AFMPB: Adaptive Fast Multipole Poisson-Boltzmann Solver

User's Guide and Programmer's Manual

Release Version Beta

Benzhuo Lu*,[1]

Xiaolin Cheng[2]

Jingfang Huang[3]

J. Andrew McCammon[4]

[1] State Key Laboratory of Scientific/Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China.

[2] Oak Ridge National Lab and the University of Tennesse.

[3] Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599-3250.

[4] Department of Chemistry and Biochemistry, Center for Theoretical Biological Physics, Department of Pharmacology, Howard Hughes Medical Institute, University of California at San Diego, La Jolla, California 92093-0365.

* Comments or requests concerning the AFMPB program can be addressed to: Dr. Ben-Zhuo Lu, Tel.: 086 01 62626492; fax: 086 01 62542285. Email:

bzlu@lsec.cc.ac.cn.

AFMPB

Adaptive Fast Multipole Poisson-Boltzmann Solver

Contact information

Benzhuo Lu (bzlu@lsec.cc.ac.cn)

State Key Laboratory of Scientific/Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing

Academy of Mathematics and Systems Science, Chinese Academy of Sciences

Beijing 100190, China

Authors:

Benzhuo Lu (bzlu@lsec.cc.ac.cn)

Xiaoling Cheng (chengx@ornl.gov)

Jingfang Huang (huang@amath.unc.edu)

James Andrew McCammon (jmccammon@ucsd.edu)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1. What is AFMPB

Adaptive Fast Multipole Poisson-Boltzmann (AFMPB) solver is a numerical simulation package for solving the linearized Poisson-Boltzmann (LPB) equation which models electrostatic interactions in biomolecule systems. In this package, a boundary integral equation (BIE) approach is applied to discretize the LPB equation. The resulting integral formulas are well conditioned for single molecule cases as well as for systems with more than one macromolecule, and are solved efficiently using Krylov subspace based iterative methods such as generalized minimal residual (GMRES) or biconjugate gradients stabilized (BiCGStab) methods. In each iteration, the convolution type matrix-vector multiplications are accelerated by a new version of the fast multipole method (FMM). The implemented algorithm is asymptotically optimal $O(N)$ both in CPU time and memory usage with optimized prefactors. This package enhances the present computational ability to treat electrostatics of large scale systems in protein-protein interactions and nano particle assembly processes. AFMPB has been tested on different biomolecule systems including the nicotinic acetylcholine receptor (nAChR), and interactions between protein Sso7d and DNA.

AFMPB development was started by Ben-Zhuo Lu (AMSS), Xiaolin Cheng (ORNL), Jingfang Huang (UNC-CH), and Andrew McCammon (UCSD) in 2004. This package is released freely under open source licenses for maximal benefit to the biophysics and biochemistry, mathematics, and other science and engineering communities. The AFMPB package utilizes two outside open source packages: the iterative solvers from the SparseKit package and the new version of fast multipole algorithms from FMMYuk and FMMLap.

1.2. Getting Started

You need to perform the following steps in order to get simulation results:

1. download the code and compile it on your platform. You need to use Intel Fortran Compiler or GNU F90 and later versions.

2. Prepare the input documents for the simulation.

3. Goto directory **job**, and prepare the file job.csh

4. Run job.csh.

5. Analyze the results.

For example, if you want to find the electrostatic potential for nAChR, after correctly compiling and installing the software, you goto the directory **job**, modify the file **job.csh** or simply use existing job-nAChR.csh, and then execute the csh file.

We will added more features to this package in future releases.

1.3. Recommended Reading

We recommend the following papers to users of this package.

- B. Lu, X. Cheng, J. Huang, and J. A. McCammon, "Order N algorithm for computation of electrostatic interactions in biomolecular systems", PNAS, December 19, 2006; 103(51): 19314 - 19319.

- Lu, Benzhuo; Cheng, Xiaolin; McCammon, J. Andrew "New-version-fast-multipole-method" accelerated electrostatic calculations in biomolecular systems. J. Comput. Phys. 226 (2007), no. 2, 1348–1366.

We recommend the advanced programmers the following papers for understanding the adaptive new version of fast multipole methods used in this package.

- Greengard, Leslie; Rokhlin, Vladimir, "A new version of the fast multipole method for the Laplace equation in three dimensions". Acta numerica, 1997, 229–269, Acta Numer., 6, Cambridge Univ. Press, Cambridge, 1997.

- Cheng, H.; Greengard, L.; Rokhlin, V. “A fast adaptive multipole algorithm in three dimensions.” J. Comput. Phys. 155 (1999), no. 2, 468–498.
- Greengard, Leslie F.; Huang, Jingfang, “A new version of the fast multipole method for screened Coulomb interactions in three dimensions”. J. Comput. Phys. 180 (2002), no. 2, 642–658.

For the Krylov subspace iterative methods, we used the “SparseKit” package developed by Kesheng John Wu and Professor Yousef Saad. The “reverse communication protocol” is a very convenient feature in this package which makes it easy to interface with our FMM accelerated matrix vector multiplication subroutines. Please check the SparseKit website for documents and source codes.

http://www-users.cs.umn.edu/~ saad/software/SPARSKIT/sparskit.html

2. METHODS

In this chapter, we discuss the technical details of the algorithm. In particular, the boundary integral equation formulation and its discretization, the adaptive new version of FMM and how it is applied to the BEM, the Krylov subspace method, and the mesh generation.

2.1. A Brief History

Although the continuum Poisson-Boltzmann (PB) equation model for these systems was introduced almost a century ago by Debye and Hückel[16] and later developed by Kirkwood,[25] its numerical solutions have only been extensively explored in the last two decades.[13,15,18,22,23,26,40,46,48] Traditional schemes include the finite difference methods where difference approximations are used on structured grids describing the computational domain, and finite element methods in which arbitrarily shaped biomolecules are discretized using elements and associated basis functions. The resulting algebraic systems for both are commonly solved using multigrid or domain decomposition accelerations for optimal efficiency. However, as the grid number (and thus the storage, number of operations, and condition number of the system) increases proportionally to the volume size, finite difference and finite element methods become less efficient and accurate for systems of large spatial sizes commonly encountered in studying either macromolecules or interacting systems in the association and dissociation processes. Alternative methods include the boundary element (BEM) and boundary integral equation (BIE) methods. In these methods, only the surfaces (compared to the 3D volume) of the molecules are discretized and hence the number of unknowns is greatly reduced. In addition, the boundary elements (BEs) form a kind of surface "conforming" mesh (because they align with the surface), which therefore allows the application of the BEM to biomolecules characterized by irregular geometries, while maintaining a high level of calculation accuracy.

However, in practical biomodeling, the BEM is the least used relative to the other methods. In earlier versions of BEM, the integral equation formulations may not have been well-conditioned and the matrix was typically stored explicitly. The resulting dense linear system was often solved using direct matrix inversion such as Gauss elimination, so that $O(N^2)$ storage and $O(N^3)$ operations were required, where N is the number of unknowns defined on the surface. This is extremely inefficient for any typical size system of interest. To improve the BEM efficiency, some later studies improved the condition of the integral formulation,[2,23,28,29] reduced the number of the boundary elements[44] or introduced novel BEM.[30] It has been dmonstrated that when the system is well-conditioned or can be effectively preconditioned, the matrix equations can be solved efficiently using iterative Krylov subspace methods which are matrix-implicit, thus eliminating the bottleneck of storage. As the number of iterations in these methods is independent of the system size for well conditioned systems, the computational cost is then dominated by the matrix-vector multiplication calculations corresponding to the N-body electrostatic particle interactions of both the Coulombic ($\kappa = 0$) and screened Coulombic ($\kappa \neq 0$) types, which require $O(N^2)$ operations using direct methods for each iteration. By introducing novel fast summation algorithms developed in the last twenty years, this cost has been reduced to $O(N \log N)$ or the asymptotically optimal $O(N)$. These algorithms incude the hierarchical "tree code",[4,6] fast Fourier transform (FFT) based algorithms such as the precorrected FFT (pFFT)[3,27,34] and the particle-mesh Ewald (PME) methods,[14] the hierarchical SVD method,[24] and FFT on multipoles.[32,33] Further improvements show that asymptotically optimal $O(N)$ complexity can be achieved by using the wavelet techniques[41,43] or the fast multipole method (FMM).[19] FMM algorithms for the screened Coulombic interaction (Yukawa potential) have also been recently developed,[9,21] which allows their direct application to the solution of the PB equation. The tree code algorithm and the FMMs based on the old scheme[19] have been implemented in former BEM PB work.[7,8,10,17,33,49] However, as revealed by previous numerical experiments, although asymptotically optimal, the original FMM[19] turns out to be less efficient for problem sizes of current interest when compared with the tree code and FFT based $O(N \log N)$ techniques, due to the huge prefactor in $O(N)$.[17]

To further accelerate the numerical solution of the PB equation, our AFMPB solver uses an efficient algorithm based on a well conditioned BIE formulation, for which the solution is accelerated by a new version of FMM first introduced by Greengard and Rokhlin[20] for the Laplace equation. By proper coupling of single and double layer potentials as in reference,[37] a Fredholm second kind integral equation formulation for the PB equation can be derived. Similar formulations were first introduced by Juffer et al.[23] who aimed to avoid the singularity problem in deriving the complete BIE forms for the linearized PB equation. The well-conditioned property has been discussed by Liang et al.,[28] and also demonstrated in the work of Boschitsch et al.[10] We extend the formulation for systems with an arbitrary number of domains in AFMPB. Compared with traditional BEM formulations, the condition number of our BIE system does not increase with the number of unknowns, hence the number of iterations in the Krylov subspace based methods is bounded. For the matrix vector multiplication in each iteration, we use the new version FMM developed for the screened Coulombic interaction (Yukawa potential).[21] Compared with the original FMM, the plane wave expansion based diagonal translation operators dramatically reduce the prefactor in the $O(N)$ new version FMM, especially in three dimensions where a break-even point of approximately 600 for 6 digits precision is numerically observed. Perhaps due to its complexity in theory and programming, we are unaware of any previous implementations of the new version FMM for the PB equation.

2.2. Boundary integral equation formulations

When Green's second identity is applied, traditional boundary integral equations for the linearized PB equation for a single domain (molecule) take the form

$$\frac{1}{2}\phi_p^{\text{int}} = \oint_S^{PV} [G_{pt}\frac{\partial \phi_t^{\text{int}}}{\partial n} - \frac{\partial G_{pt}}{\partial n}\phi_t^{\text{int}}]dS_t + \frac{1}{D_{\text{int}}}\sum_k q_k G_{pk}, \quad p \in S, \tag{2.1}$$

$$\frac{1}{2}\phi_p^{\text{ext}} = \oint_S^{PV} [-u_{pt}\frac{\partial \phi_t^{\text{ext}}}{\partial n} + \frac{\partial u_{pt}}{\partial n}\phi_t^{\text{ext}}]dS_t, \quad p \in S, \tag{2.2}$$

where ϕ_p^{int} is the interior potential at surface position p of the molecular domain Ω, $S = \partial\Omega$ is its boundary, i.e., solvent-accessible surface, ϕ_p^{ext} is the exterior potential at position p, D_{int} is the interior dielectric constant, t is an arbitrary point on the boundary, n is the outward normal vector at t, and PV represents the principal value integral to avoid the singular point when $t \to p$ in the integral equations. In the formulae, $G_{pt} = \frac{1}{4\pi|r_t - r_p|}$ and $u_{pt} = \frac{\exp(-\kappa|r_t - r_p|)}{4\pi|r_t - r_p|}$ are the fundamental solutions of the corresponding Poisson and Poisson-Boltzmann equations, respectively, r_k is the position of the kth source point charge q_k of the molecule, κ is the reciprocal of the Debye-Hückel screening length determined by the ionic strength of the solution. These equations can be easily extended to multi-domain systems in which Eq. (2.1) is enforced for each individual domain and the integration domain in Eq. (2.2) includes the collection of all boundaries.[31]

To complete the system, the solutions in the interior (Eq. (2.1)) and exterior (Eq. (2.2)) are matched by the boundary conditions $\phi^{\text{int}} = \phi^{\text{ext}}$ and $D_{\text{int}}\frac{\partial \phi^{\text{int}}}{\partial n} = D_{\text{ext}}\frac{\partial \phi^{\text{ext}}}{\partial n}$, where D_{ext} is the exterior (solvent) dielectric constant. Using these conditions, we can define $f = \phi^{\text{ext}}$ and $h = \frac{\partial \phi^{\text{ext}}}{\partial n}$ as the new unknowns and recover other quantities using boundary integrals of f and h. Unfortunately, theoretical analysis shows that the corresponding equation system for f and h is in general a Fredholm integral equation of first kind and hence ill-conditioned. i.e., when solved iteratively using Krylov subspace methods, the number of iterations increases with the number of unknowns, and the resulting algorithm becomes inefficient for large systems. Instead of this "direct formulation", Rokhlin[37] introduced a technique where the single and double layer potentials are combined in order to derive an optimized second kind Fredholm integral equation. We want to mention that a well-conditioned form actually appeared in Juffer et al.'s work[23] when they tried to derive the complete BI form for linearized PBE using a limiting process to avoid the singularity problem. The same form has been used and discussed in later BEM PB work,[10,28] and similar techniques have also been applied and discussed in engineering computations.[42]

The derivative BEM (dBEM) can be obtained by linearly combining the derivative forms of Eqs. (2.1)-(2.2):

$$(\frac{1}{2\varepsilon}+\frac{1}{2})f_p = \oint_S^{PV}[(G_{pt}-u_{pt})h_t - (\frac{1}{\varepsilon}\frac{\partial G_{pt}}{\partial n}-\frac{\partial u_{pt}}{\partial n})f_t]dS_t + \frac{1}{D_{\text{ext}}}\sum_k q_k G_{pk}, \quad p\in S, \tag{2.3}$$

$$(\frac{1}{2}+\frac{1}{2\varepsilon})h_p = \oint_S^{PV}[(\frac{\partial G_{pt}}{\partial n_0}-\frac{1}{\varepsilon}\frac{\partial u_{pt}}{\partial n_0})h_t - \frac{1}{\varepsilon}(\frac{\partial^2 G_{pt}}{\partial n_0 \partial n}-\frac{\partial^2 u_{pt}}{\partial n_0 \partial n})f_t]dS_t + \frac{1}{D_{\text{ext}}}\sum_k q_k \frac{\partial G_{pk}}{\partial n_0}, \quad p\in S. \tag{2.4}$$

where n_0 is the unit normal vector at point p, $\varepsilon = D_{\text{ext}}/D_{\text{int}}$. This set of BIEs leads to a well-conditioned system of algebraic equations, which we will adopt.

