December 2016

An Adapative Treecode-accelerated Boundary Integral Solver for Computing the Electrostatics of a Biomolecule

Andrew Joseph Szatkowski
University of Wisconsin-Milwaukee

AN ADAPTIVE TREECODE-ACCELERATED BOUNDARY INTEGRAL SOLVER FOR COMPUTING THE ELECTROSTATICS OF A BIOMOLECULE

by

Andrew J. Szatkowski

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Mathematics

at

The University of Wisconsin-Milwaukee

December 2016

ABSTRACT

AN ADAPTIVE TREECODE-ACCELERATED BOUNDARY INTEGRAL SOLVER FOR COMPUTING THE ELECTROSTATICS OF A BIOMOLECULE

by

Andrew J. Szatkowski

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

The Poisson-Boltzmann equation (PBE) is a widely-used model in the calculation of electrostatic potential for solvated biomolecules. PBE is an interface problem defined in the whole space with the interface being a molecular surface of a biomolecule, and has been solved numerically by finite difference, finite element, and boundary integral methods. Unlike the finite difference and finite element methods, the boundary integral method works directly over the whole space without approximating the whole space problem into an artificial boundary value problem. Hence, it is expected to solve PBE in higher accuracy. However, so far, it was only applied to a linear PBE model.

Recently, a solution of PBE was split into three component functions. One of them, G, is a known function that collects all the singularity points of PBE so that the other two components become continuously twice differentiable within the protein and solvent regions. Such an approach has led to efficient PBE finite element solvers. This provided motivation to study the application of this solution decomposition to the development of a new boundary integral algorithm for solving PBE.

Reformulating the interface problem of Ψ into a boundary integral equation is nontrivial because the involved flux interface condition is discontinuous. Development of a fast numerical algorithm for solving the resulted boundary integral equation is an attractive research topic. In this masters thesis, we focus on one key step of our new boundary integral algorithm: how

to solve for the second component function Ψ of the PBE solution by a boundary integral method. This work becomes important by itself because the sum of Ψ with G gives the solution of the Poisson dielectric model for the case of a biomolecule in water.

In this project, we obtain the new boundary integral equation and develop an adaptive treecode-accelerated boundary integral algorithm. We then program the new algorithm in Fortran and make various numerical tests to validate our new algorithm and program package. In particular, numerical tests performed against analytic models verify the effectiveness of the solver, and comparisons to experimental data verify its accuracy for real-world applications. In this way, it is demonstrated that this solver and solution decomposition can compute the electrostatics of a biomolecule in water with high numerical accuracy.

To
My Parents

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Completing a bachelor's degree at the University of Wisconsin - Milwaukee was an event that could only be exceeded by acceptance into their graduate program. I owe many acknowledgements for the opportunity to pursue and complete my masters work here at UWM. First and foremost, I owe a debt of gratitude to my advisor Professor Dexuan Xie. My presence in the graduate program was in no small way due to his belief in my potential as a mathematician. His guidance through this project, and ideas about mathematics and research, are lessons I will carry with me into the future. I also owe my thanks to Professor Richard Stockbridge who, as chair of the math department at the time, initiated, facilitated and inspired my journey to attain these degrees. I also extend my gratitude to Professors Gabriella Pinter and Jeb Willenbring whose work with me and others founding the UWM math club emphasized the necessity of collaborative efforts in high-level mathematics.

I would like to thank my committee as well. Professor Bruce Wade was my instructor for numerical analysis and his enthusiasm for topics in applied mathematics left a lasting impression on me. Professor Lei Wang, her presence and penetrating questions posed during seminars, allowed me to see the current research projects from new vantage points. I thank Professor Hans Volkmer, who contributed significant work to the analytic model used to verify my project. His talks provided insight into the link between the analysis and theory, and the actual implementation of an idea.

I also thank the Department of Mathematics at UWM for providing an assistantship that allowed me to pursue this degree. I have gained invaluable life lessons serving as a teaching

assistant and contributing to the education that many receive here. In this regard, I would like to thank Dr. Kyle Swanson and Dr. Kelly Kohlmetz for their guidance during my role of teaching assistant and instructor. I would also like to thank all the members of my cohort during my time here. I certainly would not have learned as much as I have, or seen as many different points of view, if it weren't for the inspired discussions we shared in our offices and classrooms. I also would like to mention the wonderful people that staff the main office who in no small may have made my job here significantly easier.

Last, but not least, I would like to mention the endless encouragement of my friends and family. Certainly without their love and support I would not have made it this far.

Chapter 1

Introduction

This chapter introduces the motivations for this project, and outlines the thesis. The motivations for this project are two-fold. First, we discuss the importance of solving the Poisson-Boltzmann equation (PBE) accurately for real-world applications for which the ramifications are profound for the field of biochemistry. Second, the importance of exploring and expanding the reach of numerical methods in the field of applied mathematics and scientific computing is highlighted. Here, a brief description of the importance of this project is given from the perspectives of both biochemistry and applied mathematics.

1.1 Motivations

Computing the electrostatics of a solvated biomolecule is a topic of vast importance in the field of biochemistry. PBE is a commonly-used implicit solvent continuum model for predicting such electrostatics. It has been a popular problem to solve in both its linearized and nonlinear forms. We also notice that as an implicit solvent model, it ignores the size of the molecules under consideration treating them as point charges, and it omits the nonlocal effect of polarization correlations among water molecules. It is to our advantage that significant work has been done to alleviate the problems with both types of oversights. The literature has seen analysis developed for the nonlocal case [19] and analysis to handle the size effects of the different molecules [20].

The PBE model is an extremely costly and difficult problem to solve. Even as computers get bigger and faster, the reduction of computing cost (both in terms of memory and other

resources consumed, and total cpu time) is a vital and important research endeavor.

There are two popular approaches for solving PBE: finite element and finite difference methods which discretize the whole space and approximate the boundary conditions at infinity, and boundary element methods which discretize the surface of the biomelecule. There exist popular program packages for solving PBE based on the aformentioned approaches such as DelPhi [14], UHBD [15], and APBS [16]. The work done in [1] (using a treecode method) and [18] (using a fast multipole method) demonstrate two solvers for the linearized PBE using boundary elements.

The primary purpose of this thesis is to develop a boundary integral solver via the solution decomposition described in [3]. The solution is a modification of the approach proposed in [1], in which the PBE system was reduced to a system of two coupled integral equations. Using the aforementioned solution decomposition, the resulting system is only one equation and half the size of the tradition boundary integral formulation.

1.2 Outline

The remainder of this thesis is organized as follows:

In Chapter 2, we review the solution decomposition of PBE. We then consider the case of water to derive a new boundary integral equation in a Fredholm equation of the second kind is described. For this derivation, the solvent domain and protein domain are treated separately. By using the boundary conditions and some known results of potential theory, these two separate treatments can be combined into one equation described on the interface, Γ. Hence, we have rewritten a 3-dimensional volume integral as a 2-dimensional integral over a surface.

In Chapter 3, the adaptive-treecode solver is described. First, we give a general overview of the treecode structure, and clarify the N-body potential form necessary to employ this technique. We describe how the 3D system of atom positions described in cartesian coordinates is identified into near and far-field clusters. The Taylor approximation scheme is described to

handle so-called particle-cluster interactions. The chapter is concluded with a description of how the system is discretized and the general algorithm that was followed to solve for Ψ using the adaptive-treecode solver.

In Chapter 4, the results of the simulations are displayed. The accuracy of the solver will be given as compared to analytic models. Historically the Born model was the only available analytic solution for comparison, but in this thesis we use the results of [9] in which an analytic solution was found for an arbitrary number of charges placed inside a sphere is given to verify the code. We then use the protein 1A63 protein to verify the time complexity improvement of the method, and a set of 17 proteins for which there exists experimental data to compare the free energy of solvation calculations.

The conclusion, as well as future work, will be given in the last chapter. In this section a summary of the results and their importance to relevant fields of study will be expatiated.

Chapter 2

Boundary Integral Equation For PBE

This chapter examines the commonly-used dimensionless PBE model. The classic definition gives the electrostatic potential over the entire domain $\mathbb{R}^3$. The steps to convert this equation into one integral on the boundary will be presented. To do so we follow the technique described in [6].

