Published in final edited form as:
SIAM J Sci Comput. 2010 January 20; 31(6): 4524–4552. doi:10.1137/090746173.

FAST MOLECULAR SOLVATION ENERGETICS AND FORCE COMPUTATION*

CHANDRAJIT BAJAJ† and **WENQI ZHAO**‡
CHANDRAJIT BAJAJ: bajaj@ices.utexas.edu; WENQI ZHAO: wzhao@ices.utexas.edu
† Department of Computer Sciences and Institute for Computational Sciences and Engineering, University of Texas at Austin, Austin, Texas, 78712
‡ Institute for Computational Sciences and Engineering, University of Texas at Austin, Austin, Texas, 78712

Abstract

The total free energy of a molecule includes the classical molecular mechanical energy (which is understood as the free energy in vacuum) and the solvation energy which is caused by the change of the environment of the molecule (solute) from vacuum to solvent. The solvation energy is important to the study of the inter-molecular interactions. In this paper we develop a fast surface-based generalized Born method to compute the electrostatic solvation energy along with the energy derivatives for the solvation forces. The most time-consuming computation is the evaluation of the surface integrals over an algebraic spline molecular surface (ASMS) and the fast computation is achieved by the use of the nonequispaced fast Fourier transform (NFFT) algorithm. The main results of this paper involve (a) an efficient sampling of quadrature points over the molecular surface by using nonlinear patches, (b) fast linear time estimation of energy and inter-molecular forces, (c) error analysis, and (d) efficient implementation combining fast pairwise summation and the continuum integration using nonlinear patches.

Keywords

generalized Born; molecular surface; fast summation; error analysis

1. Introduction

Most of the protein molecules live in the aqueous solvent environment and the stabilities of the molecules depend largely upon their configuration and the solvent type. Since the solvation energy term models the interaction between a molecule and the solvent, the computation of the molecular solvation energy (also known as molecule - solvent interaction energy) is a key issue in molecular dynamics (MD) simulations, as well as in determining the inter-molecular binding affinities “in-vivo” for drug screening. Molecular dynamics simulations where the solvent molecules are explicitly represented at atomic resolution, for example as in the popular package NAMD [1], provide direct information about the important influence of solvation. Moreover, as the total number of atoms of solvent molecules far outnumber the atoms of the solute, a larger fraction of the time is spent on computing the trajectory of the solvent molecules, even though the primary focus of the simulation is the configuration and energetics of the solute molecule. Implicit solvent models, attempt to considerably lower the cost of computation

*This research was supported in part by NSF grant: CNS-0540033, and in part by NIH grants: P20-RR020647, R01-GM074258, R01-GM073087, and R01-EB004873.

through a continuum representation (mean-field approximation) of the solvent [2]. In the implicit model, the solvation free energy G_{sol} which is the free energy change to transfer a molecule from vacuum to solvent, consists of three components: the energy to form a cavity in the solvent which is also known as the hydrophobic interactions, the van der Waals interactions between the molecule and the solvent, and the electrostatic potential energy between the molecule and the solvent (also known as polarization energy), $G_{\text{sol}} = G_{\text{cav}} + G_{\text{vdw}} + G_{\text{pol}}$. Based on the Weeks-Chandler-Andersen (WCA) perturbation theory [3,4], the non-polar solvation energies are of the form $G_{\text{cav}} + G_{\text{vdw}} = G^{(\text{rep})} + G^{(\text{rep})}$. In [5], $G_{(\text{rep})}$ is described as the weighted sum of the solvent-accessible surface area A_i of the atoms. In [31], a volume term is added: $G^{(\text{rep})}=\sum_{i=1}^{M}\gamma_i A_i+pV$, where p is the solvent pressure parameter and V is the solvent-accessible volume. In [6], the attractive van der Waals dispersion energy $G^{(\text{att})}=\sum_{i=1}^{M}G_i^{(\text{att})}$, where $G_i^{(\text{att})}=\rho_0\int u_i^{(\text{att})}(\mathbf{x}_i,\mathbf{y})\theta(\mathbf{y})\,d\mathbf{y}$, ρ_0 is the bulk density, $u_i^{(\text{att})}(\mathbf{x}_i,\mathbf{y})$ is the van der Waals dispersive component of the interaction between atom i in the solute and the volume of solvent at $\mathbf{y}$, $\theta(\mathbf{y})$ is a density distribution function for the solvent. Hence the non-polar solvation energies

$$G_{\text{cav}}+G_{\text{vdw}}=\sum_{i=1}^{M}\gamma_i A_i+pV+\rho_0\int u_i^{(\text{att})}(\mathbf{x}_i,\mathbf{y})\theta(\mathbf{y})\,d\mathbf{y}. \tag{1.1}$$

The electrostatic solvation energy is caused by the induced polarization in the solvent when the molecule is dissolved in the solvent, therefore

$$G_{\text{pol}}=\frac{1}{2}\int\varphi_{\text{reaction}}(\mathbf{r})\rho(\mathbf{r})\,d\mathbf{r}, \tag{1.2}$$

where $\varphi_{\text{reaction}} = \varphi_{\text{solvent}} - \varphi_{\text{gas-phase}}$, $\varphi(\mathbf{r})$ and $\rho(\mathbf{r})$ are the electrostatic potential and the charge density at $\mathbf{r}$, respectively.

The Poisson-Boltzmann (PB) model was developed to compute the electrostatic solvation energy by solving the equation $-\nabla(\varepsilon(\mathbf{x})\nabla\varphi(\mathbf{x})) = \rho(\mathbf{x})$ for the electrostatic potential φ. Numerical methods to solve the equation include the finite difference method [7,8], finite element method [9,10], and boundary element method [11]. However the PB methods are prohibitive for large molecules such as proteins due to the limited computational resources. As an alternative, (1.2) is approximated by a generalized Born (GB) model which is in the form of discrete sum [12]

$$G_{\text{pol}}=-\frac{\tau}{2}\sum_{i,j}\frac{q_i q_j}{\left[r_{ij}^2+R_i R_j\exp(-\frac{r_{ij}^2}{4R_i R_j})\right]^{\frac{1}{2}}}, \tag{1.3}$$

where $\tau=\frac{1}{\varepsilon_p}-\frac{1}{\varepsilon_w}$, εp and εw are the solute (low) and solvent (high) dielectric constants, q_i and R_i are the charge and effective Born radius of atom i, respectively, and r_{ij} is the distance between atoms i and j. The solvation force acting on atom α, which is part of the forces driving dynamics is computed as

$$\mathbf{F}_{\alpha}^{\mathrm{sol}} = -\frac{\partial G_{\mathrm{sol}}}{\partial \mathbf{x}_{\alpha}}. \tag{1.4}$$

Because the GB calculation is much faster than solving the PB equation, the GB model is widely used in the MD simulations. Programs which implement the GB methods include CHARMM [13], Amber [14], Tinker [15], and Impact which is now part of Schrodinger, Inc.'s FirstDiscovery program suite. Even though the GB computation is much faster than the PB model, the computation of the Born radius R_i is still slow. During the MD simulation, the Born radii need to be frequently recomputed at different time steps. Because this part of computation is too time-consuming, there are attempts to accelerate the MD simulation by computing the Born radii at a larger time step. For example, in [16] in their test of a 3 ns GB simulation of a 10-base pair DNA duplex, they change the time step of computing the Born radii and long-range electrostatic energy from 1 fs to 2 fs. This reduces the time of carrying out the simulation from 13.84 hours to 7.16 hours. From this example we can see that the calculation of the Born radii takes a large percentage of total computation time in the MD simulation. In the long dynamic runs, this decrease in the frequency of evaluating the effective Born radii are not accurate enough to conserve energy which restricts the MD simulation of the protein folding process to small time scale [17]. Hence it is demanding to calculate the Born radii and the solvation energy accurately and efficiently.