For a system with an arbitrary number, e.g. J, of separate domains (molecules) surrounded by infinite homogeneous solvent, Eq. (2.1) holds and the integration can be performed only over one molecular surface where the evaluation point p is located, while the integrand in Eq. (2.2) is the combination of all the molecular surfaces. Following the same treatment, and supposing $p \in S^i$, the derivative BIEs for multiple domains are extended as:

$$(\frac{1}{2\varepsilon}+\frac{1}{2})f_p = \overset{PV}{\oint_{S^i}}[(G_{pt}-u_{pt})h_t - (\frac{1}{\varepsilon}\frac{\partial G_{pt}}{\partial n}-\frac{\partial u_{pt}}{\partial n})f_t]dS_t$$
$$+\sum_{j\neq i}\oint_{S^j}[-u_{pt}h_t+\frac{\partial u_{pt}}{\partial n}f_t]dS_t+\frac{1}{D_{\text{ext}}}\sum_{k^i}q_{k^i}G_{pk^i}, \quad p\in S^i, i=1,...J, \tag{2.5}$$

$$(\frac{1}{2}+\frac{1}{2\varepsilon})h_p = \overset{PV}{\oint_{S^i}}[(\frac{\partial G_{pt}}{\partial n_0}-\frac{1}{\varepsilon}\frac{\partial u_{pt}}{\partial n_0})h_t - \frac{1}{\varepsilon}(\frac{\partial^2 G_{pt}}{\partial n_0 \partial n}-\frac{\partial^2 u_{pt}}{\partial n_0 \partial n})f_t]dS_t$$
$$+\sum_{j\neq i}\oint_{S^j}\frac{1}{\varepsilon}[-\frac{\partial u_{pt}}{\partial n_0}h_t+\frac{\partial^2 u_{pt}}{\partial n_0 \partial n}f_t]dS_t+\frac{1}{D_{\text{ext}}\varepsilon}\sum_{k^i}q_{k^i}\frac{\partial G_{pk^i}}{\partial n_0}, \quad p\in S^i, i=1,...J. \tag{2.6}$$

However, it is noticed that in Eqs. (2.5) and (2.6) the integrand kernels for integration on surface S^i (enclosed in the first pair of square brackets) are not the same as those on molecular surface S^j (enclosed in the second pair of square brackets). This is not convenient for application of the FMM. The FMM algorithm uses hierarchical levels of boxes to group all the evaluation points (meshes), so it would be beneficial to have similar integral formulae on all molecular surfaces for every evaluation point. If we apply Green's second theorem to domain S^j, and still let $p\in S^i, i\neq j$, it is found that the following set of equations hold

$$0=\oint_{S^j}[G_{pt}h_t-\frac{1}{\varepsilon}\frac{\partial G_{pt}}{\partial n}f_t]dS_t+\frac{1}{D_{\text{ext}}}\sum_{k^j}q_{k^j}G_{pk^j}, \quad p\in S^i, i=1,...J, \tag{2.7}$$

$$0=\oint_{S^j}[\frac{\partial G_{pt}}{\partial n_0}h_t-\frac{1}{\varepsilon}\frac{\partial^2 G_{pt}}{\partial n_0 \partial n}f_t]dS_t+\frac{1}{D_{\text{ext}}}\sum_{k^j}q_{k^j}\frac{\partial G_{pk^j}}{\partial n_0}, \quad p\in S^i, i=1,...J. \tag{2.8}$$

Combining these equations for different boundary j, it is found that Eqs. (2.5)-(2.6) have another neat form

$$
\begin{aligned}
(\frac{1}{2\varepsilon}+\frac{1}{2})f_p = \sum_j^J \int_{S^j}^{PV} [(G_{pt}-u_{pt})h_t - (\frac{1}{\varepsilon}\frac{\partial G_{pt}}{\partial n}-\frac{\partial u_{pt}}{\partial n})f_t]dS_t \\
+\frac{1}{D_{\text{ext}}}\sum_j\sum_{k^j} q_{k^j}G_{pk^j}, \quad p\in S^i, i=1,...J, \qquad (2.9)
\end{aligned}
$$

$$
\begin{aligned}
(\frac{1}{2}+\frac{1}{2\varepsilon})h_p = \sum_j^J \int_{S^j}^{PV} [(\frac{\partial G_{pt}}{\partial n_0}-\frac{1}{\varepsilon}\frac{\partial u_{pt}}{\partial n_0})h_t - \frac{1}{\varepsilon}(\frac{\partial^2 G_{pt}}{\partial n_0 \partial n}-\frac{\partial^2 u_{pt}}{\partial n_0 \partial n})f_t]dS_t \\
+\frac{1}{D_{\text{ext}}}\sum_j^J\sum_{k^j} q_{k^j}\frac{\partial G_{pk^j}}{\partial n_0}, \quad p\in S^i, i=1,...J. \qquad (2.10)
\end{aligned}
$$

Now, all the calculated points can treated uniformly by this set of equations, which is similar to the case of one molecule. This is a set of well-conditioned Fredholm second kind integral equation formulations for multi-biomolecule systems. As a matter of fact, it can be more straightforward to obtain the derivative BIEs for multi-domain cases from the single domain equations, because Eqs. (2.1)-(2.4) hold not only for a single closed boundary surface, but also for any combination of separated boundaries. Compared with Eqs. (2.5)-(2.6), Eqs. (2.9)-(2.10) add more operations in the integrals and summations. But these additional operations only account for a very small part of the whole computational cost for solving the PB equation, and the summations in Eqs. (2.9)-(2.10) are also efficiently accelerated by using the FMM. In addition, as mentioned above, the symmetrized integral formulations also make the coding convenient and easy-to-maintain. It is worth noting that for the case when the interior dielectric constants are different for different molecular domains, and are same in exterior domain, a set of formulae very similar to Eqs. (2.9)-(2.10) are still available, except for that the coefficient ε is to be replaced by ε_j because it varies for different molecular surface integrals. In this case, the FMM still applies because the Green's functions are the same, but it needs to separate the terms associated with ε_j and rescale f and h to absorb ε_j on different molecular surfaces, then use the FMM. The case with different dielectric constants was studied in a recent BEM paper.[47]

2.3. Discretization of the BIEs

Similar to the reference[31] the discretized form of the BIEs (2.9)-(2.10) can be written as:

$$(\frac{1}{2\varepsilon}+\frac{1}{2})f_p = \sum_t^T (A_{pt}h_t - B_{pt}f_t) + \frac{1}{D_{\text{ext}}}\sum_k q_k G_{pk}, \tag{2.11}$$

$$(\frac{1}{2}+\frac{1}{2\varepsilon})h_p = \sum_t^T (C_{pt}h_t - D_{pt}f_t) + \frac{1}{D_{\text{ext}}}\sum_k q_k \frac{\partial G_{pk}}{\partial n_0}, \tag{2.12}$$

where T is total number of discretized patches of the combined boundaries, which is half of the total unknowns (f or h) of the system, and here the $\sum_k$ encompasses all the source charges in the considered system. The coefficient matrices are defined as follows:

$$\begin{aligned} A_{pt} &= \int_{\Delta S_t} (G_{pt} - u_{pt})dS, & B_{pt} &= \int_{\Delta S_t} (\frac{1}{\varepsilon}\frac{\partial G_{pt}}{\partial n} - \frac{\partial u_{pt}}{\partial n})dS, \\ C_{pt} &= \int_{\Delta S_t} (\frac{\partial G_{pt}}{\partial n_0} - \frac{1}{\varepsilon}\frac{\partial u_{pt}}{\partial n_0})dS, & D_{pt} &= \int_{\Delta S_t} \frac{1}{\varepsilon}(\frac{\partial^2 G_{pt}}{\partial n_0 \partial n} - \frac{\partial^2 u_{pt}}{\partial n_0 \partial n})dS. \end{aligned} \tag{2.13}$$

where the integrations are performed on the small patch ΔS_t. To obtain above form, it is assumed that the solution f and h are constants in every small patch ΔS_t. For nearby patches p and t, Eq. (2.13) is performed by direct integration. For far field, the kernels for each patch integral are taken as constants (depending on the relative positions of p and t). The linear system can be written in a matrix form:

$$\begin{pmatrix} (\frac{1}{2\varepsilon}+\frac{1}{2})I + B & -A \\ D & (\frac{1}{2}+\frac{1}{2\varepsilon})I - C \end{pmatrix} \begin{pmatrix} f \\ h \end{pmatrix} = \begin{pmatrix} \frac{1}{D_{\text{ext}}}\sum_k q_k G_{pk} \\ \frac{1}{D_{\text{ext}}}\sum_k q_k \frac{\partial G_{pk}}{\partial n_0} \end{pmatrix} \tag{2.14}$$

where I is the identity matrix. The linear system is well-conditioned and can be solved efficiently using Krylov subspace methods. As the number of iterations is bounded, the most time consuming part becomes the convolution type matrix vector multiplication in each iteration. In next section, we discuss how this can be accelerated by the new version FMM.

2.4. New version fast multipole method

The original idea of FMM is to subdivide the summation system of N particles into hierarchical groups of particles, and the potentials produced by far-field particles for a given particle are approximated by using the multipole expansions (Figure 2.1a). The fundamental observation in the multipole expansion based methods is that the numerical rank of the far field interactions is relatively low and hence can be approximated by P terms (depending on the prescribed accuracy) of the so-called "multipole expansion",

$$\Phi(R,\theta,\phi) = \sum_{i=1}^{N} q_i \cdot \frac{1}{|\vec{R}-\vec{\rho}_i|} \approx \sum_{n=0}^{P} \sum_{m=-n}^{m=n} M_n^m \frac{Y_n^m(\theta,\phi)}{R^{n+1}} \tag{2.15}$$

where the multipole coefficients,

$$M_n^m = 8\sum_{i=1}^{N} q_i \cdot Y_n^{-m}(\alpha_i,\beta_i) \tag{2.16}$$

where the spherical harmonic function of order n and degree m is defined according to the formula,[1]

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(2n+1)(n-|m|)!}{4\pi(n+|m|)!}} \cdot P_n^{|m|}(cos\theta)e^{im\phi} \tag{2.17}$$