2.1 PBE Model

To describe the system of a solvated protein, we denote the bounded region hosting the protein structure by D_p and the surrounding solvent region by D_s. The surface of the protein, or the interface, is denoted by Γ. The space $\mathbb{R}^3$ has the decomposition

$$\mathbb{R}^3 = D_s \cup D_p \cup \Gamma \tag{2.1}$$

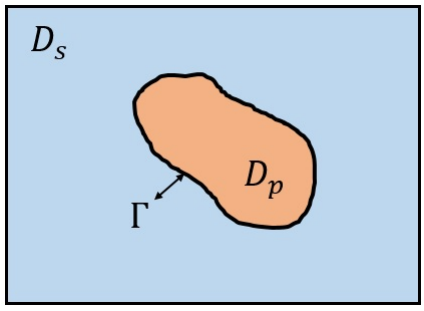

Figure 2.1: Decomposition of the whole space into the protein and solvent domains and the interface linking them

The commonly-used dimensionless PBE model is given as

$$\begin{cases} -\epsilon_p \Delta u(\mathbf{r}) = \alpha \sum_{j=1}^{n_p} z_j \delta_{r_j}, & \mathbf{r} \in D_p \\ -\epsilon_s \Delta u(\mathbf{r}) + \kappa^2 sinh(u) = 0, & \mathbf{r} \in D_s \\ u(\mathbf{s}^+) = u(\mathbf{s}^-), \quad \varepsilon_s \dfrac{\partial u(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \varepsilon_p \dfrac{\partial u(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma \\ u(\mathbf{r}) \to 0 & \text{as} |\mathbf{r}| \to \infty \end{cases} \tag{2.2}$$

where α and κ^2 are defined by

$$\alpha = \frac{10^{10} e_c^2}{\epsilon_0 k_B T}, \qquad \kappa^2 = 2 I_s \frac{10^{-17} N_A e_c^2}{\epsilon_0 k_B T}. \tag{2.3}$$

Here, e_c represents the elementary charge of an electron, ϵ_0 is the permittivity of free space, k_B is the Boltzmann constant, I_s is the ionic strength, N_A is Avogradro's number, and T is the absolute temperature in Kelvin. We also have that ϵ_s is the permittivity of the solvent region, ϵ_p is the permittivity of the protein region, and z_j is the charge of atom j. The work done in [3] allows us to decompose the PBE solution u as

$$u(\mathbf{r}) = G(\mathbf{r}) + \Psi(\mathbf{r}) + \tilde{\Phi}(\mathbf{r}) \qquad \forall \mathbf{r} \in \Omega \tag{2.4}$$

where G is given by

$$G(\mathbf{r}) = \frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} \frac{z_j}{|\mathbf{r} - \mathbf{r}_j|} \tag{2.5}$$

and $\frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}$ can be found as

$$\frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} = -\frac{\alpha}{4\pi\epsilon_p} \sum_{j=1}^{n_p} z_j \frac{(\mathbf{s} - \mathbf{r}_j) \cdot \mathbf{n}}{|\mathbf{s} - \mathbf{r}_j|^3}. \tag{2.6}$$

Notice that Ψ is a solution of the linear interface problem

$$\begin{cases} \Delta\Psi(\mathbf{r}) = 0, & \mathbf{r} \in D_p \cup D_s, \\ \Psi(\mathbf{s}^+) = \Psi(\mathbf{s}^-), & \mathbf{s} \in \Gamma, \\ \varepsilon_s \frac{\partial \Psi(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \varepsilon_p \frac{\partial \Psi(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} + (\epsilon_p - \epsilon_s) \frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \Psi(\mathbf{r}) \to 0 & \text{as } |\mathbf{r}| \to \infty, \end{cases} \tag{2.7}$$

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

$$\begin{cases} \Delta\tilde{\Phi}(\mathbf{r}) = 0, & \mathbf{r} \in D_p, \\ -\epsilon_s \Delta\tilde{\Phi}(\mathbf{r}) + \kappa^2 \sinh(G + \Psi + \tilde{\Phi}) = 0, & \mathbf{r} \in D_s, \\ \tilde{\Phi}(\mathbf{s}^+) = \tilde{\Phi}(\mathbf{s}^-), \quad \varepsilon_s \frac{\partial \tilde{\Phi}(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} = \varepsilon_p \frac{\partial \tilde{\Phi}(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})}, & \mathbf{s} \in \Gamma, \\ \tilde{\Phi}(\mathbf{r}) \to 0 & \text{as } |\mathbf{r}| \to \infty. \end{cases} \tag{2.8}$$

For the case when the solvent region is water, the ionic strength is zero and hence the term κ disappears, thus eliminating the nonlinear effects and the need to solve (2.8). This means that, for the water case, we only need to solve (2.7), because G is a known function. The solution decomposition then becomes, $u = G + \Psi$. To convert (2.7) into a boundary integral, it is necessary to consider the cases $\mathbf{r} \in D_p$, the protein domain, and $\mathbf{r} \in D_s$, the solvent domain, separately

2.2 The Protein Domain

For the boundary integral considering the interior domain, we will consider the first equation in (2.7) (i.e. we are considering the case when $\mathbf{r} \in D_p$). The fundamental solution to the Poisson equation with singularity in $\mathbf{r}$ satisfies

$$\Delta G_0(\mathbf{r}, \mathbf{s}) = -\delta_{\mathbf{r}} \tag{2.9}$$

where $\delta_{\mathbf{r}}$ is the Dirac delta measure for which it is known that as a continuous linear functional we have

$$< \delta_{\mathbf{r}}, v >= v(\mathbf{r}) \tag{2.10}$$

for any test function v, and G_0 is known to be

$$G_0(\mathbf{r}, \mathbf{s}) = \frac{1}{4\pi|\mathbf{r} - \mathbf{s}|}. \tag{2.11}$$

We multiply the first equation in (2.7) by $G_0(\mathbf{r}, \mathbf{s})$, multiply (2.9) by $\Psi(\mathbf{s})$, subtract the results from each other, and integrate over the domain D_p with respect to $\mathbf{s}$. We obtain:

$$\int_{D_p} \Psi(\mathbf{s}) \Delta G_0(\mathbf{r}, \mathbf{s}) - G_0(\mathbf{r}, \mathbf{s}) \Delta\Psi(\mathbf{s}) \, d\mathbf{s} = \int_{D_p} -\Psi(\mathbf{s}) \delta_{\mathbf{r}} d\mathbf{s}. \tag{2.12}$$

Green's second identity is defined as

$$\int_{\Omega} (u\Delta v - v\Delta u) \, dV = \int_{\partial\Omega} \left(u\frac{\partial v}{\partial n} - v\frac{\partial u}{\partial n} \right) dS, \tag{2.13}$$

and applying it to (2.12), we find that

$$\int_{\Gamma} \left[\Psi(\mathbf{s}) \frac{\partial G_0(\mathbf{r}, \mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} - G_0(\mathbf{r}, \mathbf{s}) \frac{\partial \Psi(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} \right] dS(\mathbf{s}) = -\Psi(\mathbf{r}). \tag{2.14}$$

Rearranging terms results in

$$\Psi(\mathbf{r}) = \int_{\Gamma} G_0(\mathbf{r}, \mathbf{s}) \frac{\partial \Psi(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}) - \int_{\Gamma} \Psi(\mathbf{s}) \frac{\partial G_0(\mathbf{r}, \mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}). \tag{2.15}$$

Now, let $\mathbf{r} = \mathbf{s} - \alpha\mathbf{n}(\mathbf{s})$, where $\mathbf{n}(\mathbf{s})$ is the outward normal vector at $\mathbf{s}$ and α is a positive constant, and consider the limit as $\alpha \to 0$. We define this limit as $\mathbf{s}^-$. Applying the properties

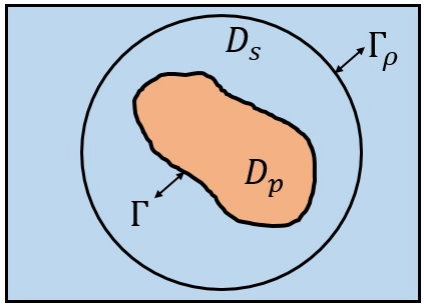

Figure 2.2: To consider the integral in the solvent domain we introduce a new interface Γ_ρ. Therefore, the integrals are well-defined, and we can take the limit as $\rho \to \infty$

of single and double layer potentials as described in [8] we obtain

$$\frac{1}{2}\Psi\left(\mathbf{s}^{-}\right) = \int_{\Gamma} G_0\left(\mathbf{s}^{-}, \mathbf{s}\right) \frac{\partial \Psi\left(\mathbf{s}\right)}{\partial \mathbf{n}\left(\mathbf{s}\right)} dS\left(\mathbf{s}\right) - \int_{\Gamma} \Psi\left(\mathbf{s}\right) \frac{\partial G_0\left(\mathbf{s}^{-}, \mathbf{s}\right)}{\partial \mathbf{n}\left(\mathbf{s}\right)} dS\left(\mathbf{s}\right), \quad \text{for } \mathbf{s} \in \Gamma. \tag{2.16}$$

2.3 The Solvent Domain

For the boundary integral considering the exterior domain, consider the first equation in (2.7). We are now considering the case when $\mathbf{r} \in D_s$. First, we must define a new domain, $D_\rho = \{\mathbf{r} || \mathbf{r}| \leq \rho\} \backslash D_p \cup \Gamma$ such that $\partial D_\rho = \Gamma_\rho \cup \Gamma$ (See Figure: 2.2). Now we follow the same steps as for the protein domain, that is

$$\int_{D_\rho} \Psi\left(\mathbf{s}\right) \Delta G_0\left(\mathbf{r}, \mathbf{s}\right) - G_0\left(\mathbf{r}, \mathbf{s}\right) \Delta \Psi\left(\mathbf{s}\right) d\mathbf{s} = \int_{D_\rho} -\Psi\left(\mathbf{s}\right) \delta_{\mathbf{r}} d\mathbf{s}. \tag{2.17}$$

Applying Green's second identity yields

$$\int_{\Gamma} \Psi\left(\mathbf{s}\right) \frac{\partial G_0\left(\mathbf{r}, \mathbf{s}\right)}{\partial \mathbf{n}_\rho\left(\mathbf{s}\right)} - G_0\left(\mathbf{r}, \mathbf{s}\right) \frac{\partial \Psi\left(\mathbf{s}\right)}{\partial \mathbf{n}_\rho\left(\mathbf{s}\right)} dS\left(\mathbf{s}\right) \tag{2.18}$$

$$+ \int_{\Gamma_\rho} \Psi\left(\mathbf{s}\right) \frac{\partial G_0\left(\mathbf{r}, \mathbf{s}\right)}{\partial \mathbf{n}_\rho\left(\mathbf{s}\right)} - G_0\left(\mathbf{r}, \mathbf{s}\right) \frac{\partial \Psi\left(\mathbf{s}\right)}{\partial \mathbf{n}_\rho\left(\mathbf{s}\right)} dS\left(\mathbf{s}\right) = -\Psi\left(\mathbf{r}\right). \tag{2.19}$$

