convenient mathematical tools in handling interface-related problems: they can be used to obtain the interface itself and all its geometry parameters in a straightforward manner.[167] Thus, it would greatly benefit discretization of the PB equation if we can express SES as a level set function. However, construction steps of SES are simply too complex to be reproduced algebraically. What we are trying to do here is to explore how to approximate SES with a level set function that is learned on modern computers. Furthermore, we do not intend to reproduce SES 100%, which is impossible, as we also aim to find a smooth function that is more friendly for interface tracking in implicit solvent simulations, such as numerical PB solutions.

Artificial neural networks (ANNs), usually simply called neural networks, are mathematical models that have been motivated by the brain function.[168-169] A simple example of an ANN structure is shown below in Figure 3.1.

The basic idea of a neuron model is that an input, $\boldsymbol{x}$, together with a bias, b, is weighted by $\boldsymbol{w}$, and then summed together,[170] as shown schematically in Figure 1. The bias, b, is a scalar value whereas the input $\mathbf{x}$ and the weights $\boldsymbol{w}$ are vectors, i.e., $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{w} \in \mathbb{R}^n$ with $n \in \mathbb{N}$ corresponding to the dimension of the input. The sum of these terms, i.e. $z = \boldsymbol{w}^T\boldsymbol{x} + b$, forms the argument of an activation function, $f()$, resulting in the output of the neuron model,

$$y = f(z) = f(\boldsymbol{w}^T\boldsymbol{x} + b)$$

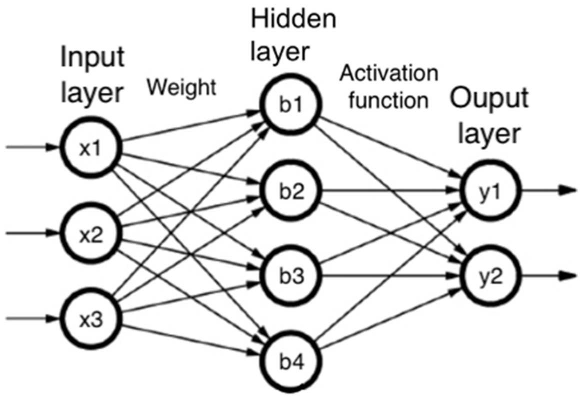


Figure 3.1. Structure of an artificial neural network with one hidden layer. Here, each circular node represents an artificial neuron, and an arrow represents a connection from the output of one artificial neuron to the input of another.

The role of the activation function, $f()$, is to perform a non-linear transformation of z. There are many activation functions in practice, such as sigmoid, Tanh, ReLU, Leaky ReLU, and so on, and the ReLU activation function is usually the most popular activation function for deep neural networks.[170]

Though its structure may seem highly complicated, a neural network can be viewed as a combination of simple elementary functions. Thus, a neural network is differentiable to any order, as all its component functions are differentiable to any order, except for a countable number of points. This is an important reason why we have chosen a machine learning approach to express the SES level set function, because it ensures that the SES level set function can be used to compute surface derivatives to any order as needed.

Due to its excellent mathematical properties, ANNs have shown great promises in a range of applications.[171] Most ANN applications fall into two categories: function approximation / regression analysis and classification problems. In this study, the ANN is applied as a classification tool because the goal is to know which region a grid point belongs to (solvent or solute region) in a PB system.

3.3 Computational Details

The training data were generated from the Amber PBSA benchmark suite of 573 biomolecular structures.[50] A modified PBSA program was used to print out the training data. The default atomic cavity radii were read from the topology files. The solvent probe radius was set to 1.4 Å, the grid spacing to 0.95 Å, and all other parameters remain as default in the PBSA module in the Amber 20 package.[172] The training software package is Keras in TensorFlow, version 2.2.4.[173] Adam was chosen as the optimizer, and the squared hinge function as the loss function. The partition ratio of training and validating sets is 9:1, and an early stopping criterion of 100 epochs was used.

The training data contains two parts: labels or the target values, and the variables. The labels are integer values of irregular grid points generated by the Amber/PBSA surface builder for the geometry-based SES, termed classical SES below.[116] This implementation follows the basic idea as outlined by You *et al.*[129] and Rocchia *et al*.[124] Specifically, a numerical representation of the solvent accessible arcs was used to map the reentry surface to the finite-difference grid. Briefly, there are two steps in assigning the integer labels. First is to determine the solvent accessible arcs that are numerically represented centers of solvent accessible probes tangential to at least two atoms simultaneously. The density of the numerical arcs is set to be high enough, so that the mapping error is negligible in numerical PB calculations. Second is to label grid points nearby the molecular surface as inside or outside with a multi-step labeling scheme.[116]

The irregular points of the outside (solvent) region are labeled as -1, and those of the inside (solute) region are labeled as +1. The variables of the training data are the tuples of coordinates and the radii of those atoms near the irregular grid points, in the unit of Å. An atom is considered "near" a grid point if the distance between the grid point and the atom is within R+2Dp, where R is the atom radius and Dp is the diameter of the probe. The coordinates of the atoms are expressed in the relative frame whose origin is the grid point of interest. The neighbor atoms were then sorted in an ascending order based on the distance between the grid point and their surfaces. In the training data, the maximum neighbor atoms are 48, so the maximum number of variables are 192. The variable dimensions are uniformly enlarged to 200, by filling in zeros for blanks. Overall, there are about 6 million entries of data generated from the Amber PBSA benchmark suite, which was separated into 6 subsets, each about 1 million entries.

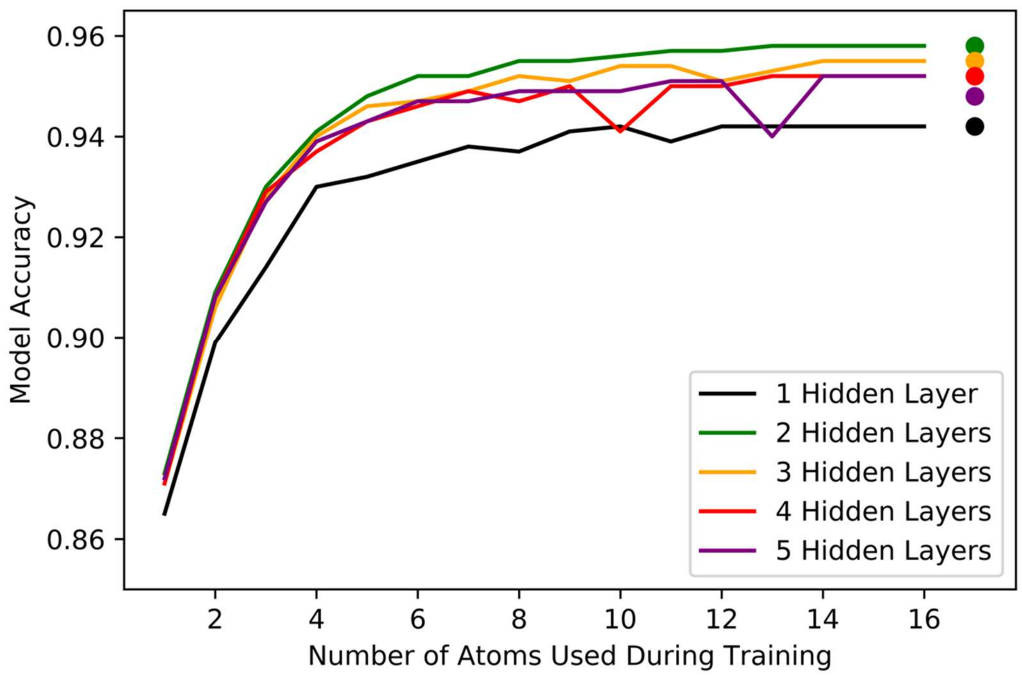


Figure 3.2. Incremental training of the model. The ANN model was set up with different numbers of hidden layers, each hidden layer is of 200 neurons. The model was trained with the first training data set. The analysis was also repeated with two additional training sets.

The training was conducted in three steps. First, the model was trained incrementally by adding one atom at a time, which means in the first round of training, the model was fed with only the first (nearest) atom’s coordinates and radius. After the training reaches stabilization, the model was fed with the first two atoms’ data, and so on. After enough number of atoms were fed to the atom, the accuracy of the model reaches certain stable level. We then fed it with all the nearby atoms to complete the training. This step is necessary to initialize the ANN model training, because for such a high-dimension (200) nonlinear system, its local minimum can be extremely deep, causing the training optimization to fail.

In this step, we also investigated how many hidden layers are needed to reach stable prediction accuracy, starting with one layer to five layers. As shown in the Figure 3.2, the accuracy of the model becomes stable at 16 atoms, no matter how many hidden layers were used in the ANN. Therefore after 16 atoms, all available nearby atoms were fed to the model. Figure 3.2 further shows that 2 hidden layers are already very good for our problem. The second and the third steps are to determine the number of neurons and to finalize the model, respectively. The training script and final model can be found at http://rayluolab.org/ml-ses/.

3.4. Results and Discussion

To evaluate the overall performance of the machine-learned SES method, we first did an accuracy test on different structures. The test data sets were generated in the same fashion as the training data sets. Except for the original training protein monomers, we also included two extra sets of biomolecules for this analysis, nucleic acid structures[174] and protein/protein complex structures from previous PBSA developments.[160] The testing results are shown in Table 3.1.

Table 3.1. Model accuracy with different structural data sets. The first six data sets are those from the training phase, and the last two sets are not included in the training phase.

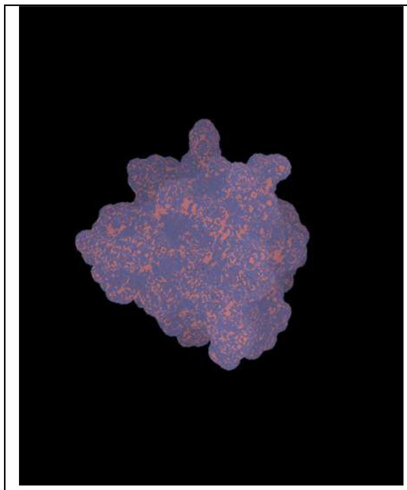
Data sets	Training data 1	Training data 2	Training data 3	Training data 4	Training data 5	Training data 6	Nucleic acid	Protein

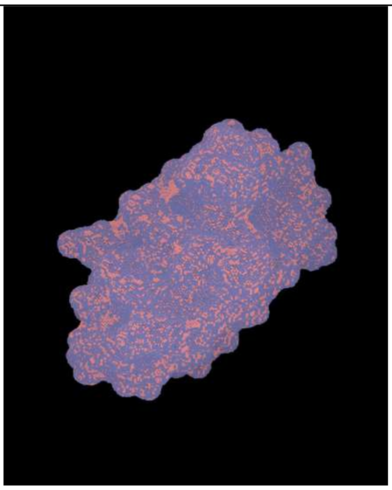
								comple x
Accuracy	95.96%	96.01%	96.02%	96.03%	96.26%	97.78%	97.07 %	95.16%

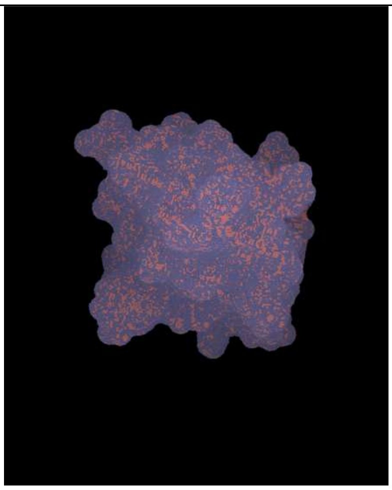
Table 3.1 shows that the machine-learned SES performs very well for all structural sets, including those not included in training. The classification accuracies are basically around 96% for all data sets, with a variance about 1%. The method performs the best for training data set 6 and the nucleic acid data set. This is because those two contains mostly smallest molecules among all tested molecules. In general, a small molecule has simpler surface geometry, so it is easier to predict their geometries. On the contrary, the protein/protein complex structures are much larger and thus have more complicated surfaces, so the method performs slightly worse on the data set, but still obtains an over 95% accuracy.

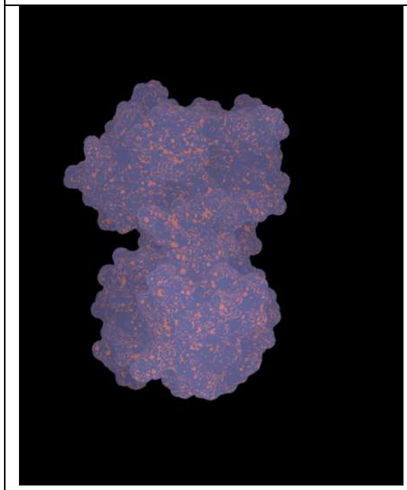
Another point worth mentioning is that the accuracy test clearly shows excellent transferability of the method, from the smaller training protein monomers to the nucleic acid structures and the larger protein/protein complex structures. The testing molecules overall do not resemble those in the training sets, such as the nucleic acids. Nevertheless, the method can successfully predict the SES of those unseen structures, showing that the method has indeed learned the fundamental rules for predicting the SES surface. In summary, the consistent accuracy confirms that the machine-learned SES method can be applied to typical computational analyses of biomolecules.

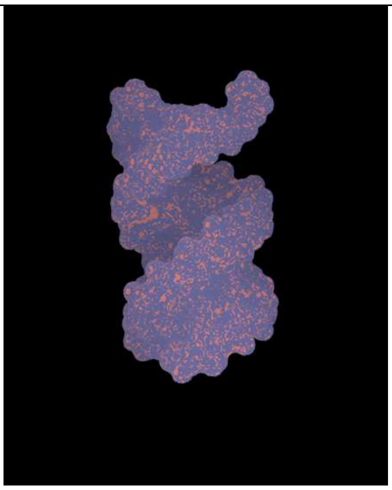
We chose six representative biomolecules and compared their molecular interfaces generated with three different methods: the newly developed machine-learned SES, the classical geometry-based SES, and a revised density function surface.[143] The superimposed surfaces of the machine-learned SES and the classical SES are shown in Figure 3.3.


(a)


(b)


(c)


(d)


(e)

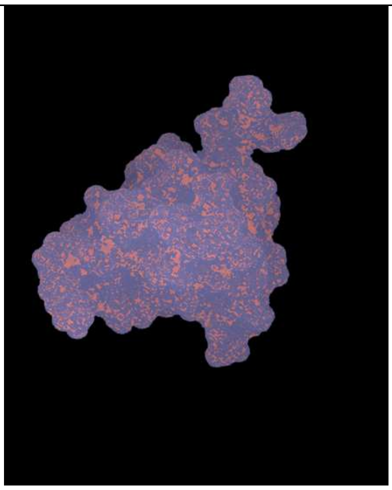

(f)

Figure 3.3. Superimposed rendering of the machine-learned SES (blue) and the classical SES (red) of representative molecules. (a) PDB ID: 1enh, all-alpha protein; (b) PDB ID: 1pgb, all-beta protein; (c) PDB ID: 1shg, alpha/beta protein; (d) PDB ID: 1w0u, protein/protein complex; (e) PDB ID: 3czw, RNA duplex; (f) PDB ID: 3fdt, protein/DNA complex.

It is clear from Figure 3.3 that the machine-learned SES excellently reproduces the classical SES for all tested molecules, including both proteins and nucleic acids, consistent with the accuracy analysis shown in above.

Because SES is a molecular surface, it should possess both translational and rotational symmetries. The translational symmetry is naturally included in the machine-learned SES, as the input coordinates are relative to the grid point and are invariant to the translational transformations. However, the rotational symmetry is in general not preserved if it is not imposed. Thus, if the machine-learned SES can reproduce the classical SES, it should be able to "learn" the rotational symmetry through the training process. Thus, whether the method possess rotational symmetry is a direct test of whether it has learned the essentials of the classic SES.

To test the rotational symmetry of the machine-learned SES, 20,000 interface points are randomly selected from the training data set and tested with different orientations imposed. Without loss of generality, the z-axis was chosen as the rotational axle to generate different orientations. Specifically, we performed the following transformation on the input coordinates of the nearby atoms for each grid point:

$$x' = xcos(\theta) - ysin(\theta)$$

$$y' = y cos(\theta) + x sin(\theta)$$

Here, θ is the rotation angel and its values are 30°, 60°, and all the way through 330°. Note that the input coordinates are already transformed with the grid point as the origin. Next the trained machine-learned SES is used to predict the grid labels, and the agreement rate with respect to the original predicted grid labels is then computed.

Table 3.2. Rotational symmetry test. The agreement rate is the percentage of transformed inputs give the same prediction (+1 or -1) as the original inputs.

Angel	30	60	90	120	150	180
Rate (%)	98.6	98.5	98.1	98.3	98.8	98.4
Angel	210	240	270	300	330	
Rate (%)	98.2	98.8	98.7	98.6	99.1	

As shown above, our machine-learned SES indeed preserves the rotational symmetry, because the agreement (aka precision) rate is already higher than the accuracy rate (~95%). As we have discussed above, this is achieved without any human intervention. The analysis shows that the machine-learned SES has learned the essentials of the classic SES very well. In addition, if we look at the actual level set function values that the machine-learned SES gives, for about 95% of the tested grid points the deviations between the rotated inputs and original inputs that are less than 0.001 and for 97% of the tested grid points the deviations between the two are less that 0.01. Thus our model not only gives the correct classification, but also gives almost the level set value for different orientations.

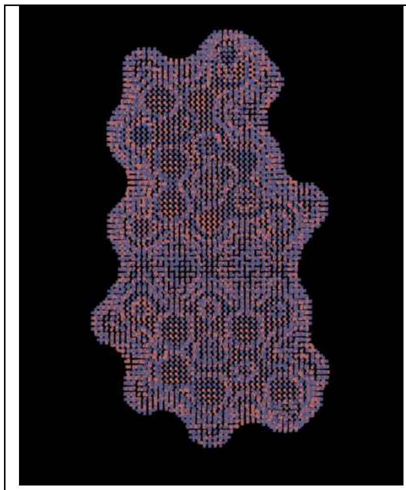
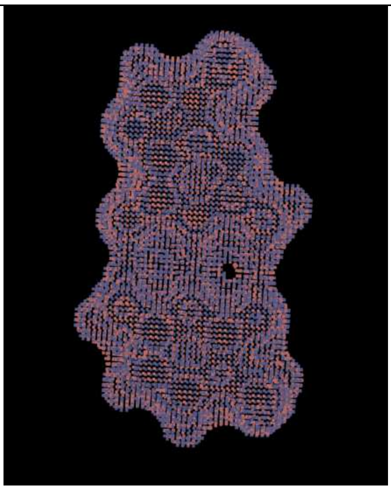
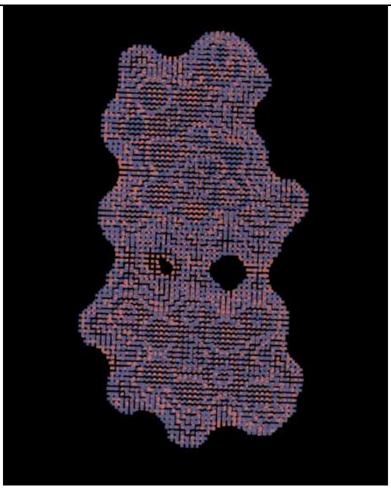
We further evaluated how the machine-learned SES behaves in binding/unbinding scenarios, with a DNA base pair (dGdC, a pair of guanine and cytosine bases) as a test case. Starting from the hydrogen bonded dGdC dimer, the two bases are manually pulled part by 0.5 Å each step to see how the two SES methods agree with each other for each of the tested conformations.

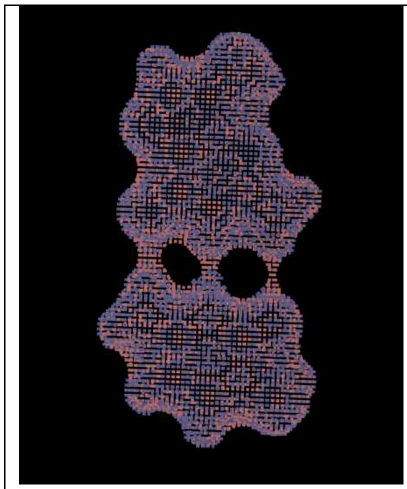
It is clear from Figure 3.4 that the machine-learned SES and the classical SES agree excellently for all base separation distances tested, consistent with the performance for single molecules shown above. For example, both surfaces first generate a small cavity on the right and then another one on the left. The two cavities gradually increase and finally break all three hydrogen bonds between the two bases, causing the two molecules to separate completely. On the other hand, the two surfaces still differ somewhat during the unbinding process. However, whether these details are of any physical significance is subject to debate. In contrast, the machine-learned SES does not contain the spikes and as it is produced by a smooth function. The binding / unbinding test shows that the machine-learned SES is numerically more stable than the classical SES in the more challenging dynamical process.

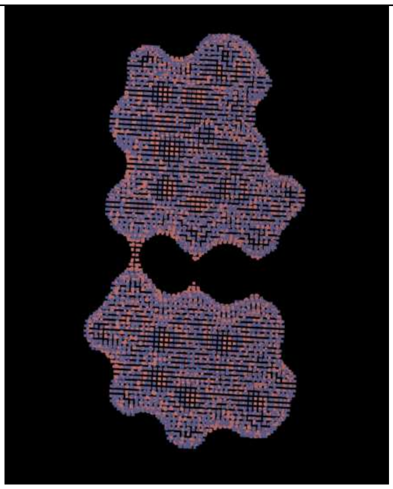
To further test the robustness of our model, we want to know how the model performs away from the interface (solvent region or deep buried molecular interior), since the ANN model was initially trained with irregular points near the solute-solvent interface.

We have tested the performance of the model in the entire solvation box, not just nearby the interface, include those grid points that are either far away outside the solute or deeply buried inside the solute. The performance of the model is shown in Figure 3.5.

Figure 3.5 shows that the model clearly does not work at grid points far away from the interface. Specifically, Figure 3.5(a) illustrates the presence of an “island” or a small cluster of predicted interface grid points, outside the interface. Even more serious failures are within the molecule interior as illustrated in Figure 3.5(c), where hundreds of small clusters of interface grid points are wrongly predicted. This is surprising because the model can predict with an accuracy level over 95% in the most difficult interface region yet made hundreds of mistakes in the much easier regions.

 (a)	 (b)	 (c)


(d)


(e)

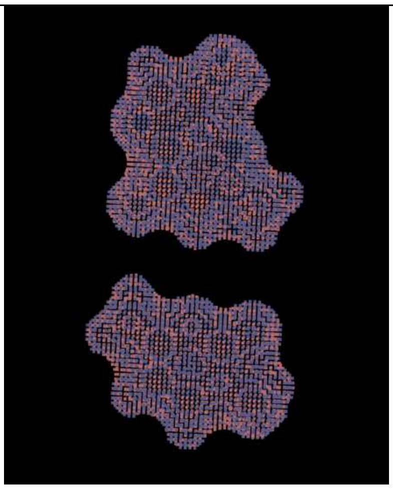
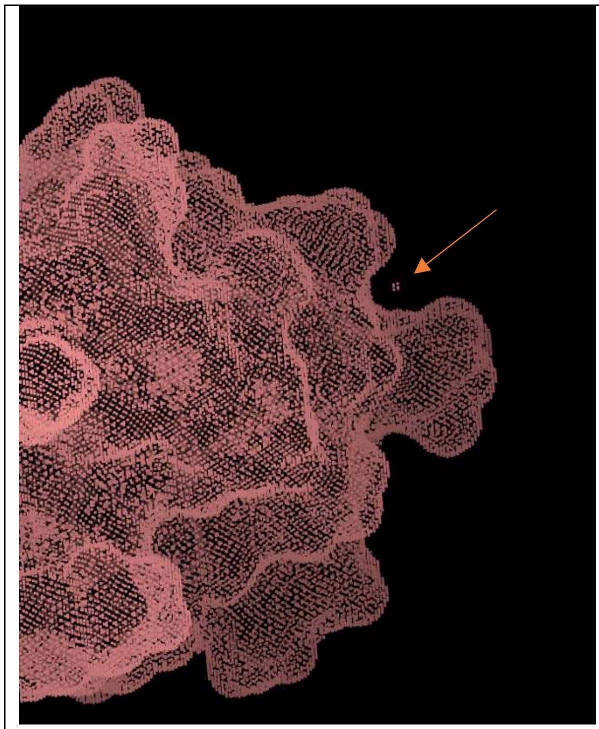
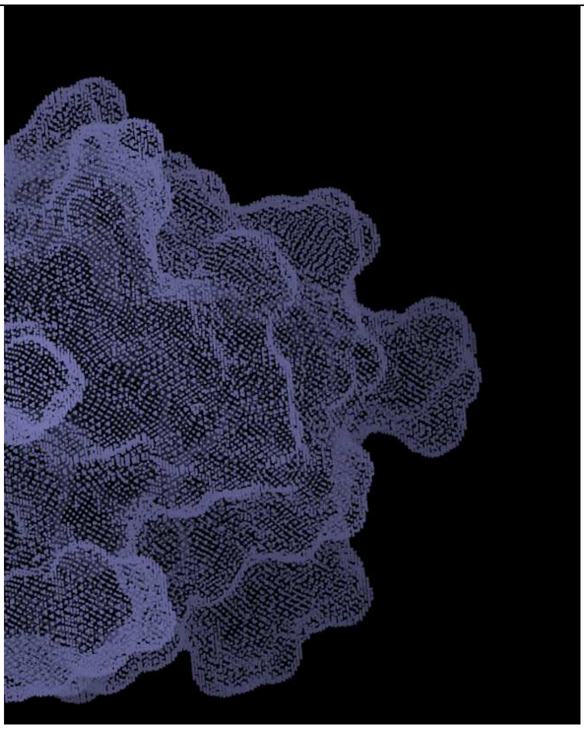
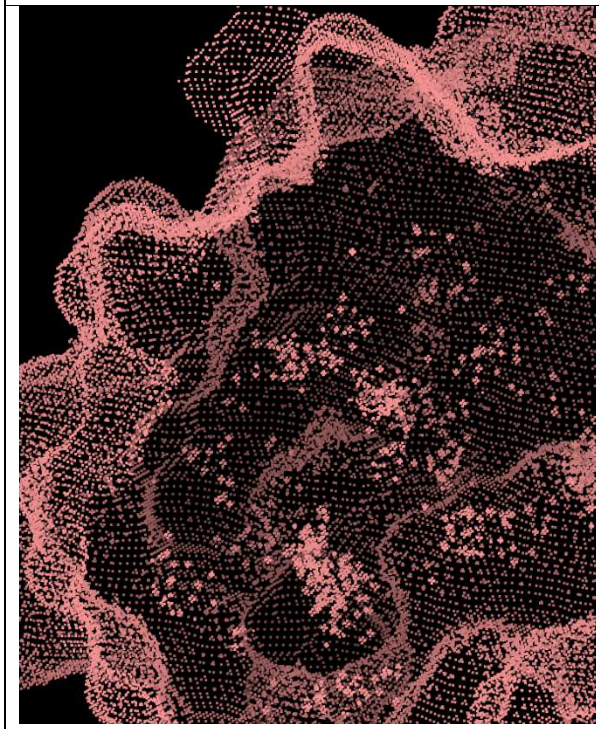
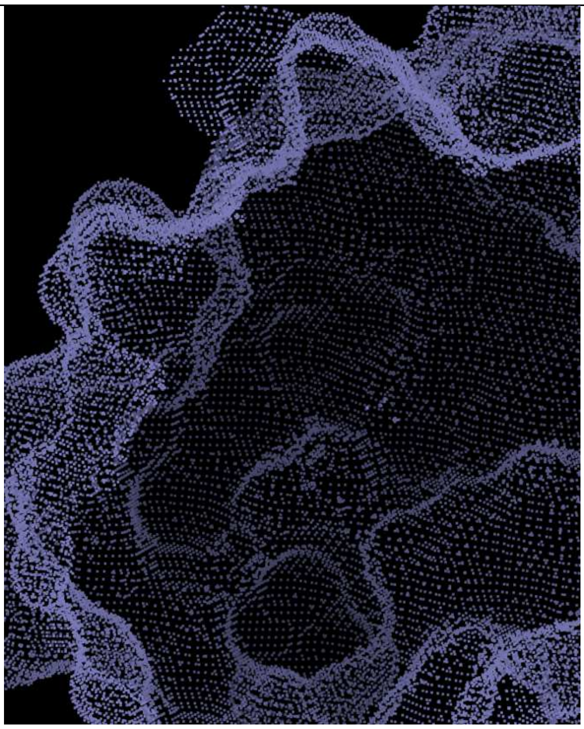

(f)

Figure 3.4. Superimposed rendering of machine-learned SES (blue) and the classical SES (red) of the dGdC complex. Starting from the hydrogen bonded structure, the two molecules are manually pulled part for 0.5 Å each step.