In this paper we develop a method for fast computation of the GB solvation energy, along with the energy derivatives for the solvation forces, based on a discrete and continuum model of the molecules (Figure 1.1). An efficient method of sampling quadrature points on the nonlinear patch is given. We also show that the error of the Born radius calculation is controlled by the size of the triangulation mesh and the regularity of the periodic function used in the fast summation algorithm. The time complexity of the forces computation is reduced from the original $O(MN + M^2)$ to nearly linear time $O(N + M + n^3 \log n + M \log M)$, where M is the number of atoms of a molecule, N is the number of integration points that we sample on the surface of the molecule when we compute the Born radius for each atom, and n is a parameter introduced in the fast summation algorithm. The fast summation method shows its advantage when it is applied to the Born radius calculations for macromolecules, where there could be tens of thousands or millions of atoms, and N could be even larger. In the fast summation method, one only need to choose a small n which is much smaller than M and N to get a good approximation, which makes the new fast summation based GB method more efficient.

The rest of the paper is organized as follows: in Section 2 we explain the geometric model that our energy and force computation are based on; we discuss in detail the energy computation in Section 3 and the force computation in Section 4; some implementation results are shown in Section 5; some details such as the fast summation algorithm and the NFFT algorithm are discussed in the appendix.

2. Geometric model

2.1. Gaussian surface

The electron density and shape are used in a similar sense in the literature with respect to the modeling of molecular surfaces or interfaces between the molecule and its solvent. The electron density of atom i at a point $\mathbf{x}$ is represented as a Gaussian function: $f(\mathbf{x}) = e^{\beta\left(\frac{|\mathbf{x}-\mathbf{x}_i|^2}{r_i^2} - 1\right)}$ where $\mathbf{x}_i$, r_i are the position of the center and radius of the atom k. If we consider the function value of 1, we see that it is satisfied at the surface of the sphere ($\mathbf{x}$: $|\mathbf{x} - \mathbf{x}_i| = r_i$). Using this model, the electron density at $\mathbf{x}$ due to a protein with M atoms is just a summation of Gaussians:

$$f(\mathbf{x})=\sum_{i=1}^{M} e^{\beta\left(\frac{|\mathbf{x}-\mathbf{x}_i|^2}{r_i^2}-1\right)} \tag{2.1}$$

where β is a parameter used to control the rate of decay of the Gaussian and known as the *blobbiness* of the Gaussian. In [18] $\beta = -2.3$, *isovalue* = 1 is indicated as a good approximation to the molecular surface.

2.2. Triangular mesh

The triangular mesh of the Gaussian surface is generated by using the dual contouring method [19,20]. In the dual contouring method a top-down octree is recursively constructed to enforce that each cell has at most one isocontour patch. The edges whose endpoints lie on different side of the isocontour are tagged as sign change edges. In each cube that contains a sign change edge, we compute the intersection points (and their unit normals) of the isocontour and the edges of the cube, denoted as p_i and n_i, and compute the minimizer point in this cube which minimizes the quadratic error function (QEF) [21]:

$$\mathrm{QEF}(x)=\sum_i [n_i \cdot (x - p_i)]^2.$$

Since each sign change edge is shared by either four cubes (uniform grid) or three cubes (adaptive grid), connecting the minimizer points of these neighboring cubes forms a quad or a triangle that approximates the isocontour. We divide the quads into triangles to generate the pure triangular mesh.

2.3. Algebraic spline molecular surface (ASMS)

The triangular mesh is a linear approximation to the Gaussian surface. In our solvation energy computation, we generate another higher order approximation called ASMS model (Figure 2.3 (f)) based on the triangular mesh to improve accuracy and efficiency [22]. Starting from the triangular mesh, we first construct a prism scaffold as follows. Let $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$ be a triangle of the mesh where $\mathbf{v}_i$, $\mathbf{v}_j$, $\mathbf{v}_k$ are the vertices of the triangle and $\mathbf{n}_i$, $\mathbf{n}_j$, $\mathbf{n}_k$ be their unit normals. Define $\mathbf{v}_l(\lambda) = \mathbf{v}_l + \lambda\mathbf{n}_l$. Then the prism is define as

$$D_{ijk}:=\{\mathbf{p}:\mathbf{p}=b_1\mathbf{v}_i(\lambda)+b_2\mathbf{v}_j(\lambda)+b_3\mathbf{v}_k(\lambda), \lambda \in I_{ijk}\},$$

where $b_1, b_2, b_3 \in [0, 1]$, $b_1 + b_2 + b_3 = 1$, and I_{ijk} is a maximal open interval such that (i) $0 \in I_{ijk}$, (ii) for any $\lambda \in I_{ijk}$, $\mathbf{v}_i(\lambda)$, $\mathbf{v}_j(\lambda)$ and $\mathbf{v}_k(\lambda)$ are not collinear, and (iii) for any $\lambda \in I_{ijk}$, $\mathbf{n}_i$, $\mathbf{n}_j$ and $\mathbf{n}_k$ point to the same side of the plane $P_{ijk}(\lambda):= \{\mathbf{p}: \mathbf{p} = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda)\}$ (Figure 2.1).

Next we define a function over the prism D_{ijk} in the cubic Bernstein-Bezier (BB) basis:

$$F(b_1, b_2, b_3, \lambda)=\sum_{i+j+k=3} b_{ijk}(\lambda)B_{ijk}^3(b_1, b_2, b_3), \tag{2.2}$$

where $B^3_{ijk}(b_1,b_2,b_3)=\frac{3!}{i!j!k!}b_1^i b_2^j b_3^k$. The ASMS denoted as Γ is the zero contour of F. The scheme for defining the coefficients b_{ijk} are defined is described in detail in [22]. In short they are defined such that

- the vertices of the triangular mesh are points on Γ;
- Γ is C^1 at the vertices of mesh;
- Γ is C^1 at the midpoints of the mesh edges.

Later, given the barycentric coordinates of a point (b_1, b_2, b_3) in triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$, we solve the equation $F(b_1, b_2, b_3, \lambda) = 0$ for λ by Newton's method. In this way we can get the corresponding point (x,y,z) on Γ:

$$(x,y,z)^{\mathrm{T}}=b_1\mathbf{v}_i(\lambda)+b_2\mathbf{v}_j(\lambda)+b_3\mathbf{v}_k(\lambda). \tag{2.3}$$

We have proved in [22] that the ASMS model is C^1 everywhere if the normals of the mesh satisfy certain symmetry conditions. The error between the ASMS and the Gaussian surface is bounded and we have shown that the ASMS converges to the Gaussian surface at the rate of $O(h^3)$ where h is the maximum edge length of the mesh.