The integral in (2.19) vanishes as $\rho \to \infty$ per the boundary conditions. Moreover, $\mathbf{n}_\rho(\mathbf{s}) = -\mathbf{n}(\mathbf{s})$ so that

$$\Psi(\mathbf{r}) = \int_\Gamma -G_0(\mathbf{r},\mathbf{s}) \frac{\partial \Psi(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}) + \int_\Gamma \Psi(\mathbf{s}) \frac{\partial G_0(\mathbf{r},\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}), \quad \mathbf{r} \in D_s. \tag{2.20}$$

In this case, we define $\mathbf{r} = \mathbf{s} + \alpha \mathbf{n}(\mathbf{s})$ and define the limit as $\alpha \to 0$ to be denoted as $\mathbf{s}^+$. Applying the results from [8] again, we see that

$$\frac{1}{2}\Psi(\mathbf{s}^+) = \int_\Gamma -G_0(\mathbf{s}^+,\mathbf{s}) \frac{\partial \Psi(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}) + \int_\Gamma \Psi(\mathbf{s}) \frac{\partial G_0(\mathbf{s}^+,\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}), \quad \text{for } \mathbf{s} \in \Gamma. \tag{2.21}$$

2.4 The new boundary integral

So far we have

$$\frac{1}{2}\Psi(\mathbf{s}^-) = \int_\Gamma G_0(\mathbf{s}^-,\mathbf{s}) \frac{\partial \Psi(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}) - \int_\Gamma \Psi(\mathbf{s}) \frac{\partial G_0(\mathbf{s}^-,\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}), \quad \text{for } \mathbf{s} \in \Gamma \tag{2.22}$$

and

$$\frac{1}{2}\Psi(\mathbf{s}^+) = \int_\Gamma -G_0(\mathbf{s}^+,\mathbf{s}) \frac{\partial \Psi(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}) + \int_\Gamma \Psi(\mathbf{s}) \frac{\partial G_0(\mathbf{s}^+,\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} dS(\mathbf{s}), \quad \text{for } \mathbf{s} \in \Gamma. \tag{2.23}$$

The goal is to combine these two equations so that the normal derivative of Ψ is no longer needed. To accomplish this task, we use the boundary condition

$$\frac{\partial G(\mathbf{s})}{\partial \mathbf{n}(\mathbf{s})} = \frac{1}{(\varepsilon_p - \varepsilon_s)} \left[\varepsilon_s \frac{\partial \Psi(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} - \varepsilon_p \frac{\partial \Psi(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} \right]. \tag{2.24}$$

Multiplying (2.22) by ε_p and (2.23) by ε_s, and adding them together results in

$$\frac{1}{2}(\varepsilon_p + \varepsilon_s)\Psi(\mathbf{s}) = \int_\Gamma G_0(\mathbf{s},\mathbf{s}') \left[\varepsilon_p \frac{\partial \Psi(\mathbf{s}^-)}{\partial \mathbf{n}(\mathbf{s})} - \varepsilon_s \frac{\partial \Psi(\mathbf{s}^+)}{\partial \mathbf{n}(\mathbf{s})} \right] dS(\mathbf{s}') \tag{2.25}$$

$$+\int_{\Gamma}\frac{\partial G_0\left(\mathbf{s},\mathbf{s}'\right)}{\partial\mathbf{n}\left(\mathbf{s}'\right)}\left(\varepsilon_s-\varepsilon_p\right)\Psi\left(\mathbf{s}'\right)dS\left(\mathbf{s}'\right). \tag{2.26}$$

Lastly, we multiply by $-\frac{1}{\varepsilon_p-\varepsilon_s}$.

$$\frac{-\left(\varepsilon_p+\varepsilon_s\right)}{2\left(\varepsilon_p-\varepsilon_s\right)}\Psi\left(\mathbf{s}\right)=\int_{\Gamma}\frac{G_0\left(\mathbf{s},\mathbf{s}'\right)}{\left(\varepsilon_p-\varepsilon_s\right)}\left[\varepsilon_s\frac{\partial\Psi\left(\mathbf{s}^+\right)}{\partial\mathbf{n}\left(\mathbf{s}\right)}-\varepsilon_p\frac{\partial\Psi\left(\mathbf{s}^-\right)}{\partial\mathbf{n}\left(\mathbf{s}\right)}\right]dS\left(\mathbf{s}'\right) \tag{2.27}$$

$$-\int_{\Gamma}\frac{\partial G_0\left(\mathbf{s},\mathbf{s}'\right)}{\partial\mathbf{n}\left(\mathbf{s}'\right)}\frac{\left(\varepsilon_s-\varepsilon_p\right)}{\left(\varepsilon_p-\varepsilon_s\right)}\Psi\left(\mathbf{s}'\right)dS\left(\mathbf{s}'\right) \tag{2.28}$$

Applying (2.24) simplifies this equation by replacing the normal derivatives of Ψ. The boundary integral equation for Ψ is

$$\frac{-\left(\varepsilon_p+\varepsilon_s\right)}{2\left(\varepsilon_p-\varepsilon_s\right)}\Psi\left(\mathbf{s}\right)=\int_{\Gamma}G_0\left(\mathbf{s},\mathbf{s}'\right)\frac{\partial G\left(\mathbf{s}'\right)}{\partial\mathbf{n}\left(\mathbf{s}'\right)}+\int_{\Gamma}\frac{\partial G_0\left(\mathbf{s},\mathbf{s}'\right)}{\partial\mathbf{n}\left(\mathbf{s}'\right)}\Psi\left(\mathbf{s}'\right)dS\left(\mathbf{s}\right),\quad\text{for }\mathbf{s}\in\Gamma. \tag{2.29}$$

Hence, the result described in [2] is verified. After the solution of the above boundary equation is found, we can calculate the values of Ψ in D_p and D_s using the following expressions:

$$\Psi(\mathbf{r})=\begin{cases}\dfrac{1}{\epsilon_p}g(\mathbf{r})+\dfrac{\epsilon_s-\epsilon_p}{\epsilon_p}\displaystyle\int_{\Gamma}\frac{\partial G_0\left(\mathbf{r},\mathbf{s}\right)}{\partial\mathbf{n}\left(\mathbf{s}\right)}\Psi\left(\mathbf{s}\right)dS\left(\mathbf{s}\right) & \text{if }\mathbf{r}\in D_p,\\[2ex] \dfrac{1}{\epsilon_s}g(\mathbf{r})+\dfrac{\epsilon_s-\epsilon_p}{\epsilon_s}\displaystyle\int_{\Gamma}\frac{\partial G_0\left(\mathbf{r},\mathbf{s}\right)}{\partial\mathbf{n}\left(\mathbf{s}\right)}\Psi\left(\mathbf{s}\right)dS\left(\mathbf{s}\right) & \text{if }\mathbf{r}\in D_s,\end{cases} \tag{2.30}$$

where

$$g(\mathbf{r})=\left(\epsilon_s-\epsilon_p\right)\int_{\Gamma}G_0\left(\mathbf{r},\mathbf{s}\right)\frac{\partial G\left(\mathbf{s}\right)}{\partial\mathbf{n}\left(\mathbf{s}\right)}dS\left(\mathbf{s}\right), \tag{2.31}$$

and

$$\frac{\partial G_0\left(\mathbf{r},\mathbf{s}\right)}{\partial\mathbf{n}\left(\mathbf{s}\right)}=\frac{\left(\mathbf{r}-\mathbf{s}\right)\cdot\mathbf{n}(\mathbf{s})}{4\pi|\mathbf{r}-\mathbf{s}|^3}. \tag{2.32}$$

Chapter 3

Adaptive Treecode-Accelerated Solver

This chapter provides a general overview of the adaptive-treecode method: the adaptive features, the role of N-body potentials, how the system was discretized, and the general algorithm are discussed.

3.1 Treecode Structure and N-body Potentials

The treecode method is a popular tool for solving N-body problems. In general, any problem that involves N-body potentials may employ a treecode technique. The general form of an N-body potential is

$$V_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} q_j K(\mathbf{x_i}, \mathbf{x_j}), \qquad i = 1, \ldots, N \tag{3.1}$$

where K is a kernel function, $\mathbf{x_i}$ and $\mathbf{x_j}$ are particle positions, and q_j is a charge associated with $\mathbf{x_j}$ [1]. In order to evaluate the potential V_i, the positions of each particle are broken down into an octree structure used in [4]. This means the root of the tree is defined to be a cube containing all the particles. The root cube is divided into eight equally-sized cubes, and each of those cubes is divided similarly (Figure: 3.1). The division process continues recursively until there are no more than N_0 particles in each cube, a parameter to be defined by the user. The smallest cubes are called the leafs of the tree. This process yields uniformly sized clusters. For the purposes of this project, an adaptive treecode is employed to utilize non-uniform clusters. The adaptive tree is different in that the cubes surrounding the particles

are resized to be the smallest cube containing all particles, and empty cubes are discarded as shown in Figure 3.2. This reduces the size of the tree, and also allows for a more accurate description of a cluster, i.e. it cuts down on empty space.

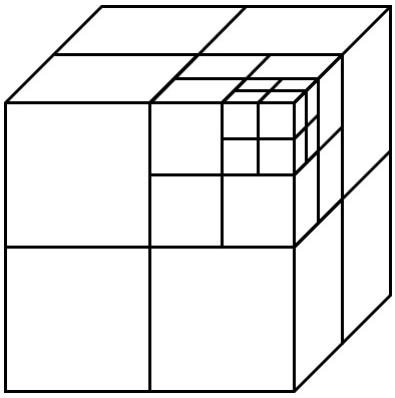

Figure 3.1: The octree structure encloses all particles within the root cube, and then successively divides each cube into eight smaller cubes.