For the Debye-Hückel (screened Coulombic) interaction, a similar expansion can be written as follows,

$$\Phi(R,\theta,\phi) = \sum_{i=1}^{N} q_i \cdot \frac{e^{-\kappa|\vec{R}-\vec{\rho}_i|}}{|\vec{R}-\vec{\rho}_i|} \approx \sum_{n=0}^{P} \sum_{m=-n}^{m=n} M_n^m \cdot k_n(\kappa R) \cdot Y_n^m(\theta,\phi) \tag{2.18}$$

where the multipole coefficients,

$$M_n^m = 8\kappa\sum_{i=1}^{N} q_i \cdot i_n(\kappa\rho_i) \cdot Y_n^{-m}(\alpha_i,\beta_i) \tag{2.19}$$

where $i_n(r)$ and $k_n(r)$ are modified spherical Bessel and modified spherical Hankel functions respectively. The modified spherical Bessel and modified spherical Hankel functions are defined in terms of the conventional Bessel function via,[1]

$$I_v(r) = i^{-v}J_v(ir), \tag{2.20}$$

$$K_v(r) = \frac{\pi}{2\sin v\pi}[I_{-v}(r) - I_v(r)], \tag{2.21}$$

$$i_n(r) = \sqrt{\frac{\pi}{2r}} I_{n+1/2}(r), \tag{2.22}$$

$$k_n(r) = \sqrt{\frac{\pi}{2r}} K_{n+1/2}(r). \tag{2.23}$$

For arbitrary distributions of particles, a hierarchical oct-tree (in 3D) is generated so each particle is associated with different boxes at different levels, and a divide-and-conquer strategy is applied to account for the far field interactions at each level in the tree structure. In the "tree code" developed by Appel,[4] and Barnes and Hut,[6] as each particle interacts with 189 boxes in its "interaction list" through P terms of multipole expansions at each level and there are $O(\log N)$ levels, the total amount of operations is approximately $189P^2N\log N$. The tree code was later improved by Greengard and Rokhlin in 1987.[19] In their original FMM, local expansions (under a different coordinate system) were introduced to accumulate information from the multipole expansions in the interaction list (Figure 2.1b).

$$\Phi(R,\theta,\phi) = \sum_{i=1}^{N} q_i \cdot \frac{1}{|\vec{R}-\vec{\rho}_i|} \approx \sum_{n=0}^{P}\sum_{m=-n}^{m=n} L_n^m \cdot R^n Y_n^m(\theta,\phi) \tag{2.24}$$

where L_n^m are local expansion coefficients.

$$L_n^m = 8\sum_{i=1}^{N} q_i \cdot \frac{Y_n^{-m}(\alpha_i,\beta_i)}{\rho_i^{n+1}} \tag{2.25}$$

For the screened Coulombic interaction, a similar expansion can be written as follows,

$$\Phi(R,\theta,\phi) = \sum_{i=1}^{N} q_i \cdot \frac{e^{-\kappa|\vec{R}-\vec{\rho}_i|}}{|\vec{R}-\vec{\rho}_i|} \approx \sum_{n=0}^{P}\sum_{m=-n}^{m=n} L_n^m \cdot i_n(\kappa R) \cdot Y_n^m(\theta,\phi) \tag{2.26}$$

where

$$L_n^m = 8\kappa\sum_{i=1}^{N} q_i k_n(\kappa\rho_i) \cdot Y_n^{-m}(\alpha_i,\beta_i), \tag{2.27}$$

As the particles only interact with boxes and other particles at the finest level, and information at higher levels is transferred using a combination of multipole and local expansions as explained in Figure 2.2, the original FMM is asymptotically optimal $O(N)$. However, because the multipole to local translation requires prohibitive $189P^4$ operations for each box, the huge prefactor makes the original FMM less competitive with the tree code and other FFT based methods. In 1997, a new version of FMM was introduced by Greengard and Rokhlin[20] for the Laplace equation. Compared with the original FMM, a plane wave expansion based diagonal translation operator is introduced and the original $189P^4$ operations were reduced to $40P^2+2P^3$.

Figure 2.1: Series expansion approximations of the function $\frac{1}{r}$. a) For any point $R(R, \theta, \phi)$ located outside of a sphere S_a of radius a, the potential generated by N charges located inside of S_a with spherical coordinates $\rho(\rho_i, \alpha_i, \beta_i)$, respectively, can be described using *multipole expansions*; b) in the opposite case, for any point $R(R, \theta, \phi)$ located inside of S_a, the potential generated by N charges located outside of S_a with spherical coordinates $\rho(\rho_i, \alpha_i, \beta_i)$, respectively, can be described using *local expansions*.

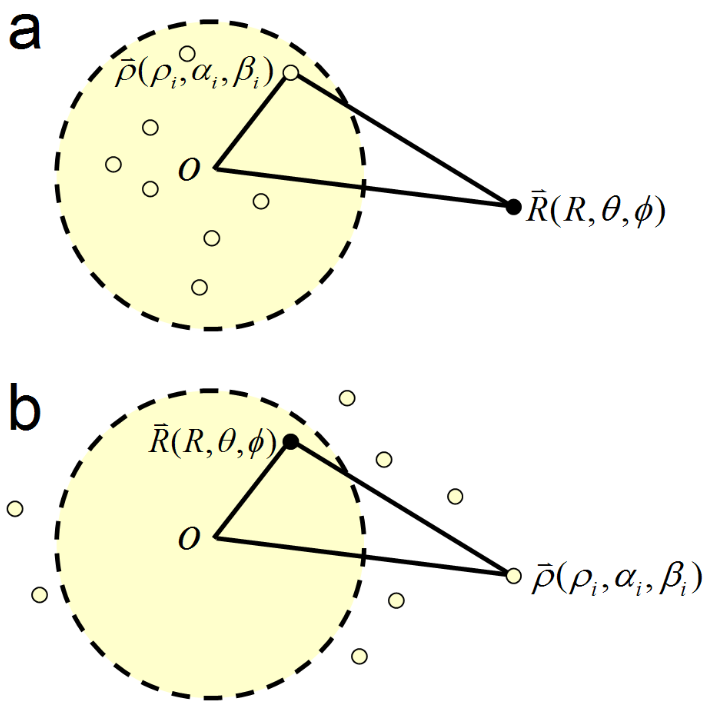
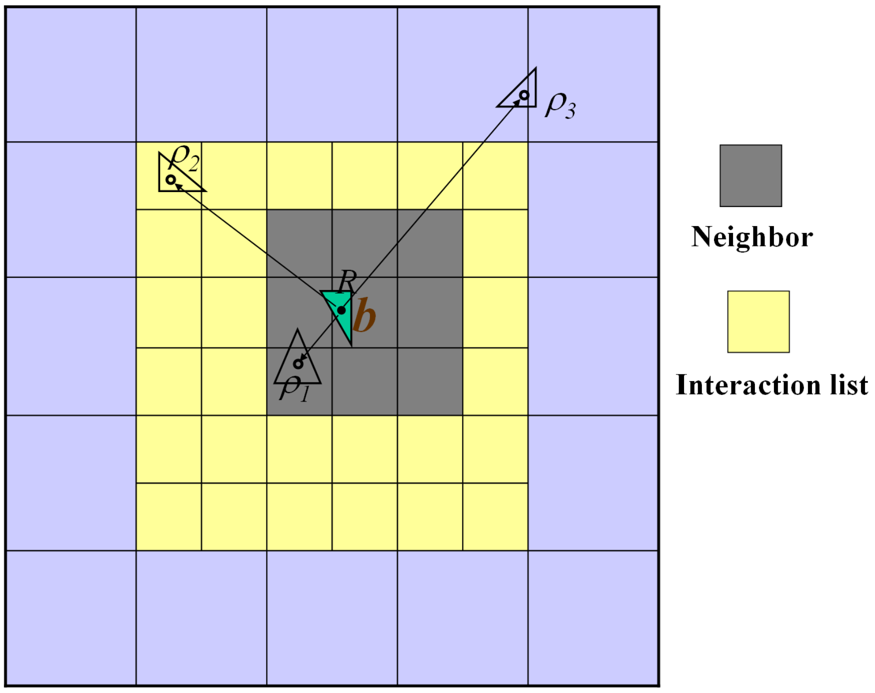

a
$\vec{\rho}(\rho_i, \alpha_i, \beta_i)$
O
$\vec{R}(R, \theta, \phi)$
b
$\vec{R}(R, \theta, \phi)$
O
$\vec{\rho}(\rho_i, \alpha_i, \beta_i)$

Figure 2.2: Schematic showing the hierarchical divided boxes for recording the neighbor boxes and interaction list in the new version FMM. The neighbor boxes (up to 27 including itself in three dimensions) of the target box *b* are darkly shaded, while its interaction list (up to 189 boxes in three dimensions) is indicated in yellow. The remaining far-field boxes are indicated in light blue. Also shown are the source points ρ_i and evaluation point *R* (field). In BEM implementation, the source particles are located at the centers of the surface triangular elements.



The incorporation of fast FMM into BEM-based PB models has been successfully pursued by several groups.[7,8,10,49] However, all past implementations have used an older scheme of the FMM algorithm. As we mentioned above, the cost associated with those types of algorithms is approximately $189P^2N\log N$ (the tree code) or $189P^3N$ (in the original FMM scheme). Although it scales better than the direct computation, considerable speed up can only be achieved for systems of over 20, 000 particles due to the large value of the prefactor. Recent work by Greengard and Rokhlin, which introduces a plane wave expansion during the repeated multipole to local transitions, significantly reduces the cost and breaks even with direct calculation for a reasonable value of N ($\sim$ 1000). The new version of FMM has subsequently been extended to screened Coulomb interactions (corresponding to the linearized PB kernel) in three dimensions.[21] Although mathematically more complicated, the new version of FMM makes it practical to be combined with the boundary element based solution of the linear PB equation. In our algorithm, we adapt the new version of FMM for the screened Coulomb interactions. Preliminary numerical experiments show that the overall break even point of the new version FMM becomes 600 with 6-digit accuracy and about 400 for 3-digit.

Before proceeding to describe how the new version of FMM is used in the context of the BEM solution of the linearlized PB equation, we first introduce how the gradient of the local expansion coefficients can be calculated in FMM. If we define $Q^{\kappa}_{n,m} = i_n(\kappa r)\cdot Y^m_n(\theta,\phi)$ (in the limiting case when $\kappa = 0$, then $Q^0_{n,m} = r^n \cdot Y^m_n(\theta,\phi)$), then a very useful recursive relationship for the gradient of $Q^{\kappa}_{n,m}$ can be expressed as linear combinations of $Q^{\kappa}_{n,m}$ of different order and degree.

$$\nabla Q^{\kappa}_{n,m} = \frac{\kappa}{2n+1}[B]\left(\begin{pmatrix} (n+m-1)(n+m)Q^{\kappa}_{n-1,m-1} \\ (n+m)Q^{\kappa}_{n-1,m} \\ Q^{\kappa}_{n-1,m+1} \end{pmatrix}\cdot\frac{1}{s} - \begin{pmatrix} (n-m+1)(n-m+2)Q^{\kappa}_{n+1,m-1} \\ -(n-m+1)Q^{\kappa}_{n+1,m} \\ Q^{\kappa}_{n+1,m+1} \end{pmatrix}\cdot s\right), \tag{2.28}$$

where s is the scaling factor to avoid under-over flow ($s = 1$, if $\kappa r > 1$ and $s = \kappa r$, if $\kappa r \leq 1$). Note that the above relationship is applicable for all $0 < m < n-1$, for $m = 0$,

$$\nabla Q^{\kappa}_{n,0} = \frac{\kappa}{2n+1}[B](\begin{pmatrix} Q^{\kappa}_{n-1,-1} \\ nQ^{\kappa}_{n-1,0} \\ Q^{\kappa}_{n-1,1} \end{pmatrix} \cdot \frac{1}{s} - \begin{pmatrix} Q^{\kappa}_{n+1,-1} \\ -(n+1)Q^{\kappa}_{n+1,0} \\ Q^{\kappa}_{n+1,1} \end{pmatrix} \cdot s), \tag{2.29}$$

for $m = n-1$,

$$\nabla Q^{\kappa}_{n,m} = \frac{\kappa}{2n+1}[B](\begin{pmatrix} (n+m-1)(n+m)Q^{\kappa}_{n-1,m-1} \\ (n+m)Q^{\kappa}_{n-1,m} \\ 0 \end{pmatrix} \cdot \frac{1}{s} - \begin{pmatrix} 6Q^{\kappa}_{n+1,m-1} \\ -2Q^{\kappa}_{n+1,m} \\ Q^{\kappa}_{n+1,m+1} \end{pmatrix} \cdot s), \tag{2.30}$$

for $m = n$,

$$\nabla Q^{\kappa}_{n,m} = \frac{\kappa}{2n+1}[B](\begin{pmatrix} (n+m-1)(n+m)Q^{\kappa}_{n-1,m-1} \\ 0 \\ 0 \end{pmatrix} \cdot \frac{1}{s} - \begin{pmatrix} 2Q^{\kappa}_{n+1,m-1} \\ -Q^{\kappa}_{n+1,m} \\ Q^{\kappa}_{n+1,m+1} \end{pmatrix} \cdot s), \tag{2.31}$$

where,

$$[B] = -\frac{1}{2} \begin{pmatrix} 1 & 0 & -1 \\ i & 0 & i \\ 0 & -2 & 0 \end{pmatrix} \tag{2.32}$$

Higher order derivatives can be easily obtained by recursive application of Eq. (2.28). For example, the second derivatives can be obtained by inserting the first order derivatives into the right side of Eq. (2.28). The recursive relationship for $Q^{\kappa}_{n,m}$ is a very useful property for applying FMM to the BEM solution of PB equation, which will become apparent in the following section.