These failures are due to the lack of similar data in the training of our initial model as only irregular grid points were fed to the training of the model. Thus, the remedy is to assembly a supplementary training set with grid points not just near interface, i.e. the irregular grid points in the initial training. In the inside region, the supplementary set includes all grid points that are inside SES but outside VDW. In the outside region, the set includes all grid points that are in the reentry cones formed by solvent accessible arcs and their associated atom pairs. Training of the initial model on the supplementary set leads to the second version of the model. Without surprise, the second version of the model can successfully classify those grid points incorrectly predicted by the initial model as shown in Figure 3.5(b)/3.5(d).

 (a)	 (b)
	

(c)	(d)

Figure 3.5. Detailed views of SES as predicted by first and second versions of the machine-learned SES model. The PDB id of the tested molecule is 1shg. (a) and (c) are from the first version of the model, outside vision and the inside vision, respectively. (b) and (d) are from a second version of the model, outside vision and the inside vision, respectively.

In summary, the extrapolation analysis also demonstrates the robust nature of the machine-learned SES, as it not only works well near the interface, but also can be applied to the entire solvation box.

To illustrate the applications of the machine-learned SES for biomolecular studies, we implemented it into the AMBER/PBSA program. It follows the same algorithm structure as the revised density function.[143] In both approaches, the goal is to assign level set function values for all grid points nearby a molecular solute.[143] The algorithm contains two parts: pre-processing and model predicting. The pre-processing part includes all the preparation needed for generating inputs for the machine-learned level set function. First, it collects all grid points that fall inside the SAS but outside the VDW of each solute atom. This step involves a loop over all solute atoms. Second, it generates a list of neighboring atoms of each grid points which were collected in the first step and then sorts them in the order of distances. The pre-processing part is relatively quick and takes only about 5% of the total time. The predicting part is the main bottleneck of the model, involving a loop over all collected grid points.

Given the current CPU implementation based on the Keras modules, we conducted a timing analysis of the method with the Amber/PBSA benchmark suite used in the training.[50] The classical SES was computed with the default SES builder in Amber/PBSA program, which is already highly optimized.[116] Figure 3.6 compares both timing data of methods. The regression shows that the machine-learned SES method is about 2.5 times as efficient as the classical SES algorithm on the tested compute node on our local cluster.

As discussed before, an advantage of the machine-learned SES is that it is naturally suitable for parallel computing, because the level set function calls for the grid points are independent of each other and thus can be distributed to different computing cores. In other words, the computation cost of the machine-learned model in principle scales almost linearly with the number of cores, so that its cost can be dramatically reduced on highly parallel platforms such as GPUs. Of course, further development to optimize the code and to port it to the GPU platform is necessary to realize its full potential, as observed for the PB solvers in our previous developments.[152, 165-166]

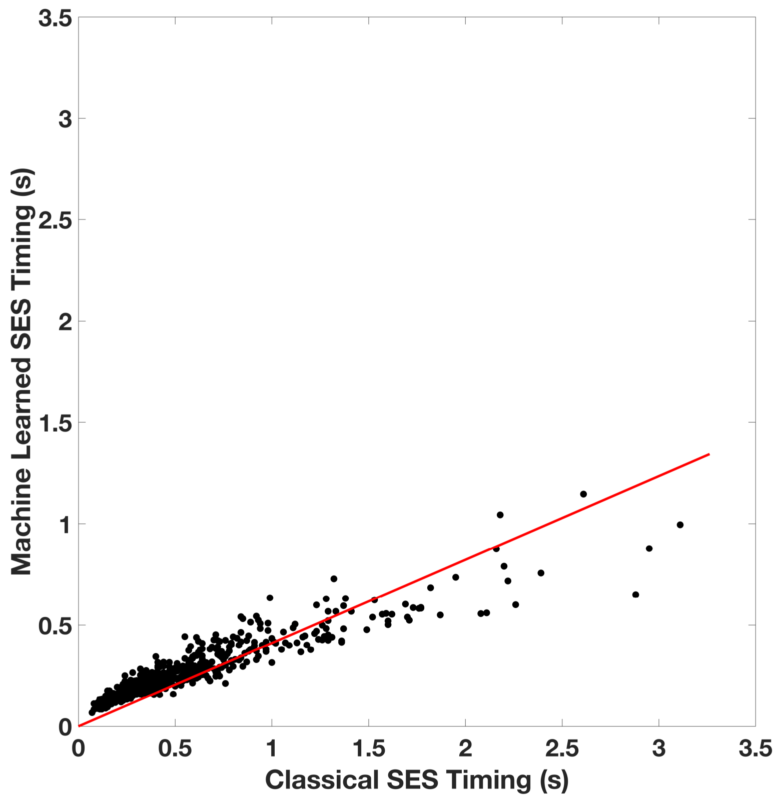
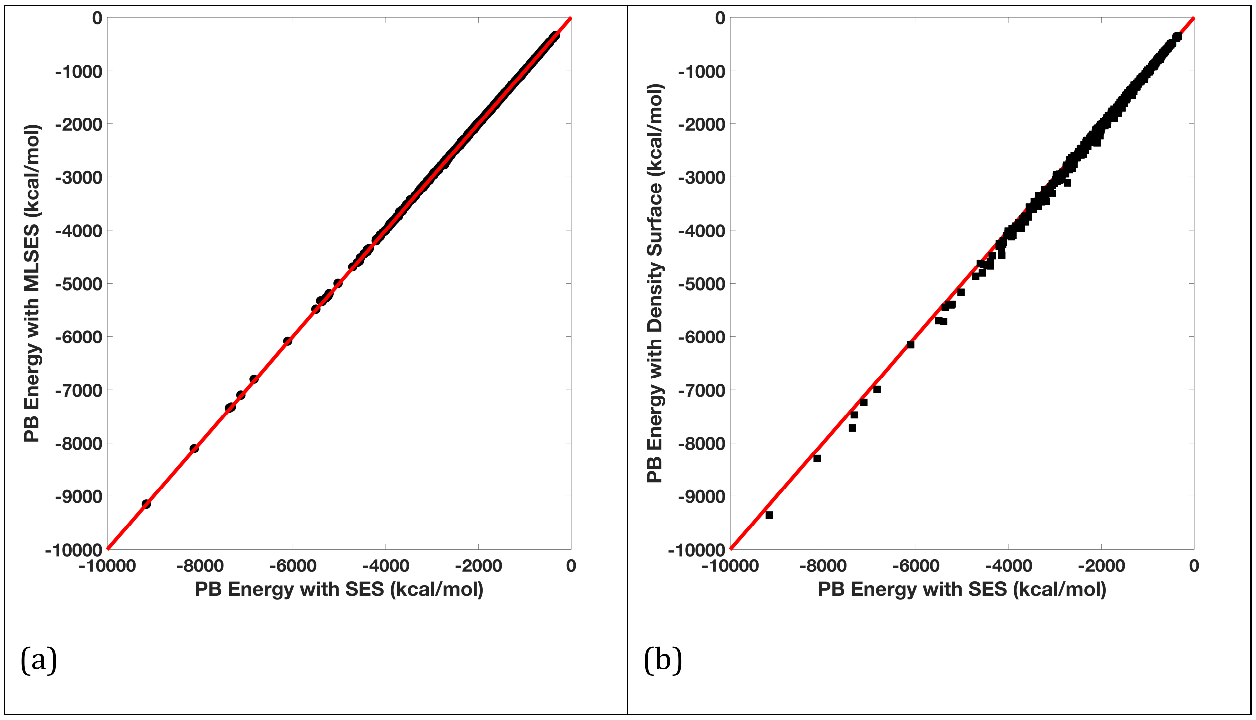
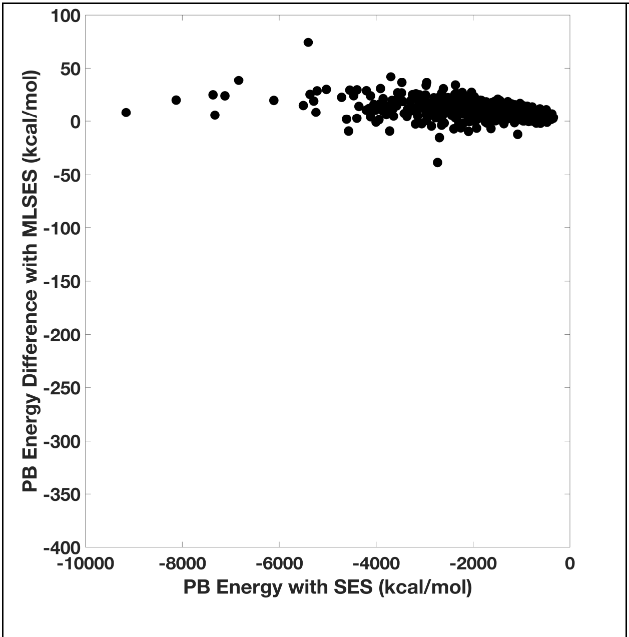


Figure 3.6. Timing comparison of the machine-learned SES and the classical SES. Both sets of data were collected on a single core of an INTEL Xeon E5-4620 CPU. The grid spacing was set to 0.95 Å, which is close to the visualization setting (1 Å) used in this study. The regression line between the two sets of timing data is y=0.4114x.

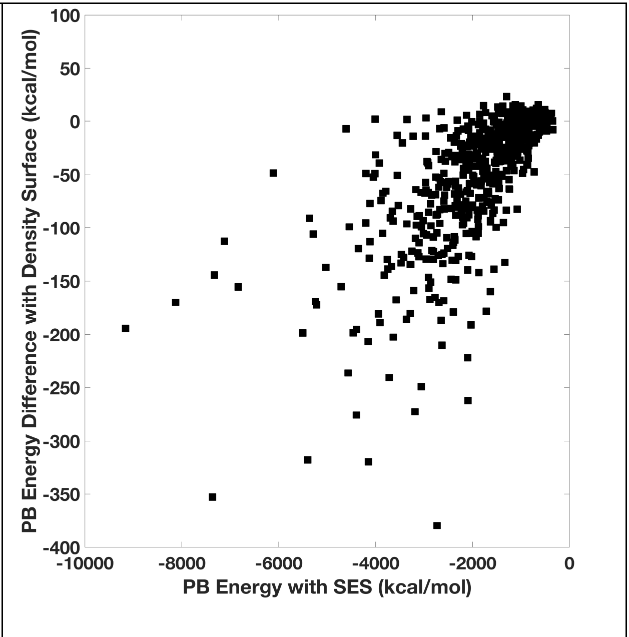
The PB reaction field energies were computed with the classical SES, the machine-learned SES, and the revised density function with otherwise identical conditions for all protein structures in the AMBER/PBSA benchmark suite.[50] Figure 3.7 shows the agreement between the computed PB energies with both machine-learned SES and the revised density function surfaces and those with the classical SES surface.

It is clear from Figure 3.7 that the machine-learned SES performs uniformly well when used for PB energy calculations. The deviations between the PB energies with the machine-learned SES and those with the classical SES are around 1%. In contrast, the average deviation is roughly 2% but the maximum deviation is over 10% between the PB energies with the density function surface and those with the classical SES. Figure 3.7(c)/(d) further shows that the PB energies from the machine-learned SES are more random around the mean of zero but the PB energies from the density function are systematically more negative. This is because a small solvent probe (0.7 Å) has to be used in the density function to achieve reasonable agreement with the PB energies with the classical SES.[143] If the default probe of 1.4 Å is used, the PB energies would be 30% more positive than those with the classical SES.





(c)



(d)

Figure 3.7. PB reaction field energies with different molecular surfaces. (a) Correlation between energies from machine-learned SES and classical SES. (b) Correlation between energies from density function and classical SES. (c) Energy differences in (a). (d) Energy differences in (b). The lines in (a) and (b) are the diagonal line "y=x" as reference.

3.5. Conclusion

In this study, we developed a new level-set formulation to generate the solvent excluded surface through a machine learning process. SES is a widely used molecular surface definition and there are at least two advantages of expressing SES in a level set formulism: to facilitate parallel computing and to make it differentiable for future applications in implicit solvent simulations.

The level set function was trained on the data generated from a set of heterogenous biomolecular structures, containing about 6 million entries. The training was conducted in three steps to determine the numbers of nearby atoms needed to define the level set, the number of hidden layers, and the number of neurons of each layer. After the final training, the machine-learned SES can predict the classical SES with an accuracy over 95% on tested proteins, nucleic acids, and complex structures.

Analysis of visualized molecular surfaces of tested biomolecules shows that the machine-learned SES agrees excellently with the classical SES. It is also clear that a previous approach based on atomic density functions is clearly insufficient, generating a molecular surface often larger than the classical SES and causing deeper crevices in the reentry regions. Besides, our machine-learned model is incredibly stable, in terms of rotational and translational variation of the molecules.

We also performed a timing analysis between the machine learned SES and the classical SES algorithm. The analysis shows that the machine-learned SES is roughly 2.5 times as efficient as the classical SES routine in AMBER/PBSA on the tested CPU platform. We expect further performance gain on massively parallel platforms such as GPUs given the ease in converting the machine-learned SES to a parallel procedure.

To further test the machine-learned SES, we implemented it into the AMBER/PBSA program to see if it can be used to reproduce the PB reaction field energies computed with the classical SES. Our analysis shows that the two sets of energies differ on average only about 1%, better than the 2% average deviation when the energies between the density-function surface and the classical SES are compared. Furthermore, the deviations in PB energies by the machine-learned SES are uniformly distributed yet the deviations in PB energies by the density-function surface often exhibit large deviations as high as 10%. This shows that the machine-learned SES is much more stable in reproducing the classical SES in realistic applications.

Chapter 4

4.1. Introduction

Atomistic simulations of biomolecules have been applied in a wide range of biological systems.[1] While additive nonpolarizable models will continue to play important roles,[2-4] nonadditive polarizable models are expected to extend our ability to study more complex biomolecular systems and processes. Nonpolarizable models typically use fixed atom-centered partial charges to model electrostatics and include polarization response to the environment (mostly in water) only in an averaged, mean-field manner. Subsequently, nonpolarizable models that provide excellent descriptions of the homogeneous bulk phase are poor models for gas-phase clusters or in nonpolar solvents. The importance of modeling nonadditive effects is well known.[5] For example, the gas-phase water dimer interaction energy is overestimated by more than 30% in the TIP5P model.[6] Similarly for large biomolecular systems, there are concerns that such models cannot correctly account for situations where the same nonpolarizable moiety is exposed to different electrostatic environments/solvents, either within a single large structure or during a simulation process. In addition, there is an inherent inconsistency in most nonpolarizable models related to their static inclusion of average bulk polarization within the potential. This results in internal energies and other properties that are derived against a gas-phase reference state, which is already “pre-polarized” for the liquid phase. These limitations lead to issues in modeling multiple important problems such as pH-dependent processes, ion-dependent interactions, order-disorder transition, enzyme reactions, and so on.

In response to the above concerns, much effort has been invested on the inclusion of explicit polarization within the molecular mechanics potentials.[7-9] Several methods are available to explicitly model polarization in molecular simulations, such as the Drude oscillator,[10, 11] fluctuating charges,[12] and induced dipoles.[6, 13, 14] The use of polarizable point dipoles is a classical approach with a long history in molecular simulation.[15] The original induced dipole model of Applequist places induced point dipoles on atom centers.[16] However, this model suffers from the so-called "polarization catastrophe": when interaction between two mutually interacting induced dipoles with atomic polarizabilities diverge at a finite distance. Thole proposed a solution by applying a damping function to induced dipole – induced dipole interactions.[17] However, a drawback to this model is that it does not prescribe how induced dipoles and permanent charges interact. A great deal of effort has been devoted to developing modern polarizable models, including the fluctuating charge models[18, 19] in the context of OPLS-AA, the fluctuating charge model and the Drude oscillator model[20-23] in the context of CHARMM, and detailed multipole expansions and more complicated MM potentials in the context of Amoeba.[24] In Amber, polarization was implemented with induced dipoles.[25] In Amber ff12pol, the induced dipoles are calculated using Thole models to avoid "polarization catastrophe".[26-29]

Another limitation of widely adopted nonpolarizable models is their use of partial atomic charges in the electrostatic models, which often lack sufficient mathematical flexibility to describe the electrostatic potential around molecules. Williams showed that optimal least-squares fitting of atom-centered partial charges resulted in relative root-mean-square errors of 3-10% over a set of grid points in a shell outside the surface of a series of small polar molecules.[30] These errors were reduced by 2-3 orders of magnitude via the use of higher atomic multipoles.[6] In Amoeba force fields, multipoles are placed on each atom, allowing better capture of electrostatic potential distribution around molecules.[31, 32] Gaussian electrostatic model (GEM) is a force field based on density fitting, which can extend to arbitrary angular momentums (multipoles).[33-35] Of course there are many other proposals to model electronic polarization in the literature.[12, 36-38]

Recently, Elking *et al.* proposed to a polarizable multipole model with Gaussian charge densities.[39] A key benefit of the polarizable Gaussian Model (pGM) is its screening of all short-range electrostatic interactions in a physically consistent manner. This is critical for stable charge-fitting in polarizable force fields when the polarization of 1-2 and 1-3 charges are included and are needed to reproduce molecular anisotropy, as we discussed before.[40] An advantage of pGM is that each atom's multipoles are represented by a single Gaussian function and its derivatives with different amplitudes. Therefore, pGM is a minimalist Gaussian polarizable model. In comparison, the GEM model[33-35] treats nuclear charges explicitly and uses Hermite Gaussian auxiliary basis sets to reproduce atomic electron density, so it has the potential to represent the short-range interactions more faithfully than the pGM model. However, because the computational cost of the nonbonded electrostatic calculation scales as the squared number of functions on each atom, the multiple functions used to represent each atom in GEM can notably increase the simulation cost. The increased number of parameters associated with the functions may also pose additional challenges in parameterization. Another major difference is that GEM,[33-35] like several other efforts, such as X-Pol[41] and Amoeba,[31, 32] uses electronic densities to model molecular polarization and other effects. In comparison, our pGM model follows the Amber tradition and uses the *ab initio* electrostatic potential (ESP) to fit the parameters of atomic partial charges and dipoles. Most macromolecular simulations with long-range electrostatic interactions are performed using periodic boundary conditions. A rigorous treatment of electrostatic interactions in periodic boundary conditions requires a careful treatment of the associated lattice sums. Thus, the widely used lattice sum methods, like particle mesh Ewald (PME), need to be extended to handle multipolar related summations. Fortunately, efficient implementations of PME of dipoles and higher multipoles are already available in widely used software package such as Amber.[42, 43] This greatly simplifies the integration of pGM with PME for molecular simulations.

In the following sections we first describe the detailed pGM electrostatics scheme with a focus on how to define the atomic Gaussian multipoles and associated analytical algorithms for force computation. This is followed by algorithmic details of interfacing pGM and PME. We then present the validation of the analytical force formulation and accuracy discussion of pGM in PME simulations. Finally, we conclude the manuscript with a brief discussion of the next steps in our development.

4.2. Theory

The Gaussian multipole model represents the charge distribution on each atom as a Gaussian-shaped multipole expansion. So, an *n*th order Gaussian multipole with the radius of $1/\beta$, located at position $\vec{R}$, is[175]

$$\rho^{(n)}(\vec{r};\vec{R}) = \Theta^{(n)} \cdot \nabla_R^{(n)} (\frac{\beta}{\sqrt{\pi}})^3 exp(-\beta^2|\vec{r}-\vec{R}|^2)$$

Here $\Theta^{(n)}$ is the *n*th rank momentum tensor and $\nabla^{(n)}$ is the *n*th rank gradient operator (Appendix A.1).

In our current polarizable Gaussian multipole (pGM) model, only monopoles and dipoles are retained, so only the first two terms are needed at each atom as shown below

$$\rho^{(0)}(\vec{r};\vec{R}) = q(\frac{\beta}{\sqrt{\pi}})^3\exp(-\beta^2|\vec{r}-\vec{R}|^2)$$

$$\rho^{(1)}(\vec{r};\vec{R}) = \vec{\mu}\ \cdot \nabla_R\ (\frac{\beta}{\sqrt{\pi}})^3\exp(-\beta^2|\vec{r}-\vec{R}|^2)$$

where the zeroth-order term represents a monopole, and the first-order term represents a dipole.

Once the charge densities are defined, the pairwise Coulombic interaction energy expressions needed for the current pGM model are as follows

(1) Monopole-Monopole:

$$q_1q_2\frac{erf(\beta_{12}R_{12})}{R^{12}}$$

(2) Monopole-Dipole:

$$q_1\vec{\mu}_2\cdot\nabla_2\frac{erf(\beta_{12}R_{12})}{R^{12}}$$

(3) Dipole-Dipole:

$$(\vec{\mu}_1\cdot\nabla_1)(\vec{\mu}_2\cdot\nabla_2)\frac{erf(\beta_{12}R_{12})}{R^{12}}$$

where *erf*() is the error function and

$$\beta_{12} = \frac{\beta_1\beta_2}{\sqrt{{\beta_1}^2+{\beta_2}^2}} \text{ and } R_{12} = |\overrightarrow{R_1}-\overrightarrow{R_2}$$

Finally it is often convenient to introduce the dipole-dipole interaction tensor $\vec{\vec{T}}_{12} = \nabla_1\nabla_2\frac{erf(\beta_{12}R_{12})}{R^{12}}$ so that the interaction can be simplified as $\vec{\mu}_1\cdot\vec{\vec{T}}_{12}\cdot\vec{\mu}_2$. Here it is worth pointing out an important convention used throughout this manuscript. All gradient operators paired with a dipole only operate on coordinates. For example, the gradient operator in $\vec{\mu}_1\cdot\nabla_1$ only operates on the atomic coordinates that follow. On the other hand, all other gradient operators that are not paired with a dipole are used in the normal sense. This convention is adopted throughout this manuscript.

It can be shown that an effective potential and corresponding effective field at atomic center $\overrightarrow{R_1}$ can be defined as

$$\phi^{effective} = (q_2 + \vec{\mu}_2 \cdot \nabla_2) \frac{erf(\beta_{12} R_{12})}{R^{12}}$$

$$E^{effective} = -(q_2 + \vec{\mu}_2 \cdot \nabla_2) \nabla_1 \frac{\mathrm{erf}(\beta_{12} R_{12})}{R^{12}}$$

due to a charge distribution at atomic center $\overrightarrow{R_2}$, so that the pairwise Coulomb energies can be reproduced when an effective point charge of q_1 and an effective point dipole $\vec{\mu}_1$ are placed at atomic center $\overrightarrow{R_1}$. The use of effective potential and field simplifies the derivation of pairwise Coulombic force calculations as to be shown below. Note that these are different from the real Coulombic potential and field due to Gaussian charges and dipoles. For example, the real potential at any location $\overrightarrow{R_1}$ due to atom 2 at $\overrightarrow{R_2}$ is

$$\phi^{real} = (q_2 + \vec{\mu}_2 \cdot \nabla_2) \frac{erf(\beta_2 R_{12})}{R^{12}}$$

In our current pGM model, interactions are modeled with both permanent and induced atomic multipoles at atomic centers, both of which are truncated at the dipole level. The framework can be easily extended to higher-order multipoles, if needed in future developments.

Permanent multipoles are the first part of the pGM model and are defined with respect to a local frame overlapped with an atom's covalent bonds. This choice is based on the fact that atomic moments result from atomic covalent bonding interactions. This is also because covalent bonding interactions are along the stiffest degrees of freedom of a molecule. Thus our design follows the logic that induced moments are meant to be responsible for changes in molecular moments due to changes in soft degrees of freedom in molecular simulations. Of course, the partition between permanent and induced moments is somewhat artificial in a moment fitting procedure. Therefore we refer permanent multipoles in our pGM model as covalent multipoles in the following discussion.

The zeroth-order covalent multipoles, i.e. covalent monopoles, are simply the atomic partial charges as in other polarizable or nonpolarizable force fields. The first-order multipoles, i.e. covalent dipoles, are expressed in linear combinations of certain basis vectors. We define the basis vectors to be along the bonding directions, or more precisely, covalent interaction directions. Thus, there may be more covalent interactions than the number of bonds needed to fully define all covalent dipoles on an atom. For example, hydrogen atoms in water are with covalent dipole moments not 100% along the H-O bonds, so virtual H-H bonds may be needed to define covalent dipoles more accurately. On the other hand, sp3 carbon atoms may have up to 4 covalent dipoles due to the presence of 4 bonds, though the presence of symmetry often reduces the number of unique covalent dipoles. The new local frame originates from a physical consideration that permanent dipole moments are primarily aligned along the covalent bonds due to the differences in electronegativity of bonded atoms. An illustration of basis vectors is shown in Figure 4.1 for O and H atoms of water, and we refer these as the covalent basis vectors (CBVs). The local frame formed by CBVs on an atom is termed its CBV frame. Another representative case is the alpha carbon atom in proteins, which has four bonds, so there can be four basis vectors in its CBV frame to define the covalent dipoles.

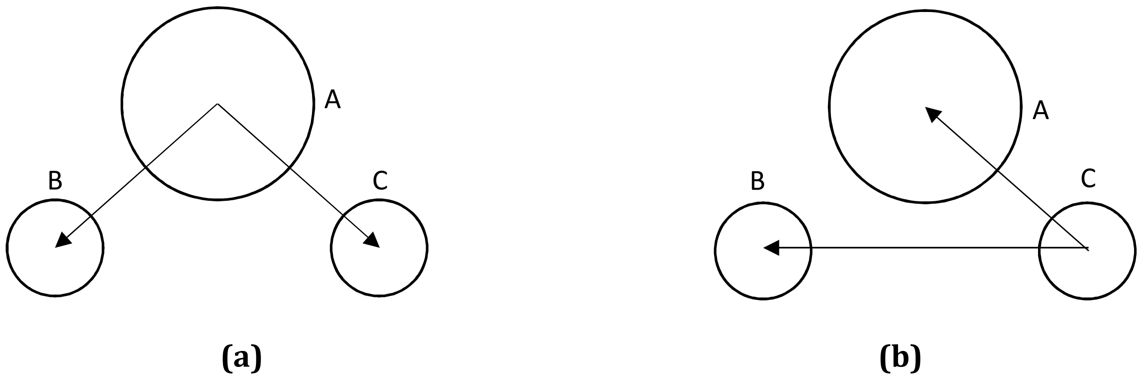


Figure 4.1. Definition of covalent basis vectors for atoms in the water molecule. (a) For covalent dipoles centered at A (oxygen), the two basis vectors are $\vec{e}^{BA}$ and $\vec{e}^{CA}$, which are defined as unit vectors along its two O-H bonds, whose two covalent dipoles are the same due to symmetry. (b) For covalent dipoles centered at C (hydrogen), the two basis vectors are $\vec{e}^{AC}$, the unit vector along the H-O bond, and $\vec{e}^{BC}$, the unit vector along the H-H virtual bond. The covalent dipoles centered on the other hydrogen atom B can be defined similarly.

The CBV frame is also chosen for the sake of simplifying force calculations because the basis vectors are directly dependent on the positions of atoms. For example, the gradient of a covalent dipole vector used extensively in force calculations can be obtained easily within the CBV frame as

$$\nabla(\vec{u}) = \nabla\left(\sum_i u_i \frac{\vec{R}_i}{|\vec{R}_i|}\right) = \sum_i u_i \left(\frac{\vec{\vec{I}}}{|\vec{R}_i|} - \frac{\vec{R}_i\vec{R}_i}{|\vec{R}_i|^3}\right)$$

where $\vec{u}$ is the permanent dipole of an atom, $\vec{R}_i$ is the vector pointing from the atom to its ith bonded atom (including virtually), $\vec{\vec{I}}$ is the identity tensor, and the summation is over all covalent interactions of the atom.