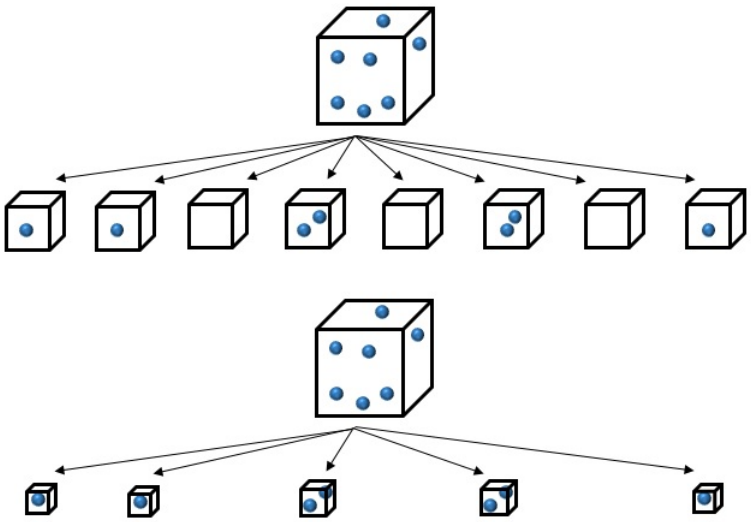

Figure 3.2: This illustration describes the adaptive nature of this treecode scheme: Cells are resized and empty cells are eliminated.

Once the tree has been created and the clusters have been determined, the summation for the potential in (3.1) can be evaluated as a sum of near-field (N_i) interactions and far-field (F_i) interactions. The far-field is determined to be the set of clusters that are well-separated

from the point $\mathbf{x_i}$. A particle $\mathbf{x_i}$ and a cluster c are defined to be well-separated if a multipole acceptance criterion (MAC) is satisfied. The MAC is defined as

$$\frac{r_c}{R} \leq \theta \tag{3.2}$$

where r_c is defined to be the cluster radius, $R = |\mathbf{x_i} - \mathbf{x_c}|$ is the particle-cluster distance and θ is a user-defined parameter between zero and one. Larger values of θ identify more clusters as far-field clusters. The cluster radius and the center of the cluster are defined in various ways. The center may be defined quite literally as the middle of the cube defining the cluster. In this way the cluster radius may naturally be the distance from the center to the corner of the cube. Since this problem considers the effects of spherical objects, it is natural for us that the center, $\mathbf{x_c}$, is determined to be the center of mass of the cluster, c, and the cluster radius is defined to be the maximum distance between the center and all bodies within the cluster, $\mathbf{r_c} = \max_{\mathbf{x_j} \in c} |\mathbf{x_j} - \mathbf{x_c}|$. See Figure 3.3.

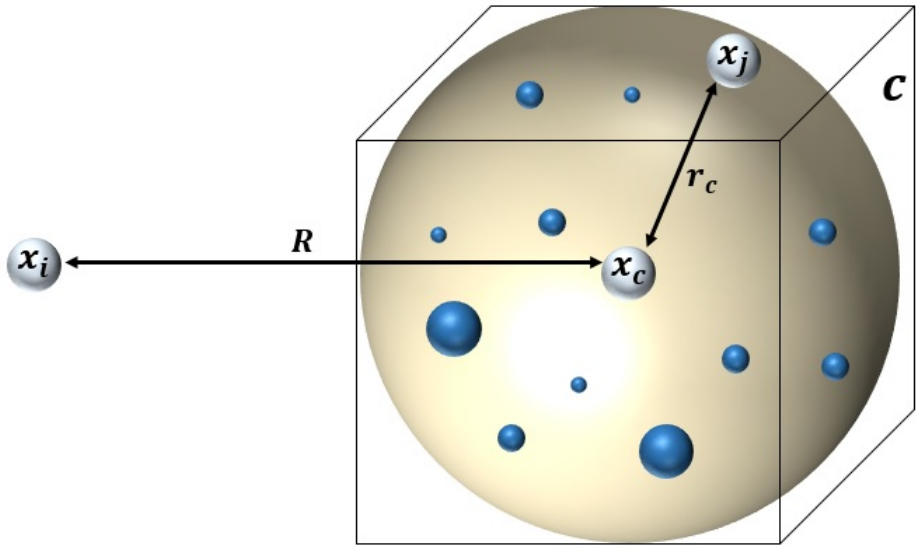

Figure 3.3: Particle-Cluster Separation: A visual representation of what how the MAC parameter determines a cluster to be well separated from a particle.

Any clusters that satisfy the MAC criterion are determined to be in the far-field. The far-field clusters can be evaluated as a p^{th} order Cartesian Taylor approximation about the center of the cluster $\mathbf{x_c}$. The summation in (3.1) is broken apart into two sums. One sum involves clusters in the near-field, that is, clusters that do not satisfy the MAC criterion. The interactions between $\mathbf{x_i}$ and these clusters are evaluated using direct summation. The second summation involves clusters determined to be well-separated from $\mathbf{x_i}$. It is these clusters that we wish to approximate using a Taylor expansion. The Taylor expansion about a point $\mathbf{y} \in \mathbb{R}^3$ of order p is defined as

$$f(\mathbf{x}) = \sum_{|\alpha|=0}^{p} \frac{1}{\alpha!} D_{\mathbf{x}}^{\alpha} f(\mathbf{y}) (\mathbf{x} - \mathbf{y})^{\alpha} \tag{3.3}$$

where

$$D_{\mathbf{x}}^{\alpha} = \partial_{\mathbf{x}}^{\alpha} = \frac{\partial^{\alpha}}{\partial \mathbf{x}^{\alpha}}. \tag{3.4}$$

Cartesian multi-index notation is defined by $\alpha = (\alpha_1, \alpha_2, \alpha_3)$, $\alpha_i \in \mathbb{N}$, $\|\alpha\| = \alpha_1 + \alpha_2 + \alpha_3$, $\alpha! = \alpha_1! \alpha_2! \alpha_3!$

The summation in (3.1) is expanded about the center of a cluster $\mathbf{x_c}$ and can be written as

$$V_i \approx \sum_{c \in N_i} \sum_{\mathbf{x_j} \in c} q_j K(\mathbf{x_i}, \mathbf{x_j}) + \sum_{c \in F_i} \sum_{\mathbf{x_j} \in c} q_j \sum_{\|\alpha\|=0}^{p} \frac{1}{\alpha!} \partial_{\mathbf{y}}^{\alpha} K(\mathbf{x_i}, \mathbf{x_c}) (\mathbf{x_j} - \mathbf{x_c})^{\alpha}. \tag{3.5}$$

Notice that the terms q_j and $(\mathbf{x_j} - \mathbf{x_c})^{\alpha}$ on the right hand side depend on j. This allows us to separate the summation involving those terms to get

$$V_i \approx \sum_{c \in N_i} \sum_{\mathbf{x_j} \in c} q_j K(\mathbf{x_i}, \mathbf{x_j}) + \sum_{c \in F_i} \sum_{\|\alpha\|=0}^{p} \frac{1}{\alpha!} \partial_{\mathbf{y}}^{\alpha} K(\mathbf{x_i}, \mathbf{x_c}) \sum_{\mathbf{x_j} \in c} q_j (\mathbf{x_j} - \mathbf{x_c})^{\alpha}. \tag{3.6}$$

We define the Taylor coefficients to be

$$a^{\alpha}(\mathbf{x_i}, \mathbf{x_c}) = \frac{1}{\alpha!} \partial_y^{\alpha} K(\mathbf{x_i}, \mathbf{x_c}) \tag{3.7}$$

and the cluster moments as

$$m_c^{\alpha} = \sum_{\mathbf{x_j} \in c} q_j (\mathbf{x_j} - \mathbf{x_c})^{\alpha} \tag{3.8}$$

so that the equation for V_i can be more succinctly written as

$$V_i \approx \sum_{c \in N_i} \sum_{\mathbf{x_j} \in c} q_j K(\mathbf{x_i}, \mathbf{x_j}) + \sum_{c \in F_i} \sum_{\|\alpha\|=0}^{p} a^{\alpha}(\mathbf{x_i}, \mathbf{x_c}) m_c^{\alpha}. \tag{3.9}$$

Just as in [1] we have the partial derivative of the kernel. We must apply an operator, $\partial_{\mathbf{x_j}}^{\mathbf{I_0}}$, to (3.9) to get

$$\partial_{\mathbf{x_j}}^{\mathbf{I_0}} V_i = \sum_{j=1}^{N} q_j \partial_{\mathbf{x_j}}^{\mathbf{I_0}} K(\mathbf{x_i}, \mathbf{x_j}), \quad i = 1, \ldots, N \tag{3.10}$$

$$\approx \sum_{c \in N_i} \sum_{\mathbf{x_j} \in c} q_j \partial_{\mathbf{x_j}}^{\mathbf{I_0}} K(\mathbf{x_i}, \mathbf{x_j}) \tag{3.11}$$

$$+ \sum_{c \in F_i} \sum_{\|\alpha\|=0}^{p} \frac{(\alpha + \mathbf{I_0})!}{\alpha!} a^{\alpha + \mathbf{I_0}}(\mathbf{x_i}, \mathbf{x_c}) m_c^{\alpha}. \tag{3.12}$$