2.5. The adaptive FMM

The fast multipole method was first designed for a cluster of particles randomly distributed in a unit box and usually a uniform probability density function (PDF) distribution is assumed. In this scenario, a uniform octree structure is easily generated, and the corresponding algorithm complexicity analysis becomes easy. However, the particle distributions are almost certain non-uniform in most real-world applications, and an adaptive FMM has to be used. For example, in the boundary element method, all nodes (particles) are located on a surface, resulting in a lot of "empty" boxes if a uniform octree is applied.

It is possible to generate a graph and use graph theory to find the optimal adaptive strategy: In the graph, all particles and boxes will be listed twice, as source particles/boxes, and as target particles/boxes. The source particles and target ones can be connected directly, or they can send/receive information form the corresponding boxes at different levels. All connections are associated with a "cost" function, and the goal is to find a "connection strategy" such that the total cost is minimized while there still exist a path connecting each source and target pair. This is further explained in Fig 2.3. In the figure, we assume source points are also target points, and a binomial tree is generated for these points. On the left of the figure, we list all the source points and the "sending" boxes; on the right, we duplicate the left part to generate all the "receiving" boxes and the target points. The source points can interact directly with target points, for example, source point #1 can directly interact with target point #15, using 1 operation. Or, it can first send its information to "sending" box #1, using p opeartions, where p is the number of terms in the multipole expansion, then "sending" box #1 communicates with target point #15. This is the strategy used in the so-called "tree" code. Or as in the fast multipole method, "sending" box can translate its multipole expansion to a new multipole expansion in box #9, and then box #9 translates the collected multipole expansion to a local expansion in the "receiving" box #12 (a member in the interaction list), which is then tranlated again to box #8 using the local to local translation operation, and this local expansion is then evaluated at target point #15. Note that there exist many ways to construct a path and connect the source and target points, therefore the question to be answered is that which strategy is optimal in efficiency. Currently, we are working on this problem and it is not clear if this question can be answered using $O(N)$ operations.

In our current solver, we follow the strategy in,[12] and construct an adaptive tree once given the particle distribution based on a simple strategy: if the box is empty, it is immediately deleted, and if it contains fewer than *nbox* particles, it will be called "childless" and will not be further divided. Note that this strategy may not generate the optimal adaptive tree structure, however we believe it is a good approximation to

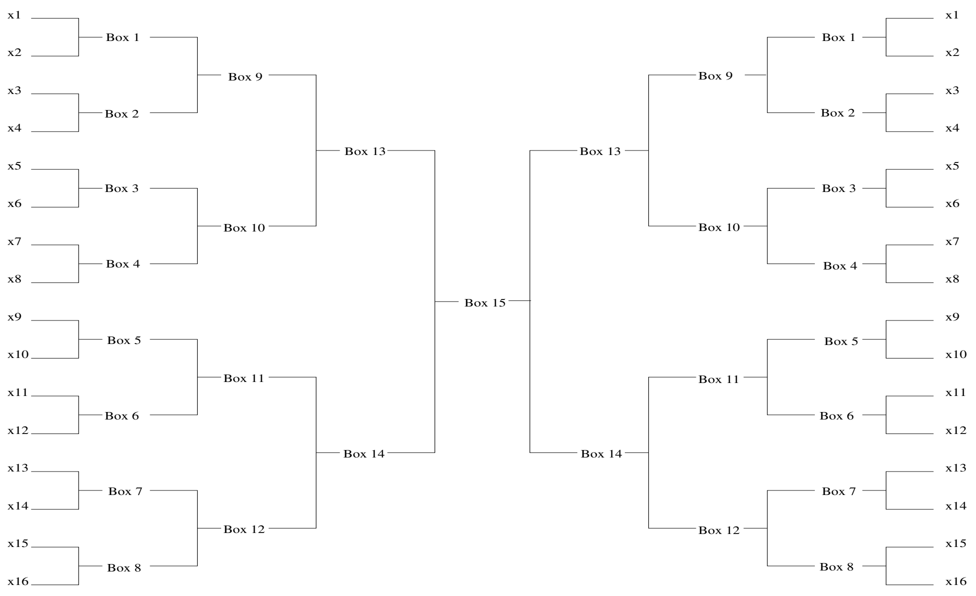


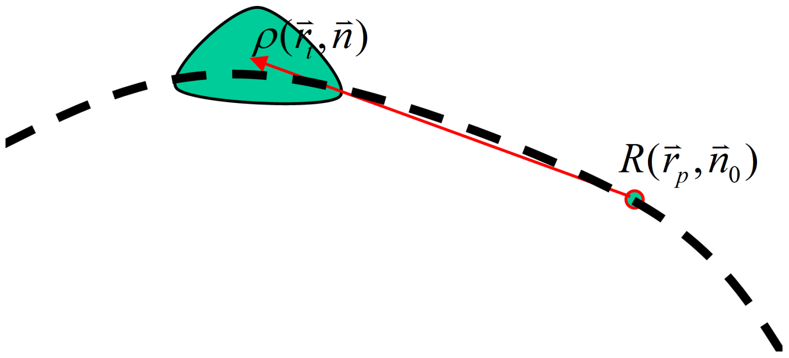
Figure 2.3: Tree communication graph

the final optimal solution.

2.6. FMM in the context of BEM

The solution of the f_i and h_i can be obtained by inverting the $2N \times 2N$ matrix in Eq. (2.14). As mentioned above, direct methods such as Gaussian elimination (LU decomposition) are too expensive in terms of both CPU time and memory. To this end, an iterative procedure will be used in the present algorithm. Another important feature of these iterative methods is that no explicit matrix needs to be stored or calculated; only the calculation of matrix-vector multiplication is required. The multiplication of a matrix (A, B, C, and D) and a vector (f and h) is analogous to calculating electrostatic potentials for $2N$ locations induced by $2N$ point charges. In the present FMM implementation, for each evaluation point p, the evaluation of the left-hand side of matrix Eq. (2.14) can be divided into two parts: 1. contributions from all of the far-field elements of element p (located outside the finest level box encompassing the evaluation point p) will be calculated using local expansions; 2, contributions from all remaining near neighbor elements (inside the same childless box that contains evaluation point p) must be evaluated directly (Figure 2.2).

Figure 2.4: Schematic showing the location of the evaluation point $R(\vec{r}_p, \vec{n}_0)$ (R_p) and a BE location ρ_t.



It is convenient to convert the normal derivatives of functions G and u at ρ into the spatial gradients of G and u at R (Figure 2.4) using the following equations,

$$\frac{\partial G}{\partial n} = -\nabla_R G(R,\rho)\cdot n, \qquad \frac{\partial^2 G}{\partial n_0 \partial n} = -n_0 \cdot \nabla_R^2 G(R,\rho)\cdot n,$$
$$\frac{\partial u}{\partial n} = -\nabla_R u(R,\rho)\cdot n, \qquad \frac{\partial^2 u}{\partial n_0 \partial n} = -n_0 \cdot \nabla_R^2 u(R,\rho)\cdot n, \tag{2.33}$$

where $n = (n_x, n_y, n_z)$ is the unit normal vector at point ρ, $n_0 = (n_{0x}, n_{0y}, n_{0z})$ is the unit normal vector at point R. Substituting Eq. (2.33) into Eqs. (2.13) yields,

$$A_{pt} = (G_{pt} - u_{pt})\Delta S_t, \tag{2.34}$$
$$B_{pt} = (-\frac{1}{\varepsilon}\nabla_R G_{pt}\cdot n + \nabla_R u_{pt}\cdot n)\Delta S_t, \tag{2.35}$$
$$C_{pt} = (\nabla_R G_{pt}\cdot n_0 - \frac{1}{\varepsilon}\nabla_R u_{pt}\cdot n_0)\Delta S_t, \tag{2.36}$$
$$D_{pt} = \frac{1}{\varepsilon}(-n_0\cdot\nabla_R^2 G_{pt}\cdot n + n_0\cdot\nabla_R^2 u_{pt}\cdot n)\Delta S_t. \tag{2.37}$$

Given an initial set of f_t and h_t at any element locations, then for any evaluation point p, the far-field contribution to the left-hand side of Eq. (2.14) can be written as local expansions that sum contributions from a collection of far-field elements (denoted as

$t \in \{L\}$),

$$\Phi_{1p} \simeq \sum_{n=0}^{P} \sum_{m=-n}^{m=n} (-\frac{1}{\varepsilon}\{\nabla_R Q^0_{n,m}(R)\}\{E^0_{n,m}\} - \{Q^0_{n,m}(R)\}\{H^0_{n,m}\}$$
$$+ \{\nabla_R Q^\kappa_{n,m}(R)\}\{E^\kappa_{n,m}\} + \{Q^\kappa_{n,m}(R)\}\{H^\kappa_{n,m}\}), \qquad (2.38)$$

$$\Phi_{2p} \simeq \sum_{n=0}^{P} \sum_{m=-n}^{m=n} (-\{n_{0x}, n_{0y}, n_{0z}\}\{\nabla_R Q^0_{n,m}(R)\}\{F^0_{n,m}\} - \frac{1}{\varepsilon}\{n_{0x}, n_{0y}, n_{0z}\}\{\nabla^2_R Q^0_{n,m}(R)\}\{E^0_{n,m}\}$$
$$+ \frac{1}{\varepsilon}\{n_{0x}, n_{0y}, n_{0z}\}\{\nabla_R Q^\kappa_{n,m}(R)\}\{F^\kappa_{n,m}\} + \frac{1}{\varepsilon}\{n_{0x}, n_{0y}, n_{0z}\}\{\nabla^2_R Q^\kappa_{n,m}(R)\}\{E^\kappa_{n,m}\}). \qquad (2.39)$$

The local expansion coefficients $\{E^0_{n,m}, H^0_{n,m}, F^0_{n,m}, E^\kappa_{n,m}, H^\kappa_{n,m}, F^\kappa_{n,m}\}$ for all of the elements $t \in \{L\}$, are

$$\{E^0_{n,m}\} = \frac{1}{4\pi}\sum_{t\in\{L\}} \begin{pmatrix} n_x f_t \\ n_y f_t \\ n_z f_t \end{pmatrix} L^0_{n,m}(\rho) dS_t \qquad \{E^\kappa_{n,m}\} = \frac{1}{4\pi}\sum_{t\in\{L\}} \begin{pmatrix} n_x f_t \\ n_y f_t \\ n_z f_t \end{pmatrix} L^\kappa_{n,m}(\rho) dS_t$$
$$\{H^0_{n,m}\} = \frac{1}{4\pi}\sum_{t\in\{L\}} h_t L^0_{n,m}(\rho) dS_t \qquad \{H^\kappa_{n,m}\} = \frac{1}{4\pi}\sum_{t\in\{L\}} h_t L^\kappa_{n,m}(\rho) dS_t$$
$$\{F^0_{n,m}\} = \frac{1}{4\pi}\sum_{t\in\{L\}} f_t L^0_{n,m}(\rho) dS_t \qquad \{F^\kappa_{n,m}\} = \frac{1}{4\pi}\sum_{t\in\{L\}} f_t L^\kappa_{n,m}(\rho) dS_t, \qquad (2.40)$$

where $n_x f_t \Delta S_t$, $n_y f_t \Delta S_t$, $n_z f_t \Delta S_t$, $h_t \Delta S_t$ can be considered as groups of effective charges respectively. In Eqs. (2.38)- (2.39), the operation between two curly braces could be scalar-scalar product, or vector-vector dot product, or matrix-vector/vector-matrix multiplication, depending on the property of the quantities in the curly braces. It is worth noting that, for both $\frac{\partial^2 G}{\partial n_0 \partial n}$ and $\frac{\partial^2 u}{\partial n_0 \partial n}$, the first derivative is with respect to R (evaluation points) and the second is with respect to ρ (source points), so there is only a little computational overhead ($< 10\%$) compared to the original non-derivative formulation.

At this point, we are ready to summarize the FMM algorithm in the context of BEM solution of the PB equation, which proceeds as follows (Figure 2.2):

1. Develop an adaptive octree structure encompassing all of the boundary elements by recursively dividing each box into eight child boxes until any child box contains no more than s BEs;

2. Compute multipole expansion coefficients for the childless boxes in the tree structure; for each parent box, form a multipole expansion by merging multipole expansions from its eight children;

3. Start at the tree's coarsest level, compute local expansion coefficients by converting the multipole expansions at any well-separated boxes (interaction list) into a local expansion around the target center and by directly adding contributions due to local near source points (neighbor boxes);

4. For each parent box, translate the local expansion to each of its children;

5. Go to step 3 until all childless boxes are reached;

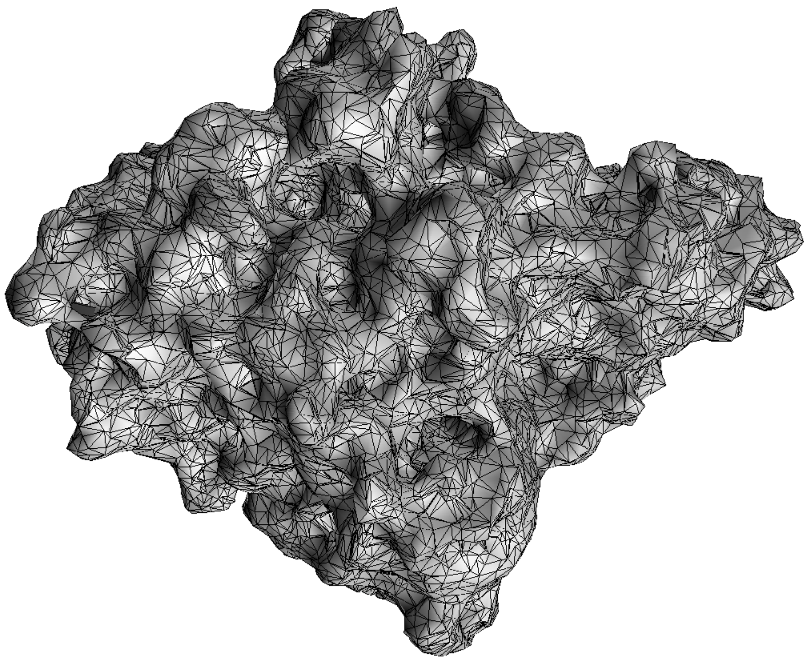
6. For each childless box, evaluate the potential at each target location from the local expansions, and compute the remaining near neighbor interactions directly.