Even if quadrupoles are not used in the current pGM model, it is instructive to outline how they are defined in the CBV frame. Given the covalent basis vectors defined above, covalent basis tensors are constructed as dyadic tensors, with each of which formed as a dyadic product of two covalent basis vectors. For example in the case of oxygen atom with two covalent basis vectors ($\vec{e}^{BA}, \vec{e}^{CA}$) in Figure 1a, there are up to four dyadic tensors ($\vec{e}^{BA}\vec{e}^{BA}, \vec{e}^{CA}\vec{e}^{CA}, \vec{e}^{BA}\vec{e}^{CA}, \vec{e}^{CA}\vec{e}^{BA}$) available to define its quadrupole.

Induced multipoles are the second part of the pGM model. Only first-order terms, i.e. induced dipoles, are used. The pGM polarization scheme can naturally avoid the well-known polarization catastrophe in point polarizable models without employing any artificial screening factors,[16] because distributed dipole densities instead of point dipoles are induced at atomic centers.[176]

In the current pGM model, the linear polarization relation is retained as follows

$$\vec{p}_i = \alpha_i \vec{E}_i^{effective} = \alpha_i (\vec{E}_i^{covalent,effective} - \sum_{j \neq i} \vec{\vec{T}}_{ij} \cdot \vec{p}_j)$$

$$\vec{E}_i^{covalent,effective} = -\sum_{j \neq i} (q_j + \vec{\mu}_j \cdot \nabla_j) \nabla_i \frac{\mathrm{erf}(\beta_{ij} R_{ij})}{R_{ij}}$$

$$\vec{\vec{T}}_{ij} = \nabla_i \nabla_j \frac{\mathrm{erf}(\beta_{ij} R_{ij})}{R_{ij}}$$

where $\vec{p}_i$ is the induced dipole and α_i is the polarizability coefficient of atom i, $\vec{E}_i^{effective}$ is the total effective electric field at atom i, which contains two parts, (1) the effective field of covalent multipoles, $\vec{E}_i^{covalent,effective}$, and (2) that of induced dipoles, $-\sum_{j \neq i} \vec{\vec{T}}_{ij} \cdot \vec{p}_j$. Note that we have used the effective electric field instead of the real electric field to define the induced dipoles in the pGM model. One reason is to ensure the symmetry of $\vec{\vec{T}}_{ij}$, which greatly reduces the complexity of force calculation later.[176] To simplify the following discussion, we drop the *effective* superscript as we plan to use effective electric fields in all subsequent discussions of energy and force calculations in the pGM model.

Another issue worth pointing out about induced dipoles is their self energies. The linear polarization itself implies a self-energy term of the form

$$U = \frac{1}{2}\frac{\vec{p}^{\,2}}{\alpha}$$

The derivation can be found in many publications.[177] However, the pGM model, due to its use of Gaussian distributions of multipoles, posts extra difficulty. For example, a Gaussian charge distribution itself has self-energy, or assembly energy, of the form,

$$U = \frac{q^2\beta}{\sqrt{2\pi}} + \frac{\beta^3(\vec{\mu} + \vec{p}\,)^2}{3\sqrt{2\pi}}$$

Clearly, the self-energy is different from above and it does not lead to a linear polarization behavior. In fact, it is difficult to assess the physical meaning for the nonlinear assembly energy, just like it is hard to discuss the physical meaning of the infinitely large assembly/self energy for a point charge. Thus, for the current model development, we do not consider self-energies beyond that.

From the introduction of the pGM model in previous sections, it is clear that the electrostatic potential energy of the system can be divided into two parts:

(1) Covalent Dipole-Covalent Dipole Interaction Energy:

$$U_{covalent-covalent} = \frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{N}(q_i + \vec{\mu}_i\cdot\nabla_i)(q_j + \vec{\mu}_j\cdot\nabla_j)\frac{\mathrm{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

(2) Induced Energy:

$$U_{induced} = U_{induced-covalent} + U_{induced-induced} + U_{self}$$

$$= -\sum_{i}^{N} \vec{p}_i \cdot \vec{E}_i^{covalent} + \frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{N} (\vec{p}_i \cdot \vec{\vec{T}}_{ij} \cdot \vec{p}_j) + \frac{1}{2}\sum_{i}^{N} \vec{p}_i \cdot (\vec{E}_i^{covalent} - \sum_{j\neq i}^{N} \vec{\vec{T}}_{ij} \cdot \vec{p}_j)$$

$$= -\frac{1}{2}\sum_{i}^{N} \vec{p}_i \cdot \vec{E}_i^{covalent}$$

where N denotes the number of atoms in the system. Thus, atomic electrostatic forces can be derived as negative gradients of the above two potential energy terms, respectively.

When computing gradients for the covalent-dipole interaction energy, it is very important to know which quantities are the variables of the virtual displacement of atom *i*. There are two types of variables: (1) pairwise distances between atom *i* and all other atoms, and (2) covalent dipoles on atom *i* and covalent dipoles on atoms covalently interacting with atom *i*. Given this classification of variables, we can group the terms in Eqn. (13) into four different parts and discuss their gradients with respect to $\vec{R}_i$, separately.

The detailed derivations are presented in Appendix A.2, and the final force expression for the covalent-dipole interaction energy is

$$\vec{F}^{i}_{covalent-covalent} = -\sum_{j}^{N} \nabla_i \left(\vec{u}_j\right) \cdot \sum_{k\neq j}^{N} \nabla_j \left(q_k + \vec{u}_k \cdot \nabla_k\right) \frac{\operatorname{erf}\left(\beta_{jk} R_{jk}\right)}{R_{jk}}$$

$$-(q_i + \vec{u}_i \cdot \nabla_i) \sum_{j\neq i}^{N} \left(q_j + \vec{u}_j \cdot \nabla_j\right) \nabla_i \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right)}{R_{ij}}$$

where N denotes the number of atoms in the system. Briefly, the first term is from the derivatives over the covalent dipoles, and the second term is from derivatives over the pairwise distances.

The derivation of forces for the induced energy is not that straightforward. We first need to express the induced dipole, $\vec{p}_i$, in terms of fields from covalent dipoles only, not as their definition. This is because $\vec{p}_i$ appears on both sides, i.e. induced dipoles mutually influence each other, so it is difficult to take their derivatives. Instead we proceed by expressing $\vec{p}_i$ as shown in Appendix A.2 as

$$p_i^s = A^{-1}{}_{ij}^{st} E_j^{covalent,t}$$

where $A^{-1}{}_{ij}^{st}$ is an $3N \times 3N$ matrix which we do not know the expression of, and *s t* and *i j* are coordinate component indices and atom indices, respectively. Next it can be rewritten as

$$U = -\frac{1}{2} A^{-1}{}_{jk}^{st} E_j^{covalent,s} E_k^{covalent,t}$$

where the Einstein's index notation is employed for *j k s t*, so that a repeated index implies a summation over all possible values of the index, i.e. it is a quadruple summation. Even if we do not know the expression of matrix A^{-1}, we can still obtain its gradient with respect to $\vec{R}_i$, or the virtual displacement of atom *i*

$$\frac{\partial A^{-1}{}_{ij}^{st}}{\partial x_k^w} = -A^{-1}{}_{ii'}^{ss'} \frac{\partial T_{i'j'}^{s't'}}{\partial x_k^w} A^{-1}{}_{j'j}^{t't}$$

where all primed indices follow the Einstein's index notation, and T is the dipole-dipole interaction tensor.

Given the above preparations, the induced part of the force can be obtained as

$$\vec{F}_{induced}^i = -\vec{p}_i \cdot \nabla_i \sum_{j \neq i}^{N} (\vec{p}_j \cdot \nabla_j) \nabla_i \frac{\mathrm{erf}(\beta_{ij} R_{ij})}{R_{ij}} + \sum_{j}^{N} \nabla_i (\vec{E}_j^{covalent}) \cdot \vec{p}_j$$

where N denotes the number of atoms in the system. Details are presented in Appendix A.2. Briefly, the first term is obtained from the derivative of matrix A^{-1} and the second term is calculated as derivative of the covalent field $\vec{E}_j^{covalent}$ as follows

$$\nabla_i(\vec{E}_j^{covalent})$$

$$= \begin{cases} -\sum_{k\neq j}^{n} \nabla_i(\vec{u}_k)\cdot\nabla_k\nabla_j \frac{\operatorname{erf}(\beta_{jk}R_{jk})}{\beta_{jk}R_{jk}} - (q_i + \vec{u}_i\cdot\nabla_i)\nabla_i\nabla_j \frac{\operatorname{erf}(\beta_{ij}R_{ij})}{\beta_{ij}R_{ij}} & if \quad j \neq i \\ -\sum_{k\neq i}^{n} \nabla_i(\vec{u}_k)\cdot\nabla_k\nabla_i \frac{\operatorname{erf}(\beta_{ik}R_{ik})}{\beta_{ik}R_{ik}} - \sum_{k\neq i}^{N}(q_k + \vec{u}_k\cdot\nabla_k)\,\nabla_i\nabla_i \frac{\operatorname{erf}(\beta_{ik}R_{ik})}{\beta_{ik}R_{ik}} & if \quad j = i \end{cases}$$

where *n* denotes all atoms that are covalently (including virtually) bonded with atom *i*, and atom *i* itself.

In practice, force calculations must be combined with an Ewald summation or particle mesh Ewald (PME) technique to handle long-range under periodic boundary condition. This is to be discussed in detail later.

The Ewald summation was introduced to compute the electrostatic energy of an infinite lattice under periodic boundary conditions.[178] The basic idea is to put a mask Gaussian charge distribution on the real charge on each atom. Then a direct-space pairwise summation is conducted to compute the electric field due to real charges masked by the Gaussian charges. This step can be executed with a reasonably short cutoff distance due to the very fast decay after applying the mask Gaussian charges. Next the field generated by the mask Gaussians can be computed efficiently by a reciprocal-space summation to bring back the original electric field due to the real charges. Finally, a correction step is used to remove interactions not needed in the original electrostatic model. Similar to its use in point charge/dipole models, a mask Gaussian distribution is also used on each moment of each atom in the pGM model.

$$\rho_i^{mask}(\vec{r};\vec{R}_i) = q_i\left(\frac{{\beta_0}^2}{\pi}\right)^{\frac{3}{2}}\exp\left(-{\beta_0}^2\left|\vec{r}-\vec{R}_i\right|^2\right) + (\vec{\mu}_i\ + \vec{p}_i)\cdot\nabla_{R_i}\left(\frac{{\beta_0}^2}{\pi}\right)^{\frac{3}{2}}\exp\left(-{\beta_0}^2\left|\vec{r}-\vec{R}_i\right|^2\right)$$

where β_0 is an adjustable parameter usually in the range of about $\frac{1}{5} \sim \frac{1}{2}\,\text{Å}^{-1}$, universal for all atoms.

Given the mask distribution is also a Gaussian function, it is straightforward to compute electrostatic potential, field, and the gradient of field of a masked pGM charge distribution, as follows

$$\phi_i = \sum_{j \neq i}^{N} [\, q_j + (\vec{\mu}_j + \vec{p}_j) \cdot \nabla_j \,] \frac{\operatorname{erf}(\beta_{ij} R_{ij}) - \operatorname{erf}(\beta_0 R_{ij})}{R_{ij}}$$

$$\vec{E}_i = -\sum_{j \neq i}^{N} [\, q_j + (\vec{\mu}_j + \vec{p}_j) \cdot \nabla_j \,] \nabla_i \frac{\operatorname{erf}(\beta_{ij} R_{ij}) - \operatorname{erf}(\beta_0 R_{ij})}{R_{ij}}$$

$$\vec{\vec{E}}_i = -\sum_{j \neq i}^{N} [\, q_j + (\vec{\mu}_j + \vec{p}_j) \cdot \nabla_j \,] \nabla_i \nabla_i \frac{\operatorname{erf}(\beta_{ij} R_{ij}) - \operatorname{erf}(\beta_0 R_{ij})}{R_{ij}}$$

For the current pGM model, no higher-order field is needed. Here, *N* represents all the atoms including those in the periodic boxes, but their influence would decay to zero very quickly due to the masking effect.

Another point worth pointing out is that the real field of the mask Gaussian multipoles is used, i.e. β_0 is used instead of $\beta_{i0} = \frac{\beta_i \beta_0}{\sqrt{{\beta_i}^2 + {\beta_0}^2}}$ in the above expressions. The mixed use is not an issue because mask multipoles do not really exist, their role is just a mathematical treatment in the Ewald summation as long as the effect is exactly cancelled out in the later step.

The reciprocal summation of the pGM model follows the same procedure as a traditional point polarizable model.[179] Thus the electrostatic potential, field, and gradient of field can be shown as

$$\phi_i = \frac{1}{\pi V}\sum_{\vec{m}\neq 0}\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{{\beta_0}^2}\right)}{\vec{m}^2}\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)S(\vec{m})$$

$$\vec{E}_i = \frac{2i}{V}\sum_{\vec{m}\neq 0}\vec{m}\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{{\beta_0}^2}\right)}{\vec{m}^2}\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)S(\vec{m})$$

$$\vec{\vec{E}}_i = -\frac{4\pi}{V}\sum_{\vec{m}\neq 0}\vec{m}\vec{m}\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{{\beta_0}^2}\right)}{\vec{m}^2}\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)S(\vec{m})$$

where the *i*'s that are not subscripts but the imaginary unit, V is the volume of the unit cell, and $\vec{m}$ is the reciprocal space vector. $S(\vec{m})$ is the structure factor

$$S(\vec{m}) = \sum_{j=1}^{N}\tilde{L}_j(\vec{m})\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)$$

$$\tilde{L}_j(\vec{m}) = q_j + 2\pi i(\vec{\mu}_j + \vec{p}_j)\cdot\vec{m}$$

Here *N* is the number of atoms in the primary simulation box only.

The PME correction term is used to handle various specific situations in a force field. For example, most force fields have masked bonded (1-2 and 1-3) atom pairs, which result in no electrostatic interactions between these pairwise atoms. Thus, the interactions among these masked pairs must be removed. However, in the current pGM model we do not have any masked pairs, so there is no need for such correction.

Another correction that needs paying attention to is the self-interaction correction. This is the only correction in the pGM model. The self-potential, self-field, gradient of self-field can be shown as[179-180]

$$\phi_i = \frac{2q_i\beta_0}{\sqrt{\pi}}$$

$$\vec{E}_i = -\frac{4(\vec{\mu}_i + \vec{p}_i){\beta_0}^3}{3\sqrt{\pi}}$$

$$\vec{\vec{E}}_i = \frac{4q_i{\beta_0}^3}{3\sqrt{\pi}}\vec{\vec{I}}$$

These terms need to be properly subtracted to obtain correct potential, field, and field gradient, respectively

In summary, the similarity between the Ewald summation in pGM model and that in the polarizable point charge/dipole model shows that the PME MD engine for the point charge/dipole model can be easily transplanted over for pGM applications with little revision. There are excellent literature discussing the details of PME for polarizable point dipole models, and can be safely omitted in this work.[179-180]

As pointed out before, analytical force expressions cannot be used directly in typical MD simulations since all summations are over infinite numbers of atoms with periodic boundary conditions. They must be combined with an Ewald summation or a PME technique to facilitate solvated-phase simulations. To bypass the infinite summations, the force expressions are reformulated in terms of fields and its derivatives, which are also the quantities that an Ewald or PME procedure would return. The key to express energy and force with fields and gradients of fields is to consider the following quantities together

$$\vec{E}_i^{covalent} = -\sum_{j\neq i}^{N}(\, q_j + \vec{\mu}_j \cdot \nabla_j\,)\nabla_i \frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

$$\vec{\vec{E}}_i^{covalent} = -\sum_{j\neq i}^{N}(\, q_j + \vec{\mu}_j \cdot \nabla_j\,)\nabla_i\nabla_i \frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

$$\vec{E}_i^{induced} = -\sum_{j\neq i}^{N} \vec{p}_j \cdot \nabla_j \nabla_i \frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

$$\vec{\vec{E}}_i^{induced} = -\sum_{j\neq i}^{N} \vec{p}_j \cdot \nabla_j \nabla_i \nabla_i \frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

where N denotes all atoms in the system, including those of the periodic boxes. Thus, these are all infinite summations.

A key step is in the computation of $\vec{F}^{i}_{induced}$, where term $\sum_j^N \nabla_i(\vec{E}_j^{covalent}) \cdot \vec{p}_j$ also has to be reformulated accordingly. Given that $\nabla_i(\vec{E}_j^{covalent})$ lists two separate terms for $j = i$ and $j \neq i$, we can rewrite $\sum_j^N \nabla_i(\vec{E}_j^{covalent}) \cdot \vec{p}_j$ as follows

$$\sum_{j}^{N} \nabla_i(\vec{E}_j^{covalent}) \cdot \vec{p}_j$$

$$= \sum_{j\neq i}^{N} \sum_{k\neq j}^{n} (-\nabla_i(\vec{u}_k)) \cdot \nabla_k(\vec{p}_j \cdot \nabla_j) \frac{\operatorname{erf}(\beta_{jk}R_{jk})}{\beta_{jk}R_{jk}}$$

$$- \sum_{j\neq i}^{N} (q_i + \vec{\mu}_i \cdot \nabla_i) \nabla_i(\vec{p}_j \cdot \nabla_j) \frac{\operatorname{erf}(\beta_{ij}R_{ij})}{\beta_{ij}R_{ij}} - \sum_{k\neq i}^{n} \nabla_i(\vec{u}_k) \cdot \nabla_k(\vec{p}_i \cdot \nabla_i) \frac{\operatorname{erf}(\beta_{ik}R_{ik})}{\beta_{ik}R_{ik}}$$

$$- \sum_{k\neq i}^{N} (q_k + \vec{\mu}_k \cdot \nabla_k) \nabla_i(\vec{p}_i \cdot \nabla_i) \frac{\operatorname{erf}(\beta_{ik}R_{ik})}{\beta_{ik}R_{ik}}$$

where both $j = i$ and $j \neq i$ terms are needed due to the outermost summation over j. Combining the first and third terms gives

$$= \sum_{j}^{N} \sum_{k\neq j}^{n} (-\nabla_i(\vec{u}_k)) \cdot \nabla_k(\vec{p}_j \cdot \nabla_j) \frac{\operatorname{erf}(\beta_{jk}R_{jk})}{\beta_{jk}R_{jk}} - \sum_{j\neq i}^{N} (q_i + \vec{\mu}_i \cdot \nabla_i) \nabla_i(\vec{p}_j \cdot \nabla_j) \frac{\operatorname{erf}(\beta_{ij}R_{ij})}{\beta_{ij}R_{ij}}$$

$$- \sum_{k\neq i}^{N} (q_k + \vec{\mu}_k \cdot \nabla_k) \nabla_i(\vec{p}_i \cdot \nabla_i) \frac{\operatorname{erf}(\beta_{ik}R_{ik})}{\beta_{ik}R_{ik}}$$

Exchanging the summation order for the first term leads to

$$= \sum_{k}^{n} \sum_{j\neq k}^{N} (-\nabla_i(\vec{u}_k)) \cdot \nabla_k(\vec{p}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta_{jk}R_{jk})}{\beta_{jk}R_{jk}} - \sum_{j\neq i}^{N} (q_i + \vec{\mu}_i \cdot \nabla_i)\nabla_i(\vec{p}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta_{ij}R_{ij})}{\beta_{ij}R_{ij}}$$

$$- \sum_{k\neq i}^{N} (q_k + \vec{\mu}_k \cdot \nabla_k)\,\nabla_i(\vec{p}_i \cdot \nabla_i)\,\frac{\mathrm{erf}(\beta_{ik}R_{ik})}{\beta_{ik}R_{ik}}$$

Substitution of the expressions of electric fields and derivatives gives

$$\sum_{j}^{N} \nabla_i(\vec{E}_j^{covalent}) \cdot \vec{p}_j = \sum_{k}^{n} \nabla_i\,(\vec{u}_k) \cdot \vec{E}_k^{induced} + q_i\vec{E}_i^{induced} + \vec{u}_i \cdot \vec{\vec{E}}_i^{induced} + \vec{p}_i \cdot \vec{\vec{E}}_i^{covalent}$$

Here, *n* means those atoms that covalently interact with atom *i*.

Given the above preparations, force can finally be expressed as follows after substitution

$$\vec{F}^{i}_{covalent-covalent} = \sum_{j}^{n} \nabla_i\,(\vec{u}_j) \cdot \vec{E}_j^{covalent} + q_i\vec{E}_i^{covalent} + \vec{u}_i \cdot \vec{\vec{E}}_i^{covalent}$$

$$\vec{F}^{i}_{induced} = \sum_{j}^{n} \nabla_i\,(\vec{u}_j) \cdot \vec{E}_j^{induced} + q_i\vec{E}_i^{induced} + \vec{p}_i \cdot \vec{\vec{E}}_i^{induced} + \vec{u}_i \cdot \vec{\vec{E}}_i^{induced} + \vec{p}_i \cdot \vec{\vec{E}}_i^{covalent}$$

Adding these two terms together, the final force expression is,

$$\vec{F}^{i} = \sum_{j}^{n} \nabla_i\,(\vec{u}_j) \cdot \vec{E}_j + q_i\vec{E}_i + (\vec{u}_i + \vec{p}_i) \cdot \vec{\vec{E}}_i$$

This shows that a key step in this algorithm is to accumulate atomic electric potential and its first and second derivatives from various components, including both reciprocal and direct summations.

4.3. Results and Discussion

To validate the pGM force expressionsabove, we constructed a small toy system of two water molecules in free space. The detailed water pGM parameters are listed in Table 4.1. These parameters were derived with an iterative RESP procedure for the pGM model with quantum mechanical ESP data from a B3LYP/aug-cc-pvtz calculation of the water dimer.

Table 4.1. Two water molecules in free space. The permanent dipole moments are expressed in the lab frame. The moments in the CBV frame can be obtained once the CBV's are defined according to Figure 1 via a least square fitting procedure.

Tested atoms	water-1 O	water-1 H1	water-1 H2	water-2 O	water-2 H1	water-2 H2
Charge (e)	-1.797045	0.898523	0.898523	-1.797045	0.898523	0.898523
Dipole moment ($e \cdot$ Å)	(1.317332, -0.120867, 1.711988)	(0.262849, -0.484213, 0.323876)	(0.276182, 0.436999, 0.376728)	(0.927376, 0.134058, -1.967337)	(-0.236591, 0.041394, -0.650790)	(0.590269, 0.013966, -0.164059)

Polarizability (Å^3)	1.448980	0.427350	0.427350	1.448980	0.427350	0.427350
Gaussian radius (Å)	0.8066249	0.7147597	0.7147597	0.8066249	0.7147597	0.7147597
Coordinates (Å)	(-1.387669, -0.006775, 0.110728)	(-1.734110, 0.790147, -0.310036)	(-1.756164, -0.740122, -0.397708)	(1.514536, 0.007522, -0.121554)	(1.919419, -0.047517, 0.751251)	(0.555921, -0.008485, 0.043101)

Two methods were used to calculate the atomic forces. The first method is to use the force expressions to calculate forces analytically. The second method is to calculate forces numerically via the finite-difference method, based on the fact that each force is the negative gradient of potential energy. Here the potential energy was computed analytically. The finite-difference coordinate displacement was set to be 1×10^{-6}Å, and the induced dipole accuracy was set to 1×10^{-9}. It is clear that the differences between the two sets of atomic forces appear only on the nineth digit after the decimal point.

There is also an indirect way to confirm the correctness of the force expression, which is to utilize the fact that the total force of the system should always be zero in any direction. If we add up all atomic forces, the system net force (in $e^2/Å^2$) is 1×10^{-9}, 2×10^{-17} and 3×10^{-17}, for x, y and z directions, respectively. The overall error here is consistent with the induction tolerance used in the testing, 1×10^{-9}.

To achieve aqueous-phase simulations, an Ewald summation or PME technique is essential for any electrostatic model. Although there are various publications discussing the accuracy of PME [179-182], we have to acknowledge the fact that the pGM model has a higher accuracy requirement than classical point-charge force fields due to the presence of dipoles. In general, higher moments would require higher PME accuracy. This can be appreciated from the perspective of two considerations. Firstly, the interaction energy between two dipoles decays faster with distance ($1/r^3$) than that between two charges ($1/r$). Secondly, the second derivatives of potential are needed to compute forces on dipoles, whereas only first derivatives, e.g. electric fields, are needed to compute forces on charges. Due to these differences, we have to carefully examine the accuracy requirement of PME methods used in our model.

To test the accuracy, we only look at the most difficult pairwise interactions, so that the errors reported below are the maximum errors in the tested water system. For the reciprocal part, we focus on the electrostatic field between the bonded O atom and H atoms, whose interactions are the strongest and thus the most difficult in PME. For the direct summation part, we focus on the electrostatic field between two H atoms. Because they have the smallest Gaussian radii, their interactions converge slowest in the direct summation. Thus, to guarantee a given accuracy level for forces in the pGM model, we need to consider both PME components.

In the following analysis we set the grid spacing to 1 Å for PME as in most biomolecular simulations and varied other parameters to see how accuracy changes in both the reciprocal and the direct summation components. Also, because our model contains both charges and dipoles, we analyzed their field separately to assess the impact of different setups on their accuracy of electric field. The test results are shown below in Table 4.3, 4.4 and 4.5.

Table 4.2. Errors of reciprocal potentials and derivatives generated by the dipoles of water-1 H1 atom on water-1 O atom at different PME setups. The analytical values were calculated with MATLAB. Interpolation order refers to the rank of the B-spline interpolation method used in PME. See Ref[182] for details.

Interpolation order		5	6	7	8	9
Ewald coefficient $\beta_0 = 0.3\text{Å}^{-1}$	potential	8.2×10^{-5}	2.8×10^{-5}	1.5×10^{-7}	3.4×10^{-6}	2.7×10^{-6}
	first derivative	3.0×10^{-4}	6.4×10^{-5}	9.2×10^{-6}	7.6×10^{-6}	3.9×10^{-6}
	second derivative	1.3×10^{-3}	3.9×10^{-4}	4.6×10^{-5}	1.6×10^{-5}	5.7×10^{-6}
Ewald coefficient $\beta_0 = 0.4\text{Å}^{-1}$	potential	1.2×10^{-4}	4.0×10^{-5}	4.3×10^{-6}	3.5×10^{-8}	5.4×10^{-6}
	first derivative	1.3×10^{-3}	4.6×10^{-4}	1.5×10^{-4}	6.5×10^{-5}	2.5×10^{-5}

	second derivative	2.6×10^{-3}	1.6×10^{-3}	4.0×10^{-4}	1.9×10^{-4}	8.1×10^{-5}
Ewald coefficient $\beta_0 = 0.5Å^{-1}$	potential	2.6×10^{-4}	1.1×10^{-4}	1.7×10^{-4}	6.1×10^{-5}	9.1×10^{-5}
	first derivative	5.1×10^{-3}	2.5×10^{-3}	1.2×10^{-3}	7.2×10^{-4}	4.0×10^{-4}
	second derivative	8.8×10^{-3}	5.5×10^{-3}	2.4×10^{-3}	1.3×10^{-3}	8.8×10^{-4}

Table 4.3. Errors of reciprocal potentials and derivatives generated by the charges of water-1 H1 atom on water-1 O atom at different PME setups. The analytical values were calculated with MATLAB. Interpolation order refers to the rank of the B-spline interpolation method used in PME. See Ref[182] for details.

Interpolation order		5	6	7	8	9
Ewald coefficient $\beta_0 = 0.3Å^{-1}$	potential	1.4×10^{-6}	3.7×10^{-8}	6.5×10^{-8}	1.4×10^{-8}	5.3×10^{-9}
	first derivative	1.8×10^{-4}	3.8×10^{-5}	5.5×10^{-6}	1.1×10^{-6}	3.4×10^{-7}
	second derivative	7.7×10^{-4}	2.1×10^{-4}	3.0×10^{-5}	7.8×10^{-6}	1.8×10^{-6}
	potential	1.4×10^{-5}	3.7×10^{-6}	1.6×10^{-6}	6.8×10^{-7}	3.2×10^{-7}

Ewald coefficient $\beta_0 = 0.4\text{Å}^{-1}$	first derivative	7.4×10^{-4}	1.1×10^{-4}	6.1×10^{-5}	2.1×10^{-5}	1.0×10^{-5}
	second derivative	2.5×10^{-3}	7.7×10^{-4}	2.1×10^{-4}	7.5×10^{-5}	3.0×10^{-5}
Ewald coefficient $\beta_0 = 0.5\text{Å}^{-1}$	potential	8.1×10^{-5}	3.9×10^{-5}	1.9×10^{-5}	1.2×10^{-5}	7.0×10^{-6}
	first derivative	2.5×10^{-3}	7.8×10^{-4}	4.5×10^{-4}	2.6×10^{-4}	1.5×10^{-4}
	second derivative	6.7×10^{-3}	2.6×10^{-3}	1.1×10^{-3}	6.3×10^{-4}	3.4×10^{-4}

Table 4.4. Errors of direct summation potentials for Gaussian potentials between two H atoms at different PME setups. These values are the difference between two error functions, $\mathrm{erf}(\beta R_c) - \mathrm{erf}(\beta_0 R_c)$. Here, R_c is the direct summation cutoff distance, and $\beta = 1/(0.7147597 \cdot \sqrt{2})\text{Å}^{-1}$ for the H atom pairs.