Since the calculation of a^{α} can be costly in terms of computation time and resources for higher orders p, we use the recurrence relations derived in [7]. Since our consideration of Ψ does not involve this term associated with the screened Coulomb potential, we take their recurrence relations and let the parameter, κ, go to zero. We end up with a coupled set of relations defined by

$$\|\alpha\| |\mathbf{x} - \mathbf{y}|^2 a^{\alpha} - (2\|\alpha\| - 1) \sum_{i=1}^{3} (x_i - y_i) a^{\alpha - \mathbf{e_i}} + (\|\alpha\| - 1) \sum_{i=1}^{3} a^{\alpha - 2\mathbf{e_i}} = 0 \tag{3.13}$$

$$\|\alpha\| b^{\alpha} = 0. \tag{3.14}$$

3.2 Discretization

In order to discretize this system we first rewrite (2.29) as

$$\frac{-\left(\varepsilon_p+\varepsilon_s\right)}{2\left(\varepsilon_p-\varepsilon_s\right)}\Psi\left(\mathbf{s}\right)-\int_\Gamma G_0\left(\mathbf{s},\mathbf{s}'\right)\frac{\partial G\left(\mathbf{s}'\right)}{\partial\mathbf{n}\left(\mathbf{s}'\right)}d\mathbf{S}'=\int_\Gamma\frac{\partial G_0\left(\mathbf{s},\mathbf{s}'\right)}{\partial\mathbf{n}\left(\mathbf{s}'\right)}\Psi\left(\mathbf{s}'\right)d\mathbf{S}',\quad\text{for }\mathbf{s}\in\Gamma \tag{3.15}$$

so that it has the form of a Fredholm integral equation of the second kind defined generally as

$$\lambda x\left(t\right)-\int_D K\left(t,s\right)x\left(s\right)ds=y\left(t\right),\quad t\in D. \tag{3.16}$$

There are two common methods of solving (3.16): *collocation methods* and *Galerkin methods*. We opt for the former, and in particular, a centroid collocation method as used in [1]. As such, we then discretize the interface, Γ, into a collection of N triangles

$$\Gamma\approx\Gamma_h=\cup_{j=1}^N\triangle_j \tag{3.17}$$

so that we have for any given function f

$$\int_{\Gamma_h}f\left(\mathbf{s},\mathbf{s}'\right)d\mathbf{s}'=\sum_{j=1}^N\int_{\triangle_j}f\left(\mathbf{s},\mathbf{s}'\right)\approx\sum_{j=1}^N f\left(\mathbf{s},\mathbf{s}'\right)A_j \tag{3.18}$$

where A_j is the area of the j^{th} triangle and $\mathbf{s}$ and $\mathbf{s}'$ are triangle centroids. Since f is representing a kernel function, it will have a singularity at $\mathbf{s}'$.

One option for handling the singularity is to remove it, but this strategy could result in significant errors. In an attempt to reduce the errors associated with removing said singularity at $\mathbf{s_i}=\mathbf{s_j}$, we take the point $\mathbf{s_j}$ to be a vertex of $\triangle_j$ and then average the contribution of each vertex. So when $\mathbf{s_i}=\mathbf{s_j}$ we treat the singularity as

$$\int_{\triangle_j}f\left(\mathbf{s_i},\mathbf{s_j}\right)d\mathbf{s_j}\approx\frac{1}{3}\sum_{k=1}^3 f\left(\mathbf{s_i},\mathbf{v_k}\right)A_j \tag{3.19}$$

where v_k is the k^{th} vertex of $\triangle_j$.

Given a triangulation of the interface, Γ, let $\mathbf{s_i},\ A_i : i = 1, \ldots, N$ denote the centroids and areas of the faces in the triangulation respectively. Here N represents the total number of faces (surface points). Then for $i = 1, \ldots, N$ (3.15) becomes

$$\frac{-(\varepsilon_p + \varepsilon_s)}{2(\varepsilon_p - \varepsilon_s)}\Psi(\mathbf{s_i}) - \sum_{j=1}^{N}\left[\frac{\partial G_0(\mathbf{s_i}, \mathbf{s_j})}{\partial \mathbf{n}(\mathbf{s_j})}\Psi(\mathbf{s_j})\right] A_j = \sum_{j=1}^{N}\left[G_0(\mathbf{s_i}, \mathbf{s_j})\frac{\partial G(\mathbf{s_j})}{\partial \mathbf{n}(\mathbf{s_j})}\right] A_j. \tag{3.20}$$

Let

$$S(\mathbf{s_i}) = \sum_{j=1}^{N}\left[G_0(\mathbf{s_i}, \mathbf{s_j})\frac{\partial G(\mathbf{s_j})}{\partial \mathbf{n}(\mathbf{s_j})}\right] A_j, \tag{3.21}$$

so that we have

$$\frac{-(\varepsilon_p + \varepsilon_s)}{2(\varepsilon_p - \varepsilon_s)}\Psi(\mathbf{s_i}) - \sum_{j=1}^{N}\left[\frac{\partial G_0(\mathbf{s_i}, \mathbf{s_j})}{\partial \mathbf{n}(\mathbf{s_j})}\Psi(\mathbf{s_j})\right] A_j = S(\mathbf{s_i}). \tag{3.22}$$

Note that (3.22) defines a linear system of the from $A\mathbf{x} = \mathbf{b}$ where $\mathbf{x} = [\Psi(\mathbf{s_i})]$ and $\mathbf{b} = [S(\mathbf{s_i})]$.

3.3 Treecode Algorithm

The tree structure is programmed so that each cluster is referenced by a pointer and contains the information of its child clusters. In this way we can recursively access the branches and leaves of the tree as needed. In order to calculate the N-body potentials we employ the following algorithm.

1. Input particle positions and charges: $\mathbf{x}_i$, q_i, for $i = 1, \ldots, N$
2. Input user parameters: θ, p, N_0
3. Construct the tree

for i=1:N **do**

Compute the potential for particle $\mathbf{x}_i$

end for

Compute potential contributions for all other particles $\mathbf{x}_j$

if MAC is satisfied **then**

compute moments of the cluster c (unless already stored)

compute particle-cluster interactions by Taylor approximation

else

if c is a leaf **then**

compute particle-cluster interactions via direct solver

else

for j=1:number of children **do**

Call Compute Potential for each child cube

end for

end if

end if

The recursive nature of the algorithm that brings down the time complexity. For an interface discretized into N triangles, each branch of the tree has at most $N/8^l$ cubes, where l describes the level of the branch (0 refers to the root cluster). The number of levels the tree has will roughly be $8^l = N$. Therefore, similar to binary search algorithms, the amount of time it will take to traverse the tree to the lowest level will be on the order of $\mathcal{O}(N \log_8 N)$.

3.4 Implementation

The adaptive-treecode accelerated boundary integral solved developed in the previous sections was programmed in Fortran 90. The system described in (3.22) is solved for Ψ using the GMRES algorithm from NetLib [23, 13] where each matrix vector multiplication Ax is done using the treecode algorithm. Setting the MAC parameter $\theta = 0$ results in the program

solving the system directly. The simulations were run on a Mac Pro, Mac OS X version 10.7.5, with 12 GB 667 MHZ DDR2 FB-DIMM memory, and a 2 x 3 GHZ Quad-Core Intel Xeon processor.

Chapter 4

Results

This section presents the results for two separate test cases. The first case ran the coded solution against a new class of analytic spherical models given in [9]. This new solution is described by

$$\Phi(\mathbf{r}) = \begin{cases} \dfrac{e_c}{\epsilon_0} \displaystyle\sum_{n=0}^{\infty} \sum_{j=1}^{n_p} z_j \mathbf{A}_{j,n} |\mathbf{r}|^n \mathbf{P}_n \left(\dfrac{\mathbf{r}_j \cdot \mathbf{r}}{|\mathbf{r}_j||\mathbf{r}|} \right) + \dfrac{e_c}{4\pi\epsilon_0\epsilon_p} \displaystyle\sum_{j=1}^{\infty} \dfrac{z_j}{|\mathbf{r} - \mathbf{r}_j|} & \text{if } \mathbf{r} \in D_p \\ \dfrac{e_c}{\epsilon_0} \displaystyle\sum_{n=0}^{\infty} \sum_{j=1}^{n_p} \dfrac{z_j \mathbf{B}_{j,n}}{|\mathbf{r}|^{n+1}} \mathbf{P}_n \left(\dfrac{\mathbf{r}_j \cdot \mathbf{r}}{|\mathbf{r}_j||\mathbf{r}|} \right) & \text{if } \mathbf{r} \in D_s, \end{cases} \tag{4.1}$$

where $\mathbf{A}_{j,n}$ and $\mathbf{B}_{j,n}$ are given by

$$\mathbf{A}_{j,n} = \frac{(\epsilon_p - \epsilon_s)(n+1)|\mathbf{r}_j|^n}{4\pi\epsilon_p a^{2n+1}[n\epsilon_p + (n+1)\epsilon_s]} \tag{4.2}$$

$$\mathbf{B}_{j,n} = \frac{(2n+1)|\mathbf{r}_j|^n}{4\pi[n\epsilon_p + (n+1)\epsilon_s]}. \tag{4.3}$$

The work in this paper allows us to expand on the Born Ball model, which is traditionally used to verify this type of computational work, and calculate an analytic solution for an arbitrary number or charges inside a sphere.

The second test case compares the calculation for the free energy of solvation against a set of experimental data reported in [10] for 17 different proteins. The protein 1A63 is used to demonstrate the time behavior of the solution.