2.7. Krylov subspace methods and mesh generation

In our algorithm, an iterative methods package for systems of linear equations called *SparseKit* is used. Several iterative schemes are available in the package including the GMRES method, BiCGStab method, and transpose-free quasi-minimal residual (TFQMR) algorithm. Preliminary numerical experiments show that the GMRES method converges faster than other methods, which agrees with existing analyses. Because the memory required by the GMRES method increases linearly with the iteration number k, and the number of multiplications scales like $\frac{1}{2}k^2N$, for large k, the GMRES procedure becomes very expensive and requires excessive memory storage. For these reasons, instead of a full orthogonalization procedure, GMRES can be restarted every k_0 steps where $k_0 < N$ is some fixed integer parameter. The restarted version is often denoted as GMRES(k_0). For other alternative methods as BiCGStab method and TFQMR algorithm, the storage required is independent of iteration number k, and the number of multiplications grows only linearly as a function of k. Currently a detailed comparison of different Krylov subspace methods is being performed and results will be reported in later papers.

There are normally three types of "surface" used to define the molecular boundary dividing the low dielectric (interior) and high dielectric (exterior) regions: the *van* *der Waals* surface is the surface area of the volume formed by placing van der Waals spheres at the center of each atom in a molecule, The *solvent-accessible surface*[35] is formed by rolling a solvent, or a probe, sphere over the van der Waals surface. The trajectory of the center of the solvent sphere defines the solvent-accessible surface. Whereas, the *solvent-excluded surface* is defined as the trajectory of the boundary of the solvent sphere in contact with the van der Waals surface. The solvent-excluded surface is also referred to as the molecular surface. In our BEM, to discretize the boundary integral equations, a triangular mesh of molecular surface is generated using the software MSMS,[39] and elements of zero and extremely small area are removed by a mesh checking procedure in our algorithm. The node density and probe radius are input parameters of MSMS to control the fineness of the outpur mesh; the typical values are $1.0/Å^2$ and 1.5 Å, respectively. Mesh generation is a fast step and takes only a few seconds for medium-sized molecules. A typical mesh of a molecule with 8362 atoms is shown in Figure 2.5, which contains 32975 vertices and 65982 triangles and is generated within 3 seconds of cpu time.

Figure 2.5: A typical surface triangulated mesh of a protein (Acetylcholinsterase).



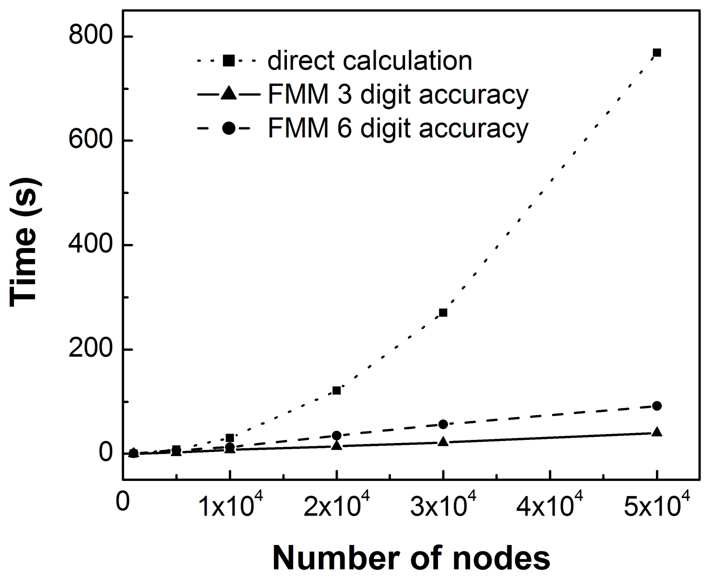
3. USAGE EXAMPLES

3.1. Computational performance of adaptive and non-adaptive solvers

As a first numerical experiment, we compared the speed and memory usage of the FMM to the direct calculation in one GMRES iteration step using the uniform version of our AFMPB solver. The position and parameters of BEs were randomly generated but uniformly distributed on the surface of a sphere of radius 40 $\mathring{A}$. As expected, the error of the FMM calculation is bounded when the number of multipole expansion terms P is set. Similar to what is observed in the original FMM implementation,[21] our algorithm breaks even with the direct calculation at about $N = 400$ for three-digit precision ($P = 9$), and $N = 600$ for six-digit precision ($P = 16$). As shown in Figure 3.1a, in contrast to the quadratic increase in direct calculation, the actual CPU time required by our fast algorithm grows approximately linearly with the number of BEs. We want to mention that we also have a non-adaptive version of AFMPB, in Figure 3.1b we display some non-linearity for the growth of memory usage for the non-adaptive version. In the non-adaptive FMM, as the number of levels increases, there is a cubic increase of number of boxes (storing the local expansion coefficients for each box is the main source of memory usage), leading to a slightly non-linear growth of memory usage.

In our calculation, the majority of computer memory is allocated to store the neighboring list and the corresponding near-field coefficients, the size of which mainly relies on the total number of BEs and the level for box subdivision. Depending on a trade-off between memory and speed, at each iterative step these coefficients can either be saved as in a memory-intensive mode or be discarded as in a memory-saving mode. In a non-adaptive FMM case, the number of neighboring boxes of a box (therefore any BE located within this box) is 27 (including itself). If we further assume that the maximum number of elements per box at the finest level is s, then it is easy to see

Figure 3.1: The CPU time (a) and memory usage (b) of our fast BEM-PB algorithm as compared to those from the direct calculation in one GMRES iteration step.



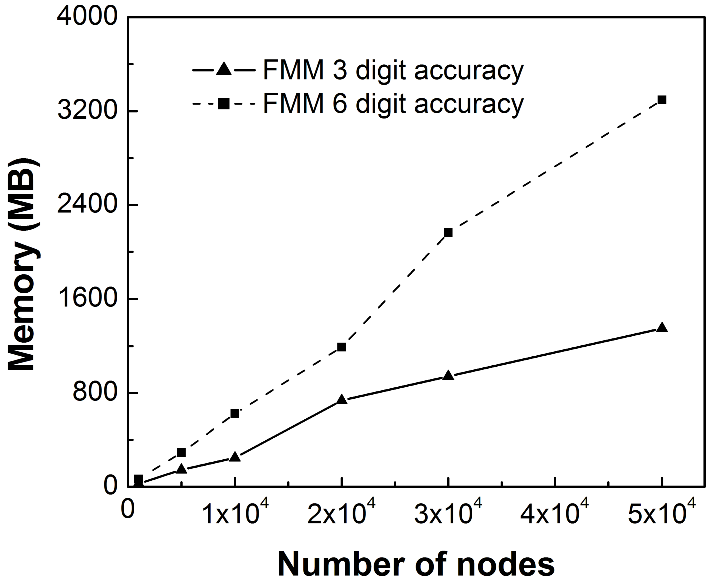


Table 3.1: Timing results for the FMM on a 10, 000-node-system (in one GMRES iteration step) using various levels and terms (P).

P	levels	T_{fmm}(s)	T_{local}(s)	T_{total}(s)
9	3	0.7	10.6	11.3
9	4	1.2	2.4	3.6
9	5	5.7	0.4	6.1
16	3	2.3	10.6	12.9
16	4	4.9	2.4	7.3
16	5	26.9	0.4	27.3

that the number of near-field elements for each BE can normally be up-bounded by a fixed number $27s$. Hence, the size of neighboring list is also up-bounded by $27sN$; this and the fact that there are at most $2N/s$ boxes in the tree structure lead to $O(N)$ overall memory usage. We want to mention that such an estimate for the adaptive code is difficult and depends on the adaptive strategy.

When using the FMM, it is important to keep a load balance between the number of BEs in the local list (calculated directly) and the number of BEs in the far-field (calculated using local expansions). If the number of local BEs is too large, then the advantage of using multipole expansions is not fully taken. Conversely, if the number of local BEs is too small, then more boxes will be needed, which usually means more operations of expansions. We assessed the performance of the FMM on a 10, 000 BE system (again in a single GMRES iteration step) using various levels and terms (P); results are presented in Table 1. The total timing T_{total} is broken into the T_{fmm} for far-field calculation and T_{local} for local direct calculation. For both three- and six-digit accuracy, the optimal level is 4. Having more levels (more boxes, fewer local BEs) and fewer levels (fewer boxes, more local BEs) both slow down the overall speed because of the unbalanced T_{fmm} and T_{total}. Generally, the optimal level of box subdivision depends on number of terms P, so that the maximum number of BEs in the lowest level box s is comparable to $P^{\frac{3}{2}}$. [a] Of radius 50 $\AA$ with a point charge +50e located at the center. The exact Born solvation energy $E_{\text{solvation}}$ of the cavity is -4046.0 (energy is in kcal/mol).

To assess the performance of the FMM BEM algorithm in solving the PB equation,

Table 3.2: BEM performance on a spherical cavity case with different surface mesh sizes[a]

Number of Elements	$T_{\text{direct BEM}}$ (s)	$T_{\text{FMM BEM}}$ (s)	level	Iteration steps	$E_{\text{solvation}}$ (error %)	Error (%) in f	h
320	0.13	0.18	2	5	-4227.5 (4.5)	6.6	5.6
1280	1.56	0.82	3	5	-4134.5 (2.2)	2.8	2.5
5120	19.67	3.39	3	5	-4088.6 (1.1)	1.4	1.1
20480	247.20	15.86	4	5	-4066.5 (0.5)	0.7	0.6
81920	3122.10	87.96	5	5	-4050.6 (0.3)	0.2	0.4

we next calculate the Born solvation energy of a point charge +50 e located at the center of a spherical cavity with a radius of 50 Å. The surface is discretized at various resolution levels by recursively subdividing an icosahedron. Table 2 summarizes the timing results (on a Dell dual 2.0 GHz P4 desktop with 2 GB memory) and some related control parameters using a FMM accelerated BEM (denoted by FMM BEM) and a direct BEM without invoking any fast algorithms (denoted by direct BEM). Due to memory constraints, the PC can not handle higher levels of subdivision on the sphere (more than 300k BEs). As for the efficiency, we noticed that regardless of the surface resolution, all the GMRES iteration steps are below 5, which numerically confirms that the derivative BEM formulations are well-conditioned. The CPU time for the new version of FMM linearly increases with the number of BEs, while it quadratically increases for the direct integration method. Note that whenever switching to a higher level of box division, there will be a small jump of CPU time due to the increased boxes, which leads to some deviation from performance linearity for the FMM. For a system with 81,920 surface elements, the $O(N)$ new version FMM is approximately 40 times faster than the direct method.

The numerical error of our BEM algorithm is on the first order of the grid size of the mesh. In the spherical case in Table 2, when the mesh is refined to a higher level, the number of BEs is quadrupled, and the size of each element is reduced by half. The relative errors of the calculated energy, f, and h compared with the analytical results also show that the computational accuracies are nearly linearly improved upon the refinement of mesh scale. More discussion on the accuracies of BEMs can be found in ref. 31. *Using the same level=4 for all FMM calculations in BEM.

Table 3.3: Comparison between BEM* and APBS$^+$. F denotes the number of faces, V the vertices; a and b denote the memory-intensive and memory-saving calculation modes, respectively

Methods	Mesh	Memory (MB)		CPU (s)		$E_{solvation}$	Iteration
		a	b	a	b	(kcal/mol)	steps
BEM	8894 F, 4449 V	224	54	22	35	-556.1	14
	12044 F, 6024 V	289	59	26	53	-540.3	13
	15046 F, 7525 V	350	63	32	75	-534.6	13
	18046 F, 9025 V	411	67	36	98	-525.5	13
	21430 F, 10717 V	481	72	44	129	-522.0	13
APBS	$65 \times 65 \times 65$	78		39		-552.1	–
	$97 \times 65 \times 97$	150		64		-542.3	–
	$127 \times 97 \times 127$	341		131		-531.0	–
	$161 \times 129 \times 161$	742		258		-525.0	–
	$225 \times 161 \times 225$	1784		599		-522.8	–

$^+$APBS using focusing procedure, and solving the PB equation two times in each solvation energy calculation. When a much finer mesh (321 x 321 x 321, which can not be handled in a 2 GB memory PC) is used to run APBS again, a solvation energy 521.1 kcal/mol is obtained. This could be taken as a referece solvation energy.