Cutoff distance (Å)	7	8	9	10	11
$\beta_0 = 0.3\text{Å}^{-1}$	3.0×10^{-3}	6.9×10^{-4}	1.3×10^{-4}	2.2×10^{-5}	3.1×10^{-6}
$\beta_0 = 0.4\text{Å}^{-1}$	7.5×10^{-5}	6.0×10^{-6}	3.6×10^{-7}	1.5×10^{-8}	4.9×10^{-10}
$\beta_0 = 0.5\text{Å}^{-1}$	7.4×10^{-7}	1.5×10^{-8}	2.0×10^{-10}	1.5×10^{-12}	7.3×10^{-15}

It is clear from the above analyses that the pGM model demands a higher accuracy level than classical point-charge models. This is as expected for any electrostatic model with dipoles or higher moments. For the reciprocal part, the field generated by dipoles are more difficult to handle than that of charges in PME. Comparing Tables 4.3 and 4.4, we can see that the errors of dipole fields are about twice larger than that of charge fields. Furthermore, Table 4.3 and 4.4 show that the second derivatives are the most difficult in PME. Thus, to ensure the accuracy of the reciprocal summation of PME, we need to make sure the second derivatives reach a specified accuracy level. For example, if we use 5×10^{-5} as the accuracy threshold, which is a common choice, we have to set Ewald $\beta_0 = 0.3\text{Å}^{-1}$ and the interpolation order 7 or higher in the PME setup (Table 4.3). Of course, use of smaller grid spacing would increase the accuracy but with a higher overhead in reciprocal summation. We will explore the best tradeoff in accuracy and efficiency for realistic biomolecular systems in a later publication. Situations are similar for the direct summation part. To reach a common accuracy threshold of 5×10^{-5}, if we set $\beta_0 = 0.3\text{Å}^{-1}$ as in the reciprocal part, the direct space cutoff should be set to a relatively longer cutoff distance of 10 Å, as shown in Table 4.5. Of course, a choice of larger β_0 (i.e. 0.35Å^{-1}) would allow a commonly used cutoff distance of 9 Å. However, this would require a higher interpolation order to achieve the similar level of accuracy.

Given all the accuracy consideration above, we performed a pure water simulation to test the energy conservation behavior in an NVE run with the PME treatment. The electrostatic parameters were derived from those in Table 1 and transplanted onto the TIP3P water model. We used 512 water molecules in a truncated octahedron box of 27.5 Å. The dimension of the particle mesh grid is 30^3, so the grid spacing is a bit less than 1 Å. The PME $\beta_0 =$

0.35Å^{-1}, the real space cutoff was set as 9 Å, and the interpolation order was 8 so that the overall PME error was less than 5×10^{-5}.

We first tested a range of induction tolerance criteria, ranging from 10^{-3} to 10^{-6}. Our experiments show that 10^{-3} and 10^{-4} are clearly not sufficient for the induction iteration, leading to decreasing energy throughout the MD simulations. This is consistent with previous findings in the developments of polarizable point dipole models.[180] Specifically, the energy in the NVE run of 10^{-3} drifts too fast, so that it is already out of the plotting range at the 100th step, the very first data point. The rest of the energy plots over the simulation time are shown below in Figure 4.2. The initial testing shows that the energy convergence became much better after we tighten the iteration tolerance to 10^{-5}. Though the total energy still drifts down a little, but much slower. Finally, after we tighten it to 10^{-6}, the total energy is basically conserved. Of course, the total energy fluctuation does exist.

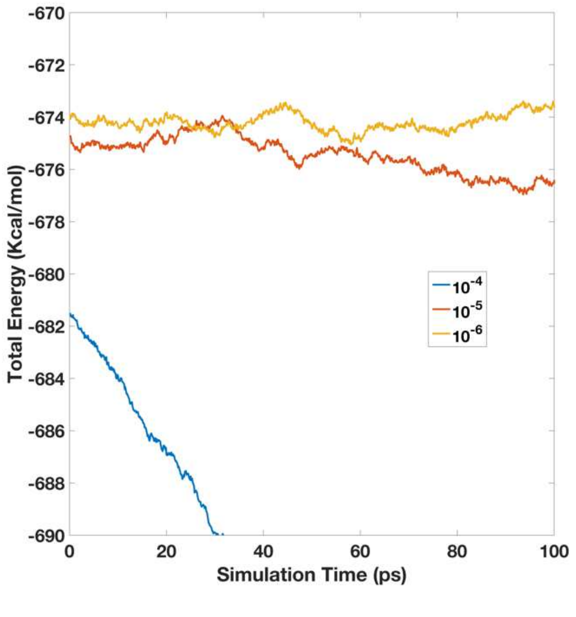


Figure 4.2. Total energy versus simulation time for the 512-water box simulation. Here, all the simulations are performed with a 5×10^{-5} PME accuracy, but with different induction iteration tolerance ($1 \times 10^{-4} \sim 1 \times 10^{-6}$).

Next we also studied the influences of the PME setup on the energy conservation. To compare with the NVE run with the high PME accuracy above. We collected a comparable NVE run with the same induction tolerance of 10^{-6}, but with a somewhat lower PME setting. The real space cutoff was set as 8 Å and the interpolation order was set as 6 but others remained to be the same, which leads to a lower PME accuracy of $\sim 5 \times 10^{-4}$. The total energy is more positive because there are fewer van der Waals pairs. As shown in Figure 4.3, the total energy also drifts noticeably, though it becomes more positive over time. In summary, our experiment shows that high enough accuracy in both the PME calculation and the induction iteration is necessary for a polarizable dipole model with permanent dipoles to achieve energy conservation. Furthermore, we expect even higher accuracy is necessary if higher moments, i.e. quadrupoles, are used in future pGM developments as higher derivatives are needed from the PME calculation.

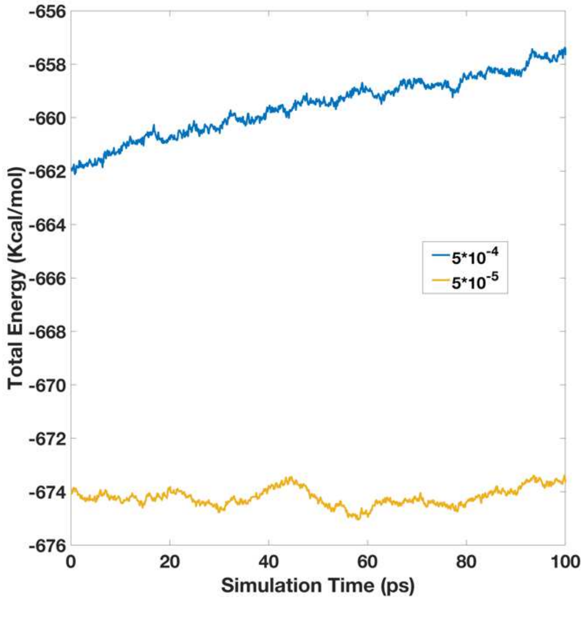


Figure 4.3. Total energy versus simulation time for the 512-water box simulation. Here, both simulations were performed with a 1×10^{-6} induction iteration tolerance, but with different PME accuracy, low for 5×10^{-4} and high for 5×10^{-5}.

4.4. Conclusion

In this work, we proposed an efficient formulation for the polarizable Gaussian Multipole (pGM) model for biomolecular simulations. Firstly, a local frame based on the covalent basis vectors (CBV) /tensors was used to set up the permanent (covalent) multipoles on all atoms. The CBV frame nicely allows the intrinsic molecular flexibility during simulations and facilitates an efficient expression of the electrostatic forces in the closed form. Based on the new CBV local frame, we then derived the analytical force expressions for the pGM model. Finally, we outlined how to interface the pGM electrostatics seamlessly with the PME implementation for molecular simulations under the periodic boundary conditions.

To validate the analytical force expression for the pGM model defined on the CBV frame, we studied the accuracy of the analytical atomic forces with a finite-different force analysis for a water dimer. The analysis shows a very good consistency between the analytical and numerical forces, with an error comparable to the finite difference uncertainty. In addition, total analytical and numerical forces of the water dimer are very close to zero with an error consistent with the induction iteration tolerance.

Next, we analyzed the PME setups necessary for accurate pGM energy and force calculations. It was found that the pGM model requires higher accuracy than the classical point-charge models due to the presence of dipoles. This is because the electrostatic field generated by dipoles are much more difficult to interpolate than that of charges in PME, and the error of the dipole field is about twice that of the charge field. In addition, the second derivative of potential is needed, which is the more difficult to compute accurately in PME to ensure accurate pGM forces.

To validate the overall electrostatic framework for the reformulated pGM model, we conducted an NVE simulation for a small water box of 512 water molecules. Our results show that, to achieve energy conservation, it is important to ensure enough accuracy on both PME and induced dipoles. With a 5×10^{-5} accuracy on PME and a 1×10^{-6} tolerance for induced dipoles, the tested NVE water simulation in the pGM model was shown to conserve energy reasonably well. Future development will be necessary to improve the efficiency of the pGM model in both PME setup and induction iteration to bring out the potential of the pGM model.

Chapter 5

5.1. Introduction

Atomistic simulations of biomolecules have been applied to various biological systems.[2] While additive nonpolarizable models continue to play dominant roles,[3-5] nonadditive polarizable models are emerging as a tool extending our capability of studying more complex biomolecular systems and processes. Nonpolarizable models typically use fixed atomic partial charges to model electrostatics. As a result, they only incorporate polarization response to the environment (mostly in water) in a mean-field manner. Thus, nonpolarizable models with excellent descriptions of the homogeneous bulk phase can be poor models treating gas-phase clusters or in cases where large scale conformational changes are accompanied by changes in dielectric environment.[6] Another limitation of the nonpolarizable models is the use of partial atomic charges. This treatment often lacks sufficient mathematical flexibility to accurately describe the electrostatic potential nearby molecules. Williams showed that optimal least-squares fitting of atom-centered partial charges resulted in relative root-mean-square errors of 3-10% in reproducing quantum electrostatic potentials at grid points in a shell outside the surface of a series of small polar molecules.[7] In summary, the importance of modeling polarization effects and the limitation of point charge models have been well documented.[6, 8]

To address the above issues, extensive efforts have been invested in the development of explicit polarizable models.[9-11] Several methods have been proposed including explicit polarization in the molecular mechanics potentials, such as the Drude oscillator,[12-13] fluctuating charges,[14] and induced dipoles.[6, 15-16] Among these new models, implementation of polarizable point dipoles was originally proposed by Applequist[17]. It is a classical approach with a long history in molecular simulation.[18] However, this model suffers from the so-called "polarization catastrophe",[19] which requires proper screenings for short range electrostatics.

Recently, Elking *et al.* proposed the polarizable multipole model with Gaussian charge densities.[20] A key benefit of the polarizable Gaussian Multipole model (pGM) is its screening of all short-range electrostatic interactions in a physically consistent manner, as we discussed previously.[21] We have presented an analytical formulation for the pGM model that is more suitable for molecular dynamics simulation due to the lack of explicit torque computation.[1] In addition, we further exploited a new local frame, the covalent basis vectors (CBVs) to preserve the physical symmetry and to better accommodate the fact that permanent dipoles are primarily resulted from covalent bonds. Subsequent numerical tests show that, with enough accuracy involving both pGM and Particle Mesh Ewald (PME), NVE simulation can be realized with a similar energy drift as in classic point charge models.[1]

However, most of the real-world biological systems resemble constant pressure ensembles, which requires our pGM model to be extended to NPT simulations. Andersen first proposed a new constant pressure method in which the volume was allowed to fluctuate, with its average value being determined by the balance between the internal stress in a system and the externally set pressure.[22] Following his pioneer works, Nose´ and Klein[23] and Darden et al.[24] indicated how to extend the formulation to include the case of the long range charge-charge interactions. Toukmaji et al. introduced an efficient approach to include induced dipolar interactions,[25] and Sagui et al. extended the formulation to multipolar situations.[26] Recently, these virials for point induced dipoles and multipoles have been implemented for

AMOEBA in Tinker-HP and Tinker9 (GPU).[27] Additionally, there have been interesting discussions about pressure calculation related to Ewald surface correction.[28-29] However, how to rigorously treat internal stress in a system with both permanent and induced dipoles is still unclear.

In this work, we present an analytical formulation of stress tensor for both flexible and rigid molecular systems modeled with the pGM model. We performed a finite difference test and confirmed that both are rigorously correct. Finally, we implemented the proposed pGM formalism in the context of PME into the SANDER program in Amber. Our NPT simulations of a 512-water system showed the algorithm is stable as indicated by the time evolutions of pressure, temperature, density, and total system energy.

5.2. Method

The total electrostatic energy of the system, as shown in our previous work[183], is defined as:

$$U_{pGM} = \sum_{i}^{N} \frac{1}{2}(q_i + \vec{\mu}_i \cdot \nabla_i)\phi_i^0 + \sum_{i}^{N} \frac{1}{2}(\vec{p}_i \cdot \nabla_i)\phi_i^0$$

Here, q_i, $\vec{\mu}_i$, and $\vec{p}_i$ are the charge, covalent dipole (i.e. permanent dipole) and induced dipole on atom *i*, respectively, and ϕ_i^0 is the electrostatic potential on atom *i* that is generated only by charges and covalent dipoles in the system. Later in this paper, we will denote ϕ_i, as the electrostatic potential on atom *i* that is generated by charges, covalent dipoles and *induced dipoles* in the system.

Due to the periodic boundary condition, ϕ_i^0 has to be obtained through certain lattice summation technique, and here the Ewald summation technique is used.

According to the Ewald summation,

$$\phi_i^0 = \frac{1}{\pi V}\sum_{\vec{m}\neq 0}\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}\left[\left(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j\right)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\}$$

$$+\sum_{j\neq i}^{\infty}(q_j + \vec{\mu}_j\cdot\nabla_j)\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right) - \mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}} - \lim_{\vec{R}_j\to\vec{R}_i}(q_i + \vec{\mu}_i\cdot\nabla_j)\frac{\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}$$

Here, V is the volume of the box, and $\vec{m}$ is the reciprocal lattice vector, and $\vec{R}_i$ and $R_{ij} = \left|\vec{R}_j - \vec{R}_i\right|$ are the coordinate of atom *i* and its distance to atom *j*, respectively. In addition, β_0 is the Ewald coefficient, whose value is usually between $\frac{1}{5}\sim\frac{1}{2}\,\text{Å}^{-1}$ for typical biomolecular applications, and β_{ij} equals to $\frac{\beta_i\beta_j}{\sqrt{{\beta_i}^2+{\beta_j}^2}}$, where β_i and β_j are the Gaussian parameters for atoms *i* and *j*, respectively.

Below we will use the following notation convention: all gradients paired with a dipole (covalent or induced) only apply to atom coordinates. For example, the gradient operator in $\vec{\mu}_i\cdot\nabla_i$ only acts on the coordinates of atom *i*. On the other hand, all other gradient operators (not paired with a dipole) follow the standard convention.

Stress Tensor

From its definition, stress tensor (virial) is the derivative of system Lagrangian with respect to the system size parameters, except for a constant pre-factor. In a typical biomolecular simulation, the simulating box can be described by three column vectors that represent the edges of the box, $\vec{u}_1$, $\vec{u}_2$ and $\vec{u}_3$. Thus, a matrix *h* can be constructed from these three unit-cell vectors, defined as the system shape parameters,

$$h = (\vec{u}_1, \vec{u}_2, \vec{u}_3)$$

Thus, stress tensor $\vec{\vec{V}}$ can be expressed as:

$$\vec{\vec{V}} \cdot (h^{-1})^T = \frac{\partial L}{\partial h} = \frac{\partial T}{\partial h} - \frac{\partial U}{\partial h}$$

Here, L represents the system Lagrangian, superscript "T" denotes transpose of the matrix, T and U are kinetic and potential energies, respectively. The derivative with respect to matrix h is equivalent to differentiating L against each component of h, with the fractional coordinates of each atom held constant.

Because the kinetic energy expression is the same for a pGM system and a classic point charge system, the related virial terms are the same, so we omit their derivations here. There are many different components in the potential energy, but only the electrostatic part is different from other force fields in the current pGM model. Thus, we only need to derive the electrostatic components of the virial. This is to say, we are deriving the following quantity in this work,

$$\vec{\vec{V}}_{pGM} = -\frac{\partial U_{pGM}}{\partial h} \cdot (h)^T$$

To do so, we first need to identify which quantities in energy are dependent on matrix h.

Stress Tensor in a pGM System

Quantities that obviously depend on h are as follows:

1. Atomic coordinates, $\vec{R}_i = s_{i1}\vec{u}_1 + s_{i2}\vec{u}_2 + s_{i3}\vec{u}_3$. Here, s's are fractional coordinates and are constants when computing the derivatives with respect to h.

2. Atomic covalent dipoles, $\vec{\mu}_i = \sum_k^n u_{ik} \frac{\vec{R}_{ik}}{|\vec{R}_{ik}|} = \sum_k^n u_{ik} \frac{\vec{R}_k - \vec{R}_i}{|\vec{R}_{ik}|}$. Here, n refers the number of atoms covalently connected to atom i (this notation will be used thoroughly in the paper) and u_{ik} is the dipole moment along the specific direction $\vec{R}_k - \vec{R}_i$, which is a constant. Noted that the definition of $\vec{R}_{ik} \equiv \vec{R}_k - \vec{R}_i$ is used, throughout this manuscript.
3. System volume, $V = \det(h)$.
4. Reciprocal lattice vector, $\vec{m} = (k_1, k_2, k_3) \cdot h^{-1}$, where k_1, k_2, k_3 are integer constants.

The derivatives of these quantities with respect to h are given in **Appendix** B.1.

Moreover, the induced dipole, $\vec{p}_i$, is clearly dependent on h. However, there is no explicit expression of $\vec{p}_i$, as $\vec{p}_i$ is computed through a self-consistent iteration. In the following, we show how to obtain its derivatives.

As shown in our previous work,[183] the induced dipole can be expressed in the periodic system as

$$\vec{p}_i = \alpha_i\left(\vec{E}_i^0 + \vec{E}_i^{induced}\right) = \alpha_i(\vec{E}_i^0 - \sum_j^N \vec{\vec{T}}_{ij} \cdot \vec{p}_j)$$

$$\vec{E}_i^0 = -\nabla_i \phi_i^0$$

$$T_{ij}^{\alpha\beta} =$$

$$\partial_i^\alpha \partial_j^\beta \begin{cases} \frac{1}{\pi V} \sum_{\vec{m} \neq 0} \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \exp\left(2\pi i \vec{m} \cdot \vec{R}_j\right) \exp\left(-2\pi i \vec{m} \cdot \vec{R}_i\right) + \frac{\mathrm{erf}(\beta_{ij} R_{ij}) - \mathrm{erf}(\beta_0 R_{ij})}{R_{ij}} & if\ i \neq j \\ \frac{1}{\pi V} \sum_{\vec{m} \neq 0} \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \exp\left(2\pi i \vec{m} \cdot \vec{R}_j\right) \exp\left(-2\pi i \vec{m} \cdot \vec{R}_i\right) - \lim_{\vec{R}_j \to \vec{R}_i} \frac{\mathrm{erf}(\beta_0 R_{ij})}{R_{ij}} & if\ i = j \end{cases}$$

Here $\alpha, \beta = 1, 2, 3$ refer to x, y, z directions, respectively. The expression of $T_{ij}^{\alpha\beta}$ comes from Ewald summation of $\partial_i^\alpha \partial_j^\beta \frac{\mathrm{erf}(\beta_{ij}R_{ij})}{R_{ij}}$. Thus the summation for $i \neq j$ includes all images in the periodic system. Introduction of

$$A_{ij}^{\alpha\beta} = \frac{1}{\alpha_i}(\delta_{ij}^{\alpha\beta} + \alpha_i \cdot T_{ij}^{\alpha\beta})$$

leads to the following short-hand expression for the induced dipole[183]

$$\vec{p}_i = \sum_j^N \vec{\vec{A}}_{ij}^{-1} \cdot \vec{E}_j^0$$

In above we have transferred the dependence of $\vec{p}_i$ on h to the dependence of $\vec{\vec{A}}_{ij}^{-1}$ and $\vec{E}_j^0$ on h, and the dependence of $\vec{E}_j^0 = -\nabla_j \phi_j^0$ is already known. Thus, what is needed is to derive the dependence of $\vec{\vec{A}}_{ij}^{-1}$ on h.

Given $A \cdot A^{-1} = 1$, we have

$$\frac{\partial (A^{-1})_{ij}^{\alpha\beta}}{\partial h} = -(A^{-1})_{ii'}^{\alpha\alpha'} \frac{\partial A_{i'j'}^{\alpha'\beta'}}{\partial h} (A^{-1})_{j'j}^{\beta'\beta} = -(A^{-1})_{ii'}^{\alpha\alpha'} \frac{\partial T_{i'j'}^{\alpha'\beta'}}{\partial h} (A^{-1})_{j'j}^{\beta'\beta}$$

where the Einstein's index notation is employed for i', j', α', β'. We can now proceed to compute the derivatives for the energy term related to $\vec{p}_i$ in energy as follows,

$$\frac{\partial - \sum_i^N \frac{1}{2} \vec{p}_i \cdot \vec{E}_i^0}{\partial h} = \frac{1}{2} E_i^{0\alpha} (A^{-1})_{ii'}^{\alpha\alpha'} \frac{\partial T_{i'j'}^{\alpha'\beta'}}{\partial h} (A^{-1})_{j'j}^{\beta'\beta} E_j^{0\beta} - \sum_i^N \vec{p}_i \cdot \frac{\partial \vec{E}_i^0}{\partial h}$$

$$= \frac{1}{2} p_{i'}^{\alpha'} \frac{\partial T_{i'j'}^{\alpha'\beta'}}{\partial h} p_{j'}^{\beta'} - \sum_i^N \vec{p}_i \cdot \frac{\partial \vec{E}_i^0}{\partial h}$$

Again, the Einstein's index notation is employed here for i, j, α, β i', j', α', β'.

With the above preparations, we can calculate the electrostatic energy part of the virial from its definition,

$$\vec{\vec{V}} \cdot (h^{-1})^T = -\frac{\partial U}{\partial h_{\mu\nu}} = -(\frac{\partial U_{rec}}{\partial h_{\mu\nu}} + \frac{\partial U_{dir}}{\partial h_{\mu\nu}} + \frac{\partial U_{self}}{\partial h_{\mu\nu}})$$

$$\frac{\partial U_{rec}}{\partial h_{\mu\nu}}$$

$$= \frac{\partial \sum_i^N \frac{1}{2}(q_i + \vec{\mu}_i \cdot \nabla_i) \frac{1}{\pi V} \sum_{\vec{m}\neq 0} \left\{ \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \sum_j^N \left[\left(q_j + 2\pi i \vec{m} \cdot \vec{\mu}_j\right) \exp\left(2\pi i \vec{m} \cdot \vec{R}_j\right) \right] \exp\left(-2\pi i \vec{m} \cdot \vec{R}_i\right) \right\}}{\partial h_{\mu\nu}}$$

$$+ \frac{1}{2} p_{i'}^{\alpha'} \frac{\partial\, \partial_{i'}^{\alpha'} \partial_{j'}^{\beta'} \frac{1}{\pi V} \sum_{\vec{m}\neq 0} \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \exp\left(2\pi i \vec{m} \cdot \vec{R}_{j'}\right) \exp\left(-2\pi i \vec{m} \cdot \vec{R}_{i'}\right)}{\partial h_{\mu\nu}} p_{j'}^{\beta'}$$

$$+ \sum_i^N \vec{p}_i$$

$$\cdot \frac{\partial \nabla_i \frac{1}{\pi V} \sum_{\vec{m}\neq 0} \{ \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \sum_j^N \left[\left(q_j + 2\pi i \vec{m} \cdot \vec{\mu}_j\right) \exp\left(2\pi i \vec{m} \cdot \vec{R}_j\right) \right] \exp\left(-2\pi i \vec{m} \cdot \vec{R}_i\right) \}}{\partial h_{\mu\nu}}$$

$$\frac{\partial U_{dir}}{\partial h_{\mu\nu}} = \frac{\partial \sum_i^N \frac{1}{2}(q_i + \vec{\mu}_i \cdot \nabla_i) \sum_{j\neq i}^{\infty}(q_j + \vec{\mu}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta_{ij}R_{ij}) - \mathrm{erf}(\beta_0 R_{ij})}{R_{ij}}}{\partial h_{\mu\nu}}$$

$$+ \frac{1}{2} p_{i'}^{\alpha'} \frac{\partial\, \partial_{i'}^{\alpha'} \partial_{j'}^{\beta'} \frac{\mathrm{erf}(\beta_{i'j'}R_{i'j'}) - \mathrm{erf}(\beta_0 R_{i'j'})}{R_{i'j'}}}{\partial h_{\mu\nu}} p_{j'}^{\beta'}$$

$$+ \sum_i^N \vec{p}_i \cdot \frac{\partial \nabla_i \sum_{j\neq i}^{\infty}(q_j + \vec{\mu}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta_{ij}R_{ij}) - \mathrm{erf}(\beta_0 R_{ij})}{R_{ij}}}{\partial h_{\mu\nu}}$$

$$\frac{\partial U_{self}}{\partial h_{\mu\nu}} = \frac{\partial \sum_i^N \frac{1}{2}(q_i + \vec{\mu}_i \cdot \nabla_i)[-\lim_{\vec{R}_j \to \vec{R}_i} (q_i + \vec{\mu}_i \cdot \nabla_j) \frac{\mathrm{erf}(\beta_0 R_{ij})}{R_{ij}}]}{\partial h_{\mu\nu}}$$

$$+ \frac{1}{2} p_{i'}^{\alpha'} \frac{\partial\, \partial_{i'}^{\alpha'} \partial_{j'}^{\beta'} [-\lim_{\vec{R}_j \to \vec{R}_i} \frac{\mathrm{erf}(\beta_0 R_{ij})}{R_{ij}}]}{\partial h_{\mu\nu}} p_{j'}^{\beta'}$$

$$+ \sum_i^N \vec{p}_i \cdot \frac{\partial \nabla_i [-\lim_{\vec{R}_j \to \vec{R}_i} (q_i + \vec{\mu}_i \cdot \nabla_j) \frac{\mathrm{erf}(\beta_0 R_{ij})}{R_{ij}}]}{\partial h_{\mu\nu}}$$