3. Fast solvation energy computation

3.1. Method

Similarly to what is done for other GB models, we use (1.3) as the electrostatic solvation energy function. Before we compute (1.3), we need to first compute the effective Born radius R_i for every atom which reflects the depth a charge buried inside the molecule (Figure: 3.1). An atom buried deep in a molecule has a larger Born radius, whereas an atom near the surface has a smaller radius. Hence surfactant atoms have a stronger impact on the polarization. Given a discrete van der Waals (vdW) atom model, as long as we know R_i for each atom, we can compute (1.3) by using the fast multipole method (FMM) [23] with the time complexity $O(M \log M)$. However the Born radii computation is not easy and is very time-consuming. There are various ways of computing the Born radius as summarized in [24]. These methods can be divided into two categories: volume integration based methods and surface integration based methods. In general, the surface integration methods are more efficient than the volume integration methods due to the decreased dimension. So we adopt the surface integration method given in [25] to compute the Born radius:

$$R_i^{-1}=\frac{1}{4\pi}\int_\Gamma \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\mathrm{d}S \qquad i=1,\ldots,M, \tag{3.1}$$

where Γ is the molecule-solvent interface, $\mathbf{x}_i$ is the center of atom i, and $\mathbf{n}(\mathbf{r})$ is the unit normal on the surface at $\mathbf{r}$ and we use ASMS as the model of Γ.

Applying the Gaussian quadrature, We compute (3.1) numerically:

$$R_i^{-1}=\frac{1}{4\pi}\sum_{k=1}^{N} w_k\frac{(\mathbf{r}_k-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k-\mathbf{x}_i|^4} \qquad i=1,\ldots,M, \tag{3.2}$$

where w_k and $\mathbf{r}_k$ are the Gaussian integration weights and nodes on Γ (Figure 3.2). $\mathbf{r}_k$ are computed by mapping the Gaussian nodes of a master triangle to the algebraic patch via the

transformation $\mathscr{T}$. Let $\mathbf{r}_k^0$ and w_k^0 be one of the Gaussian nodes and weights on the master triangle. Then the corresponding node $\mathbf{r}_k$ and weight w_k are $\mathbf{r}_k = \mathscr{T}(\mathbf{r}_k^0)$ and $w_k = w_k^0 |J(\mathscr{T})|$ where $|J(\mathscr{T})|$ is the Jacobian determinant of $\mathscr{T}$.

We formalize (3.2) in two steps. First we split it into two parts:

$$R_i^{-1} = \frac{1}{4\pi} \sum_{k=1}^{N} \frac{w_k \mathbf{r}_k \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} - \frac{1}{4\pi} \sum_{k=1}^{N} \frac{w_k \mathbf{x}_i \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4}. \tag{3.3}$$

Then we split the second summation in (3.3) into three components:

$$\sum_{k=1}^{N} \frac{w_k \mathbf{x}_i \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} = x_i \sum_{k=1}^{N} \frac{w_k n_x^k}{|\mathbf{r}_k - \mathbf{x}_i|^4} + y_i \sum_{k=1}^{N} \frac{w_k n_y^k}{|\mathbf{r}_k - \mathbf{x}_i|^4} + z_i \sum_{k=1}^{N} \frac{w_k n_z^k}{|\mathbf{r}_k - \mathbf{x}_i|^4}. \tag{3.4}$$

The first summation in (3.3) and the three summations in (3.4) without the coefficients in front are of the common form:

$$G(\mathbf{x}_i) = \sum_{k=1}^{N} c_k g(\mathbf{x}_i - \mathbf{r}_k) \quad i = 1, \ldots, M, \tag{3.5}$$

with the kernel function $g(\mathbf{x} - \mathbf{r}_k) = \frac{1}{|\mathbf{x} - \mathbf{r}_k|^4}$ and the coefficient $c_k = w_k \mathbf{r}_k \cdot \mathbf{n}(\mathbf{r}_k)$, $w_k n_x^k, w_k n_y^k, w_k n_z^k$, respectively. (3.5) can be efficiently computed by using the fast summation algorithm introduced in [26] with complexity $O(M + N + n^3 \log n)$, where n is a parameter used in the fast summation algorithm.

3.2. Fast summation

The fast summation algorithm is published in [26]. For convenience, we discuss this algorithm in this section briefly. The fast summation algorithm is often applied to compute the summations of the form

$$G(\mathbf{x}_i) = \sum_{k=1}^{N} c_k g(\mathbf{x}_i - \mathbf{r}_k), \quad i = 1, \ldots, M, \tag{3.6}$$

where the kernel function g is a fast decaying function. Cutting off the tail of g, one can assume that the support of g is bounded. In our Born radii computation, since the distance between $\mathbf{x}_i$ and $\mathbf{r}_k$ is no less than the smallest radius of the atoms, there is no singularity in g. Without loss of generality, we assume $\mathbf{x} - \mathbf{r}_k \in \Pi := [-\frac{1}{2}, \frac{1}{2}]^3$. After duplicating g in the other intervals, g can be extended to be a periodic function of period one in $\mathbb{R}^3$ and this periodic function can be decomposed into the Fourier series:

$$g(\mathbf{x} - \mathbf{r}_k) = \sum_{\omega \in I_\infty} g_\omega e^{2\pi i \omega \cdot (\mathbf{x} - \mathbf{r}_k)}, \tag{3.7}$$

where $I_\infty := \{(\omega_1, \omega_2, \omega_3) \in \mathbb{Z}^3\}$ and $g_\omega = \int_\Pi g(\mathbf{x}) e^{-2\pi i \boldsymbol{\omega}\cdot\mathbf{x}}\, d\mathbf{x}$. We approximate (3.7) by a truncated series:

$$g(\mathbf{x} - \mathbf{r}_k) \approx \sum_{\omega \in I_n} g_\omega e^{2\pi i(\mathbf{x}-\mathbf{r}_k)\cdot\omega}, \tag{3.8}$$

where $I_n := \{(\omega_1, \omega_2, \omega_3) \in \mathbb{Z}^3 : -\frac{n}{2} \le \omega_i < \frac{n}{2}\}$. We compute the Fourier coefficients g_ω numerically by

$$g_\omega = \frac{1}{n^3} \sum_{\mathbf{j} \in I_n} g\left(\frac{\mathbf{j}}{n}\right) e^{-2\pi i \omega\cdot\mathbf{j}/n}, \quad \boldsymbol{\omega} \in I_n. \tag{3.9}$$

by using the fast Fourier transform (FFT) algorithm with complexity $O(n^3 \log n)$.

Plugging (3.8) into (3.6), we get

$$\begin{aligned} G(\mathbf{x}_i) \approx \sum_{k=1}^{N} c_k \left(\sum_{\omega \in I_n} g_\omega e^{2\pi i(\mathbf{x}_i - \mathbf{r}_k)\cdot\omega} \right) &= \sum_{\omega \in I_n} g_\omega \left(\sum_{k=1}^{N} c_k e^{-2\pi i \omega\cdot\mathbf{r}_k} \right) e^{2\pi i \omega\cdot\mathbf{x}_i} \\ &= \sum_{\omega \in I_n} g_\omega a_\omega e^{2\pi i \omega\cdot\mathbf{x}_i} \end{aligned} \tag{3.10}$$

where

$$a_\omega = \sum_{k=1}^{N} c_k e^{-2\pi i \omega\cdot\mathbf{r}_k}. \tag{3.11}$$

(3.10) is computed by using the NFFT algorithm with complexity $O(n^3 \log n + M)$ and (3.11) is computed by the NFFT^T algorithm with complexity $O(n^3 \log n + N)$. Hence the total complexity of computing (3.6) is $O(N + M + n^3 \log n)$, which is significantly faster than the the trivial $O(MN)$ summation method once the number of terms in the Fourier series n is much smaller than M and N. We explain the NFFT algorithm and the NFFT^T algorithm in Appendix A and B, respectively.