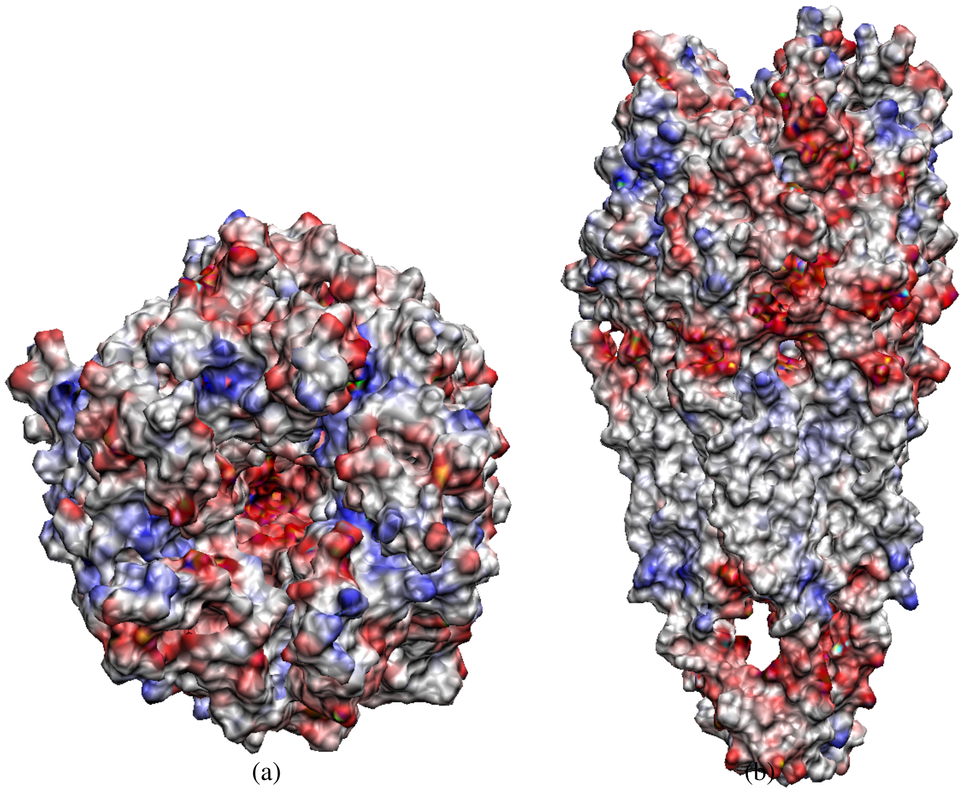
To further illustrate the performance of our fast BIE technique on protein electrostatic calculations, we computed the electrostatic solvation energies of fasciculinII, a 68 residue protein, and compared the algorithm performance with the multigrid finite difference algorithm, as implemented in the widely used software APBS[5] (see Table 3). We want to mention that the two program codes employ very different algorithms and data structures, hence an exact comparison between them would be difficult. Also, APBS is designed primarily for massively parallel computing; it has an integrated mesh generation routine, while the current BEM only runs on a single CPU, and needs a pre-generated mesh as an input. Two sets of meshes at different resolutions were generated for BEM and APBS calculations respectively. Similar convergence trends are observed for both energy calculations. At low mesh resolution (with small number of nodes and faces), the BEM seems to require more memory than APBS does. The reason is that we use the same level of 4 of box subdivision for all the BEM calculations, which consumes a large portion of the total memory, and may not be optimal for small systems. When system size increases, the memory usage shows a slower increase, as does the CPU time cost. It should also be noted that APBS solves the PB equation twice to obtain the solvation energy, while BEM only solves it once. However, if the potentials and forces at the "volume" grid points away from the surface are needed, they are readily available in APBS solutions, while in BEM it is necessary to calculate again by integrating the PB equation solutions on the boundary.

3.2. Electrostatics of the nicotinic acetylcholine receptor (nAChR)

nAChR is one of ligand-gated ion channels that mediate fast synaptic transmission between cells. The roles of electrostatic interactions in governing the agonist binding, ion conduction and anesthetic action in nAChR have been implicated in many previous studies. As a test of our PB solver, we calculated the electrostatic potentials of the human α7 nAChR. The receptor structure including both the extra-cellular and intra-cellular domains was built up by homology modeling based on the cryo-electron microscopy structure of *Torpedo* nAChR (PDB code: 2BG9).[45] The modeled structure contains 1880 residues, has a total length of about 160 $\AA$ and a diameter of about 40 $\AA$ parallel to the membrane surface. The BEM calculation was performed with 194428 triangular elements and 97119 vertices.

In Figure 3.2, the molecular surface of nAChR is colored according to electrostatic potentials such that the most negative region is in red while the most positive region is in blue. The interior of the channel vestibule formed by the pentameric assembly of the ligand-binding domains shows very negative potentials. This would be expected to increase the local concentration of permeant cations (i.e. Na+ and K+ ions), and is consistent with earlier suggestions.[11] Moreover, deeper inside the channel, more negative potentials are observed, which reach the minimum roughly in the middle of pore. The existence of an electrostatic potential gradient across the channel pore may facilitate passage of ions through the channel. The surface potentials can be divided into two regions: the membrane-spanning domain that is dominated by positive potentials, and the extra/intra-cellular domains that are dominated by negative potentials. The strong negative potentials on the extra-cellular surface are expected to impose electrostatic steering attraction to positive ligands (e.g. acetylcholine) and cations.

Figure 3.2: The surface potential map of nAChR from different views. The increasing potential from negative to positive value is represented by changing the color from red to blue.



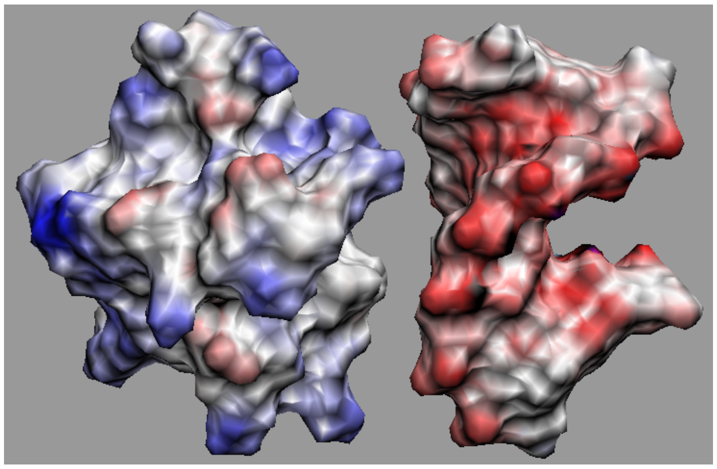
We also performed the calculation in the zero ionic strength condition. The results turn out to be very different where the surface potentials are almost all negative (data not shown). The difference indicates that the ionic strength has a great impact on the electrostatic character of nAChR.

3.3. Electrostatic interactions between Sso7d and DNA

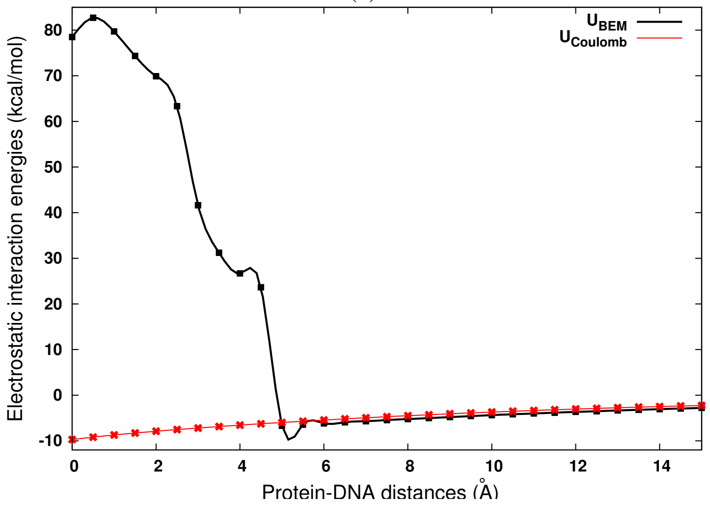
We studied the electrostatic interactions between two molecules: Sso7d and DNA based on a crystal structure (PDB code: 1AZQ).[36] Sso7d is a small chromosomal protein from the hyperthermophilic archaeabacteria *Sulfolobus solfataricus*. The protein has high thermal, acid and chemical stability. It binds DNA without marked sequence preference. In the crystal structure, Sso7d has 66 residues in complex with a short double-stranded DNA with 8 base pairs. Sso7d binds in the minor groove of DNA and causes sharp kink in DNA. The protein-DNA complexes are normally highly charged. Sso7d is positively charged (+6), whereas the complex is negatively-charged (-8) overall due to the additional 14 negative charges carried by the DNA phosphate groups. To investigate the role of electrosatics in the Sso7d-DNA association process, the interactions between Sso7d and DNA at different separation distances are calculated. These structures are generated by displacing DNA away from the binding site along the center-to-center direction in the Sso7d-DNA complex.

The BEM calculation is performed on a two-domain system if two molecules move away and two separate meshes are generated or on a single domain system if two molecules are close enough to 'merge' and only a single mesh generated. For intermolecular electrostatics, the present BEM method provides the full PB interaction energy that takes into account both the desolvation and mutual polarization contributions from the two molecules. Figure 3.3a shows the electrostatic potentials mapped on the molecular surfaces of Sso7d and DNA at a separation of 10 $\AA$. The potential surfaces exhibit good electrostatic complementarity at the binding interface. Electrostatic attraction governs the intermolecular interaction at distances larger than $\sim 5\AA$ (Figure 3.3b, black line). Nevertheless, the electrostatic interaction becomes repulsive at close distances. A closer inspection of the complex structure suggests that a signifi-

Figure 3.3: Electrostatics of Sso7d-DNA. (a) The surface potential map of Sso7d and DNA at separation of 10 $\mathring{A}$. (b) The electrostatic interaction energies as functions of separation along the center-to-center unbinding direction. U_{BEM} is the full electrostatic interaction energy determined by our BEM, and U_{Coulomb} is the sum of all the atomic pair screened Coulomb interactions between Sso7d and DNA. The curves connect the calculation points (denoted by the diamond and star symbols) consecutively by fitting with cubic splines.



(a)



(b)

cant component of the binding free energy is due to the non-electrostatic interactions, made in large part by the interfacial hydrophobic residues.[36] The origin of the large unfavorable electrostatic interaction at close separations can be attributed to the electrostatic desolvation, an effect due to the unfavorable exclusion of the high dielectric solvent around one protein when the other one approaches. As a comparison, we also calculate the screened Coulomb interactions by summing up all the atomic pair contributions between Sso7d and DNA (see Figure 3.3b, red line). In this treatment, it is found that the interactions are all attractive across the whole separations. The values are close to the full BEM calculations at long distances, but the desolvation effects are obviously missed at close distances.

The electrostatic interaction characteristics displayed in Figure 3.3 are very similar to the acetylcholinesterase and fasciculinII complex system as has been demonstrated previously,[29] which also shows long-range attraction ($>\sim 5\AA$) and close-range repulsion. Another common observation of these two systems is that the electrostatic interactions start abruptly increasing from around 5Å. This is the distance where the water molecules between the two molecules are squeezed out, and the two molecules begin to collapse into a compact complex structure. At the same time, the generated molecular surface meshes of the two molecules also begin to merge into a single mesh. This implies that the interfacial non-polar interaction, hydrophobic packing, and possibly local conformational rearrangement upon binding take effects from around 5Å of association and become dominant binding forces in the final stage of complex formation.