After careful derivation (see **Appendix** B.2. for details), the following results can be obtained:

$$\frac{\partial U_{rec}}{\partial h_{\mu\nu}} = -\frac{1}{2\pi V}(h^{-1})^{T}_{\mu\nu}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}\left[q_j + 2\pi i\vec{m}\right.\right.$$

$$\left.\left.\cdot\left(\vec{\mu}_j + \vec{p}_j\right)\right]\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}\left[q_i - 2\pi i\vec{m}\cdot(\vec{\mu}_i + \vec{p}_i)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{m}^{\gamma}\right)\sum_{j}^{N}\left[q_j + 2\pi i\vec{m}\right.\right.$$

$$\left.\left.\cdot\left(\vec{\mu}_j + \vec{p}_j\right)\right]\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}\left[q_i - 2\pi i\vec{m}\cdot(\vec{\mu}_i + \vec{p}_i)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}\left[q_j + 2\pi i\vec{m}\right.\right.$$

$$\cdot\left(\vec{\mu}_j + \vec{p}_j\right)\Big]\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}\sum_{k}^{n}\left[\left(-2\pi i\vec{m}\cdot\vec{R}_{ik}\mu_{ik}\right)\left(-\frac{\vec{R}^{\mu}_{ik}\vec{S}^{\nu}_{ik}}{\left|\vec{R}_{ik}\right|^3}\right)\right]\exp\Big(-2\pi i\vec{m}$$

$$\left.\cdot\vec{R}_i\Big)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}\left[q_j + 2\pi i\vec{m}\right.\right.$$

$$\left.\left.\cdot\left(\vec{\mu}_j + \vec{p}_j\right)\right]\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}\left[(-2\pi i)\left(-\vec{m}^{\mu}h^{-1}_{\nu\gamma}p_i^{\gamma}\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$= -\frac{1}{2}(h^{-1})^{T}_{\mu\nu} \cdot \sum_{i}^{N}\left[q_i(\phi_{rec})_i - (\vec{\mu}_i + \vec{p}_i)\cdot(\vec{E}_{rec})_i\right]$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{m}^{\gamma}\right)S(\vec{m})S(-\vec{m})\right\}$$

$$+\sum_{i}^{N}\sum_{k}^{n}\left[\left(-(\vec{E}_{rec})_i\cdot\vec{R}_{ik}\mu_{ik}\right)\left(-\frac{\vec{R}^{\mu}_{ik}\vec{S}^{\nu}_{ik}}{\left|\vec{R}_{ik}\right|^3}\right)\right]+\sum_{i}^{N}(\vec{E}_{rec})^{\mu}_{i}h^{-1}_{\nu\gamma}p^{\gamma}_{i}$$

where $S(\vec{m})$ is the structure factor, defined as $\sum_j^N\left[q_j + 2\pi i\vec{m}\cdot\left(\vec{\mu}_j+\vec{p}_j\right)\right]\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)$.

$$\frac{\partial U_{dir}}{\partial h_{\mu\nu}} = -\sum_{i}^{N}\sum_{k}^{n}u_{ik}\left[\frac{\left(\vec{E}_{dir}\right)^{\mu}_{i}\vec{S}^{\nu}_{ik}}{\left|\vec{R}_{ik}\right|} - \left(\vec{E}_{dir}\right)_i\cdot\vec{R}_{ik}\frac{\vec{R}^{\mu}_{ik}\vec{S}^{\nu}_{ik}}{\left|\vec{R}_{ik}\right|^3}\right]$$

$$-\frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{\infty}\left\{q_i\left(\vec{E}_{dir}\right)^{\mu}_{j\to i}(\vec{S}^{\nu}_{i}-\vec{S}^{\nu}_{j}) + \left[(\vec{u}_i+\vec{p}_i)\cdot\left(\vec{\vec{E}}_{dir}\right)_{j\to i}\right]^{\mu}(\vec{S}^{\nu}_{i}-\vec{S}^{\nu}_{j})\right\}$$

$$\frac{\partial U_{self}}{\partial h_{\mu\nu}} = -\sum_{i}^{N}\sum_{k}^{n}u_{ik}\left[\frac{\left(\vec{E}_{self}\right)^{\mu}_{i}\vec{S}^{\nu}_{ik}}{\left|\vec{R}_{ik}\right|} - \left(\vec{E}_{self}\right)_i\cdot\vec{R}_{ik}\frac{\vec{R}^{\mu}_{ik}\vec{S}^{\nu}_{ik}}{\left|\vec{R}_{ik}\right|^3}\right]$$

Here $\vec{E}_{rec}$, $\vec{E}_{dir}$ and $\vec{E}_{self}$ refer to the electrostatic fields resulting from the reciprocal part, direct part, or self-part of an Ewald summation, respectively. And the notation for ϕ and $\vec{\vec{E}}$ are similar. Furthermore, the following expressions for electric field and its derivatives are used:

$$\left(\vec{E}_{dir}\right)_{j\to i} = -(q_j + \vec{\mu}_j\cdot\nabla_j)\nabla_i\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)-\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}$$

$$\left(\vec{\vec{E}}_{dir}\right)_{j\to i} = -(q_j + \vec{\mu}_j\cdot\nabla_j)\nabla_i\nabla_i\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)-\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}$$

Adding up all three terms above and removing the common factor $-(h^{-1})^T$, the final expression of the electrostatic energy contribution to virial is as follows:

$$V_{\mu\nu} = \frac{1}{2}\delta_{\mu\nu} \cdot \sum_{i}^{N} \left[q_i(\phi_{rec})_i - (\vec{\mu}_i + \vec{p}_i) \cdot \left(\vec{E}_{rec}\right)_i\right]$$

$$- \frac{1}{\pi V} \sum_{\vec{m} \neq 0} \left\{ \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \frac{1 + \frac{\pi^2 \vec{m}^2}{\beta_0^2}}{\vec{m}^2} \vec{m}^{\mu} \vec{m}^{\nu} S(\vec{m}) S(-\vec{m}) \right\} - \sum_{i}^{N} \left(\vec{E}_{rec}\right)_i^{\mu} p_i^{\nu}$$

$$+ \sum_{i}^{N} \sum_{k}^{n} u_{ik} \left[\frac{\left(\vec{E}_{dir} + \vec{E}_{self}\right)_i^{\mu} \vec{R}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|} - \left(\vec{E}_{rec} + \vec{E}_{dir} + \vec{E}_{self}\right)_i \cdot \vec{R}_{ik} \frac{\vec{R}_{ik}^{\mu} \vec{R}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|^3} \right]$$

$$+ \frac{1}{2} \sum_{i}^{N} \sum_{j \neq i}^{\infty} \left\{ q_i \left(\vec{E}_{dir}\right)_{j \to i}^{\mu} (\vec{R}_i^{\nu} - \vec{R}_j^{\nu}) + \left[(\vec{u}_i + \vec{p}_i) \cdot \left(\vec{\vec{E}}_{dir}\right)_{j \to i} \right]^{\mu} (\vec{R}_i^{\nu} - \vec{R}_j^{\nu}) \right\}$$

Stress Tensor of Rigid Molecules

Molecular dynamics simulations of bio-molecular systems are often conducted under certain constraints. The most common are distance constraints,

$$\left(\vec{R}_i - \vec{R}_j\right)^2 - d_{ij}^2 = 0$$

When molecules are under this kind of constraints, a molecule-based scaling in NPT simulations is often used to vary the box dimensions instead of atom-based scaling. In doing so, the atomic coordinate can be expressed as

$$\vec{R}_i = \vec{R}_{i0} + \vec{d}_i = s_{i0,1}\vec{u}_1 + s_{i0,2}\vec{u}_2 + s_{i0,3}\vec{u}_3 + \vec{d}_i$$

where $\vec{R}_{i0}$ is the molecular center to which atom i belongs (usually the mass center of the molecule), $\vec{s}_{i0}{}'s$ are the fractional coordinates of the center, and $\vec{d}_i$ is the relative displacement of atom i from the center. Obviously, $\vec{d}_i$ is a constant vector during the box dimension scaling.

Thus, the virial tensor of a pGM system under this rigid body condition with distance constraints is

$$
\begin{aligned}
V_{\mu\nu} = \frac{1}{2}\delta_{\mu\nu} \cdot \sum_{i}^{N} \left[q_i(\phi_{rec})_i - (\vec{\mu}_i + \vec{p}_i) \cdot \left(\vec{E}_{rec}\right)_i \right] \\
- \frac{1}{\pi V} \sum_{\vec{m} \neq 0} \left\{ \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \frac{1 + \frac{\pi^2 \vec{m}^2}{\beta_0^2}}{\vec{m}^2} \vec{m}^{\mu} \vec{m}^{\nu} S(\vec{m}) S(-\vec{m}) \right\} \\
- \sum_{i}^{N} \left(\vec{E}_{rec}\right)_i^{\mu} \left(\vec{\mu}_i^{\nu} + \vec{p}_i^{\nu} + q_i \vec{d}_i^{\nu}\right) - \sum_{i}^{N} \left(\vec{\mu}_i^{\gamma} + \vec{p}_i^{\gamma}\right) \left(\vec{\vec{E}}_{rec}\right)_i^{\gamma\mu} \vec{d}_i^{\nu} \\
+ \frac{1}{2} \sum_{i}^{N} \sum_{j \neq i}^{\infty} \{ q_i \left(\vec{E}_{dir}\right)_{j \to i}^{\mu} (\vec{R}_{i0}^{\nu} - \vec{R}_{j0}^{\nu}) + \left[(\vec{u}_i + \vec{p}_i) \cdot \left(\vec{\vec{E}}_{dir}\right)_{j \to i} \right]^{\mu} (\vec{R}_{i0}^{\nu} - \vec{R}_{j0}^{\nu}) \}
\end{aligned}
$$

Here index γ follows the Einstein's summation notation. The derivation of the above expression is similar to what was described before for flexible molecules, so it is omitted here. The detail can be found in **Appendix B.3**.

Virial Correction of Screened System

In the current pGM setup, the electrostatic interactions are not screened, meaning all 1-2, 1-3, and 1-4 electrostatic interactions are considered at their full strength. This choice is based on the consideration of polarizability anisotropy.[184] However, if electrostatic interactions between certain atoms are screened, the electrostatic virial correction can be defined as follows.

(1) For flexible molecules

$$\left(V_{\mu\nu}\right)_{correction} = \sum_{i}^{N}\sum_{k}^{n} u_{ik}\left[\frac{\left(-\vec{E}_{screen}\right)_{i}^{\mu}\vec{R}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|} - \left(-\vec{E}_{screen}\right)_{i}\cdot\vec{R}_{ik}\frac{\vec{R}_{ik}^{\mu}\vec{R}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|^{3}}\right]$$

$$+\frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{*(i)}\{q_i\left(-\vec{E}_{screen}\right)_{j\to i}^{\mu}(\vec{R}_{i}^{\nu}-\vec{R}_{j}^{\nu}) + \left[(\vec{u}_i+\vec{p}_i)\cdot\left(-\vec{\vec{E}}_{screen}\right)_{j\to i}\right]^{\mu}(\vec{R}_{i}^{\nu}$$

$$-\vec{R}_{j}^{\nu})\}$$

(2) For rigid molecules

$$\left(V_{\mu\nu}\right)_{correction} = \frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{*(i)}\{q_i\left(-\vec{E}_{screen}\right)_{j\to i}^{\mu}(\vec{R}_{i0}^{\nu}-\vec{R}_{j0}^{\nu}) + \left[(\vec{u}_i+\vec{p}_i)\cdot\left(-\vec{\vec{E}}_{screen}\right)_{j\to i}\right]^{\mu}(\vec{R}_{i0}^{\nu}$$

$$-\vec{R}_{j0}^{\nu})\}$$

Here $\vec{E}_{screen}$ and $\vec{\vec{E}}_{screen}$ are the electric field and its derivative generated by the screened interactions, and $*\,(i)$ represents the screened atom pairs of atom *i*.

Derivation of the above equations is given in **Appendix B.4**, which is similar to what was presented in our previous development.[183] However, it is important to point out that the screening has to be consistent throughout the model. In other words, we must screen the same interactions when computing both induced dipoles and forces. This is to ensure that the energy and corresponding forces are consistent under the same screening scheme.

5.3 Results and Discussion

Finite Difference Validation

As we presented above, there are the electrostatic virial expressions of flexible and rigid pGM molecular systems. To confirm the theoretical derivation, we performed a finite difference test, starting from the virial definition of Lagrangian, to assess whether the derivation is correct. Here a cubic box with 512 water molecules as in our previous study was used,[183-184] and the dimension of the box is 33 Å. To guarantee high-precision energy calculation, the PME setup uses the following parameters: coefficient β_0 = 0.3 $Å^{-1}$, B-spline interpolation order = 9, FFT grid spacing = 0.33 Å, and direct space cutoff = 14.4 Å. The induced dipole convergence was also set to a very tight criterium of 10^{-12}. The setup leads to an energy accuracy level of $\sim 10^{-12}$, so that the finite difference test can be carried successfully.

It is clear from Figure 5.1 that both finite-difference virial values approach their respective analytical values in roughly linear fashions as the finite difference step sizes decrease, demonstrating that the numerical values converge to the analytical values. This finding confirms that the analytical virial expressions are correct when used to compute virials of the tested molecular system. Note also that the finite-difference steps in the rigid body system (Fig. 5.1b) are 10-time larger than the ones used in the flexible system (Fig. 5.1a). This is because the virials of the rigid body system are much smaller ($\sim$1/50) than those of the flexible system due to large intramolecular contributions that are sensitive to the distance variations in the finite difference calculations. Hence, rather small finite difference step sizes were needed. Such large intramolecular contributions were absent in the rigid-body system and a somewhat larger step sizes were used for numerical accuracy. The linear convergence trends illustrated in both the flexible and rigid body systems confirm the correctness of virial. To further validate the result, we conducted the same finite difference test on another box with 4096 water molecules. The PME was set to even higher accuracy: coefficient β_0 = 0.4 $Å^{-1}$, B-spline interpolation order = 16, FFT grid spacing = 0.33 Å, and direct space cutoff = 15.0 Å. The induced dipole convergence was also set to 10^{-12}. The test results are extremely similar to that of Figure 5.1.

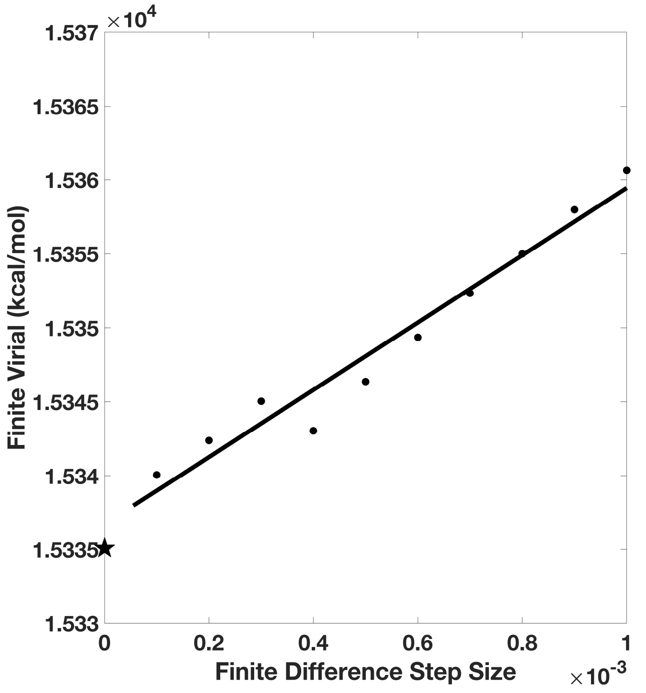
NPT Simulation of Water Box

To illustrate that the analytical formulation can achieve stable NPT simulations, we performed a test run for 1 ns of the water box containing 512 pGM water molecules.[183-184] The rigid-body formula was used in this test, and the water models were simply created from the standard TIP3P geometry by adding permanent and induced dipole moments and by setting the van der Waals parameters to be 9% of those in the standard TIP3P waters, to give a relatively reasonable density. We note that this pGM water model is yet to be optimized to reproduce any physical water properties. Here a setup of lower energy accuracy was used for reasonable simulation throughput, with the Ewald coefficient β_0 = 0.4 $Å^{-1}$, B-spline interpolation order = 8, FFT grid spacing = 0.5 Å, and direct space cutoff = 9 Å. The induced dipole convergence criterium was also raised to 10^{-6}. The simulation time step was chosen to be 1fs. The temperature was set to 300 K and the pressure 1.0 bar. The Berendsen thermostat and barostat were used and all other dynamics simulation parameters were set as default from the Amber SANDER program. The job was run on a single core of an INTEL Xeon E5-4620 CPU with a wall-clock time of 6.9 days.

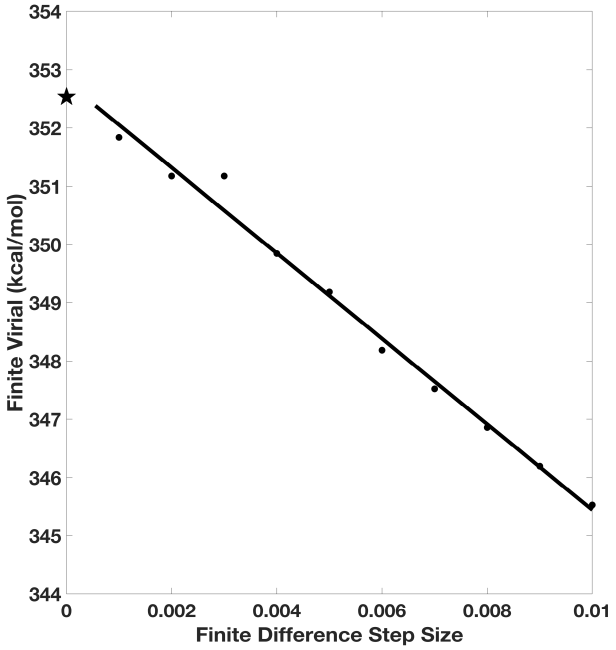
Figure 5.2 shows that the pGM/PME runs well under the NPT condition. Both density and pressure fluctuate around the equilibrium values, and the fluctuation ranges were similar to those observed for the classical point-charge models. The simulation was stable, no "polarization catastrophe" event was observed even if all interactions, including 1-2 and 1-3 interactions, were included.

It is interesting to point out that the electrostatic component of the pGM water model is much smaller (~1/10) than the TIP3P water model, even though the atomic charges are usually higher in polarizable models. This in part explains why we need to reduce the van der Waals parameters to 9% of the original TIP3P values, to match the much smaller electrostatic virial component so that the system would eventually equilibrate to a reasonable state. Indeed, the average density and pressure values are 0.99 g/cm^3 and 1.0 bar during the 1 ns run, respectively. As a supporting evidence for the correct NPT implementation for the pGM model, we also implemented a Monte Calo pressure regulation scheme for the pGM model, as no virial (pressure) calculation is needed in this regulation scheme.[185] The simulation followed the exact same setup and was found to stabilize at the same state, with averaged density values of 0.99 g/cm^3.

The much-reduced electrostatic virial component is worth noting but a reasonable behavior. The induced dipoles in the pGM model are actively playing the role of electrostatic screening. As a result, the electrostatic interactions between water molecules are largely reduced, leading to much lower electrostatic virial component. This phenomenon shows that the polarizable pGM model is fundamentally different from classical point-charge models. Thus, its parameterization shall require extensive development and validation. Nevertheless, we reiterate that the goal of this development is not to develop a pGM water model, but to establish a constant pressure simulation protocol for polarizable methods such as the pGM model.

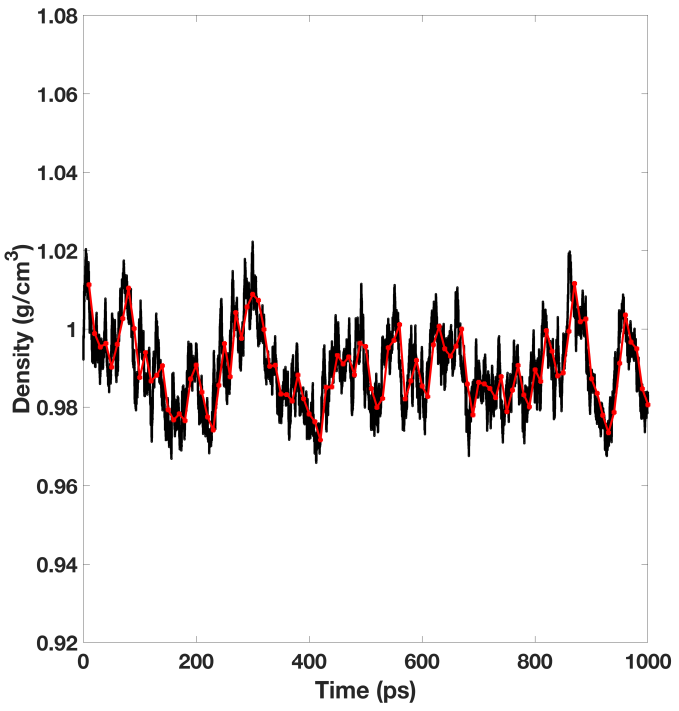

×10^4
1.537
1.5365
1.536
1.5355
1.535
1.5345
1.534
1.5335
1.533
Finite Virial (kcal/mol)
0
0.2
0.4
0.6
0.8
1
Finite Difference Step Size
×10^-3

(a)

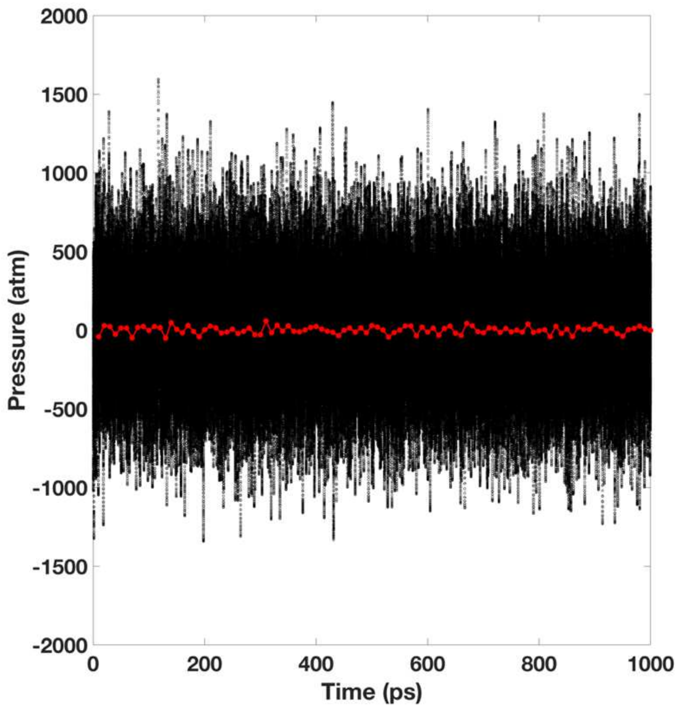

354
353
352
351
350
349
348
347
346
345
344
Finite Virial (kcal/mol)
0
0.002
0.004
0.006
0.008
0.01
Finite Difference Step Size

(b)

Figure 5.1. Finite-difference validation of the analytical virial expressions in the *xx* direction. (a) is for the flexible case and (b) is for the rigid body case. *x* axis refers to proportional changes of simulation box dimension in the *x* direction. The star on the *y* axis is the analytical virial value. The finite-difference values were obtained.



(a)



(b)

Figure 5.2. Time evolutions of density and pressure of the NPT simulation for the 512-water box. The red lines are the running averages of every 10 ps.

Another interesting observation upon inspecting Fig 5.2 is that the pGM water model exhibits a longer correlation time in density than the TIP3P water model, with pGM water's density correlation time as long as 200~300 ps, rather than tens of picoseconds as is usually observed for point-charge water models. This effect could either result from the imperfect calibration of the pGM water tested here, the use of a barostat coupling constant that is too weak for the pGM model, or the inclusion of the explicit polarization in the pGM model, which requires longer time to equilibrate. Obviously, the point charge models also have certain level of polarization effect from reorientation of the molecules, but they certainly do not acquire any electronic polarization that is explicitly modeled with the induced dipoles. All these phenomena strongly suggest that the parametrization of pGM models requires extra care and extensive validations.

5.4. Conclusion

In this work, we presented the derivation and implementation of an analytical method for achieving constant pressure simulation for the pGM model. Specifically, we derived the stress tensor expression for both flexible and rigid body molecular system in the pGM electrostatic model, the trace of which gives the expression for internal stress. In addition, we showed how to correct the stress tensor if a short-range screening strategy is used in the force field development. Once the internal stress is obtained, by balancing it with the externally applied pressure through a certain type of barostat, a constant pressure simulation is readily realized.

Since the formulation of the stress tensor in pGM model is rather complex, we first performed a finite difference test to validate its correctness. We showed that for both flexible and rigid-body systems, the finite-difference virial approaches the analytical values when decreasing the differentiation step size, confirming the correctness of the derived expressions.

The tensor expression was implemented into the Amber/SANDER program and tested with a small box of 512 pGM water molecules. In the calculations the revised TIP3P water geometry was applied, and the TIP3P van der Waals parameters were scaled down to 9% of the original values after imposing the pGM electrostatic interactions. After equilibration, the system was found to reach a reasonable density of 0.99 g/cm^3 and pressure of 1.0 bar under the room temperature during the production run of 1 ns. A constant pressure simulation of the system with Monte Carlo pressure regulation scheme stabilized at the same state. This shows that the analytical stress tensor formula and its implementation are successful.

Several interesting differences were observed between the pGM water model and the TIP3P water model from which the pGM model was currently derived. Notably, the electrostatic virial was much smaller, requiring much smaller van der Waals parameters to reach reasonable water density. The density correlation time was also much longer. The observation strongly suggests that the pGM model, as a polarizable model, is fundamentally different compared to classic point charge models. Thus, the parametrization of the pGM model requires extra care and extensive validation, which is a major focus of our next step research.

Chapter 6

Electrostatic interactions play crucial roles in biophysical processes such as protein-protein and protein-ligand interactions. Accurate and efficient treatment of electrostatics is thus vital in computational analyses of biomolecular structures and dynamics. In our multi-scaled modeling framework, the solute is represented in the atomic resolution or the coarse-grained resolution and the solvent is represented either in the atomic detail or in a continuum. Once completing the multi-scaled framework, we will have all-atom and coarse-grained solutes in one hand and explicit and implicit solvents in the other. Our solute models and solvent models can be combined in different manners for various accuracy and efficiency requirements.

When the solvent is in the continuum representation, the Poisson-Boltzmann modeling is necessary due to the presence of dielectric interface in the solution system, where the solute region is treated as a low dielectric constant region with a number of point charges/dipoles located at atomic centers, and the solvent is treated as a high dielectric constant region. Poisson-Boltzmann equation (PBE)-based implicit solvent model has been widely used in biomolecular applications.

Immersed interface method (IIM) is a promising high-accuracy numerical scheme for the Poisson-Boltzmann model. However, the IIM suffers from instability and slow convergence for typical applications. In this study, we introduced both analytical interface and surface regulation into IIM to address these issues. The analytical interface setup leads to better accuracy and its convergence closely follows a quadratic manner as predicted by theory. The surface regulation further speeds up the convergence for nontrivial biomolecules. In addition, uncertainties of the numerical energies for tested systems are also reduced by about half. More interestingly, the analytical setup significantly improves the linear solver efficiency and stability by generating more precise and better-conditioned linear systems. Finally, we implemented the bottleneck linear system solver on GPUs to further improve the efficiency of the method, so it can be widely used for practical biomolecular applications.

However, methods like IIM are very liberal in the use of grid points nearby the molecular surface, making them difficult to use on high-performance computing platforms. Alternatively, the harmonic average (HA) method has been used to approximate dielectric interface conditions near the molecular surface with surprisingly good convergence and is well suited for high-performance computing. By adopting a 7-point stencil, the HA method is advantageous in generating simple 7-banded coefficient matrices, which greatly facilitate linear system solution with dense data parallelism, on high-performance computing platforms such as graphics processing unit (GPU). However, the HA method is limited due to its lower accuracy. Therefore, it would be of great interest for high-performance applications to develop more accurate methods while retaining the simplicity and effectiveness of the 7-point stencil discretization scheme. In this study, we have developed two new algorithms based on the spirit of the HA method by introducing more physical interface relations and imposing the discretized Poisson's equation to the second order, respectively. Our testing shows that, for typical biomolecules, the new methods significantly improve the numerical accuracy to that comparable to the second-order solvers, and with ~65% overall efficiency gain on widely available high-performance GPU platforms.

Except for solving Poisson Boltzmann equations, a vital step in almost all implicit solvent models is to determine the solvent-solute interface, and the solvent excluded surface (SES) is the most widely used interface definition in these models. However, classical algorithms used for computing SES are geometry-based, thus neither suitable for parallel implementations nor convenient for obtaining surface derivatives. To address the limitations, we explored a machine learning strategy to obtain a level-set formulation for the SES. The training process was conducted in three steps, eventually leading to a model with over 95% agreement with the classical SES. Visualization of tested molecular surfaces shows that the machine-learned SES overlaps with the classical SES on almost all situations. Further analyses shows that the machine-learned SES is incredibly stable in terms of both rotational and translational variation of the molecules. Our timing analysis shows that the machine-learned SES is roughly 2.5 times as efficient as the classical SES routine implemented in Amber/PBSA on a tested CPU platform. We expect further performance gain on massively parallel platforms such as GPUs given the ease in converting the machine-learned SES to a parallel procedure. We also implemented the machine-learned SES into the Amber/PBSA program to study its performance on reaction field energy calculation. The analysis shows that the two sets of reaction field energies are highly consistent with 1% deviation on average. Given its level-set formulation, we expect the machine-learned SES to be applied in molecular simulations that require either surface derivatives or high efficiency on parallel computing platforms.