3.3. Error analysis

The numerical analysis of the error introduced during the computation of (3.1) can be decomposed as follows: (i) the sum of a quadrature error E_Q; (ii) some "fast computation" error in the evaluation of the quadrature itself. The latter error is then decomposed in three terms, which correspond to different steps in the numerical procedure. They are the truncation error E_{FS} when we truncate the Fourier series (3.7) into finite terms, NFFT^T errors $E_{\boldsymbol{\omega}}$ when we compute the coefficients (3.11), and an NFFT error E_{NFFT} when we finally evaluate (3.10) by the NFFT algorithm.

Let I_i and $\tilde{I}_i$ denote the exact integration and the numerical output of (3.1) for atom i, respectively. Then We have

$$I_i = \tilde{I}_i + E_{\text{NFFT}} + E_{\text{NFFT}^{\text{T}}} + E_{\text{FS}} + E_{\text{Q}}.$$

Let $\|E\|_\infty = \max_i |I_i - \tilde{I}_i|$. We have

$$\|E\|_\infty \le \|E_{\text{Q}}\|_\infty + \|E_{\text{FS}}\|_\infty + \|E_{\text{NFFT}}\|_\infty + \|E_{\text{NFFT}^{\text{T}}}\|_\infty. \tag{3.12}$$

Next we will analyze each individual error $\|E_{\text{Q}}\|_\infty$, $\|E_{\text{FS}}\|_\infty$, $\|E_{\text{NFFT}}\|_\infty$, and $\|E_{\text{NFFT}^{\text{T}}}\|_\infty$.

3.3.1. Quadrature error—Let Γ_e be one of the algebraic patches on the molecular surface Γ. Suppose Γ_e is built based on a triangle $e := [\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k]$. Any point $(b_1, b_2, b_3) \in e$ can be mapped to a point $\mathbf{r}(b_1, b_2) \in \Gamma_e$. The integration (3.1) over Γ_e is

$$\begin{aligned} I_e &= \int_{\Gamma_e} \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4} \mathrm{d}S \\ &= \iint_{\Omega_0} \frac{(\mathbf{r}(b_1,b_2)-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r}(b_1,b_2))}{|\mathbf{r}(b_1,b_2)-\mathbf{x}_i|^4} |J| \mathrm{d}b_1 \mathrm{d}b_2 \end{aligned} \tag{3.13}$$

where Ω_0 is the canonical triangle, (b_1, b_2, b_3) is the barycentric coordinates of the points in Ω_0 and $|J|$ is the Jacobian. Let $f(b_1, b_2)$ denote the integrand in (3.13). As we discuss in Appendix C, $f(b_1, b_2) \in C^\infty(\Omega_0)$. Suppose we use an s-th order quadrature rule on element e, then

$$I_e = \iint_{\Omega_0} f(b_1, b_2) \mathrm{d}b_1 \mathrm{d}b_2 = \sum_{k=1}^{s_e} w_k f(b_1^k, b_2^k) + E. \tag{3.14}$$

We expand $f(b_1, b_2)$ in a Taylor series around a point $(b_1', b_2', b_3') \in \Omega_0$:

$$f(b_1, b_2) = P_s(b_1, b_2) + R_s(b_1, b_2), \tag{3.15}$$

where $P_s(b_1, b_2)$ is a polynomial of degree s:

$$P_s(b_1, b_2) = f(b_1', b_2') + \frac{1}{s!}\left[(b_1 - b_1')\frac{\partial}{\partial b_1} + (b_2 - b_2')\frac{\partial}{\partial b_2}\right]^s f(b_1', b_2') \tag{3.16}$$

and the residue R_s is

$$R_s(b_1, b_2) = \frac{1}{(s+1)!}\left[(b_1 - b_1')\frac{\partial}{\partial b_1} + (b_2 - b_2')\frac{\partial}{\partial b_2}\right]^{s+1} f(b_1^*, b_2^*), \quad (b_1^*, b_2^*) \in \Omega_0. \tag{3.17}$$

Then the error E becomes

$$E=\iint_{\Omega_0} R_s(b_1,b_2)\mathrm{d}b_1\mathrm{d}b_2-\sum_{k=1}^{s_e} w_k R_s(b_1^k,b_2^k).$$

Let $W_k = \max(|w_k|)$, we get

$$|E| \le \iint_{\Omega_0} |R_n(b_1,b_2)|\mathrm{d}b_1\mathrm{d}b_2+W_k\sum_{k=1}^{3}|R_n(b_1^k,b_2^k)|.$$

Within Ω_0, $|b_1-b_1'| \le 1$ and $|b_2-b_2'| \le 1$, hence

$$|R_s(b_1,b_2)| \le \frac{1}{(s+1)!}[|\frac{\partial}{\partial b_1}|+|\frac{\partial}{\partial b_2}|]^{s+1} f(b_1^*,b_2^*), \tag{3.18}$$

where $|\frac{\partial}{\partial b}|.$ denotes $|\frac{\partial .}{\partial b}|$. By the chain rule,

$$\frac{\partial}{\partial b_1}=\frac{\partial}{\partial x}\frac{\partial x}{\partial b_1}+\frac{\partial}{\partial y}\frac{\partial y}{\partial b_1}+\frac{\partial}{\partial z}\frac{\partial z}{\partial b_1}.$$

According to (2.3), we have $\frac{\partial x}{\partial b_1}=v_1^x-v_3^x+\lambda(n_1^x-n_3^x)$. Let h_{max} be the maximum edge length of the triangular mesh, $\lambda_{max} = \max\{|\lambda|\}$, and $h = \max(h_{max}, \lambda_{max})$. Then we have $|\frac{\partial x}{\partial b_1}| \le 2h$. Similarly, we can get the same bound for the derivatives of x, y, z with respect to b_1 and b_2. Therefore

$$|R_s(b_1,b_2)| \le \frac{(2h)^{s+1}}{(s+1)!}[|\frac{\partial}{\partial x}|+|\frac{\partial}{\partial y}|+|\frac{\partial}{\partial z}|]^{s+1}\ \tilde{f}\,(x^*,y^*,z^*) \le C\frac{(2h)^{s+1}}{(s+1)!} \tag{3.19}$$

where $(x^*,y^*,z^*)=b_1^*\mathbf{v}_i(\lambda)+b_2^*\mathbf{v}_j(\lambda)+b_3^*\mathbf{v}_k(\lambda)$, $\tilde{f}\,(x^*,y^*,z^*)=f(b_1^*,b_2^*)$, and the constant $C=\max_{(x,y,z)\in\Gamma}|D^{s+1}\ \tilde{f}\,(x,y,z)|<\infty$. Noticing that the area of Ω_0 is 1/2, we can write

$$|E| \le (\frac{1}{2}+s_eW_k)\frac{2^{s+1}}{(s+1)!}Ch^{s+1}. \tag{3.20}$$

Even though a greater number of quadrature nodes correspond to the higher order of accuracy, the increase in complexity is a limiting factor. Meanwhile, since the ASMS error is of the order h^3, there is no point in a very accurate approximation of (3.20) to too high an order. As a trade-off, we use a two dimensional 3-point Gaussian quadrature over the triangle Ω_0 which is of order 2 [27]. So $s = 2$ and $s_e = 3$. The nodes are $(\frac{1}{6},\frac{1}{6},\frac{1}{3})$ and its permutations. $W_k=\frac{1}{3}$ for k = 1,2,3. Then