4. USING AFMPB

4.1. Compile and installation

After the user downloads the package and extracts it to a local computer, the following directories can be found:

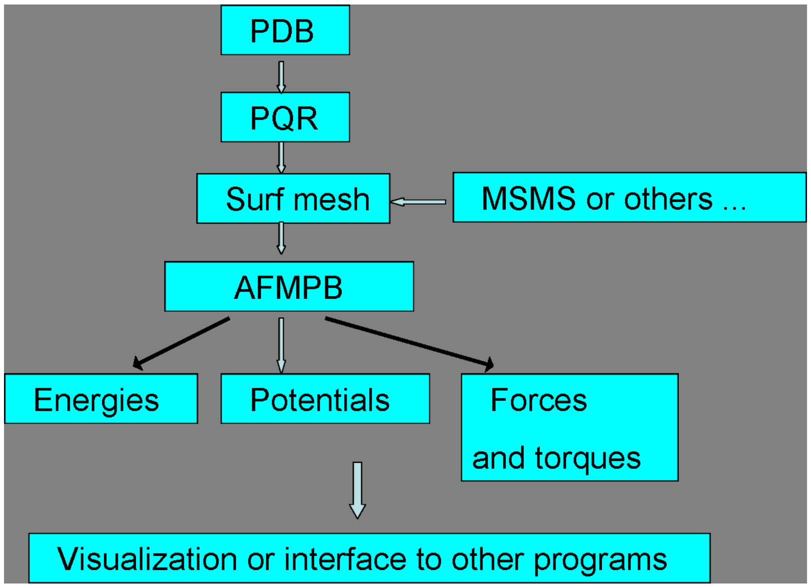
- Doc: contains the references and this manual.
- job-examples: contains a README file, samples of the c-shell job files *job.csh* to run the package, and the generated input/output files. A subdirectory ./Two-proteins contains similar input/output files but is for the test case of a two-molecule system.
- FBEM: the driver and the boundary element codes of the AFMPB package. It also contains one subroutine iters.f, the Krylov subspace iterative solvers from SPARSKIT.[38] The default Krylov solver is GMRES, though the user can also use the restarted GMRES, TFQMR, or BiCGStab.
- fmm3d_node: the fast multipole method library for the Laplace and Yukawa potentials.
- tools: pre- and post- processing utility programs. Additional tools will be added to this directory for calculation of different parameters.

For compiling, check the file **Makefile** for further information. The package has been successfully compiled using the Intel©compiler for Linux, and the GNU©F95 compiler.

4.2. General organization of AFMPB calculation

The program diagram is shown in Fig. 4.1.

Figure 4.1: AFMPB flow chart



AFMPB can be considered as a main driver for calculating the electrostatics in biomolecular systems. It requires molecular structure, charge and mesh information from the pre-processing tools as inputs, and outputs potential, various energies and force results for visualization and/or further processing. To generate the "boundary elements", AFMPB currently accepts mesh data from the package *MSMS*[39] and other programs that can generate molecular surface mesh in the *OFF* format. For post-processing, we use VMD for visualization. The solver can also be coupled with multi-scale time stepping schemes to simulate the dynamics of molecular systems under the influence of electrostatic interactions.

4.3. General usage description

Pre-processing, AFMPB job run, and post-processing are usually required for biomolecular electrostatic analysis. Some utility tools are supplied at the directory *./tools* for pre-/post- processings and visualization. In the *./job* directory, there are several example job scripts for calculations at different cases, which can be a useful start to use AFMPB package.

4.4. Running AFMPB calculation

A command line style is adopted to run AFMPB. A simple way to run AFMPB is "*./afmpb*", with all the default input/output files in the working directory. A typical job with specified options can be like

afmpb -i inp.dat -o out.log -s surfpot.dat.

-*i* means to be followed by the control input file,

-*o* an output log file,

-*s* is followed by an output file that contains surface nodal potential.

AFMPB also supports potential calculations at exterior points by using two more options:

-*vm* followed by a file name that contains exterior points to be calculated.

-*vp* a file storing the potential file at the exterior points.

4.5. The input files

To run AFMPB, users are required to prepare input files including (a) a control file (*inp.dat* by default), (b) *PQR* file(s) containing atomic charges and their locations, (c) molecular surface mesh file(s), and an optional (d) off-surface points location file where the potentials need to be evaluated. For the output, a log file (*out.log* by default) will be generated automatically after every AFMPB run, and users may also request a surface potential file (default *surfpot.dat*) by setting a write-control parameter in *inp.dat*. All the input files and output log file (*out.log*) are in free format. The job-control input file can also be generated by the job shell script.

4.5.1. The control input file

An example control input file

————–

# nmol, number of molecules

1

```
# di, de, ion concentration(mM), temperature
2.0 80. 150 300.0
# meshfmt: 0 icosohedron, 1 raw, 2 msms, 3 matlab sphere, 4 mc, 5 off
2
# output key: ipotw(surf pot),iforce,iinterE,iselfE (solvation),ipotdx(vol pot)
1 0 0 1 0
# pqr and mesh files (repeat nmol lines for multi-molecules)
pqr1.dat mesh1.dat
————————-
```

The comment lines explains the meaning of each variable.

4.5.2. PQR file

The PQR format is a modification of the PDB format which adds charge and radius parameters to existing PDB file. The *PQR* file can be generated from a standard *PDB* file using the software *pdb2pqr*. *pdb2pqr* also adds missing hydrogen atoms in the *PDB* file, optimizes the hydrogen bond network, and assigns atomic charges and radii based on various force field parameter sets.

4.5.3. Moelcular surface and surface mesh

The molecular surface mesh file is traditionally generated from the *PQR* file. There exist at least three types of "molecular surface" to define the boundary between the low (interior) and high dielectric (exterior) regions: the *van der Waals* surface, the *solvent-accessible surface*,[35] and the *solvent-excluded surface*. The *van der Waals* surface is the union of the surface area formed by placing van der Waals spheres at the center of each atom in a molecule. When rolling a solvent (or a probe) sphere over the van der Waals surface, the trajectory of the solvent sphere center defines the *solvent-accessible surface*, while the trajectory of the boundary of the solvent sphere in closest contact with the van der Waals surface defines the *solvent-excluded surface*. The solvent-excluded surface is also referred to as the molecular surface in bimolecular studies. In the AFMPB solver, a triangular mesh of the molecular surface can be generated using the software *MSMS* as described in.[39] Two important parameters in *MSMS* are the node density and probe radius that control the resolution of the output mesh, and the typical values are set to $1.0/\AA^2$ and 1.5 Å, respectively. The mesh from *MSMS* can be further improved by removing the elements of extremely small area using a mesh checking procedure provided in the AFMPB package, though this step is usually not necessary. A typical mesh of a molecule with 8362 atoms is shown in Fig 2.5.

The surface mesh file contains node coordinates and how they connect to form triangles. The current version of AFMPB allows a few slightly different mesh formats, including the standard *OFF* format and the so-called *MSMS* format. A comment line in *inp.dat*

”# meshfmt: 0 icosohedron, 1 raw, 2 msms, 3 matlab sphere, 4 mc, 5 off”

indicates to choose an optional mesh format. The *OFF* format is used by many mesh generating tools, and can be viewed using visualization software such as “GEOMVIEW”, while the *MSMS* format can be conveniently generated by using a script tool provided in our package (see the section “Utility Tools”). The script will call the software MSMS[39] that is widely used in biomolecular studies. A sample *MSMS* format file is as follows:

```
5358 10712
-18.162 1.138 29.523 -0.309 -0.779 0.546
-17.185 -0.033 30.761 -0.960 0.001 -0.280
...
16.334 2.571 23.760 -0.000 1.000 0.000
1 5 3645
3645 5 3646
...
3614 3620 3613
```

In this file, the first line describes the total numbers of nodes and triangular elements. Starting from the second line, the *xyz* coordinates and optional *xyz* components of the normal direction for each node are given, followed by a list indicating the indices of three nodes for each triangular element.

The AFMPB solver requires that the normal vectors point outward (as defined in the current *OFF* and *MSMS* formats) and the nodes are ordered counter-clockwisely. In case the normal vectors are not available in the input surface mesh files, which happens when other file formats are used as input files, functions are provided in the solver to calculate these quantities.

4.5.4. A sample shell script

In the following, we provide a simple shell file for executing the AFMPB package.

A Shell file *job.csh* for AFMPB

```
#!/bin/csh
# xxx, 20xx
# Set up directories.
    set DIR=../
    set OUT=.

# Set up molecule data and mesh.
    set pqr1=fas2.pqr # mol1 pqr
    set mesh1=fas2-mod.dat-d1.2-r1.5 # mol1 msms mesh
# Prepare input data file.
    cat << _END > inp.dat
    # nmol, number of molecules
    1
    # di, de, ion concentration(mM), temperature
    2.0 80. 150 300.0
    # meshfmt: 0 icosohedron, 1 raw, 2 msms, 3 matlab sphere, 4 mc, 5 off
```

```
2
# output key: ipotw(surf pot),iforce,iinterE,iselfE (solvation),ipotdx(vol pot)
1 0 0 1 0
# pqr and mesh files (repeat nmol lines for multi-molecules)
$pqr1 $mesh1
_END
```

Then, simply execute "*./job.csh*" to run AFMPB with all the variables and options setted in the script.

4.6. The output file

The output log file *out.log* records useful information during code execution, including the number of iterations of the iterative Krylov solver, the CPU time and memory usage information, the total/solvation/interaction/Coulombic energies, and so on. The surface potentials and forces are recorded in a formatted file (*surfpot.dat*), which can be extracted for further analysis, and can be visualized with VMD using a provided TCL script file (see the subsection "Utility Tools").

4.7. Analyzing AFMPB calculation

The calculation results for energies and forces (between molecules) can be extracted from *out.log*. The surface and off-surface potential files can be used for further analysis. The solvation energies for individual molecules are also stored in *out.log* if the corresponding output options are switched on in the input file *inp.dat*. The solvation energy is computed by solving the PBE only once, which is different from most finite difference based methods. For multi-molecule systems, the binding energy, the total forces and torques acting on individual molecules due to the other molecules (excluding itself) can also be calculated and outputed to *out.log* by setting the corresponding control variables in *inp.dat*.

4.8. Utility Tools

The directory *./tools* contains tools for file format conversion, mesh generation and refinement, and data analysis and visualization. In the current release of the solver, two scripts are provided: *pqrmsms.csh* and *showSmoothMesh.tcl*. The c-shell script *pqrmsms.csh* generates a *MSMS* molecular surface mesh from a *PQR* file. Before using the script file, the program *MSMS* should be installed and the path should be correctly set. Given a *PQR* file (e.g. *fas2.pqr* in the directory), the surface mesh can be generated by running the following command:

./pqrmsms.csh fas2 1.2 1.5.

The last two variables specify the node density (in unit of 1 per $\AA^2$) and probe radius in unit of $\AA$, respectively.

The second tool, a TCL script *showSmoothMesh.tcl*, is used together with the visualization program *VMD* to display the surface potential data file. *VMD* should be installed before running the script. A sample execution to visualize the $fas2$ structure and the resulting surface potentials follows:

vmd fas2.pqr -e showSmoothMesh.tcl -args ../job-examples/surfpot.dat.

5. Programmer's Notes

5.1. Programming Language

Most of the codes are in Fortran 77 style, including the FMM library subroutines and the iterative solvers from SparseKit. However, we also use two commands from Fortran 90 and later versions for dynamical memory allocations.

5.2. Special functions

One function which may be machine dependent is the subroutine for get the current CPU clock information for timing purposes. The users should check their platform and compiler and write such a subroutine. Check second.f for details.

6. Variables and Data Structure

6.1. Important variables in header files

The following important variables are defined in different header files

- defgeom.h: defines variables to describe the geometry of the molecules.
- files.h: defines the unit for different input and output files. The following are used for data input files: inp, outp, sufp, vmesh, volp.

  The following units are used:

  - ivmesh=9 for volume mesh input if one needs potential evaluation.
  - ismesh=7 for surface input.
  - iupqr=30 for pqr input.
  - iusurfp for surface potential output.
  - iuvolp=29 for volume potential output.
- fmmtree.h: in the fast multipole method, the size of many variables and vectors can be determined before one knows the adaptive tree structure for the input geometry. In this file, all the "fixed" length variables are defined here, including many variables for storing the precomputed data. One common block is defined here for adaptive tree structure.
- membem.h: this file defines the variables for the geometry and input/output variables in the boundary element method. The common block geomol defines the numbers of molecules, node points, elements, and singular charges. The common block membem defines the pointer to a huge working array where different geometry and input/output variables are stored.

- memfmm.h: this files defines the pointers for the fast multipole method variables. All adjusted variables are allocated once the adaptive tree structure is determined, and these pointers are generated for integer, real, and complex variables, respectively.
- parm.h: important physical and code parameters. The physical parameters include:
  - di
  - de
  - dei
  - conctr
  - kap
  - tempr
  - totmem
  - untfactor
  - pi, piq, pih,pi2,pi4,pi8: π and its factors.

  The following parameters are used for code control.

  - cut1, cut2
  - sigm:
  - ipotdx:
  - iflag: the current solver only supports "free space" boundary condition (iflag=0). We plan to add the periodic boundary conditions for a big box in future implementations (iflag=1).
  - lw: as the tree structure is adaptive, one can not determine the size of the memory where the structure will be stores. Hence initially a large amount of memory space is allocated to generate the adaptive tree. If this number is too small, error message will be provided asking the users to increase this number.