Another aspect of our effort on modeling electrostatics is to develop a novel polarizable atomic model. Here we focus on a recently proposed polarizable Gaussian Multipole (pGM) model for biomolecular simulations. A key benefit of pGM is its screening of all short-range electrostatic interactions in a physically consistent manner, which is critical for stable charge-fitting and is needed to reproduce molecular anisotropy. Another advantage of pGM is that each atom's multipoles are represented by a single Gaussian function or its derivatives, allowing for more efficient electrostatics than other Gaussian-based models. In this study we present an efficient formulation for the pGM model defined with respect to a local frame formed with a set of covalent basis vectors. The covalent basis vectors are chosen to be along each atom's covalent bonding directions. The new local frame allows molecular flexibility during molecular simulations and facilitates an efficient formulation of analytical electrostatic forces without explicit torque computation. Subsequent numerical tests show that analytical atomic forces agree excellently with numerical finite-difference forces for the tested system. Finally, the new pGM electrostatics algorithm is interfaced with the PME implementation in Amber for molecular simulations under the periodic boundary conditions. To validate the overall pGM/PME electrostatics, we conducted an NVE simulation for a small water box of 512 water molecules. Our results show that, to achieve energy conservation in the polarizable model, it is important to ensure enough accuracy on both PME and induction iteration. It is hoped that the reformulated pGM model will facilitate the development of future force fields based on the pGM electrostatics for applications in biomolecular systems and processes where polarization plays crucial roles.

After the basic framework of pGM model has been built, we move forward to derive the pGM internal stress tensor for constant pressure MD simulations with the pGM electrostatics. Three different formulations are presented for the flexible, rigid, and short-range screened systems, respectively. The analytical formulations were implemented in the SANDER program in the Amber package and were first validated with the finite-difference method for two different boxes of pGM water molecules. This is followed by a constant temperature and constant pressure MD simulation for a box of 512 pGM water molecules. Our results show that the simulation system stabilized at a physically reasonable state and maintained the balance with the externally applied pressure. In addition, several fundamental differences were observed between the pGM and classic point charge models in terms of the simulation behaviors, indicating more extensive parametrization is necessary to utilizing the pGM electrostatics.

For future directions, we will continue working on the multi-scaled models of electrostatics. At the atomic level, we plan to vastly accelerate our current pGM implementation, including exploring fast convergence algorithms and utilizing advanced computing hardware, e.g., GPUs. At the coarse-grain level, we will develop models targeting large molecular systems with united-atom and residue-level resolutions. At the continuum (solvent) level, we are working on adding induction interactions back into Poisson-Boltzmann model, so that it will be able to model solute-solvent interaction more accurately, without losing its advantage in efficiency. In addition, we will continue to apply developed models to interesting biomedical problems in the field of molecular biology.

Reference

1. Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L., Comparison of simple potential functions for simulating liquid water. *The Journal of chemical physics* **1983,** *79* (2), 926-935.

2. Berendsen, H. J.; Postma, J. P.; van Gunsteren, W. F.; Hermans, J., Interaction models for water in relation to protein hydration. In *Intermolecular forces*, Springer: 1981; pp 331-342.

3. Mahoney, M. W.; Jorgensen, W. L., A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions. *The Journal of Chemical Physics* **2000,** *112* (20), 8910-8922.

4. Clementi, E.; Kistenmacher, H.; Kolos, W.; Romano, S., Non-Additivity in Water-Ion-Water Interactions. *Theor. Chim. Acta* **1980,** *55* (4), 257-266.

5. Ren, P.; Ponder, J. W., Polarizable atomic multipole water model for molecular mechanics simulation. *The Journal of Physical Chemistry B* **2003,** *107* (24), 5933-5947.

6. Halgren, T. A.; Damm, W., Polarizable force fields. *Current opinion in structural biology* **2001,** *11* (2), 236-242.

7. Rick, S. W.; Stuart, S. J., Potentials and algorithms for incorporating polarizability in computer simulations. *Reviews in computational chemistry* **2002,** *18*, 89-146.

8. Ponder, J. W.; Case, D. A., Force Fields for Protein Simulations. In *Advances in Protein Chemistry*, Academic Press: 2003; Vol. 66, pp 27-85.

9. Lamoureux, G.; MacKerell Jr, A. D.; Roux, B., A simple polarizable model of water based on classical Drude oscillators. *The Journal of chemical physics* **2003,** *119* (10), 5185-5197.

10. Yu, H.; Hansson, T.; van Gunsteren, W. F., Development of a simple, self-consistent polarizable model for liquid water. *The Journal of chemical physics* **2003,** *118* (1), 221-234.

11. Rick, S. W.; Stuart, S. J.; Berne, B. J., Dynamical fluctuating charge force fields: Application to liquid water. *The Journal of chemical physics* **1994,** *101* (7), 6141-6156.

12. Caldwell, J.; Dang, L. X.; Kollman, P. A., Implementation of nonadditive intermolecular potentials by use of molecular dynamics: development of a water-water potential and water-ion cluster interactions. *Journal of the American Chemical Society* **1990,** *112* (25), 9144-9147.

13. Burnham, C. J.; Li, J.; Xantheas, S. S.; Leslie, M., The parametrization of a Thole-type all-atom polarizable water model from first principles and its application to the study of water clusters (n= 2–21) and the phonon spectrum of ice Ih. *The Journal of chemical physics* **1999,** *110* (9), 4566-4581.

14. Vesely, F. J., N-particle dynamics of polarizable Stockmayer-type molecules. *Journal of Computational Physics* **1977,** *24* (4), 361-371.

15. Applequist, J.; Carl, J. R.; Fung, K.-K., Atom dipole interaction model for molecular polarizability. Application to polyatomic molecules and determination of atom polarizabilities. *Journal of the American Chemical Society* **1972,** *94* (9), 2952-2960.

16. Thole, B. T., Molecular polarizabilities calculated with a modified dipole interaction. *Chemical Physics* **1981,** *59* (3), 341-350.

17. Kaminski, G. A.; Stern, H. A.; Berne, B. J.; Friesner, R. A.; Cao, Y. X. X.; Murphy, R. B.; Zhou, R. H.; Halgren, T. A., Development of a polarizable force field for proteins via ab initio quantum chemistry: First generation model and gas phase tests. *Journal of Computational Chemistry* **2002,** *23* (16), 1515-1531.

18. Friesner, R. A., Modeling Polarization in Proteins and Protein–ligand Complexes: Methods and Preliminary Results. In *Advances in Protein Chemistry*, Academic Press: 2005; Vol. 72, pp 79-104.

19. Patel, S.; Mackerell, A. D.; Brooks, C. L., CHARMM fluctuating charge force field for proteins: II - Protein/solvent properties from molecular dynamics simulations using a nonadditive electrostatic model. *Journal of Computational Chemistry* **2004,** *25* (12), 1504-1514.

20. Lamoureux, G.; Harder, E.; Vorobyov, I. V.; Roux, B.; MacKerell, A. D., A polarizable model of water for molecular dynamics simulations of biomolecules. *Chemical Physics Letters* **2006,** *418* (1-3), 245-249.

21. Lopes, P. E. M.; Lamoureux, G.; Roux, B.; MacKerell, A. D., Polarizable empirical force field for aromatic compounds based on the classical drude oscillator. *Journal of Physical Chemistry B* **2007,** *111* (11), 2873-2885.

22. Jiang, W.; Hardy, D. J.; Phillips, J. C.; MacKerell, A. D., Jr.; Schulten, K.; Roux, B., High-Performance Scalable Molecular Dynamics Simulations of a Polarizable Force Field Based on Classical Drude Oscillators in NAMD. *Journal of Physical Chemistry Letters* **2011,** *2* (2), 87-92.

23. Ponder, J. W.; Wu, C.; Ren, P.; Pande, V. S.; Chodera, J. D.; Schnieders, M. J.; Haque, I.; Mobley, D. L.; Lambrecht, D. S.; DiStasio, R. A., Jr.; Head-Gordon, M.; Clark, G. N. I.; Johnson, M. E.; Head-Gordon, T., Current Status of the AMOEBA Polarizable Force Field. *Journal of Physical Chemistry B* **2010,** *114* (8), 2549-2564.

24. Cieplak, P.; Caldwell, J.; Kollman, P. A., Molecular Mechanical Models for Organic and Biological Systems Going Beyond the Atom Centered Two Body Additive Approximation: Aqueous Solution Free Energies of Methanol and N-Methyl Acetamide, Nucleic Acid Base, and Amide Hydrogen Bonding and Chloroform/Water Partition Coefficients of the Nucleic Acid Bases. *J Comput Chem* **2001,** *22* (10), 1048-1057.

25. Wang, J.; Cieplak, P.; Li, J.; Hou, T.; Luo, R.; Duan, Y., Development of Polarizable Models for Molecular Mechanical Calculations I: Parameterization of Atomic Polarizability. *Journal of Physical Chemistry B* **2011,** *115* (12), 3091-3099.

26. Wang, J.; Cieplak, P.; Li, J.; Wang, J.; Cai, Q.; Hsieh, M.; Lei, H.; Luo, R.; Duan, Y., Development of Polarizable Models for Molecular Mechanical Calculations II: Induced Dipole Models Significantly Improve Accuracy of Intermolecular Interaction Energies. *Journal of Physical Chemistry B* **2011,** *115* (12), 3100-3111.

27. Wang, J.; Cieplak, P.; Cai, Q.; Hsieh, M. J.; Wang, J. M.; Duan, Y.; Luo, R., Development of Polarizable Models for Molecular Mechanical Calculations. 3. Polarizable Water Models Conforming to Thole Polarization Screening Schemes. *Journal of Physical Chemistry B* **2012,** *116* (28), 7999-8008.

28. Wang, J. M.; Cieplak, P.; Li, J.; Cai, Q.; Hsieh, M. J.; Luo, R.; Duan, Y., Development of Polarizable Models for Molecular Mechanical Calculations. 4. van der Waals Parametrization. *Journal of Physical Chemistry B* **2012,** *116* (24), 7088-7101.

29. Williams, D. E., Representation of the molecular electrostatic potential by atomic multipole and bond dipole models. *Journal of computational chemistry* **1988,** *9* (7), 745-763.

30. Ponder, J. W.; Wu, C.; Ren, P.; Pande, V. S.; Chodera, J. D.; Schnieders, M. J.; Haque, I.; Mobley, D. L.; Lambrecht, D. S.; DiStasio, R. A., Jr.; Head-Gordon, M.; Clark, G. N.; Johnson, M. E.; Head-Gordon, T., Current status of the AMOEBA polarizable force field. *J Phys Chem B* **2010,** *114* (8), 2549-64.

31. Shi, Y.; Xia, Z.; Zhang, J.; Best, R.; Wu, C.; Ponder, J. W.; Ren, P., The Polarizable Atomic Multipole-based AMOEBA Force Field for Proteins. *J Chem Theory Comput* **2013,** *9* (9), 4046-4063.

32. Cisneros, G. A.; Piquemal, J.-P.; Darden, T. A., Generalization of the Gaussian electrostatic model: Extension to arbitrary angular momentum, distributed multipoles, and speedup with reciprocal space methods. *The Journal of chemical physics* **2006,** *125* (18), 184101.

33. Piquemal, J. P.; Cisneros, G. A.; Reinhardt, P.; Gresh, N.; Darden, T. A., Towards a force field based on density fitting. *Journal of Chemical Physics* **2006,** *124* (10).

34. Duke, R. E.; Starovoytov, O. N.; Piquemal, J. P.; Cisneros, G. A., GEM*: A Molecular Electronic Density-Based Force Field for Molecular Dynamics Simulations. *Journal of Chemical Theory and Computation* **2014,** *10* (4), 1361-1365.

35. Rappe, A. K.; Goddard III, W. A., Charge equilibration for molecular dynamics simulations. *The Journal of Physical Chemistry* **1991,** *95* (8), 3358-3363.

36. Stern, H. A.; Rittner, F.; Berne, B.; Friesner, R. A., Combined fluctuating charge and polarizable dipole models: Application to a five-site water potential function. *The Journal of chemical physics* **2001,** *115* (5), 2237-2251.

37. Guillot, B.; Guissani, Y., How to build a better pair potential for water. *The Journal of Chemical Physics* **2001,** *114* (15), 6720-6733.

38. Klapper, I.; Hagstrom, R.; Fine, R.; Sharp, K.; Honig, B., Focusing of Electric Fields in the Active Site of Copper-Zinc Superoxide Dismutase Effects of Ionic Strength and Amino Acid Modification. *Proteins Structure Function and Genetics* **1986,** *1* (1), 47-59.

39. Davis, M. E.; McCammon, J. A., Solving the Finite-Difference Linearized Poisson-Boltzmann Equation - a Comparison of Relaxation and Conjugate-Gradient Methods. *J. Comput. Chem.* **1989,** *10* (3), 386-391.

40. Nicholls, A.; Honig, B., A Rapid Finite-Difference Algorithm, Utilizing Successive over-Relaxation to Solve the Poisson-Boltzmann Equation. *J. Comput. Chem.* **1991,** *12* (4), 435-445.

41. Luty, B. A.; Davis, M. E.; McCammon, J. A., Solving the Finite-Difference Nonlinear Poisson-Boltzmann Equation. *J. Comput. Chem.* **1992,** *13* (9), 1114-1118.

42. Holst, M.; Saied, F., Multigrid Solution of the Poisson-Boltzmann Equation. *J. Comput. Chem.* **1993,** *14* (1), 105-113.

43. Forsten, K. E.; Kozack, R. E.; Lauffenburger, D. A.; Subramaniam, S., Numerical-Solution of the Nonlinear Poisson-Boltzmann Equation for a Membrane-Electrolyte System. *J. Phys. Chem.* **1994,** *98* (21), 5580-5586.

44. Im, W.; Beglov, D.; Roux, B., Continuum Solvation Model: computation of electrostatic forces from numerical solutions to the Poisson-Boltzmann equation. *Comput. Phys. Commun.* **1998,** *111* (1-3), 59-75.

45. Rocchia, W.; Alexov, E.; Honig, B., Extending the applicability of the nonlinear Poisson-Boltzmann equation: Multiple dielectric constants and multivalent ions. *J. Phys. Chem. B* **2001,** *105* (28), 6507-6514.

46. Luo, R.; David, L.; Gilson, M. K., Accelerated Poisson-Boltzmann calculations for static and dynamic systems. *J. Comput. Chem.* **2002,** *23* (13), 1244-1253.

47. Bashford, D., An Object-Oriented Programming Suite for Electrostatic Effects in Biological Molecules. *Lecture Notes in Computer Science* **1997,** *1343*, 233-240.

48. Lu, Q.; Luo, R., A Poisson-Boltzmann dynamics method with nonperiodic boundary condition. *J. Chem. Phys.* **2003,** *119* (21), 11035-11047.

49. Prabhu, N. V.; Zhu, P. J.; Sharp, K. A., Implementation and testing of stable, fast implicit solvation in molecular dynamics using the smooth-permittivity finite difference Poisson-Boltzmann method. *Journal of Computational Chemistry* **2004,** *25* (16), 2049-2064.

50. Wang, J.; Luo, R., Assessment of Linear Finite-Difference Poisson-Boltzmann Solvers. *Journal of Computational Chemistry* **2010,** *31* (8), 1689-1698.

51. Cai, Q.; Hsieh, M. J.; Wang, J.; Luo, R., Performance of Nonlinear Finite-Difference Poisson-Boltzmann Solvers. *Journal of Chemical Theory and Computation* **2010,** *6* (1), 203-211.

52. Li, L.; Li, C.; Sarkar, S.; Zhang, J.; Witham, S.; Zhang, Z.; Wang, L.; Smith, N.; Petukh, M.; Alexov, E., DelPhi: a comprehensive suite for DelPhi software and associated resources. *BMC biophysics* **2012,** *5* (1), 9.

53. Sharp, K.; Honig, B., Lattice models of electrostatic interactions–the finite-difference Poisson-Boltzmann method. *Chem. Scr. A* **1989,** *29*, 71-74.

54. Cortis, C. M.; Friesner, R. A., Numerical solution of the Poisson-Boltzmann equation using tetrahedral finite-element meshes. *J. Comput. Chem.* **1997,** *18* (13), 1591-1608.

55. Baker, N.; Holst, M.; Wang, F., Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems. *J. Comput. Chem.* **2000,** *21* (15), 1343-1352.

56. Holst, M.; Baker, N.; Wang, F., Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples. *J. Comput. Chem.* **2000,** *21* (15), 1319-1342.
57. Shestakov, A. I.; Milovich, J. L.; Noy, A., Solution of the nonlinear Poisson-Boltzmann equation using pseudo-transient continuation and the finite element method. *J. Colloid Interface Sci.* **2002,** *247* (1), 62-79.
58. Chen, L.; Holst, M. J.; Xu, J. C., The finite element approximation of the nonlinear Poisson-Boltzmann equation. *SIAM Journal on Numerical Analysis* **2007,** *45*, 2298-2320.
59. Xie, D.; Zhou, S., A new minimization protocol for solving nonlinear Poisson–Boltzmann mortar finite element equation. *BIT Numerical Mathematics* **2007,** *47* (4), 853-871.
60. Friedrichs, M.; Zhou, R. H.; Edinger, S. R.; Friesner, R. A., Poisson-Boltzmann analytical gradients for molecular modeling calculations. *J Phys. Chem. B* **1999,** *103* (16), 3057-3061.
61. Bond, S. D.; Chaudhry, J. H.; Cyr, E. C.; Olson, L. N., A First-Order System Least-Squares Finite Element Method for the Poisson-Boltzmann Equation. *J Comput. Chem.* **2010,** *31* (8), 1625-1635.
62. Lu, B.; Holst, M. J.; McCammon, J. A.; Zhou, Y. C., Poisson-Nernst-Planck equations for simulating biomolecular diffusion-reaction processes I: Finite element solutions. *J Comput Phys* **2010,** *229* (19), 6979-6994.
63. Lu, B. Z.; Zhou, Y. C., Poisson-Nernst-Planck Equations for Simulating Biomolecular Diffusion-Reaction Processes II: Size Effects on Ionic Distributions and Diffusion-Reaction Rates. *Biophys J* **2011,** *100* (10), 2475-2485.
64. Miertus, S.; Scrocco, E.; Tomasi, J., Electrostatic Interaction of a Solute with a Continuum - a Direct Utilization of Abinitio Molecular Potentials for the Prevision of Solvent Effects. *Chem. Phys.* **1981,** *55* (1), 117-129.
65. Hoshi, H.; Sakurai, M.; Inoue, Y.; Chujo, R., Medium Effects on the Molecular Electronic-Structure .1. the Formulation of a Theory for the Estimation of a Molecular Electronic-Structure Surrounded by an Anisotropic Medium. *J. Chem. Phys.* **1987,** *87* (2), 1107-1115.
66. Zauhar, R. J.; Morgan, R. S., The Rigorous Computation of the Molecular Electric-Potential. *J. Comput. Chem.* **1988,** *9* (2), 171-187.
67. Rashin, A. A., Hydration Phenomena, Classical Electrostatics, and the Boundary Element Method. *J. Phys. Chem.* **1990,** *94* (5), 1725-1733.
68. Yoon, B. J.; Lenhoff, A. M., A Boundary Element Method for Molecular Electrostatics with Electrolyte Effects. *J. Comput. Chem.* **1990,** *11* (9), 1080-1086.
69. Juffer, A. H.; Botta, E. F. F.; Vankeulen, B. A. M.; Vanderploeg, A.; Berendsen, H. J. C., The Electric-Potential of a Macromolecule in a Solvent - a Fundamental Approach. *J. Comput. Phys.* **1991,** *97* (1), 144-171.
70. Zhou, H. X., Boundary-Element Solution of Macromolecular Electrostatics - Interaction Energy between 2 Proteins. *Biophys. J.* **1993,** *65* (2), 955-963.
71. Bharadwaj, R.; Windemuth, A.; Sridharan, S.; Honig, B.; Nicholls, A., The Fast Multipole Boundary-Element Method for Molecular Electrostatics - an Optimal Approach for Large Systems. *J. Comput. Chem.* **1995,** *16* (7), 898-913.
72. Purisima, E. O.; Nilar, S. H., A Simple yet Accurate Boundary-Element Method for Continuum Dielectric Calculations. *J. Comput. Chem.* **1995,** *16* (6), 681-689.
73. Liang, J.; Subramaniam, S., Computation of molecular electrostatics with boundary element methods. *Biophys. J.* **1997,** *73* (4), 1830-1841.
74. Vorobjev, Y. N.; Scheraga, H. A., A fast adaptive multigrid boundary element method for macromolecular electrostatic computations in a solvent. *J. Comput. Chem.* **1997,** *18* (4), 569-583.
75. Totrov, M.; Abagyan, R., Rapid boundary element solvation electrostatics calculations in folding simulations: Successful folding of a 23-residue peptide. *Biopolymers* **2001,** *60* (2), 124-133.

76. Boschitsch, A. H.; Fenley, M. O.; Zhou, H. X., Fast boundary element method for the linear Poisson-Boltzmann equation. *J. Phys. Chem. B* **2002,** *106* (10), 2741-2754.
77. Lu, B. Z.; Cheng, X. L.; Huang, J. F.; McCammon, J. A., Order N algorithm for computation of electrostatic interactions in biomolecular systems. *Proc. Natl. Acad. Sci. U. S. A.* **2006,** *103* (51), 19314-19319.
78. Lu, B. Z.; Cheng, X. L.; Hou, T. J.; McCammon, J. A., Calculation of the Maxwell stress tensor and the Poisson-Boltzmann force on a solvated molecular surface using hypersingular boundary integrals. *J Chem. Phys.* **2005,** *123* (8).
79. Lu, B. Z.; Zhang, D. Q.; McCammon, J. A., Computation of electrostatic forces between solvated molecules determined by the Poisson-Boltzmann equation using a boundary element method. *J Chem. Phys.* **2005,** *122* (21).
80. Lu, B. Z.; Cheng, X. L.; Huang, J. F.; McCammon, J. A., An Adaptive Fast Multipole Boundary Element Method for Poisson-Boltzmann Electrostatics. *J Chem. Theory Comput.* **2009,** *5* (6), 1692-1699.
81. Bajaj, C.; Chen, S. C.; Rand, A., An Efficient Higher-Order Fast Multipole Boundary Element Solution for Poisson-Boltzmann-Based Molecular Electrostatics. *Siam J Sci Comput* **2011,** *33* (2), 826-848.
82. Dominy, B. N.; Brooks, C. L., Development of a generalized born model parametrization for proteins and nucleic acids. *Journal Of Physical Chemistry B* **1999,** *103* (18), 3765-3773.
83. Onufriev, A.; Bashford, D.; Case, D. A., Modification of the generalized Born model suitable for macromolecules. *Journal Of Physical Chemistry B* **2000,** *104* (15), 3712-3720.
84. Srinivasan, J.; Trevathan, M. W.; Beroza, P.; Case, D. A., Application of a pairwise generalized Born model to proteins and nucleic acids: inclusion of salt effects. *Theoretical Chemistry Accounts* **1999,** *101* (6), 426-434.
85. Tsui, V.; Case, D. A., Molecular dynamics simulations of nucleic acids with a generalized born solvation model. *J Am Chem Soc* **2000,** *122* (11), 2489-2498.
86. Gallicchio, E.; Zhang, L. Y.; Levy, R. M., The SGB/NP hydration free energy model based on the surface generalized born solvent reaction field and novel nonpolar hydration free energy estimators. *Journal Of Computational Chemistry* **2002,** *23* (5), 517-529.
87. Lee, M. S.; Salsbury, F. R.; Brooks, C. L., Novel generalized Born methods. *J Chem Phys* **2002,** *116* (24), 10606-10614.
88. Onufriev, A.; Case, D. A.; Bashford, D., Effective Born radii in the generalized Born approximation: The importance of being perfect. *Journal Of Computational Chemistry* **2002,** *23* (14), 1297-1304.
89. Gallicchio, E.; Levy, R. M., AGBNP: An analytic implicit solvent model suitable for molecular dynamics simulations and high-resolution modeling. *Journal of Computational Chemistry* **2004,** *25* (4), 479-499.
90. Onufriev, A.; Bashford, D.; Case, D. A., Exploring protein native states and large-scale conformational changes with a modified generalized born model. *Proteins-Structure Function and Bioinformatics* **2004,** *55* (2), 383-394.
91. Li, X. F.; Hassan, S. A.; Mehler, E. L., Long dynamics simulations of proteins using atomistic force fields and a continuum representation of solvent effects: Calculation of structural and dynamic properties. *Proteins-Structure Function and Bioinformatics* **2005,** *60* (3), 464-484.
92. Chen, J. H.; Im, W. P.; Brooks, C. L., Balancing solvation and intramolecular interactions: Toward a consistent generalized born force field. *J Am Chem Soc* **2006,** *128* (11), 3728-3736.
93. Chocholousova, J.; Feig, M., Implicit solvent simulations of DNA and DNA-protein complexes: Agreement with explicit solvent vs experiment. *Journal Of Physical Chemistry B* **2006,** *110* (34), 17240-17251.
94. Davis, M. E.; Mccammon, J. A., Dielectric Boundary Smoothing in Finite-Difference Solutions of the Poisson Equation - an Approach to Improve Accuracy and Convergence. *J Comput. Chem.* **1991,** *12* (7), 909-912.

95. LeVeque, R. J.; Li, Z., The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.* **1994,** *31*, 1019-1044.
96. Wang, J.; Cai, Q.; Li, Z.-L.; Zhao, H.-K.; Luo, R., Achieving energy conservation in Poisson–Boltzmann molecular dynamics: Accuracy and precision with finite-difference algorithms. *Chemical physics letters* **2009,** *468* (4-6), 112-118.
97. Liu, X.; Wang, C.; Wang, J.; Li, Z.; Zhao, H.; Luo, R., Exploring a charge-central strategy in the solution of Poisson's equation for biomolecular applications. *Physical Chemistry Chemical Physics* **2013**.
98. Wang, C.; Wang, J.; Cai, Q.; Li, Z.; Zhao, H.-K.; Luo, R., Exploring accurate Poisson–Boltzmann methods for biomolecular simulations. *Comput. Theor. Chem.* **2013,** *1024*, 34-44.
99. Zhou, Y. C.; Zhao, S.; Feig, M.; Wei, G. W., High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *J. Comput. Phys.* **2006,** *213* (1), 1-30.
100. Geng, W. H.; Yu, S. N.; Wei, G. W., Treatment of charge singularities in implicit solvent models. *J Chem. Phys.* **2007,** *127* (11).
101. Yu, S. N.; Wei, G. W., Three-dimensional matched interface and boundary (MIB) method for treating geometric singularities. *Journal of Computational Physics* **2007,** *227* (1), 602-632.
102. Zhou, Y. C.; Feig, M.; Wei, G. W., Highly accurate biomolecular electrostatics in continuum dielectric environments. *Journal of Computational Chemistry* **2008,** *29*, 87-97.
103. Zheng, Q.; Yang, S. Y.; Wei, G. W., Biomolecular surface construction by PDE transform. *Int J Numer Meth Bio* **2012,** *28* (3), 291-316.
104. Boschitsch, A. H.; Fenley, M. O., Hybrid boundary element and finite difference method for solving the nonlinear Poisson–Boltzmann equation. *Journal of computational chemistry* **2004,** *25* (7), 935-955.
105. Boschitsch, A. H.; Fenley, M. O., A fast and robust Poisson–Boltzmann solver based on adaptive Cartesian grids. *Journal of chemical theory and computation* **2011,** *7* (5), 1524-1540.
106. Li, L.; Li, C.; Zhang, Z.; Alexov, E., On the dielectric "constant" of proteins: smooth dielectric function for macromolecular modeling and its implementation in DelPhi. *Journal of chemical theory and computation* **2013,** *9* (4), 2126-2136.
107. Chakravorty, A.; Jia, Z.; Peng, Y.; Tajielyato, N.; Wang, L.; Alexov, E., Gaussian-Based Smooth Dielectric Function: A Surface-Free Approach for Modeling Macromolecular Binding in Solvents. *Frontiers in molecular biosciences* **2018,** *5*, 25.
108. Dzubiella, J.; Swanson, J. M. J.; McCammon, J. A., Coupling nonpolar and polar solvation free energies in implicit solvent models. *J Chem. Phys.* **2006,** *124* (8).
109. Dzubiella, J.; Swanson, J. M. J.; McCammon, J. A., Coupling hydrophobicity, dispersion, and electrostatics in continuum solvent models. *Phys. Rev. Lett.* **2006,** *96* (8).
110. Cheng, L. T.; Dzubiella, J.; McCammon, J. A.; Li, B., Application of the level-set method to the implicit solvation of nonpolar molecules. *J Chem. Phys.* **2007,** *127* (8).
111. Chen, Z.; Zhao, S.; Chun, J.; Thomas, D. G.; Baker, N. A.; Bates, P. W.; Wei, G. W., Variational approach for nonpolar solvation analysis. *J Chem. Phys.* **2012,** *137* (8).
112. Thomas, D. G.; Chun, J.; Chen, Z.; Wei, G. W.; Baker, N. A., Parameterization of a geometric flow implicit solvation model. *J Comput. Chem.* **2013,** *34* (8), 687-695.
113. Xiao, L.; Cai, Q.; Li, Z.; Zhao, H.; Luo, R., A multi-scale method for dynamics simulation in continuum solvent models. I: Finite-difference algorithm for Navier–Stokes equation. *Chemical Physics Letters* **2014,** *616-617* (Supplement C), 67-74.
114. Li, Z.; Xiao, L.; Cai, Q.; Zhao, H.; Luo, R., A semi-implicit augmented IIM for Navier–Stokes equations with open, traction, or free boundary conditions. *Journal of Computational Physics* **2015,** *297* (Supplement C), 182-193.