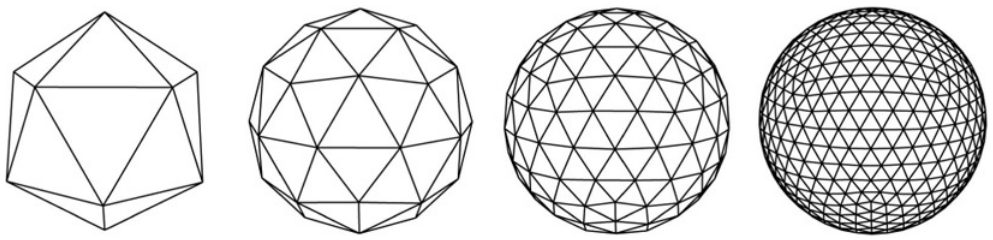

Figure 4.1: This illustration shows the geodesic spherical mesh generated by repeatedly refining an icosahedron

4.1 Analytic Sphere Model

The analytic sphere model provides a large number of test cases with which to verify the code in addition to the traditional Born Ball Model (a single point charge placed at the origin of spherical region). The program package for this model takes the atom locations and charges from a PQR file, adjusts the atom coordinates so the center of the atom structure is at the origin of a cartesian grid, and scales the magnitude of each atom position to be, at most, 80 percent of a user-specified radius. This code was then modified to generate the resulting calculations for Ψ at the centroids of a provided geodesic spherical mesh. In this way we can compare the analytic results to the numeric results at the exact same mesh points.

Fortran 90 was used to create a spherical mesh generator to run these analytic models. The program creates a spherical mesh by repeatedly refining an icosahedron scaled to a user-specified radius, as in Figure 4.1. Therefore, the same mesh data could be supplied to the analytic model that was supplied to our solver. The tests were performed using the PQR files for nine different proteins to serve as a random sample of point charges within a sphere. Thus, we test nine cases of an assortment of atoms, ranging from 488 to 8732 atoms, inside a sphere representing the protein surface of radius 2. We used the proteins defined by their protein data bank (PDB) [17] IDs: 2LZX, 1AJJ, 1FXD, 1HPT, BPTI, 1SVR, 1A63, 1CID, 1A7M, and 1F6W.

The PDB file format provides the macromolecular structure data. It includes all the necessary information to describe the 3D shape of a protein in terms of atom type, atom

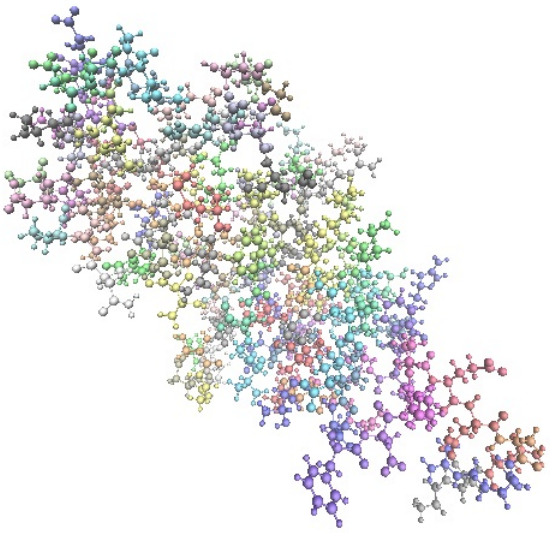

Figure 4.2: A VMD generated graphic of the atom structure of protein ID 1A63

position, atom charge, atom radius, etc. This file, for our purposes, is converted into a PQR by the software PDB2PQR available through [16] which contains only the information on atom position, atom charge, and atom radius. For example, the protein 2LZX describes the structure of Asteropsin B from a marine sponge and is proposed as a scaffolding in oral peptide drug adminstration. The protein 1A63 is the RNA binding domain of E.Coli rho factor. Using the PDB file we can generate a graphic to see the structure of these proteins using the VMD software. In Figure: 4.2 we see the atoms of protein 1A63.

We use two different error measurements to gauge the accuracy of our model. The absolute error we define by

$$e_{abs} = \left(\sum_{j=1}^{N} |\Psi^a(x_j) - \Psi(x_j)|^2 \right)^{1/2} \tag{4.4}$$

and the relative error defined by

$$e_{rel} = \left(\frac{\sum_{j=1}^{N} |\Psi^a(x_j) - \Psi(x_j)|^2}{\sum_{j=1}^{N} |\Psi^a(x_j)|^2} \right)^{1/2} \tag{4.5}$$

where Ψ^a represents the analytic solution, and Ψ represents the numerical solution. The results were compiled for each protein as shown in Table 4.1. The errors for each icosahedral

refinement level were then averaged across the nine proteins simulated in the analytic model. This gives a better picture of the performance of the code written in compared to the traditional Born model which considers only one point charge at the center of a sphere.

Protein 2LZX (488 atoms)		
Number of Surfaces	**Absolute Error**	**Relative Error**
20	27.78978	1.73018E-001
80	26.92327	8.47123E-002
320	28.55830	4.46659E-002
1,280	29.80057	2.32597E-002
5,120	30.49947	1.18959E-002
20,480	33.48362	6.52899E-003
81,920	75.22487	3.33392E-003
327,680	279.16305	1.36082E-003

Table 4.1: A display of the error results from the analytic case of test protein 2LZX for 7 different icosahedral refinements

The graphs depicted in Figure 4.3 and in Figure 4.4 show the results of this averaging. Both plots were generated using MATLABs "loglog" function which allows for a more condensed representation of the data because it spans very large (or very small) numbers. We see an exponential trend in the absolute error, which coincides with the fact that the number of data points in the vector is increasing exponentially with each icosahedral refinement. Comparatively, the relative error is decreasing, indicating the distance between data points is decreasing in comparison to the size of the analytic solution as a whole.

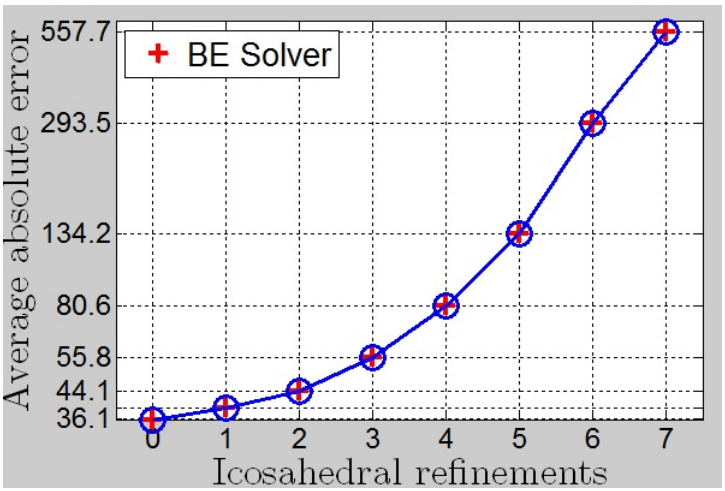

Figure 4.3: The average of the absolute error between the numerical solution and the analytic solution computed for nine different proteins

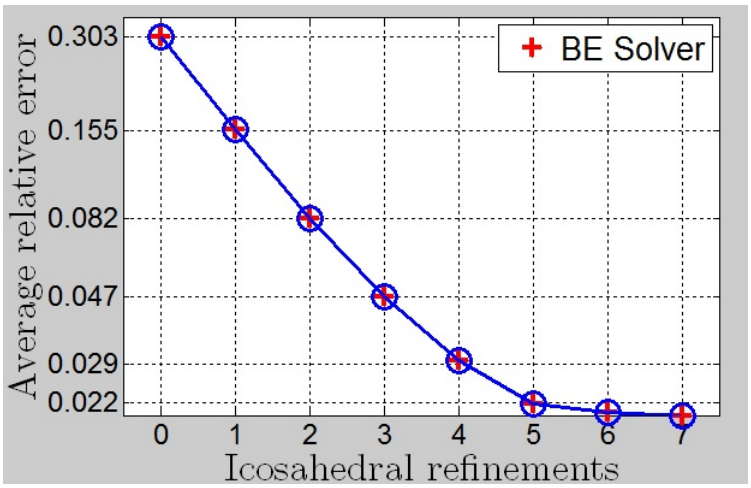

Figure 4.4: The average of the relative error between the numerical solution and the analytic solution computed for nine different proteins

4.2 Protein Simulation

The next step of compiling results comes from investigating how the solver handles these calculations for real world proteins. For this part of the testing, the point charges of the proteins were not centered, rescaled, or otherwise modified. In an effort to verify the solver's