$$|E| \leq 2Ch^3. \tag{3.21}$$

Suppose there are N_e patches on Γ, then $|E_Q| \leq 2N_eCh^3$. So we have the same bound

$$\left\|E_Q\right\|_\infty \leq 2N_eCh^3. \tag{3.22}$$

3.3.2. Fast summation error—According to the fast summation method described in Section 3.2, the Fourier series is truncated into a finite series

$$E^i_{FS} := \sum_{k=1}^{N} c_k \left(\sum_{\omega \in I_\infty \setminus I_n} g_\omega e^{2\pi i \omega \cdot (\mathbf{x}_i - \mathbf{r}_k)} \right) = \sum_{k=1}^{N} c_k T^i_k$$

where T^i_k denotes the truncation error of the Fourier series. Hence

$$|E^i_{FS}| \leq \left\|\mathbf{c}\right\|_\infty \sum_{k=1}^{N} |T^i_k| \tag{3.23}$$

where $\left\|\mathbf{c}\right\|_\infty := \max_{k=1,\ldots,N} |c_k|$,

$$|T^i_k| = \left| \sum_{\omega \in I_\infty \setminus I_n} g_\omega e^{2\pi i \omega \cdot (\mathbf{x}_i - \mathbf{r}_k)} \right| \leq \sum_{\omega \in I_\infty \setminus I_n} |g_\omega| \tag{3.24}$$

with

$$g_\omega = \int_\Pi g(\mathbf{x}) e^{-2\pi i \omega \cdot \mathbf{x}} d\mathbf{x} \tag{3.25}$$

and g being the the kernel function in the fast summation. In the Born radii calculation, $g(\mathbf{x}) = \frac{1}{|\mathbf{x}|^4}$. As defined in Section 3.2, Π is bounded and excludes 0. Let $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$. Then we rewrite $\Sigma_{\boldsymbol{\omega} \in I_\infty \setminus I_n} |g_{\boldsymbol{\omega}}|$ as

$$\sum_{\omega \in I_\infty \setminus I_n} |g_\omega| = \sum_{i,j,k,=0,1} \sum_{\omega_1 = n+1}^{\infty} \sum_{\omega_2 = n+1}^{\infty} \sum_{\omega_3 = n+1}^{\infty} |g_{(-1)^i \omega_1 (-1)^j \omega_2 (-1)^k \omega_3}|. \tag{3.26}$$

By successive integration by parts for each dimension, we get

$$g_{\omega_1\omega_2\omega_3}=\left(\frac{-i}{2\pi\omega_1}\right)^{m_1}\left(\frac{-i}{2\pi\omega_2}\right)^{m_2}\left(\frac{-i}{2\pi\omega_3}\right)^{m_3}\int_{\Pi}D^m g(\mathbf{x})e^{-2\pi i\omega\cdot\mathbf{x}}\mathrm{d}\mathbf{x},$$

where $m = m_1 + m_2 + m_3$ and $D^m g=(\frac{\partial^{m_1}}{\partial x^{m_1}}+\frac{\partial^{m_2}}{\partial y^{m_2}}+\frac{\partial^{m_3}}{\partial z^{m_3}})g$. Therefore

$$|g_{\omega_1\omega_2\omega_3}| \le \frac{1}{(2\pi)^m\omega_1^{m_1}\omega_2^{m_2}\omega_3^{m_3}}\int_{\Pi}|D^m g(\mathbf{x})|\mathrm{d}\mathbf{x}.$$

Let $\mu_m = \int_\Pi |D^m g(\mathbf{x})|\, \mathrm{d}\mathbf{x}$. We obtain $|g_{\omega_1\omega_2\omega_3}| \le \frac{\mu_m}{(2\pi)^m\omega_1^{m_1}\omega_2^{m_2}\omega_3^{m_3}}$. For the other terms in (3.26) we have the same upper bound. If we assume $m_1, m_2, m_3 \ge 2$, then

$$\begin{aligned}|T_k^i| &\le \frac{8\mu_m}{(2\pi)^m}\left(\sum_{\omega_1=n+1}^{\infty}\frac{1}{\omega_1^{m_1}}\right)\left(\sum_{\omega_2=n+1}^{\infty}\frac{1}{\omega_2^{m_2}}\right)\left(\sum_{\omega_3=n+1}^{\infty}\frac{1}{\omega_3^{m_3}}\right)\\ &\le \frac{8\mu_m}{(2\pi)^m}\left(\int_n^{\infty}\frac{1}{\omega_1^{m_1}}\mathrm{d}\omega_1\right)\left(\int_n^{\infty}\frac{1}{\omega_2^{m_2}}\mathrm{d}\omega_2\right)\left(\int_n^{\infty}\frac{1}{\omega_3^{m_3}}\mathrm{d}\omega_3\right)\\ &=\frac{8\mu_m}{(2\pi)^m(m_1-1)(m_2-1)(m_3-1)n^{m-3}}.\end{aligned}$$

For $m_1 = m_2 = m_3$, we have

$$|T_k^i| \le \frac{8\mu_6}{(2\pi)^6 n^3}. \tag{3.27}$$

Then for (3.24), we have

$$|E_{\mathrm{FS}}^i| \le \|\mathbf{c}\|_\infty \frac{8\mu_6 N}{(2\pi)^6 n^3}. \tag{3.28}$$

In fact, the right hand side of (3.28) is independent of i. Therefore we get

$$\|E_{\mathrm{FS}}\|_\infty \le \|\mathbf{c}\|_\infty \frac{8\mu_6 N}{(2\pi)^6 n^3}. \tag{3.29}$$

3.3.3. NFFT error—The error analysis of the NFFT algorithm is thoroughly discussed at the end of Appendix A. This error estimation is derived based on the analysis in [28]. In summary, the NFFT error is split into the *aliasing error* E^1_{NFFT} and the *truncation error* E^2_{NFFT} [28]:

$$\|E_{\mathrm{NFFT}}\|_\infty \le \|E^1_{\mathrm{NFFT}}\|_\infty + \|E^2_{\mathrm{NFFT}}\|_\infty.$$

The error bounds of E^1_{NFFT} and E^2_{NFFT} are

$$\left\|E^1_{\text{NFFT}}\right\|_\infty \le \left\|\widehat{G}\right\|_1 \max_{\omega\in I_n} \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \left|\frac{C_{\omega+\mathbf{i}\sigma n}(\xi)}{C_\omega(\xi)}\right|, \tag{3.30}$$

$$\left\|E^2_{\text{NFFT}}\right\|_\infty \le \frac{1}{\sigma^3 n^3} \max_{\omega\in I_n}(C^{-1}_\omega(\xi)) \left\|\widehat{G}\right\|_1 \max_i \sum_{\mathbf{l}\in I_{\sigma n}} \left|\xi(\mathbf{x}_i - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}_i - \frac{\mathbf{l}}{\sigma n})\right|, \tag{3.31}$$

where ξ is a 1-periodic window function defined in Appendix A, $C_{\boldsymbol{\omega}}(\xi)$ are the Fourier coefficients of ξ, and η is a truncated version of ξ. In the fast summation method (3.10), $\|\hat{G}\|_1 = \Sigma_{\boldsymbol{\omega}\in I_n}|g_{\boldsymbol{\omega}}a_{\boldsymbol{\omega}}|$, where $g_{\boldsymbol{\omega}}$ and $a_{\boldsymbol{\omega}}$ are defined in Section 3.2. Combining (3.30) and (3.31), one obtains

$$\left\|E_{\text{NFFT}}\right\|_\infty \le C(\xi, m, \sigma)\left\|\widehat{G}\right\|_1. \tag{3.32}$$