115. Xiao, L.; Luo, R., Exploring a multi-scale method for molecular simulation in continuum solvent model: Explicit simulation of continuum solvent as an incompressible fluid. *The Journal of Chemical Physics* **2017,** *147* (21), 214112.
116. Wang, J.; Cai, Q.; Xiang, Y.; Luo, R., Reducing Grid Dependence in Finite-Difference Poisson-Boltzmann Calculations. *J. Chem. Theory Comput.* **2012,** *8* (8), 2741-2751.
117. Botello-Smith, W. M.; Liu, X. P.; Cai, Q.; Li, Z. L.; Zhao, H. K.; Luo, R., Numerical Poisson-Boltzmann model for continuum membrane systems. *Chem. Phys. Lett.* **2013,** *555*, 274-281.
118. Wang, J.; Cai, Q.; Li, Z.-L.; Zhao, H.-K.; Luo, R., Achieving energy conservation in Poisson-Boltzmann molecular dynamics: Accuracy and precision with finite-difference algorithms. *Chem. Phys. Lett.* **2009,** *468* (4-6), 112-118.
119. Li, Z.; Ito, K., *The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains*. Siam: 2006; Vol. 33.
120. Richards, F. M., AREAS, VOLUMES, PACKING, AND PROTEIN STRUCTURE. *Annu. Rev. Biophys. Bioeng.* **1977,** *6* (1), 151-176.
121. Connolly, M. L., Solvent-accessible surfaces of proteins and nucleic acids. *Science* **1983,** *221* (4612), 709-713.
122. Connolly, M. L., Analytical molecular surface calculation. *Journal of applied crystallography* **1983,** *16* (5), 548-558.
123. Gilson, M. K.; Sharp, K. A.; Honig, B. H., Calculating the Electrostatic Potential of Molecules in Solution - Method and Error Assessment. *J. Comput. Chem.* **1988,** *9* (4), 327-335.
124. Rocchia, W.; Sridharan, S.; Nicholls, A.; Alexov, E.; Chiabrera, A.; Honig, B., 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. *J. Comput. Chem.* **2002,** *23* (1), 128-137.
125. Swanson, J. M. J.; Mongan, J.; McCammon, J. A., Limitations of atom-centered dielectric functions in implicit solvent models. *J. Phys. Chem. B* **2005,** *109* (31), 14769-14772.
126. Tan, C.; Yang, L.; Luo, R., How well does Poisson-Boltzmann implicit solvent agree with explicit solvent? A quantitative analysis. *J. Phys. Chem. B* **2006,** *110* (37), 18680-18687.
127. Wang, J.; Tan, C.; Chanco, E.; Luo, R., Quantitative analysis of Poisson–Boltzmann implicit solvent in molecular dynamics. *Physical Chemistry Chemical Physics* **2010,** *12* (5), 1194-1202.
128. Zauhar, R. J.; Morgan, R. S., Computing the Electric-Potential of Biomolecules - Application of a New Method of Molecular-Surface Triangulation. *J. Comput. Chem.* **1990,** *11* (5), 603-622.
129. You, T.; Bashford, D., An analytical algorithm for the rapid determination of the solvent accessibility of points in a three-dimensional lattice around a solute molecule. *Journal of Computational Chemistry* **1995,** *16* (6), 743-757.
130. Eisenhaber, F.; Argos, P., Improved strategy in analytic surface calculation for molecular systems: Handling of singularities and computational efficiency. *Journal of Computational Chemistry* **1993,** *14* (11), 1272-1280.
131. Varshney, A.; Brooks, F. P.; Wright, W. V., Computing smooth molecular surfaces. *IEEE Computer Graphics and Applications* **1994,** *14* (5), 19-25.
132. Edelsbrunner, H.; Mücke, E. P., Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)* **1994,** *13* (1), 43-72.
133. Totrov, M.; Abagyan, R., The contour-buildup algorithm to calculate the analytical molecular surface. *Journal of structural biology* **1996,** *116* (1), 138-143.
134. Sanner, M. F.; Olson, A. J.; Spehner, J. C., Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers* **1996,** *38* (3), 305-320.
135. Krone, M.; Bidmon, K.; Ertl, T., Interactive visualization of molecular surface dynamics. *IEEE transactions on visualization and computer graphics* **2009,** *15* (6), 1391-1398.

136. Lindow, N.; Baum, D.; Prohaska, S.; Hege, H. C. In *Accelerated visualization of dynamic molecular surfaces*, Computer Graphics Forum, Wiley Online Library: 2010; pp 943-952.

137. Krone, M.; Grottel, S.; Ertl, T. In *Parallel contour-buildup algorithm for the molecular surface*, 2011 IEEE Symposium on Biological Data Visualization (BioVis). IEEE: 2011; pp 17-22.

138. Parulek, J.; Viola, I. In *Implicit representation of molecular surfaces*, 2012 IEEE Pacific Visualization Symposium, IEEE: 2012; pp 217-224.

139. Jurčík, A.; Parulek, J.; Sochor, J.; Kozlikova, B. In *Accelerated visualization of transparent molecular surfaces in molecular dynamics*, 2016 IEEE Pacific Visualization Symposium (PacificVis), IEEE: 2016; pp 112-119.

140. Kozlíková, B.; Krone, M.; Falk, M.; Lindow, N.; Baaden, M.; Baum, D.; Viola, I.; Parulek, J.; Hege, H. C. In *Visualization of biomolecular structures: State of the art revisited*, Computer Graphics Forum, Wiley Online Library: 2017; pp 178-204.

141. Grant, J. A.; Pickup, B. T., A Gaussian Description of Molecular Shape. *J. Phys. Chem.* **1995,** *99* (11), 3503-3510.

142. Grant, J. A.; Pickup, B. T.; Nicholls, A., A Smooth Permittivity Function for Poisson–Boltzmann Solvation Methods. *J. Comput. Chem.* **2001,** *22*, 608.

143. Ye, X.; Wang, J.; Luo, R., A Revised Density Function for Molecular Surface Calculation in Continuum Solvent Models. *J. Chem. Theory Comput.* **2010,** *6* (4), 1157-1169.

144. Chakravorty, A.; Jia, Z.; Peng, Y.; Tajielyato, N.; Wang, L.; Alexov, E., Gaussian-Based Smooth Dielectric Function: A Surface-Free Approach for Modeling Macromolecular Binding in Solvents. *Front. Mol. Biosci.* **2018,** *5* (25).

145. Li, C.; Jia, Z.; Chakravorty, A.; Pahari, S.; Peng, Y.; Basu, S.; Koirala, M.; Panday, S. K.; Petukh, M.; Li, L.; Alexov, E., DelPhi Suite: New Developments and Review of Functionalities. *J. Comput. Chem.* **2019,** *40* (28), 2502-2508.

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

147. Chern, I.-L.; Liu, J.-G.; Wang, W.-C., Accurate Evaluation of Electrostatics for Macromolecules in Solution. *Methods and Applications of Analysis* **2003,** *10*, 309-328.

148. Rocchia, W.; Sridharan, S.; Nicholls, A.; Alexov, E.; Chiabrera, A.; Honig, B., 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* **2002,** *23* (1), 128-137.

149. Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R.; Merz, K. M.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A., A 2ND GENERATION FORCE-FIELD FOR THE SIMULATION OF PROTEINS, NUCLEIC-ACIDS, AND ORGANIC-MOLECULES. *J. Am. Chem. Soc.* **1995,** *117* (19), 5179-5197.

150. Case, D. A.; Brozell, S. R.; Cerutti, D. S.; Cheatham, T. E., III; Cruzeiro, V. W. D.; Darden, T. A.; Duke, R. E.; Ghoreishi, D.; Gohlke, H.; Goetz, A. W.; Greene, D.; Harris, R.; Homeyer, N.; Izadi, S.; Kovalenko, A.; Lee, T. S.; LeGrand, S.; Li, P.; Lin, C.; Liu, J.; Luchko, T.; Luo, R.; Mermelstein, D. J.; Merz, K. M.; Miao, Y.; Monard, G.; Nguyen, H.; Omelyan, I.; Onufriev, A.; Pan, F.; Qi, R.; Roe, D. R.; Roitberg, A.; Sagui, C.; Schott-Verdugo, S.; Shen, J.; Simmerling, C. L.; Swails, J.; Walker, R. C.; Wang, J.; Wei, H.; Wolf, R. M.; Wu, X.; Xiao, L.; York, D. M.; Kollman, P. A., AMBER 2018. University of California, San Francisco., 2018.

151. Nvidia NVIDIA CUDA Sparse Matrix library. https://developer.nvidia.com/cusparse (accessed October 1st, 2016).

152. Qi, R.; Botello-Smith, W. M.; Luo, R., Acceleration of Linear Finite-Difference Poisson–Boltzmann Methods on Graphics Processing Units. *J. Chem. Theory Comput.* **2017,** *13* (7), 3378-3387.

153. Wang, C.; Xiao, L.; Luo, R., Numerical interpretation of molecular surface field in dielectric modeling of solvation. *Journal of computational chemistry* **2017,** *38* (14), 1057-1070.

154. Nguyen, D. D.; Wang, B.; Wei, G. W., Accurate, robust, and reliable calculations of Poisson–Boltzmann binding energies. *Journal of computational chemistry* **2017,** *38* (13), 941-948.
155. Wang, C.; Nguyen, P. H.; Pham, K.; Huynh, D.; Le, T. B. N.; Wang, H.; Ren, P.; Luo, R., Calculating protein–ligand binding affinities with MMPBSA: Method and error analysis. *Journal of computational chemistry* **2016,** *37* (27), 2436-2446.
156. Wei, H.; Luo, R.; Qi, R., An efficient second-order poisson–boltzmann method. *Journal of computational chemistry* **2019**.
157. Botello-Smith, W. M.; Luo, R., Applications of MMPBSA to Membrane Proteins I: Efficient Numerical Solutions of Periodic Poisson–Boltzmann Equation. *J. Chem. Inf. Model.* **2015,** *55* (10), 2187-2199.
158. Wang, C. H.; Wang, J.; Cai, Q.; Li, Z. L.; Zhao, H. K.; Luo, R., Exploring accurate Poisson-Boltzmann methods for biomolecular simulations. *Comput. Theor. Chem.* **2013,** *1024*, 34-44.
159. Wei, H.; Luo, R.; Qi, R., An efficient second-order poisson–boltzmann method. *J. Comput. Chem.* **2019,** *40* (12), 1257-1269.
160. Cai, Q.; Ye, X.; Wang, J.; Luo, R., On-the-Fly Numerical Surface Integration for Finite-Difference Poisson-Boltzmann Methods. *J. Chem. Theory Comput.* **2011,** *7* (11), 3608-3619.
161. Cai, Q.; Ye, X.; Luo, R., Dielectric pressure in continuum electrostatic solvation of biomolecules. *Physical Chemistry Chemical Physics* **2012,** *14* (45), 15917-15925.
162. Xiao, L.; Cai, Q.; Ye, X.; Wang, J.; Luo, R., Electrostatic forces in the Poisson-Boltzmann systems. *Journal of Chemical Physics* **2013,** *139* (9).
163. Xiao, L.; Wang, C. H.; Ye, X.; Luo, R., Charge Central Interpretation of the Full Nonlinear PB Equation: Implications for Accurate and Scalable Modeling of Solvation Interactions. *Journal of Physical Chemistry B* **2016,** *120* (33), 8707-8721.
164. Wang, C. H.; Xiao, L.; Luo, R., Numerical interpretation of molecular surface field in dielectric modeling of solvation. *Journal of Computational Chemistry* **2017,** *38* (14), 1057-1070.
165. Qi, R.; Luo, R., Robustness and Efficiency of Poisson–Boltzmann Modeling on Graphics Processing Units. *J. Chem. Inf. Model.* **2019,** *59* (1), 409-420.
166. Wei, H.; Luo, A.; Qiu, T.; Luo, R.; Qi, R., Improved Poisson–Boltzmann Methods for High-Performance Computing. *Journal of chemical theory and computation* **2019,** *15* (11), 6190-6202.
167. Osher, S.; Fedkiw, R., *The Level Set Methods and Dynamic Implicit Surfaces*. 2004; Vol. 57, p xiv+273.
168. Hornik, K., Approximation capabilities of multilayer feedforward networks. *Neural networks* **1991,** *4* (2), 251-257.
169. Shrestha, A.; Mahmood, A., Review of deep learning algorithms and architectures. *IEEE Access* **2019,** *7*, 53040-53065.
170. Emmert-Streib, F.; Yang, Z.; Feng, H.; Tripathi, S.; Dehmer, M., An Introductory Review of Deep Learning for Prediction Models With Big Data. *Front. Artif. Intell.* **2020,** *3* (4).
171. Ching, T.; Himmelstein, D. S.; Beaulieu-Jones, B. K.; Kalinin, A. A.; Do, B. T.; Way, G. P.; Ferrero, E.; Agapow, P.-M.; Zietz, M.; Hoffman, M. M., Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface* **2018,** *15* (141), 20170387.
172. Case, D. A.; Belfon, K.; Ben-Shalom, I. Y.; Brozell, S. R.; Cerutti, D. S.; T.E. Cheatham, I.; Cruzeiro, V. W. D.; Darden, T. A.; Duke, R. E.; Giambasu, G.; Gilson, M. K.; Gohlke, H.; Goetz, A. W.; Harris, R.; Izadi, S.; Izmailov, S. A.; Kasavajhala, K.; Kovalenko, A.; Krasny, R.; Kurtzman, T.; Lee, T. S.; LeGrand, S.; Li, P.; Lin, C.; Liu, J.; Luchko, T.; Luo, R.; Man, V.; Merz, K. M.; Miao, Y.; Mikhailovskii, O.; Monard, G.; Nguyen, H.; Onufriev, A.; Pan, F.; Pantano, S.; Qi, R.; Roe, D. R.; Roitberg, A.; Sagui, C.; Schott-Verdugo, S.; Shen, J.; Simmerling, C. L.; Skrynnikov, N. R.; Smith, J.; Swails, J.; Walker, R. C.; Wang, J.; Wilson, L.; Wolf, R. M.; Wu, X.; Xiong, Y.; Xue, Y.; York, D. M.; Kollman, P. A. *Amber 2020*; University of California, San Francisco: 2020.
173. Chollet, *Keras*. Github: 2015.

174. Cai, Q.; Hsieh, M.-J.; Wang, J.; Luo, R., Performance of Nonlinear Finite-Difference Poisson-Boltzmann Solvers. *J. Chem. Theory Comput.* **2010,** *6* (1), 203-211.
175. Elking, D.; Darden, T.; Woods, R. J., Gaussian induced dipole polarization model. *Journal of Computational Chemistry* **2007,** *28* (7), 1261-1274.
176. Elking, D. Polarizable force fields. University of Georgia, 2007.
177. Böttcher, C. J. F.; van Belle, O. C.; Bordewijk, P.; Rip, A., *Theory of electric polarization*. Elsevier Science Ltd: 1978; Vol. 2.
178. Ewald, P. P., Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Annalen der Physik* **1921,** *369* (3), 253-287.
179. Sagui, C.; Pedersen, L. G.; Darden, T. A., Towards an accurate representation of electrostatics in classical force fields: Efficient implementation of multipolar interactions in biomolecular simulations. *The Journal of chemical physics* **2004,** *120* (1), 73-87.
180. Toukmaji, A.; Sagui, C.; Board, J.; Darden, T., Efficient particle-mesh Ewald based approach to fixed and induced dipolar interactions. *The Journal of chemical physics* **2000,** *113* (24), 10913-10927.
181. Darden, T.; York, D.; Pedersen, L., Particle mesh Ewald: An N· log (N) method for Ewald sums in large systems. *The Journal of chemical physics* **1993,** *98* (12), 10089-10092.
182. Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G., A smooth particle mesh Ewald method. *The Journal of chemical physics* **1995,** *103* (19), 8577-8593.
183. Wei, H.; Qi, R.; Wang, J.; Cieplak, P.; Duan, Y.; Luo, R., Efficient formulation of polarizable Gaussian multipole electrostatics for biomolecular simulations. *The Journal of Chemical Physics* **2020,** *153* (11), 114116.
184. Wang, J.; Cieplak, P.; Luo, R.; Duan, Y., Development of Polarizable Gaussian Model for Molecular Mechanical Calculations I: Atomic Polarizability Parameterization To Reproduce ab Initio Anisotropy. *Journal of Chemical Theory and Computation* **2019,** *15* (2), 1146-1158.
185. Åqvist, J.; Wennerström, P.; Nervall, M.; Bjelic, S.; Brandsdal, B. O., Molecular dynamics simulations of water and biomolecules with a Monte Carlo constant pressure algorithm. *Chemical physics letters* **2004,** *384* (4-6), 288-294.

Appendix A

A.1 Tensor Format of Boys Serial

In this study, Boys functions up to rank three were used and are listed below as reference. Higher ranked tensors and Boys functions can be found in literatures.[44, 49]

Boys functions up to rank three are

$$B_0(x) = \frac{\mathrm{erf}(x)}{x}$$

$$B_1(x) = \frac{\mathrm{erf}(x)}{x^3} - \frac{2}{\sqrt{\pi}} e^{-x^2} \frac{1}{x^2}$$

$$B_2(x) = \frac{3\mathrm{erf}(x)}{x^5} - \frac{2}{\sqrt{\pi}} e^{-x^2} \frac{1}{x^4} (3 + 2x^2)$$

$$B_3(x) = \frac{15\mathrm{erf}(x)}{x^7} - \frac{2}{\sqrt{\pi}} e^{-x^2} \frac{1}{x^6} (15 + 10x^2 + 4x^4)$$

The associated tensors are

$$\nabla \frac{\mathrm{erf}(\beta R)}{R} = -\vec{R}\beta^3 B_1(\beta R)$$

$$\nabla\nabla \frac{\mathrm{erf}(\beta R)}{R} = \hat{x}_p \hat{x}_q (R_p R_q \beta^5 B_2(\beta R) - \delta_{pq} \beta^3 B_1(\beta R))$$

$$\nabla\nabla\nabla \frac{\mathrm{erf}(\beta R)}{R} = \hat{x}_p \hat{x}_q \hat{x}_r (\left(\delta_{pq} R_r + \delta_{pr} R_q + \delta_{rq} R_p\right) \beta^5 B_2(\beta R) - R_p R_q R_r \beta^7 B_3(\beta R))$$

A.2 Force derivation

We precede in two steps, covalent-covalent interactions and induced interactions as shown in section 2.3. First, we consider interaction energies due to covalent multipoles interacting with covalent multipoles.

The system can be split into two groups of atoms, bonded atoms and nonbonded atoms. Bonded group are those atoms bonded to the atom to be considered (including itself), and the nonbonded group are the rest. In the bonded group, the atoms can be further split into two subgroups: the atom that is currently under consideration, termed the bonded-moving atom below; the other atoms in the bonded group are termed bonded-non-moving atoms. A total of three groups of atoms can be classified.

Thus, we can rewrite the covalent-covalent interaction energy as the following four parts.

1) Nonbonded atoms interacting with bonded-non-moving atoms,

$$U = \sum_{i}^{bonded-non-moving} \sum_{j}^{nonbonded} (q_i + \vec{\mu}_i \cdot \nabla_i)(q_j + \vec{\mu}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta_{ij} R_{ij})}{R_{ij}}$$

2) Nonbonded atoms interacting with bonded-moving atom *i*,

$$U = \sum_{j}^{nonbonded} (q_i + \vec{\mu}_i \cdot \nabla_i)(q_j + \vec{\mu}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta_{ij} R_{ij})}{R_{ij}}$$

3) Bonded-non-moving atoms interacting with bonded-non-moving atoms,

$$U = \frac{1}{2} \sum_{i}^{bonded-non-moving} \sum_{j \neq i}^{bonded-non-moving} (q_i + \vec{\mu}_i \cdot \nabla_i)(q_j + \vec{\mu}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta_{ij} R_{ij})}{R_{ij}}$$

4) Bonded-non-moving atoms interacting with bonded-moving atom *i*,

$$U = \sum_{j}^{bonded-non-moving} (q_i + \vec{\mu}_i \cdot \nabla_i)(q_j + \vec{\mu}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta_{ij} R_{ij})}{R_{ij}}$$

Apparently, there should be a fifth part of interaction energy, nonbonded atoms interacting with nonbonded atoms. However, this part of energy does not change in the force calculation, so we omit its expression here. Of course, atom *i*'s self-interaction is also ignored as discussed in the text.

Next force on bonded-moving atom (*i*) can be derived as the negative gradient of the above energy terms. When computing the gradient, it is worth pointing out that nothing varies on the nonbonded atoms, only the dipole directions vary on the bonded-non-moving atoms, and both dipole directions and positions of the bonded-moving atoms vary. The above four energy parts thus lead to the following four force components, respectively,

$$(\vec{F}_i)_1 = -\sum_{k}^{bonded-non-moving}\sum_{j}^{nonbonded}\nabla_i(\vec{\mu}_k)\cdot\nabla_k(q_j+\vec{\mu}_j\cdot\nabla_j)\frac{\operatorname{erf}(\beta_{kj}R_{kj})}{R_{kj}}$$

$$(\vec{F}_i)_2 = -\sum_{j}^{nonbonded}\nabla_i(\vec{\mu}_i)\cdot\nabla_i(q_j+\vec{\mu}_j\cdot\nabla_j)\frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

$$-\sum_{j}^{nonbonded}(q_i+\vec{\mu}_i\cdot\nabla_i)(q_j+\vec{\mu}_j\cdot\nabla_j)\nabla_i\frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

$$(\vec{F}_i)_3 = -\sum_{k}^{bonded-non-moving}\sum_{j\neq k}^{bonded-non-moving}\nabla_i(\vec{\mu}_k)\cdot\nabla_k(q_j+\vec{\mu}_j\cdot\nabla_j)\frac{\operatorname{erf}(\beta_{kj}R_{kj})}{R_{kj}}$$

$$(\vec{F}_i)_4 = -\sum_{j}^{bonded-non-moving}\nabla_i(\vec{\mu}_i)\cdot\nabla_i(q_j+\vec{\mu}_j\cdot\nabla_j)\frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

$$-\sum_{j}^{bonded-non-moving}(q_i+\vec{\mu}_i\cdot\nabla_i)\nabla_i(\vec{\mu}_j)\cdot\nabla_j\frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

$$-\sum_{j}^{bonded-non-moving}(q_i+\vec{\mu}_i\cdot\nabla_i)(q_j+\vec{\mu}_j\cdot\nabla_j)\nabla_i\frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

Summing up all four components, the final force expression is,

$$\vec{F}_i = -\sum_{k}^{n}\sum_{j\neq k}^{N}\nabla_i(\vec{\mu}_k)\cdot\nabla_k(q_j+\vec{\mu}_j\cdot\nabla_j)\frac{\operatorname{erf}(\beta_{kj}R_{kj})}{R_{kj}}$$

$$-\sum_{j\neq i}^{N}(q_i+\vec{\mu}_i\cdot\nabla_i)(q_j+\vec{\mu}_j\cdot\nabla_j)\nabla_i\frac{\operatorname{erf}(\beta_{ij}R_{ij})}{R_{ij}}$$

Here, *n* and *N* follows the same notation as section 2.3, number of atoms in the bonded group and the system, respectively.

Second, we consider energies caused by induced dipoles. As stated before, the induced energy contains three parts, induced dipoles interacting with covalent multipoles, induced dipoles interacting with induced dipoles, and induced dipole self-energy $\frac{1}{2}\vec{p}\cdot\vec{E}$. From section 2.3, we know that the total induced energy is,

$$\frac{1}{2}\sum_{i}^{N} -\vec{p}_i \cdot \vec{E}_i^0$$

where $\vec{E}_i^0$ is the electric field on atom *i* only by covalent multipoles.

The induced dipoles are determined by the total electric field,

$$\vec{p}_i = \alpha_i \vec{E}_i = \alpha_i(\vec{E}_i^0 - \sum_{j\neq i}^{N} \vec{\vec{T}}_{ij} \cdot \vec{p}_j)$$

$$\vec{\vec{T}}_{ij} = \nabla_i \nabla_j \frac{\operatorname{erf}(\beta_{ij} R_{ij})}{R_{ij}}$$

Changing the above expressions into the component format, and applying the Einstein's index notation, we obtain

$$p_i^s = \alpha_i(E_i^{0,s} - T_{ij}^{st} p_j^t)$$

Rearrangement leads to

$$\left(\frac{1}{\alpha_j}\delta_{ij}^{st} + T_{ij}^{st}\right) p_j^t = A_{ij}^{st} p_j^t = E_i^{0,s}$$

If we assume A_{ij}^{st} is inversible, and its inverse matrix is ${A^{-1}}_{ij}^{st}$, we have

$$p_i^s = {A^{-1}}_{ij}^{st} E_j^{0,t}$$

${A^{-1}}_{ij}^{st}$ is a $3N \times 3N$ matrix, symmetrical for both atom index and component index,

$$A^{-1}{}_{ij}^{st} = A^{-1}{}_{ij}^{ts} = A^{-1}{}_{ji}^{st}$$

The gradient of A^{-1} is,

$$\frac{\partial A^{-1}{}_{ij}^{st}}{\partial x_k^w} = -A^{-1}{}_{ii'}^{ss'} \frac{\partial A\,{}_{i'j'}^{s't'}}{\partial x_k^w} A^{-1}{}_{i'j}^{s't} = -A^{-1}{}_{ii'}^{ss'} \frac{\partial T\,{}_{i'j'}^{s't'}}{\partial x_k^w} A^{-1}{}_{j'j}^{t't}$$

where w refers to coordinate indices (x^1, x^2, x^3). It is obvious that i' or j' has to be equal to k for T to have a nonzero value. We have

$$\frac{\partial T\,{}_{i'j'}^{s't'}}{\partial x_k^w} = \frac{\partial T\,{}_{kj'}^{s't'}}{\partial x_k^w} + \frac{\partial T\,{}_{i'k}^{s't'}}{\partial x_k^w}$$

Based on above relations, the force expressed as the negative gradient of the induced energy is

$$F_i^w = \frac{\partial}{\partial x_i^w}\left(\frac{1}{2}\vec{p}_j \cdot \vec{E}_j^0\right) = \frac{\partial}{\partial x_i^w}\left(\frac{1}{2}A^{-1}{}_{jk}^{st} E_j^{0,s} E_k^{0,t}\right) = \frac{1}{2}\frac{\partial A^{-1}{}_{jk}^{st}}{\partial x_i^w} E_j^{0,s} E_k^{0,t} + A^{-1}{}_{jk}^{st} E_j^{0,s} \frac{\partial E_k^{0,t}}{\partial x_i^w}$$

$$= -\frac{1}{2}A^{-1}{}_{jj'}^{ss'} \frac{\partial T\,{}_{j'k'}^{s't'}}{\partial x_i^w} A^{-1}{}_{k'k}^{t't} E_j^{0,s} E_k^{0,t} + p_k^t \frac{\partial E_k^{0,t}}{\partial x_i^w} = -\frac{1}{2} p_{j'}^{s'} \frac{\partial T\,{}_{j'k'}^{s't'}}{\partial x_i^w} p_{k'}^{t'} + p_k^t \frac{\partial E_k^{0,t}}{\partial x_i^w}$$

$$= -p_i^{s'} \frac{\partial T\,{}_{ik'}^{s't'}}{\partial x_i^w} p_{k'}^{t'} + p_k^t \frac{\partial E_k^{0,t}}{\partial x_i^w}$$

Rewriting the above component format into the vector/tensor format, we have

$$\vec{F}_i = -\sum_{j\neq i}^{N} \vec{p}_i \cdot \nabla_i (\vec{p}_j \cdot \nabla_j) \nabla_i \frac{\operatorname{erf}(\beta_{ij} R_{ij})}{R_{ij}} + \sum_{j}^{N} \nabla_i (\vec{E}_j^0) \cdot \vec{p}_j$$

The next step is to evaluate $\nabla_i(\vec{E}_j^0)$. Following the similar strategy used in covalent-covalent interactions, we split the system into two groups: non-moving atoms and moving atom (i.e. atom i).