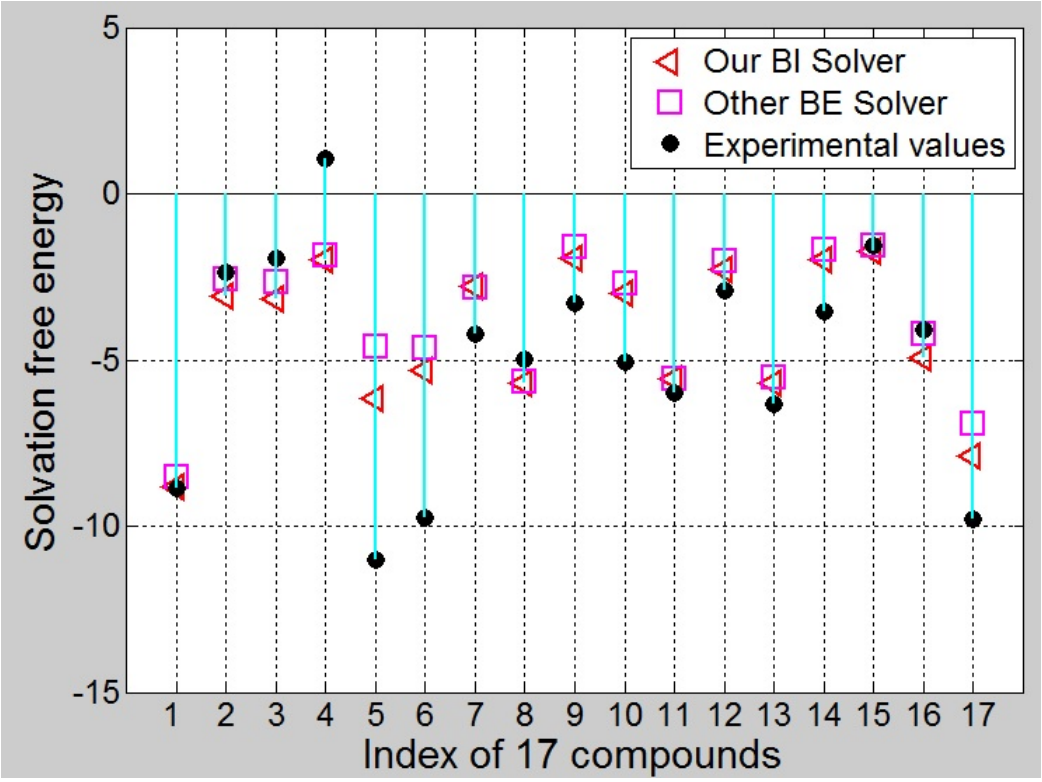

Figure 4.5: A comparison of the free energy of solvation calculations between our boundary integral solver, another boundary integral solver, and experimental data for 17 different proteins

accuracy for the complex surfaces of biomolecules, these test simulations were performed on the protein as is described in its PQR file.

Mesh generation for the surface of a protein is a much more involved issue and one of great research importance. For this project, we use the mesh generation software MSMS [12] to generate a discretized triangular surface for the protein. The MSMS software essentially creates this surface by rolling a sphere of a user-specified "probe-radius" around the atoms in the protein. We define this probe-radius as 1.4 angstroms. It is a typical value to be chosen as it approximates the radius of a water molecule.

The surface of a protein can be described in a number of ways. As described in [11] there is the Van der Waals surface, the topological boundary for the overlapping spheres for each atom, the Solvent Accessible Surface (SAS), which is defined by the center of a sphere as it rolls over the van der Waals surface, and the Solvent Excluded Surface (SES) which is much

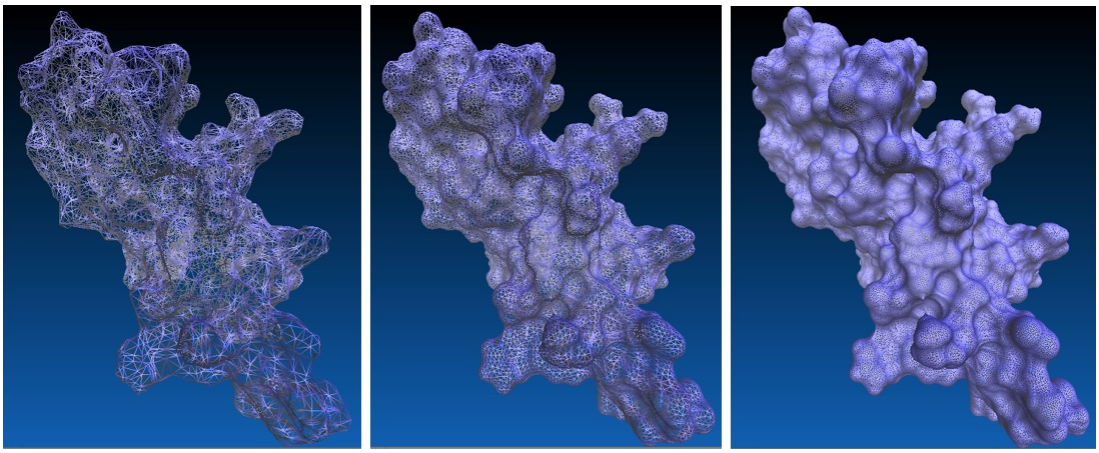

Figure 4.6: A VMD generated image showing three different surface meshes generated by the MSMS software. From left to right they were created using densities of 1, 5 and 20.

more complicated to define. MSMS creates a SAS for the protein in question to a certain so-called "surface density". Providing the software with a higher density yields a finer mesh. In Figure 4.6 we see three different meshes produced by the MSMS software. From left to right we have densities 1, 5, and 10, with a total number of surface points of 20,489, 70,201, and 265,947 respectively

We first test the efficacy of our solver for real-world applications by running simulations on a set of 17 different compounds to compute the electrostatic free energy of solvation for which experimental data was reported in [10]. This calculation is of particular interest to those studying solvated biomolecules. We use the formula as given in [5] to convert our dimensionless solution into the physical units for this model be meaningful in real world applications. This will generates a solvation energy with units of kcal/mol. The free energy of solvation, denoted E_{es}, is given by

$$E_{es} = \frac{N_A}{4184}\frac{k_B T}{2e_c}\sum_{j=1}^{n_p} z_j \Psi(\mathbf{r}_j) \tag{4.6}$$

where we use (4.1) to calculate Ψ at each atom position. Figure 4.5 compares the results of our solver to the experimental data alongside the results produced by another boundary integral solver for the Linearized PBE [1]. It can be seen that both codes provide reasonable

estimates to the measured data, but in the majority of the cases our solver produced more accurate results. The results are also displayed in Table 4.2 which also shows the absolute value of the difference between the calculated and experimental values.

Index	Experimental	Our Solver	Difference	Other Solver	Difference
1	-8.84	-8.83	0.01	-8.53	0.34
2	-2.38	-3.08	0.70	-2.59	0.21
3	-1.93	-3.17	1.24	-2.67	0.74
4	1.07	-1.97	3.04	-1.85	2.92
5	-11.01	-6.18	4.83	-4.62	6.39
6	-9.76	-5.32	4.44	-4.65	5.11
7	-4.23	-2.78	1.45	-2.84	1.39
8	-4.97	-5.68	0.71	-5.67	0.70
9	-3.28	-1.92	1.36	-1.61	1.67
10	-5.05	-3.00	2.05	-2.68	2.37
11	-6.00	-5.58	0.42	-5.59	0.41
12	-2.93	-2.29	0.64	-2.03	0.90
13	-6.34	-5.70	0.64	-5.53	0.81
14	-3.54	-1.99	1.55	-1.69	1.85
15	-1.55	-1.71	0.16	-1.55	0
16	-4.08	-4.95	0.87	-4.21	0.13
17	-9.81	-7.88	1.93	-6.90	2.91

Table 4.2: Results of the simulation run for 17 compounds with our solver, another solver, and comparisons to the experimental data

The last aspect of this solver that we would like to highlight is the reduction in solution time. For this purpose we use the protein 1A63 and the 17th protein from the above set labeled 17_s5_Zap. For these two proteins we compute the electrostatic potential for a series of increasing MSMS surface densities by both treecode and direct solver. For the treecode case we set the MAC parameter, $\theta = 0.8$, and we set $\theta = 0$ for the direct solver case. The results are shown in Table 4.3 and Table 4.4. Figure 4.7 graphically displays the data for protein 1A63 and Figure 4.8 graphically displays the results for protein 17_s5_Zap.

Protein 1A63 (2065 atoms)				
	Treecode		Direct Solver	
Surface Points	**Solve Time**	**Iterations**	**Solve Time**	**Iterations**
20,489	55.37	23	848.71	23
30,301	71.17	19	1824.38	19
70,201	173.16	17	9378.4	17
132,159	494.13	24	29,104.2	15

Table 4.3: Solution time for the electrostatics of protein 1A63 using treecode and using the direct solver alongside the total number of GMRES iterations.

Protein 17_S5_Zap (9 atoms)				
	Treecode		Direct Solver	
Surface Points	**Solve Time**	**Iterations**	**Solve Time**	**Iterations**
206	8.35E-003	5	1.70E-002	5
312	1.81E-002	5	3.88E-002	5
760	8.42E-002	5	0.22	5
1,456	0.21	5	0.77	5
3,020	0.47	5	3.63	5
6,310	1.18	5	16.34	5
12,996	2.82	5	69.99	5

Table 4.4: Solution time for the electrostatics of protein 17_S5_Zap using treecode and using the direct solver alongside the total number of GMRES iterations

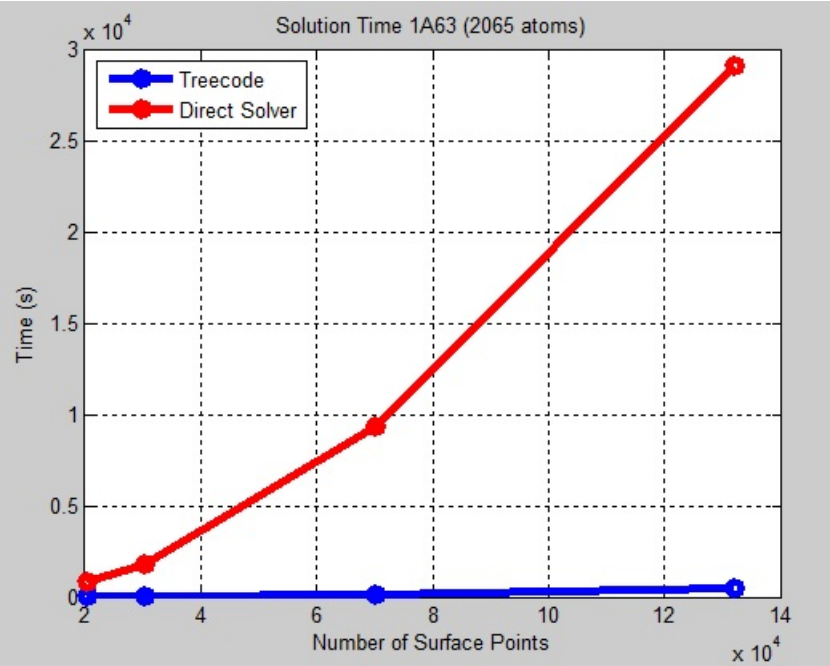

Figure 4.7: A comparison of solution time between a direct solver and the treecode accelerated solver for the protein 1A63 (2065) atoms versus number of surface points.