In [26], the coefficient $C(\xi, m, \sigma)$ is given for some special ξ. They are

- Gaussian, $\xi(\mathbf{x}) = (\pi b)^{-1/2} e^{-\|\sigma n\mathbf{x}\|^2/b}$, where $b := \frac{2\sigma}{2\sigma-1}\frac{m}{\pi}$, the coefficient $C(\xi, m, \sigma) = 4e^{-m\pi(1-1/(2\sigma-1))}$;
- cardinal central B-splines [29], $\xi(\mathbf{x}) = M_{2m}(\sigma n\mathbf{x})$, the coefficient $C(\xi, m, \sigma) = 4(\frac{1}{2\sigma-1})^{2m}$;
- powers of sinc function, $\xi(\mathbf{x}) = \frac{n(2\sigma-1)}{2m}\text{sinc}^{2m}\left(\frac{(2\sigma-2)n\pi\mathbf{x}}{2m}\right)$, the coefficient $C(\xi, m, \sigma) = \frac{1}{m-1}\left(\frac{2}{\sigma^{2m}} + (\frac{\sigma}{2\sigma-1})^{2m}\right)$;
- Kaiser-Bessel function [30]

$$\xi(\mathbf{x}) = \frac{1}{\pi}\begin{cases} \dfrac{\sinh\left(b\sqrt{m^2-(\sigma n)^2\left\|\mathbf{x}\right\|^2}\right)}{\sqrt{m^2-(\sigma n)^2\left\|\mathbf{x}\right\|^2}}, & \|x\| \le \frac{m}{\sigma n}, \\ \dfrac{\sinh\left(b\sqrt{(\sigma n)^2\left\|\mathbf{x}\right\|^2-m^2}\right)}{\sqrt{(\sigma n)^2\left\|\mathbf{x}\right\|^2-m^2}}, & \text{otherwise,} \end{cases}$$

$$C(\xi, m, \sigma) = 5\pi^2 m^{3/2}\sqrt[4]{1-\tfrac{1}{\sigma}}\, e^{-m2\pi\sqrt{1-1/\sigma}}.$$

3.3.4. NFFT$^{\mathrm{T}}$ error—As we mentioned in Section 3.2, (3.11) is computed by the NFFT$^{\mathrm{T}}$ algorithm and then they are plugged in (3.10) for the following evaluation of the summation. So the NFFT error $E_{\text{NFFT}^{\mathrm{T}}}$ is

$$E_{\text{NFFT}^{\mathrm{T}}} = \sum_{\omega\in I_n} g_\omega E_\omega e^{2\pi i\omega\cdot\mathbf{x}_i}, \tag{3.33}$$

where $E_{\boldsymbol{\omega}}$ denotes the error of the NFFT$^{\mathrm{T}}$ algorithm and $g_{\boldsymbol{\omega}}$ is the same as is defined in (3.25). Then we have

$$|E_{\mathrm{NFFT^T}}| \le \sum_{\omega\in I_n}|g_\omega E_\omega| \le \max_{\omega\in I_n}|E_\omega|\sum_{\omega\in I_n}|g_\omega| = \|E_\omega\|_\infty \|g\|_1 \tag{3.34}$$

with $\|E_\omega\|_\infty := \max_{\omega\in I_n}|E_\omega|$ and $\|g\|_1 := \sum_{\omega\in I_n}|g_\omega|$.

As we discussed in Appendix B, the NFFT^T error $E_{\boldsymbol{\omega}}$ is decomposed into the aliasing error (E_ω^1) and the truncation error (E_ω^2), $E_\omega = E_\omega^1 + E_\omega^2$. So

$$\|E_\omega\|_\infty \le \|E_\omega^1\|_\infty + \|E_\omega^2\|_\infty,$$

where $\|E_\omega^1\|_\infty = \max_{\omega\in I_n}|E_\omega^1|$ and $\|E_\omega^2\|_\infty = \max_{\omega\in I_n}|E_\omega^2|$. Based on the error bounds derived in Appendix B,

$$\|E_\omega^1\|_\infty \le \|\mathbf{c}\|_1 \max_{\omega\in I_n}\sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}}\frac{C_{\omega+\mathbf{i}\sigma n}(\xi)}{C_\omega(\xi)} \tag{3.35}$$

and

$$\|E_\omega^2\|_\infty \le \frac{1}{(\sigma n)^3}\|\mathbf{c}\|_1 \max_{\omega\in I_n}(C_\omega^{-1}(\xi))\max_k \sum_{\mathbf{l}\in I_{\sigma n}}|\xi(\frac{\mathbf{l}}{\sigma n}-\mathbf{r}_k)-\eta(\frac{\mathbf{l}}{\sigma n}-\mathbf{r}_k)| \tag{3.36}$$

where $\|\mathbf{c}\|_1 = \sum_{k=1}^N |c_k|$. Comparing (3.35) with (3.30) and comparing (3.36) with (3.31) yield the error estimation of $E_{\boldsymbol{\omega}}$ which is similar to E_{NFFT}:

$$\|E_\omega\|_\infty \le C(\xi,m,\sigma)\|\mathbf{c}\|_1.$$

Hence

$$|E_{\mathrm{NFFT^T}}| \le C(\xi,m,\sigma)\|\mathbf{c}\|_1\|g\|_1. \tag{3.37}$$

The inequality (3.37) is independent of *i*, therefore,

$$\|E_{\mathrm{NFFT^T}}\|_\infty \le C(\xi,m,\sigma)\|\mathbf{c}\|_1\|g\|_1. \tag{3.38}$$

4. Fast solvation force computation

The solvation force acting at the center of atom α, which is part of the forces driving dynamics is

$$\mathbf{F}_{\alpha}^{\text{elec}} = -\frac{\partial G_{\text{sol}}}{\partial \mathbf{x}_{\alpha}}. \tag{4.1}$$

Partition the solvation energy into polar and non-polar parts:

$$\frac{\partial G_{\text{sol}}}{\partial \mathbf{x}_{\alpha}} = \frac{\partial}{\partial \mathbf{x}_{\alpha}}(G_{\text{cav}} + G_{\text{vdw}}) + \frac{\partial G_{\text{pol}}}{\partial \mathbf{x}_{\alpha}} = \gamma \frac{\partial S_{A}}{\partial \mathbf{x}_{\alpha}} + \frac{\partial G_{\text{pol}}}{\partial \mathbf{x}_{\alpha}}. \tag{4.2}$$

The non-polar force is proportional to the derivatives of the volume and/or the surface area with respect to the atomic coordinates. There has been previous work on analytically computing the derivatives of the area/volume [31,32,33]. To compute the polar force, we first define

$$G_{ij} = q_i q_j / \left(r_{ij}^2 + R_i R_j \exp\left(-\frac{r_{ij}^2}{4R_i R_j}\right)\right)^{1/2}. \tag{4.3}$$

Then

$$G_{\text{pol}} = -\tau \sum_{i=1}^{M} \sum_{j=i+1}^{M} G_{ij} - \frac{\tau}{2} \sum_{i=1}^{M} G_{ii}. \tag{4.4}$$