When computing the derivative of field on a non-moving atom *j*, it is worth pointing out that other nonbonded non-moving atoms are not influenced by the virtual displacement of atom *i*, so only bonded non-moving atoms are considered below.

$$\nabla_i\left(\vec{E}_j^0\right) = \nabla_i\left(\sum_{k\neq j}^{bonded-non-moving} -\nabla_j(q_k + \vec{\mu}_k \cdot \nabla_k)\frac{\mathrm{erf}\left(\beta_{kj}R_{kj}\right)}{R_{kj}} - \nabla_j(q_i + \vec{\mu}_i \cdot \nabla_i)\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}}\right)$$

$$= \sum_{k\neq j}^{bonded-non-moving} -\nabla_i(\vec{\mu}_k)\cdot\nabla_k\nabla_j\frac{\mathrm{erf}\left(\beta_{kj}R_{kj}\right)}{R_{kj}} - \nabla_i(\vec{\mu}_i)\cdot\nabla_i\nabla_j\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}}$$

$$- (q_i + \vec{\mu}_i \cdot \nabla_i)\nabla_i\nabla_j\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}}$$

$$= \sum_{k\neq j}^{n} -\nabla_i(\vec{\mu}_k)\cdot\nabla_k\nabla_j\frac{\mathrm{erf}\left(\beta_{kj}R_{kj}\right)}{R_{kj}} - (q_i + \vec{\mu}_i \cdot \nabla_i)\nabla_i\nabla_j\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}}$$

Here *n* represents the number of atoms in the bonded group, including atom *i*.

Next we compute the derivative of field on the moving atom, i.e. atom *i*, as follows

$$\nabla_i\left(\vec{E}_i^0\right) = \nabla_i\left(\sum_{j\neq i}^{n} -\nabla_i\left(q_j + \vec{\mu}_j \cdot \nabla_j\right)\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}} - \sum_{j}^{nonbonded} \nabla_i\left(q_j + \vec{\mu}_j \cdot \nabla_j\right)\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}}\right)$$

$$= \sum_{j\neq i}^{n} -\nabla_i\left(\vec{\mu}_j\right)\cdot\nabla_j\nabla_i\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}} - \sum_{j\neq i}^{n}\left(q_j + \vec{\mu}_j \cdot \nabla_j\right)\nabla_i\nabla_i\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}}$$

$$- \sum_{j\neq i}^{nonbonded}\left(q_j + \vec{\mu}_j \cdot \nabla_j\right)\nabla_i\nabla_i\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}}$$

$$= -\sum_{j\neq i}^{n} \nabla_i\left(\vec{\mu}_j\right)\cdot\nabla_j\nabla_i\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}} - \sum_{j\neq i}^{N}\left(q_j + \vec{\mu}_j \cdot \nabla_j\right)\nabla_i\nabla_i\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)}{R_{ij}}$$

Here *N* represents the number of all atoms in the system.

Appendix B

B.1. Several Useful Derivatives Against *h*

$$\frac{\partial \vec{R}^{\alpha}}{\partial h_{\mu\nu}} = \delta_{\alpha\mu}\vec{S}^{\nu}$$

$$\frac{\partial V}{\partial h_{\mu\nu}} = V(h^{-1})^{T}_{\mu\nu}$$

$$\frac{\partial (h^{-1})_{\alpha\beta}}{\partial h_{\mu\nu}} = -(h^{-1})_{\alpha\mu}(h^{-1})_{\nu\beta}$$

$$\frac{\partial \vec{m}^{\alpha}}{\partial h_{\mu\nu}} = -\vec{m}^{\mu}(h^{-1})_{\nu\alpha}$$

B.2. Virial Derivation

From previous discussions in the main text,

$$\frac{\partial U}{\partial h_{\mu\nu}} = \frac{\partial U_{rec}}{\partial h_{\mu\nu}} + \frac{\partial U_{dir}}{\partial h_{\mu\nu}} + \frac{\partial U_{self}}{\partial h_{\mu\nu}}$$

Since for the direct part and the self-part the only variable that depends on h is the atom coordinates $\vec{R}$, we have the following:

$$\frac{\partial U_{dir}}{\partial h_{\mu\nu}} = \sum_i^N \frac{\partial \vec{\mu}_i}{\partial h_{\mu\nu}} \cdot \nabla_i \sum_{j\neq i}^{\infty} \left(q_j + \vec{\mu}_j \cdot \nabla_j\right) \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right) - \operatorname{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}$$

$$+ \sum_i^N \vec{p}_i \cdot \nabla_i \sum_{j\neq i}^{\infty} \frac{\partial \vec{\mu}_j}{\partial h_{\mu\nu}} \cdot \nabla_j \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right) - \operatorname{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}$$

$$+ \frac{1}{2} \sum_i^N \left(q_i + \vec{\mu}_i \cdot \nabla_i\right) \sum_{j\neq i}^{\infty} \left(q_j + \vec{\mu}_j \cdot \nabla_j\right) \frac{\partial \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right) - \operatorname{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}}{\partial h_{\mu\nu}}$$

$$+ \frac{1}{2} \sum_i^N \vec{p}_i \cdot \nabla_i \sum_{j\neq i}^{\infty} \vec{p}_j \cdot \nabla_j \frac{\partial \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right) - \operatorname{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}}{\partial h_{\mu\nu}}$$

$$+ \sum_i^N \vec{p}_i \cdot \nabla_i \sum_{j\neq i}^{\infty} \left(q_j + \vec{\mu}_j \cdot \nabla_j\right) \frac{\partial \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right) - \operatorname{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}}{\partial h_{\mu\nu}}$$

$$= \sum_i^N \frac{\partial \vec{\mu}_i}{\partial h_{\mu\nu}} \cdot \nabla_i \sum_{j\neq i}^{\infty} \left[q_j + (\vec{\mu}_j + \vec{p}_j) \cdot \nabla_j\right] \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right) - \operatorname{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}$$

$$+ \frac{1}{2} \sum_i^N \left[q_i + (\vec{\mu}_i + \vec{p}_i) \cdot \nabla_i\right] \sum_{j\neq i}^{\infty} \left[q_j + (\vec{\mu}_j + \vec{p}_j) \cdot \nabla_j\right] \frac{\partial \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right) - \operatorname{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}}{\partial h_{\mu\nu}}$$

$$= - \sum_i^N \frac{\partial \vec{\mu}_i}{\partial h_{\mu\nu}} \cdot \vec{E}_{dir}$$

$$+ \frac{1}{2} \sum_i^N \left[q_i + (\vec{\mu}_i + \vec{p}_i) \cdot \nabla_i\right] \sum_{j\neq i}^{\infty} \left[q_j + \left(\vec{\mu}_j + \vec{p}_j\right) \cdot \nabla_j\right] \frac{\partial \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right) - \operatorname{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}}{\partial \vec{R}_i}$$

$$\cdot \left(\frac{\partial \vec{R}_i}{\partial h_{\mu\nu}} - \frac{\partial \vec{R}_j}{\partial h_{\mu\nu}}\right)$$

$$= -\sum_{i}^{N}\sum_{k}^{n} u_{ik}\left[\frac{\left(\vec{E}_{dir}\right)_i^{\mu}\vec{S}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|} - \left(\vec{E}_{dir}\right)_i \cdot \vec{R}_{ik}\frac{\vec{R}_{ik}^{\mu}\vec{S}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|^3}\right]$$

$$-\frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{\infty}\{q_i\left(\vec{E}_{dir}\right)_{j\to i}^{\mu}(\vec{S}_i^{\nu} - \vec{S}_j^{\nu}) + \left[(\vec{u}_i + \vec{p}_i)\cdot\left(\vec{\vec{E}}_{dir}\right)_{j\to i}\right]^{\mu}(\vec{S}_i^{\nu} - \vec{S}_j^{\nu})\}$$

Similarly, for the self-part,

$$\frac{\partial U_{self}}{\partial h_{\mu\nu}} = -\sum_{i}^{N}\sum_{k}^{n} u_{ik}\left[\frac{\left(\vec{E}_{self}\right)_i^{\mu}\vec{S}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|} - \left(\vec{E}_{self}\right)_i \cdot \vec{R}_{ik}\frac{\vec{R}_{ik}^{\mu}\vec{S}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|^3}\right]$$

$$-\lim_{\vec{R}_j\to\vec{R}_i}\frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{\infty}\left\{q_i\left(\vec{E}_{self}\right)_{j\to i}^{\mu}(\vec{S}_i^{\nu} - \vec{S}_j^{\nu}) + \left[(\vec{u}_i + \vec{p}_i)\cdot\left(\vec{\vec{E}}_{self}\right)_{j\to i}\right]^{\mu}(\vec{S}_i^{\nu} - \vec{S}_j^{\nu})\right\}$$

$$= -\sum_{i}^{N}\sum_{k}^{n} u_{ik}\left[\frac{\left(\vec{E}_{self}\right)_i^{\mu}\vec{S}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|} - \left(\vec{E}_{self}\right)_i \cdot \vec{R}_{ik}\frac{\vec{R}_{ik}^{\mu}\vec{S}_{ik}^{\nu}}{\left|\vec{R}_{ik}\right|^3}\right]$$

However, the above approach cannot be applied for the reciprocal part of the virial. Thus, we must resort to a brute force calculation,

$$\frac{\partial U_{rec}}{\partial h_{\mu\nu}} = (1) + (2) + (3)$$

(1)

$$= \frac{\partial \sum_i^N \frac{1}{2}(q_i + \vec{\mu}_i\cdot\nabla_i)\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N\left[\left(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j\right)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}}{\partial h_{\mu\nu}}$$

$$(2) = \frac{1}{2} p_{i'}^{\alpha'} \frac{\partial\, \partial_{i'}^{\alpha'} \partial_{j'}^{\beta'} \frac{1}{\pi V} \sum_{\vec{m}\neq 0} \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \exp\left(2\pi i \vec{m}\cdot \vec{R}_{j'}\right) \exp\left(-2\pi i \vec{m}\cdot \vec{R}_{i'}\right)}{\partial h_{\mu\nu}} p_{j'}^{\beta'}$$

(3)

$$= \sum_{i}^{N} \vec{p}_i$$

$$\cdot \frac{\partial \nabla_i \frac{1}{\pi V} \sum_{\vec{m}\neq 0} \{ \frac{\exp\left(-\frac{\pi^2 \vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2} \sum_j^N \left[\left(q_j + 2\pi i \vec{m}\cdot \vec{\mu}_j\right) \exp\left(2\pi i \vec{m}\cdot \vec{R}_j\right) \right] \exp\left(-2\pi i \vec{m}\cdot \vec{R}_i\right) \}}{\partial h_{\mu\nu}}$$

For each part we derive the following expressions:

$$(1) = -\frac{1}{2\pi V}(h^{-1})^{T}_{\mu\nu}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}(q_i - 2\pi i\vec{m}\cdot\vec{\mu}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{m}^{\gamma}\right)\sum_{j}^{N}(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}(q_i - 2\pi i\vec{m}\cdot\vec{\mu}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}\sum_{k}^{n}\left[\left(-2\pi i\vec{m}\cdot\vec{R}_{ik}\mu_{ik}\right)\left(-\frac{\vec{R}^{\mu}_{ik}\vec{S}^{\nu}_{ik}}{\left|\vec{R}_{ik}\right|^3}\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$(2) = -\frac{1}{2\pi V}(h^{-1})^T_{\mu\nu}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N(2\pi i\vec{m}\cdot\vec{p}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i(-2\pi i\vec{m}\right.$$

$$\left.\cdot\,\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^\mu h^{-1}_{\nu\gamma}\vec{m}^\gamma\right)\sum_j^N(2\pi i\vec{m}\right.$$

$$\left.\cdot\,\vec{p}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N(-2\pi i\vec{m}\cdot\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N(2\pi i\vec{m}\right.$$

$$\left.\cdot\,\vec{p}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N\left[(-2\pi i)\left(-\vec{m}^\mu h^{-1}_{\nu\gamma}p_i^\gamma\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$(3) = -\frac{1}{\pi V}(h^{-1})^T_{\mu\nu}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N(-2\pi i\vec{m}\right.$$

$$\left.\cdot\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{2}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^\mu h^{-1}_{\nu\gamma}\vec{m}^\gamma\right)\sum_j^N(q_j + 2\pi i\vec{m}\right.$$

$$\left.\cdot\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N(-2\pi i\vec{m}\cdot\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N(2\pi i\vec{m}\right.$$

$$\left.\cdot\vec{p}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N\sum_k^n\left[\left(-2\pi i\vec{m}\cdot\vec{R}_{ik}\mu_{ik}\right)\left(-\frac{\vec{R}^\mu_{ik}\vec{S}^\nu_{ik}}{\left|\vec{R}_{ik}\right|^3}\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N(q_j + 2\pi i\vec{m}\right.$$

$$\left.\cdot\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N\left[(-2\pi i)\left(-\vec{m}^\mu h^{-1}_{\nu\gamma}p_i^\gamma\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

By adding together expressions for (1), (2) and (3), we arrive at the final result for $\frac{\partial U_{rec}}{\partial h_{\mu\nu}}$, which is shown in before.

B.3. Virial of Rigid Body Systems

As in the derivation presented in the **Appendix** B.2, for the direct part and the self-part, the only variable that depends on h is the atom coordinate $\vec{R}$. However, a major difference is that, for the rigid molecules, the covalent dipole $\vec{u}$ is a constant vector now. Thus, the derivation complexity is greatly reduced,

$$\frac{\partial U_{dir}}{\partial h_{\mu\nu}} = -\frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{\infty}\{q_i\left(\vec{E}_{dir}\right)_{j\rightarrow i}^{\mu}(\vec{S}_{i0}^{\nu} - \vec{S}_{j0}^{\nu}) + \left[(\vec{u}_i + \vec{p}_i)\cdot\left(\vec{\vec{E}}_{dir}\right)_{j\rightarrow i}\right]^{\mu}(\vec{S}_{i0}^{\nu} - \vec{S}_{j0}^{\nu})\}$$

$$\frac{\partial U_{self}}{\partial h_{\mu\nu}} = 0$$

As for the reciprocal part, we follow the procedure presented in the **Appendix** B.2,

$$(1) = -\frac{1}{2\pi V}(h^{-1})^{T}_{\mu\nu}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}(q_i - 2\pi i\vec{m}\right.$$

$$\left.\cdot\,\vec{\mu}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{m}^{\gamma}\right)\sum_{j}^{N}(q_j + 2\pi i\vec{m}\right.$$

$$\left.\cdot\,\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}(q_i - 2\pi i\vec{m}\cdot\vec{\mu}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}[-2\pi i\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{\mu}_j^{\gamma}\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\right.$$

$$-\left(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j\right)2\pi i\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{d}_j^{\gamma}\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)]\sum_{i}^{N}(q_i - 2\pi i\vec{m}$$

$$\left.\cdot\,\vec{\mu}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$(2) = -\frac{1}{2\pi V}(h^{-1})^{T}_{\mu\nu}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}\left(2\pi i\vec{m}\cdot\vec{p}_j\right)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}(-2\pi i\vec{m}\cdot\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{m}^{\gamma}\right)\sum_{j}^{N}\left(2\pi i\vec{m}\cdot\vec{p}_j\right)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}(-2\pi i\vec{m}\cdot\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}\left(2\pi i\vec{m}\cdot\vec{p}_j\right)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}\left[(-2\pi i)\left(-\vec{m}^{\mu}h^{-1}_{\nu\gamma}p_i^{\gamma}\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_{j}^{N}\left(2\pi i\vec{m}\cdot\vec{p}_j\right)(-2\pi i\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{d}_j^{\gamma})\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_{i}^{N}(-2\pi i\vec{m}\cdot\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$(3) = -\frac{1}{\pi V}(h^{-1})^T_{\mu\nu}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N (q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N(-2\pi i\vec{m}\right.$$

$$\left.\cdot\,\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{2}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^\mu h^{-1}_{\nu\gamma}\vec{m}^\gamma\right)\sum_j^N(q_j + 2\pi i\vec{m}\right.$$

$$\left.\cdot\,\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N(-2\pi i\vec{m}\cdot\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N\left[-2\pi i\vec{m}^\mu h^{-1}_{\nu\gamma}\vec{\mu}_j^\gamma\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\right.\right.$$

$$\left.-\left(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j\right)2\pi i\vec{m}^\mu h^{-1}_{\nu\gamma}\vec{d}_j^\gamma\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\right]\sum_i^N(-2\pi i\vec{m}$$

$$\left.\cdot\,\vec{p}_i)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N(q_j + 2\pi i\vec{m}\right.$$

$$\left.\cdot\,\vec{\mu}_j)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\sum_i^N(-2\pi i\vec{m}\cdot\vec{p}_i)(2\pi i\vec{m}^\mu h^{-1}_{\nu\gamma}\vec{d}_i^\gamma)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\right\}$$

By adding final expressions for (1), (2) and (3), we arrive at the result for $\frac{\partial U_{rec}}{\partial h_{\mu\nu}}$,

$$\frac{\partial U_{rec}}{\partial h_{\mu\nu}} = -\frac{1}{2}(h^{-1})^{T}_{\mu\nu} \cdot \sum_{i}^{N}\left[q_i(\phi_{rec})_i - (\vec{\mu}_i + \vec{p}_i)\cdot\left(\vec{E}_{rec}\right)_i\right]$$

$$+\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\left\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\left(-\frac{1+\frac{\pi^2\vec{m}^2}{\beta_0^2}}{\vec{m}^2}\right)\left(-\vec{m}^{\mu}h^{-1}_{\nu\gamma}\vec{m}^{\gamma}\right)S(\vec{m})S(-\vec{m})\right\}$$

$$+\sum_{i}^{N}\left(\vec{E}_{rec}\right)_i^{\mu}h^{-1}_{\nu\gamma}\left(\vec{\mu}_i^{\gamma}+\vec{p}_i^{\gamma}+q_i\vec{d}_i^{\gamma}\right)+\sum_{i}^{N}\left(\vec{\mu}_i^{\gamma}+\vec{p}_i^{\gamma}\right)\left(\vec{\vec{E}}_{rec}\right)_i^{\gamma\mu}h^{-1}_{\nu\xi}\vec{d}_i^{\xi}$$

Finally, adding up all above terms and removing the common factor $-(h^{-1})^T$, the final expression is the one shown before.

B.4. Screening Correction

If certain short-range electrostatic interactions are screened, the pGM system energy can be expressed as,

$$U = \sum_{i}^{N}\frac{1}{2}(q_i + \vec{\mu}_i\cdot\nabla_i)\phi_i^{0*} + \sum_{i}^{N}\frac{1}{2}(\vec{p}_i^{*}\cdot\nabla_i)\phi_i^{0*}$$

Here, ϕ_i^{0*} represents the electrostatic potential on atom *i* that is created by only charges and covalent dipoles that are not screened in the system, and $\vec{p}_i^{*}$ means the induced dipole on atom *i* that is created under the screening condition. Specifically,

$$\phi_i^{0*} = \frac{1}{\pi V}\sum_{\vec{m}\neq 0}\{\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\sum_j^N\left[\left(q_j + 2\pi i\vec{m}\cdot\vec{\mu}_j\right)\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\right]\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)\}$$

$$+\sum_{\substack{j\notin *(i)\\ j\neq i}}^{\infty}(q_j + \vec{\mu}_j\cdot\nabla_j)\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)-\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}} - \lim_{\vec{R}_j\to\vec{R}_i}\left(q_i+\vec{\mu}_i\cdot\nabla_j\right)\frac{\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}$$

$$-\sum_{j\in *(i)}(q_j + \vec{\mu}_j\cdot\nabla_j)\frac{\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}$$

$$\vec{p}_i^* = \alpha_i(\vec{E}_i^{0*} - \sum_j^N \vec{\vec{T}}_{ij}^*\cdot\vec{p}_j^*)$$

$$\vec{E}_i^{0*} = -\nabla_i\phi_i^{0*}$$

$$T_{ij}^{*\alpha\beta}$$

$$= \partial_i^\alpha\partial_j^\beta\begin{cases}\frac{1}{\pi V}\sum_{\vec{m}\neq 0}\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)+\frac{\mathrm{erf}\left(\beta_{ij}R_{ij}\right)-\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}\\ if\ i\neq j, j\notin *\ (i)\\ \frac{1}{\pi V}\sum_{\vec{m}\neq 0}\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)-\lim_{\vec{R}_j\to\vec{R}_i}\frac{\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}\quad if\ i=j\\ \frac{1}{\pi V}\sum_{\vec{m}\neq 0}\frac{\exp\left(-\frac{\pi^2\vec{m}^2}{\beta_0^2}\right)}{\vec{m}^2}\exp\left(2\pi i\vec{m}\cdot\vec{R}_j\right)\exp\left(-2\pi i\vec{m}\cdot\vec{R}_i\right)-\frac{\mathrm{erf}\left(\beta_0 R_{ij}\right)}{R_{ij}}\quad if\ j\in *\ (i)\end{cases}$$

Here, $*\ (i)$ represent the screened atom pairs of atom *i*.

Before we calculate the correction term under screening conditions for virial, we want first to show the electrostatic force expression under screening. Following our previous paper, it can be easily shown that,

$$\vec{F}_i^{covalent-covalent*} = \sum_j^n \nabla_i \left(\vec{u}_j\right) \cdot \vec{E}_j^{covalent*} + q_i \vec{E}_i^{covalent*} + \vec{u}_i \cdot \vec{\vec{E}}_i^{covalent*}$$

The asterisk here means that the quantities are calculated under the screening condition.

The induced part, however, requires some extra attention,

$$\vec{F}_i^{induced} = \vec{p}_i^* \cdot \vec{\vec{E}}_i^{induced*} + \sum_j^N \nabla_i(\vec{E}_j^{covalent*}) \cdot \vec{p}_j^*$$

Here,

$$\sum_{j}^{N} \nabla_i(\vec{E}_j^{covalent*}) \cdot \vec{p}_j^*$$

$$= \sum_{j \neq i}^{N} \sum_{\substack{k \neq j \\ k \notin *(j)}}^{n} (-\nabla_i(\vec{u}_k)) \cdot \nabla_k\left(\vec{p}_j^* \cdot \nabla_j\right) \frac{\operatorname{erf}(\beta_{jk} R_{jk})}{\beta_{jk} R_{jk}}$$

$$- \sum_{\substack{j \neq i \\ j \notin *(i)}}^{N} (q_i + \vec{\mu}_i \cdot \nabla_i) \nabla_i\left(\vec{p}_j^* \cdot \nabla_j\right) \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right)}{\beta_{ij} R_{ij}}$$

$$- \sum_{\substack{k \neq i \\ k \notin *(i)}}^{n} \nabla_i(\vec{u}_k) \cdot \nabla_k(\vec{p}_i^* \cdot \nabla_i) \frac{\operatorname{erf}(\beta_{ik} R_{ik})}{\beta_{ik} R_{ik}}$$

$$- \sum_{\substack{k \neq i \\ k \notin *(i)}}^{N} (q_k + \vec{\mu}_k \cdot \nabla_k) \nabla_i(\vec{p}_i^* \cdot \nabla_i) \frac{\operatorname{erf}(\beta_{ik} R_{ik})}{\beta_{ik} R_{ik}}$$

$$= \sum_{j}^{N} \sum_{\substack{k \neq j \\ k \notin *(j)}}^{n} \left(-\nabla_i(\vec{u}_k)\right) \cdot \nabla_k\left(\vec{p}_j^* \cdot \nabla_j\right) \frac{\operatorname{erf}\left(\beta_{jk} R_{jk}\right)}{\beta_{jk} R_{jk}}$$

$$- \sum_{\substack{j \neq i \\ j \notin *(i)}}^{N} (q_i + \vec{\mu}_i \cdot \nabla_i) \nabla_i\left(\vec{p}_j^* \cdot \nabla_j\right) \frac{\operatorname{erf}\left(\beta_{ij} R_{ij}\right)}{\beta_{ij} R_{ij}}$$

$$- \sum_{\substack{k \neq i \\ k \notin *(i)}}^{N} (q_k + \vec{\mu}_k \cdot \nabla_k) \nabla_i(\vec{p}_i^* \cdot \nabla_i) \frac{\operatorname{erf}(\beta_{ik} R_{ik})}{\beta_{ik} R_{ik}}$$

$$= \sum_{k}^{n} \sum_{\substack{j \neq k \\ j \notin *(k)}}^{N} \left(-\nabla_i(\vec{u}_k)\right) \cdot \nabla_k\left(\vec{p}_j^* \cdot \nabla_j\right) \frac{\operatorname{erf}\left(\beta_{jk} R_{jk}\right)}{\beta_{jk} R_{jk}} + q_i \vec{E}_i^{induced*} + \vec{u}_i \cdot \vec{\vec{E}}_i^{induced*}$$

$$+ \vec{p}_i^* \cdot \vec{\vec{E}}_i^{covalent*}$$

$$= \sum_{k}^{n} \nabla_i \left(\vec{u}_k\right) \cdot \vec{E}_k^{induced*} + q_i \vec{E}_i^{induced*} + \vec{u}_i \cdot \vec{\vec{E}}_i^{induced*} + \vec{p}_i^{*} \cdot \vec{\vec{E}}_i^{covalent*}$$

In the derivation here, we have used the fact that if $j \in * \ (i)$ then $i \in * \ (j)$ for any *i* and *j*.

So, the result for the force under the screening condition is,

$$\vec{F}_i^{*} = \sum_{j}^{n} \nabla_i \left(\vec{u}_j\right) \cdot \vec{E}_j^{*} + q_i \vec{E}_i^{*} + \left(\vec{u}_i + \vec{p}_i^{*}\right) \cdot \vec{\vec{E}}_i^{*}$$

The asterisk here means that the quantities are calculated under the screening condition.

To derive the correction of screening for virial, we notice that the only variable in the correction term that depends on h is the atom coordinate $\vec{R}$, so, the virial correction is:

$$\frac{\partial U_{correction}}{\partial h_{\mu\nu}} = \frac{\partial \sum_i^N \frac{1}{2} (q_i + \vec{\mu}_i \cdot \nabla_i) \sum_{j \neq i}^{\infty} (q_j + \vec{\mu}_j \cdot \nabla_j) \frac{-\operatorname{erf}(\beta_{ij} R_{ij})}{R_{ij}}}{\partial h_{\mu\nu}}$$

$$+ \frac{1}{2} p_{i'}^{*\alpha'} \frac{\partial\, \partial_{i'}^{\alpha'} \partial_{j'}^{\beta'} \frac{-\operatorname{erf}(\beta_{i'j'} R_{i'j'})}{R_{i'j'}}}{\partial h_{\mu\nu}} p_{j'}^{*\beta'}$$

$$+ \sum_{i}^{N} \vec{p}_i^{*} \cdot \frac{\partial \nabla_i \sum_{j \neq i}^{\infty} (q_j + \vec{\mu}_j \cdot \nabla_j) \frac{-\operatorname{erf}(\beta_{ij} R_{ij})}{R_{ij}}}{\partial h_{\mu\nu}}$$

Following a similar derivation as in **Appendix** B.2, we can easily obtain the correction term for the virial expression under screening.