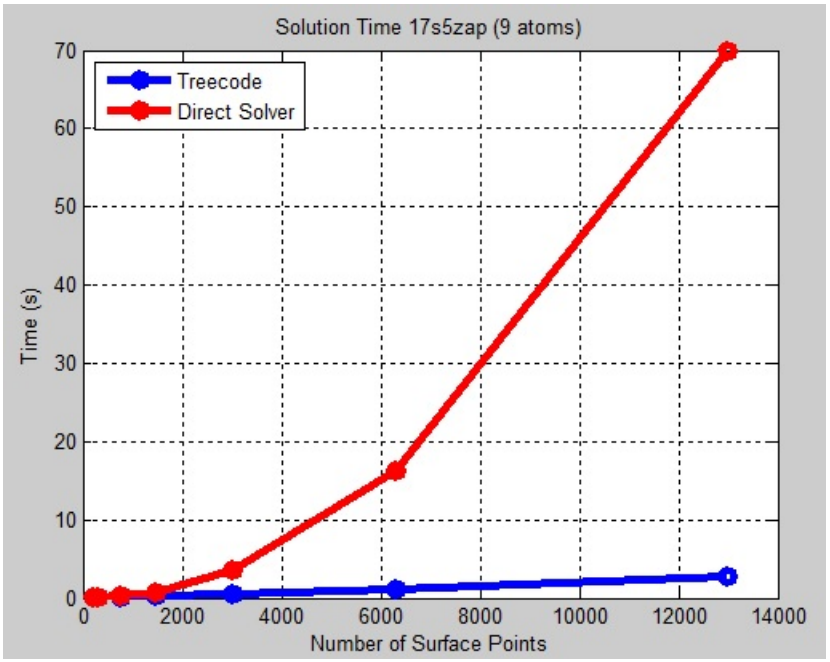

Figure 4.8: A comparison of solution time between a direct solver and the treecode accelerated solver for the protein 17 (9) atoms versus number of surface points

Chapter 5

Conclusions and Future Work

This chapter illustrates the importance of the results mentioned in the previous chapter. In particular, we discuss the validity of the tests and the theory presented in this thesis. Moreover, the importance of this project will be made clear as a first step towards a potential larger project that tackles all three components of the solution decomposition mentioned in Chapter 2. Further investigation of the techniques and theory of boundary element and hybrid solution methods for the nonlinear case would also prove a significant contribution to the literature and the fields concerned.

5.1 Conclusions

In this thesis an adaptive treecode-accelerated boundary integral solver was presented for PBE. Specifically, we applied the boundary integral method to a solution decomposition method for the water case.

The derivation for the single boundary integral was presented using Green's functions and the boundary conditions. We made explicit the basic idea of the treecode structure for accelerating the calculation of N-body potentials (matrix-vector products), and in this way, a simple algorithm was presented.

The solver was tested against a new class of analytic solutions. This provided the advantage of computing the entire solution vector at specific mesh points for a number of test cases, rather than simply comparing one calculation for solvation energy. In this way we were able to analyze the error reduction of the method. The sphere was generated using

a refined icosahedron programmed in Fortran 90. We note that for the averaged nine test cases, the relative error reduced by about a factor of two with each refinement.

We also tested the solution to verify the reduction in time complexity using protein 1A63. The graphs and tabled results clearly show that the solution time of the direct solver increases at least quadratically. In sharp contrast, the solution time when the treecode is employed behaves logarithmically.

The method applied to the solution decomposition behaved as expected. We see a good demonstration of the reduction in time complexity, as well as the accuracy of the solver. More importantly, we have a solid framework from which to construct an advanced solver for the entire nonlinear problem. The improved numerical accuracy from the boundary-integral approach should be particularly beneficial in further studies.

5.2 Future Work

The potential for projects stemming from these results is profound and numerous. The solvability of PBE using a boundary element technique with the solution decomposition provides new challenges. The nonlinear PBE has already been solved by Finite Difference, Finite Element, and even Hybrid Finite Element and Finite Difference Methods. We have noted the solution of the linearized PBE by boundary elements, employing both treecode and fast multipole methods. Therefore, improving the current model, and extending these results to the nonlinear case of particular research interest and importance.

There are two main issues with this implementation that could be improved. The first was the quadrature technique we employed. There have been boundary element methods presented using curved boundary elements [22]. With any quadrature scheme we will have to handle the case of the singularity of the kernel function that arises in the integral. It would certainly be a worthwhile endeavor to investigate different approaches for handling the singularity as well as different quadrature schemes for discretizing the problem. The second, is that the right hand side of the linear system to be solved is in and of itself an

N-body problem. That is, the set up of the system requires $\mathcal{O}(N^2)$ operations. This could be remedied with a treecode type algorithm, or perhaps the solution would manifest itself in a hybrid boundary/finite element method over which the right hand side could be calculated quickly with more existing grid-based methods.

Hence, the primary difficulty for considering the nonlinear case is that the system can not be reduced to strictly boundary integral equations. That is, we will need to consider integrals defined over the whole space. Any future work will require a hybrid method that can calculate the necessary volume integrals and integrate them with the boundary solution. The solution of $\tilde{\Phi}$ also affords the opportunities to include the research associated with Kernel-Free techniques as already investigated in [21].

We note that this project has very important results for the fields it concerns as well as for the advancement and embellishment of scientific computing methods. The results presented are encouraging and lay the groundwork to investigate the aforementioned techniques.

Chapter 6

Bibliography

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

[2] D. Xie, L. R. Scott, A new boundary integral equation for molecular electrostatics with charges over whole space, BIT Numerical Mathematics 51 (4) (2011) 1051–1071.

[3] J. Li, D. Xie, A new linear poisson-boltzmann equation and finite element solver by solution decomposition approach, Communications in Mathematical Sciences, Vol. 13, No.2, pages 315-325.

[4] J. Barnes, P. Hut, A hierarchial $\mathcal{O}(N \log N)$ force-calculation algorithm, Nature 324 (1986) 446-449

[5] Y. Jiang, Y. Xie, J. Ying, D. Xie, Z. Yu, SDPBS web server for calculation of electrostatics of ionic solvated biomolecules, Molecular Based Mathematical Biology, Vol. 3, pages 179-196, 2015

[6] A. Juffer, E. Botta, B. van Keulen, A. van der Ploeg, H. Berendsen, The electric potential of a macromolecule in a solvent: a fundamental approach, Journal of Computational Physics. 97 (1991) 144?171.

[7] P. Li, H. Johnston, R. Krasny, A Cartesian treecode for screened Coulomb interactions, Journal of Computational Physics. 228 (2009) 3858–3868.

[8] C. Johnson, Numerical Solution of Partial Differential Equations by the Finite Element Method, Chapter 10, 214–224, Dover Publishing, 1987, 2009

[9] D. Xie, H. Volkmer, J. Yang, Analytical Solutions of Two Nonlocal Poisson Dielectric Test Models with Spherical Solute Region Containing Multiple Point Charges, Physical Review E, Vol. 93, pages 319-334, 2016

[10] A. Nicholls, D. Mobley, J. Guthrie, J. Chodera, C. Bayly, M. Cooper, V. Pande, Predicting Small-Molecule Solvation Free Energies: An Informal Blind Test for Computational Chemistry, Journal of Medicinal Chemistry. 51 (2008) 769–779

[11] M. Sanner, A. Olson, Reduced surface: an efficient way to compute molecular surfaces, Biopolymers, Vol. 38, (3), 305-320, 1996

[12] MSMS website. mgl.scripps.edu/people/sanner/html/msms_home.html.

[13] Netlib repository. http://www.netlib.org/.

[14] Delphi Webserver. http://compbio.clemson.edu/sapp/delphi_webserver/

[15] UHBD Software. http://www.swmath.org/software/8088

[16] APBS. http://www.poissonboltzmann.org/docs/calculating/

[17] RCSB Protein Data Bank. http://www.rcsb.org/

[18] AFMPB Documentation. http://lsec.cc.ac.cn/ lubz/Publication/Manual.pdf

[19] D. Xie and Y. Jiang, A Nonlocal Modified Poisson-Boltzmann Equation and Finite Element Solver for Computing Electrostatics of Biomolecules, Journal of Computational Physics, Vol. 322, pages 1-20, 2016.

[20] J. Li and D. Xie, An Effective Minimization Protocol for Solving a Size-modified Poisson-Boltzmann Equation for Biomolecule in Ionic Solvent, International Journal of Numerical Analysis and Modeling, Vol. 12, No. 2, pages 286-301, 2015.

[21] W. Ying, A Kernel-free boundary integral method for the nonlinear poisson-boltzmann equation. Submitted to Journal of Computational Physics (in revision), 2014

[22] M. Altman, J. Bardhan, J. White,B. Tidor, Accurate solution of multi-region continuum biomolecule ectrostatic problems using the linearized poisson-boltzmann equation with curved boundary elements, Wily InterScience, 2008

[23] Y. Saad, M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonysmmetric linear systems, SIAM J. Sci. Stat. Comput, Vol 7, No 3, July 1986.