Differentiating (4.4) w.r.t. **x**, one gets

$$\frac{\partial G_{\text{pol}}}{\partial \mathbf{x}_{\alpha}} = -\tau \sum_{i=1}^{M} \sum_{j=i+1}^{M} \frac{\partial G_{ij}}{\partial \mathbf{x}_{\alpha}} - \frac{\tau}{2} \sum_{i=1}^{M} \frac{\partial G_{ii}}{\partial \mathbf{x}_{\alpha}}, \tag{4.5}$$

where

$$\frac{\partial G_{ij}}{\partial \mathbf{x}_{\alpha}} = \frac{\partial G_{ij}}{\partial r_{ij}} \frac{\partial r_{ij}}{\mathbf{x}_{\alpha}} + \frac{\partial G_{ij}}{\partial R_i} \frac{\partial R_i}{\mathbf{x}_{\alpha}} + \frac{\partial G_{ij}}{\partial R_j} \frac{\partial R_j}{\mathbf{x}_{\alpha}}. \tag{4.6}$$

From (4.3), one can easily compute $\frac{\partial G_{ij}}{\partial r_{ij}}$ and $\frac{\partial G_{ij}}{\partial R_i}$, which are

$$\frac{\partial G_{ij}}{\partial r_{ij}} = q_i q_j \left(r_{ij}^2 + R_i R_j e^{-\frac{r_{ij}^2}{4R_i R_j}}\right)^{-\frac{3}{2}} \left(\frac{1}{4} e^{-\frac{r_{ij}^2}{4R_i R_j}} - 1\right) r_{ij},$$

$$\frac{\partial G_{ij}}{\partial R_i} = -\frac{q_i q_j}{8R_i} \left(r_{ij}^2 + R_i R_j e^{-\frac{r_{ij}^2}{4R_i R_j}}\right)^{-\frac{3}{2}} e^{-\frac{r_{ij}^2}{4R_i R_j}} (4R_i R_j + r_{ij}^2).$$

$\frac{\partial r_{ij}}{\partial \mathbf{x}_\alpha}$ is nonzero if i or $j = \alpha$ which will be $\frac{\partial r_{\alpha j}}{\partial \mathbf{x}_\alpha} = \frac{\mathbf{x}_\alpha - \mathbf{x}_j}{r_{\alpha j}}$. In (4.6) the computation of $\frac{\partial R_i}{\mathbf{x}_\alpha}$ for $i = 1, \ldots, M$ is not trivial. Because Γ depends on the position of the atoms, it is not easy to compute the derivative of R_i directly from (3.1). To solve this problem, we convert the integration domain back to the volume:

$$R_i^{-1} = \frac{1}{4\pi} \int_{\text{ex}} \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} d\mathbf{r}. \tag{4.7}$$

Then by defining a volumetric density function to distinguish the exterior from the interior of the molecule, we may have an integration domain that is independent of $\{\mathbf{x}_i\}$. One way of defining the volumetric function is given in [34] where they first define a density function for each of the atoms

$$\chi_i(\mathbf{r}) = \begin{cases} 1, & \left\| \mathbf{r} - \mathbf{x}_i \right\| \leq a_i \\ 0, & \left\| \mathbf{r} - \mathbf{x}_i \right\| > a_i \end{cases}$$

and then define the volumetric function by following the inclusion-exclusion principle

$$\rho(\mathbf{r}) = \sum_i \chi_i - \sum_{i<j} \chi_i \chi_j + \sum_{i<j<k} \chi_i \chi_j \chi_k - \sum_{i<j<k<l} \chi_i \chi_j \chi_k \chi_l + \ldots. \tag{4.8}$$

There are some nice properties of this model. For example, the exterior region of the molecule is well characterized by $\rho = 0$ and two atoms i and j are disconnected if for any $\mathbf{r} \in \mathbb{R}^3$, $\chi_i(\mathbf{r})$ $\chi_j(\mathbf{r}) = 0$. The drawback of this model is that function χ is not smooth, which makes it inapplicable to the derivative computation. Therefore we smoothen χ by introducing a cubic spline near the atom boundary:

$$\rho_i(x) = \begin{cases} 1, & x \leq a_i \\ \frac{2}{w^3}(x - a_i)^3 - \frac{3}{w^2}(x - a_i)^2 + 1, & a_i < x < a_i + w \\ 0, & x \geq a_i + w \end{cases} \tag{4.9}$$

with $x = \| \mathbf{r} - \mathbf{x}_i \|$. The region defined by $\rho_i \neq 0$ is regarded as the interior of atom i and this region converges to the van der Waals volume of the atom as w goes to 0. In the SES model, two atoms are considered to be completely separated if the distance between the centers is greater than the sum of the radii plus the probe diameter. Otherwise they can be connected by the reentrant surface of the rolling probe. By setting $w = 1.4$ Å, atoms i and j are disconnected in the same sense as in the SES model iff $\rho_i(\mathbf{r})\rho_j(\mathbf{r}) = 0$, for any $\mathbf{r} \in \mathbb{R}^3$. In addition to this modification, we neglect the cases that more than four atoms overlap simultaneously. Therefore the molecular volumetric density function becomes

$$\rho(\mathbf{r}) = \sum_i \rho_i - \sum_{i<j} \rho_i \rho_j + \sum_{i<j<k} \rho_i \rho_j \rho_k - \sum_{i<j<k<l} \rho_i \rho_j \rho_k \rho_l. \tag{4.10}$$

We define the complementary function $\bar{\rho} = 1 - \rho$. It is easy to show that within the VWS of the molecule, $\bar{\rho}$ is always 0, beyond the SAS, $\bar{\rho}$ is always 1, in between, $0 < \bar{\rho} < 1$. Then (4.7) can be rewritten as

$$R_i^{-1}=\frac{1}{4\pi}\int_{\mathbb{R}^3}\frac{\overline{\rho}(\mathbf{r},\{\mathbf{x}_j\})}{|\mathbf{r}-\mathbf{x}_i|^4}\mathrm{d}\mathbf{r}. \tag{4.11}$$

Differentiating both sides of (4.11), one gets

$$-\frac{1}{R_i^2}\frac{\partial R_i}{\partial \mathbf{x}_\alpha}=\frac{1}{4\pi}\int_{\mathbb{R}^3}\frac{\partial}{\partial \mathbf{x}_\alpha}\left(\frac{\overline{\rho}(\mathbf{r},\{\mathbf{x}_j\})}{|\mathbf{r}-\mathbf{x}_i|^4}\right)\mathrm{d}\mathbf{r}. \tag{4.12}$$

So

$$\frac{\partial R_i}{\partial \mathbf{x}_\alpha}=-\frac{R_i^2}{4\pi}\left(\int_{\mathbb{R}^3}\frac{\frac{\partial}{\partial \mathbf{x}_\alpha}\overline{\rho}(\mathbf{r},\{\mathbf{x}_j\})}{|\mathbf{r}-\mathbf{x}_i|^4}\mathrm{d}\mathbf{r}+\int_{\mathrm{ex}}\frac{\partial}{\partial \mathbf{x}_\alpha}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\mathrm{d}\mathbf{r}\right). \tag{4.13}$$