- nbox: in our adaptive strategy, we ask that the max number of particles in each childless box is less than nbox. Reducing this parameter will generate more levels (which means more boxes but less direct interaction list).
- epsclose: when the distance between two particles is small than this value, the program will complain that the two particles are too close to each other.
- nterms, nlambs: the number of multipole and exponential expansions in the FMM code. Currently only 3 and 6 digits accuracy are allowed, corresponding to nterms=mlambs=9 and nterms=nlambs=18, respectively.

6.2. Important variables in the code

A data printing package is provided (see prini.f) for outputing integer, real and complex type variables. The output unit is initiallized by calling "prini(num1, num2)" where num1 and num2 are two unit numbers for output. If set to 0, then it will not output those data. See the file main.f for further information on calling prini().

In subroutine solvpb.f, several parameters are defined for the Krylov subspace methods (see vector ipar). Currently GMRES is used. Users can change to other Krylov subspace methods by changing gmres. However, the selected solver should not ask for the transpose matrix vector product.

7. Important Subroutines

7.1. Boundary Element Code

main.f: the main driver for AFMPB.
bempb (bempb.f): the main subroutine for the LPB equation solver. It sets up the equation and calculates all required quantities, including different energies.
solvpb (solvpb.f): this subroutine iteratively solves LPB equation solver.
rdcomm (rdcomm.f): this subroutine reads the command line input files.
elmgeom (elmgeom.f): this subroutine computes the required geometry information of the molecules.
getselfmtrx (getselfmtrx.f): this subroutine computes the local direct interaction coefficients.
val_gndgnlap (val_gndgnlap.f): this subroutine computes the Coulomb potentials when sources and targets are separated.
val_eneyukst (val_eneyukst.f): this subroutine computes the Screened Coulomb potentials when sources and targets are separated.

7.2. Iterative Package from SparseKit

There is only one file used from the SparseKit, check iters.f. The solvers which can be used by AFMPB include GMRES, TFQMR, BCGSTab, FGMRES, and DQGMRES.

7.3. Fast Multipole Methods

The following files are the main interface subroutines between the fast multipole algorithms and the boundary element codes. **ladapfmm (slapadap.f)**: this subroutine computes the Laplace interaction, single layer potential, the sources are the same as the targets.

ldnadap (slapdn.f): this subroutine computes the Laplace interaction, double layer potential, the sources are the same as the targets.

ladapst (slapst.f): this subroutine computes the Laplace interaction, the sources are different from the targets.

yadapfmm (syukadap.f): this subroutine computes the Yukawa interaction, single layer potential, the sources are the same as the targets.

ydnadap (syukdn.f): this subroutine computes the Yukawa interaction, double layer potential, the sources are the same as the targets.

yadapst (syukst.f): this subroutine computes the Yukawa interaction, the sources are different from the targets.

8. Frequently Asked Questions

1. **Where can I download this package?** You may find the package from the website http://lsec.cc.ac.cn/l̃ubz/afmpb.html at LSEC of China, and a mirror site at Prof. McCammon's group website at UCSD http://mccammon.ucsd.edu/.

2. **More questions will be reported after a period of practice of the package**

Any suggestions, bug reports, please let us know.

Acknowledgments

We would like to express our gratitude to Prof. Leslie F. Greengard at the Courant Institute of Mathematical Sciences, Professor Vladimir Rokhlin at Yale, and members in their groups.

And finally, we would like to thank our US sponsors, NSF, NIH, HHMI, NIH, NBCR, CTBP, the U.S. Department of Energy Field Work Proposal ERKJE84, and Chinese sponsors, the Academy of Mathematics and Systems Science of Chinese Academy of Sciences, the State Key Laboratory of Scientific/Engineering Computing, and NSFC,

BIBLIOGRAPHY

[1] M. Abramowitz, I. A. Stegun, Handbook of Mathematical Functions, Dover Publications, New York, 1965.

[2] M. Altman, J. Bardhan, J. White, B. Tidor, An accurate surface formulation for biomolecule electrostatics in non-ionic solutions., Conf Proc IEEE Eng Med Biol Soc 7 (NIL) (2005) 7591–5.

[3] M. D. Altman, J. P. Bardhan, B. Tidor, J. K. White, FFTSVD: a fast multiscale boundary-element method solver suitable for Bio-MEMS and biomolecule simulation, IEEE Trans. Comput-Aided Des. Integr. Circuits Syst. 25 (2) (2006) 274–284.

[4] A. W. Appel, An efficient program for many-body simulations, SIAM J. Sci. Stat. Comput. 6 (1985) 85–103.

[5] N. A. Baker, D. Sept, S. Joseph, M. J. Holst, J. A. McCammon, Electrostatics of nanosystems: application to microtubules and the ribosome, Proc. Natl. Acad. Sci. U.S.A. 98 (2001) 10037–10041.

[6] J. Barnes, P. Hut, A hierarchical O(n log n) force-calculation algorithm, Nature 324 (4) (1986) 446 – 449.

[7] R. Bharadwaj, A. Windemuth, S. Sridharan, B. Honig, A. Nicholls, The fast multipole boundary-element method for molecular electrostatics - an optimal approach for large systems, J. Comput. Chem. 16 (7) (1995) 898–913.

[8] A. J. Bordner, G. A. Huber, Boundary element solution of the linear Poisson-Boltzmann equation and a multipole method for the rapid calculation of forces on macromolecules in solution, J. Comput. Chem. 24 (3) (2003) 353–367.

[9] A. H. Boschitsch, M. O. Fenley, W. K. Olson, A fast adaptive multipole algorithm for calculating screened Coulomb (Yukawa) interactions, J. Comput. Phys. 151 (1) (1999) 212–241.

[10] A. H. Boschitsch, M. O. Fenley, H. X. Zhou, Fast boundary element method for the linear Poisson-Boltzmann equation, J. Phys. Chem. B 106 (10) (2002) 2741–2754.

[11] C. E. Capener, H. J. Kim, Y. Arinaminpathy, M. S. P. Sansom, Ion channels: structural bioinformatics and modelling, Hum. Mol. Genet. 11 (20) (2002) 2425–2433.

[12] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, J. Comput. Phys. 155 (2) (1999) 468–498.

[13] C. M. Cortis, R. A. Friesner, An automatic three-dimensional finite element mesh generation system for the Poisson-Boltzmann equation, J. Comput. Chem. 18 (13) (1997) 1570–1590.

[14] T. Darden, D. York, L. Pedersen, Particle mesh ewald: An $n\log(n)$ method for Ewald sums in large systems, J. Chem. Phys. 98 (12) (1993) 10089–10092.

[15] M. E. Davis, J. A. McCammon, Electrostatics in biomolecular structure and dynamics, Chem. Rev. 90 (3) (1990) 509–521.

[16] P. Debye, E. Huckel, Zur theorie der elektrolyte, Phys. Zeitschr. 24 (1923) 185–206.

[17] F. Figueirido, R. M. Levy, R. H. Zhou, B. J. Berne, Large scale simulation of macromolecules in solution: combining the periodic fast multipole method with multiple time step integrators, J. Chem. Phys. 106 (23) (1997) 9835–9849.

[18] M. K. Gilson, A. Rashin, R. Fine, B. Honig, On the calculation of electrostatic interactions in proteins, J. Mol. Biol. 184 (3) (1985) 503–516.

[19] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (2) (1987) 325–348.

[20] L. Greengard, V. Rokhlin, A new version of the fast multipole method for the laplace equation in three dimensions, Acta Numerica 6 (1997) 229–269.

[21] L. F. Greengard, J. F. Huang, A new version of the fast multipole method for screened coulomb interactions in three dimensions, J. Comput. Phys. 180 (2) (2002) 642–658.

[22] M. Holst, N. A. Baker, F. Wang, Adaptive multilevel finite element solution of the Poisson-Boltzmann equation i: algorithms and examples, J. Comput. Chem. 21 (2000) 1319–1342.

[23] A. H. Juffer, E. F. F. Botta, B. A. M. Vankeulen, A. Vanderploeg, H. J. C. Berendsen, The electric-potential of a macromolecule in a solvent - a fundamental approach, J. Comput. Phys. 97 (1) (1991) 144–171.

[24] S. Kapur, D. E. Long, IES3: Efficient electrostatic and electromagnetic simulation, IEEE Comput. Sci. Eng. 5 (4) (1998) 60–67.

[25] J. G. Kirkwood, On the theory of strong electrolyte solutions, J. Chem. Phys. 2 (1934) 767–781.

[26] I. Klapper, R. Hagstrom, R. Fine, K. Sharp, B. Honig, Focusing of electric fields in the active site of Cu-Zn superoxide dismutase: effects of ionic strength and amino-acid modification, Proteins 1 (1) (1986) 47–59.

[27] S. S. Kuo, M. D. Altman, J. P. Bardhan, B. Tidor, J. K. White, Fast methods for simulation of biomolecule electrostatics, in: ICCAD ’02: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design, ACM Press, New York, NY, USA, 2002.

[28] J. Liang, S. Subramaniam, Computation of molecular electrostatics with boundary element methods, Biophys. J. 73 (4) (1997) 1830–1841.

[29] B. Z. Lu, X. L. Cheng, J. F. Huang, J. A. McCammon, Order *N* algorithm for computation of electrostatic interactions in biomolecular systems, Proc. Natl. Acad. Sci. U. S. A. 103 (51) (2006) 19314–19319.

[30] B. Z. Lu, J. A. McCammon, Improved boundary element methods for Poisson-Boltzmann electrostatic potential and force calculations, J. Chem. Theory. Comput. 3 (3) (2007) 1134–1142.

[31] B. Z. Lu, D. Q. Zhang, J. A. McCammon, Computation of electrostatic forces between solvated molecules determined by the poisson-boltzmann equation using a boundary element method, J. Chem. Phys. 122 (21) (2005) 214102.

[32] E. T. Ong, K. H. Lee, K. M. Lim, A fast algorithm for three-dimensional electrostatics analysis: fast fourier transform on multipoles (FFTM), Int. J. Numer. Methods Eng. 61 (5) (2004) 633–656.

[33] E. T. Ong, K. M. Lim, K. H. Lee, H. P. Lee, A fast algorithm for three-dimensional potential fields calculation: fast Fourier transform on multipoles, J. Comput. Phys. 192 (1) (2003) 244–261.

[34] J. R. Phillips, J. K. White, A precorrected-FFT method for electrostatic analysis of complicated 3-D structures, IEEE Trans. Comput-Aided Des. Integr. Circuits Syst. 16 (10) (1997) 1059–1072.

[35] F. M. Richards, Areas, volumes, packing and protein structure, Annual Review in Biophysics and Bioengineering 6 (1977) 151–176.

[36] H. Robinson, Y. G. Gao, B. S. Mccrary, S. P. Edmondson, J. W. Shriver, A. H. J. Wang, The hyperthermophile chromosomal protein Sac7d sharply kinks DNA, Nature 392 (6672) (1998) 202–205.

[37] V. Rokhlin, Solution of acoustic scattering problems by means of second kind integral equations, Wave Motion 5 (3) (1983) 257–272.

[38] Y. Saad, M. H. Schultz, Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comp. 7 (3) (1986) 856–869.

[39] M. F. Sanner, A. J. Olson, J. C. Spehner, Reduced surface: an efficient way to compute molecular surfaces, Biopolymers 38 (3) (1996) 305–320.

[40] K. A. Sharp, B. Honig, Electrostatic interactions in macromolecules - theory and applications, Annu. Rev. Biophys. Biophys. Chem. 19 (1990) 301–332.

[41] W. Shi, J. Liu, N. Kakani, T. Yu, A fast hierarchical algorithm for 3-D capacitance extraction, in: DAC '98: Proceedings of the 35th annual conference on Design automation, ACM Press, New York, NY, USA, 1998.

[42] M. Tanaka, V. Sladek, J. Sladek, Regularization techniques applied to boundary element method, AMSE Appl. Mech. Rev. 47 (1994) 457–499.

[43] J. Tausch, J. White, A multiscale method for fast capacitance extraction, in: DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation, ACM Press, New York, NY, USA, 1999.

[44] M. Totrov, R. Abagyan, Rapid boundary element solvation electrostatics calculations in folding simulations: successful folding of a 23-residue peptide, Biopolymers 60 (2) (2001) 124–133.

[45] N. Unwin, Refined structure of the nicotinic acetylcholine receptor at 4 angstrom resolution, J. Mol. Biol. 346 (4) (2005) 967–989.

[46] J. Warwicker, H. C. Watson, Calculation of the electric-potential in the active-site cleft due to alpha-helix dipoles, J. Mol. Biol. 157 (4) (1982) 671–679.

[47] W. Xin, A. H. Juffer, A boundary element formulation of protein electrostatics with explicit ions, J. Comput. Phys. 223 (2007) 416–435.

[48] R. J. Zauhar, R. S. Morgan, A new method for computing the macromolecular electric-potential, J. Mol. Biol. 186 (4) (1985) 815–820.

[49] R. J. Zauhar, A. Varnek, A fast and space-efficient boundary element method for computing electrostatic and hydration effects in large molecules, J. Comput. Chem. 17 (7) (1996) 864–877.