For the first integral in (4.13),

$$\frac{\partial}{\partial \mathbf{x}_\alpha}\overline{\rho}=-\frac{\partial}{\partial \mathbf{x}_\alpha}\rho=-\frac{\partial \rho_\alpha}{\partial \mathbf{x}_\alpha}\left(1-\sum_j\rho_j+\sum_{j<k}\rho_j\rho_k-\sum_{j<k<l}\rho_j\rho_k\rho_l\right)=-\frac{\partial \rho_\alpha}{\partial \mathbf{x}_\alpha}g_\alpha,$$

where j, k, l are the atoms overlapping with atom α, $g = 1 - \Sigma_j \rho_j + \Sigma_{j<k} \rho_j\rho_k - \Sigma_{j<k<l} \rho_j\rho_k\rho_l$, and

$$\frac{\partial \rho_i}{\partial \mathbf{x}_\alpha}(\mathbf{r})=\begin{cases}0, & x\le a_\alpha\\ \left(\frac{6}{w^3}(x-a_\alpha)^2-\frac{6}{w^2}(x-a_\alpha)\right)\frac{\mathbf{x}_\alpha-\mathbf{r}}{x}, & a_\alpha<x<a_\alpha+w\\ 0, & x\ge a_\alpha+w\end{cases}$$

with $x = ||\mathbf{r} - \mathbf{x}_\alpha||$. Noticing that $\frac{\partial \rho_\alpha}{\partial \mathbf{x}_\alpha}\neq 0$ only if $a_\alpha < |\mathbf{r} - \mathbf{x}_\alpha| < a_\alpha + w$, the first integral in (4.13) is simplified as

$$\int_{|\mathbf{r}-\mathbf{x}_\alpha|=a_\alpha}^{|\mathbf{r}-\mathbf{x}_\alpha|=a_\alpha+w}-\frac{\partial \rho_\alpha}{\partial \mathbf{x}_\alpha}g_\alpha(\mathbf{r})\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\mathrm{d}\mathbf{r}. \tag{4.14}$$

The integration domain of (4.14) is a regular spherical shell of the width w around atom α (Figure 4.1(a)). We switch to the spherical coordinate system:

$$\begin{cases}x=x_\alpha+(a_\alpha+r)\cos\theta\sin\varphi\\ y=y_\alpha+(a_\alpha+r)\sin\theta\sin\varphi\\ z=z_\alpha+(a_\alpha+r)\cos\varphi\end{cases}$$

where $(r, \theta, \varphi) \in [0, w] \times [0, 2\pi] \times [0, \pi]$. We sample r, θ, φ by using the 2-point Gaussian quadrature nodes in each dimension. For all the atoms in the molecule, they share the same set of sampling points (r, θ, φ).

The second integral in (4.13) is nonzero if $i \equiv \alpha$. In that case

$$\int_{\text{ex}} \frac{\partial}{\partial \mathbf{x}_i} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^4} d\mathbf{r} = -\int_{\text{ex}} \frac{\partial}{\partial \mathbf{r}} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^4} d\mathbf{r}. \tag{4.15}$$

We compute each component of (4.15) individually and convert to the surface integration (Figure: 4.1(b)) by the divergence theorem:

$$-\int_{\text{ex}} \frac{\partial}{\partial x} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^4} d\mathbf{r} = \int_{\Gamma} \frac{n_x(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4} dS \simeq \sum_{k=1}^{N} \frac{w_k n_x^k}{|\mathbf{r}_k-\mathbf{x}_i|^4}, \tag{4.16}$$

$$-\int_{\text{ex}} \frac{\partial}{\partial y} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^4} d\mathbf{r} = \int_{\Gamma} \frac{n_y(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4} dS \simeq \sum_{k=1}^{N} \frac{w_k n_y^k}{|\mathbf{r}_k-\mathbf{x}_i|^4}, \tag{4.17}$$

$$-\int_{\text{ex}} \frac{\partial}{\partial z} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^4} d\mathbf{r} = \int_{\Gamma} \frac{n_z(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4} dS \simeq \sum_{k=1}^{N} \frac{w_k n_z^k}{|\mathbf{r}_k-\mathbf{x}_i|^4}, \tag{4.18}$$

where the quadrature weights and points (w_k, $\mathbf{r}_k$) and the unit normals (n_x^k, n_y^k, n_z^k) are the same as those used in Section 3. We compute (4.16), (4.17), and (4.18) by directly applying the fast summation method with the coefficients $c_k = w_k n_x^k, w_k n_y^k, w_k n_z^k$, respectively. Since the same algorithm is used in the Born radius derivative calculation, the error analysis is similar to the error analysis of the Born radius calculation except that a quadrature error of the integration over the shell region needs to be added.

To compute the force acting on each of the M atoms, we need to compute (4.15) for $i = 1, \ldots, M$. By using the fast summation algorithm, the computational complexity of this part is $O(N + M + n^3 \log n)$, the same as the energy computation. To compute (4.14), since the shell integration domain is narrow, only a small number of atoms have non-zero densities in this region, therefore the complexity of computing (4.14) for a fixed α for $i = 1, \ldots, M$ is $O(M)$. Moreover, since the integrand in (4.14) is very small if atom i and atom α are far apart, we use a cut-off distance d_0 in our computation and compute (4.14) only if $d(i, \alpha) \leq d_0$. Therefore the overall time complexity of computing (4.13) is $O(N + M + n^3 \log n)$.

5. Results

We compare the polarization energy computed based on the fast summation algorithm and the trivial summation in Table 5.1 for four proteins (PDB ID: 1CGI_l, 1BGX, 1DE4, 1N2C). An ASMS model is constructed for each protein with N_e number of patches. A three-point Gaussian quadrature is used on each algebraic patch. We also compare the overall computation time of the two methods. As we see from the table, for the small proteins (e.g. 1CGI_l), the fast summation method is slower than the trivial summation. However as the protein size gets larger (e.g. 1BGX, 1DE4, 1N2C), the fast summation is apparently faster than the trivial summation without losing too much accuracy. The relative error ε between the fast summation and the trivial summation is small. As for the trade-off between efficiency and accuracy, since in the

current research of the MD simulation efficiency is more concerned, the fast-summation-based GB is superior to the trivial GB method.

In Figure 5.1 we compare G_{pol} computed by the fast summation based GB and the trivial summation method along with their computation time for proteins of various sizes. For all these proteins, we generate the ASMS of the same number of patches (in our test we use 20,000 patches for each protein). We choose the fixed parameters $n = 30$, $m = 4$, and $\sigma = 2$ for all the proteins. We observe that the G_{pol} computed by the fastsum GB is close to that computed by the trivial GB methods and the error gets larger as the molecule gets bigger. Even though the error analysis in Section 3.3 does not show that the error depends on the size of the molecule, the analysis is based on the assumption that the kernel function is defined on the domain $[-\frac{1}{2}, \frac{1}{2}]^3$. To ensure that $\mathbf{x}_i - \mathbf{r}_k$, $i = 1, \ldots, M$, $k = 1, \ldots, N$ are all within this range, we scale the molecule. The larger the molecule, the larger the scaling factor. Later on when we scale back to the original coordinates by multiplying the scaling factor, the error gets amplified. As we expect, computation time of the fastsum GB increases as M becomes large but is much faster than the traditional GB method.

In Figure 5.2, we compare G_{pol} computed by the fast summation based GB versus the trivial summation method and the computation time for a test protein 1JPS where we generate the ASMS with different numbers of patches. We use the same values for the parameters n, m, and σ as in the previous test. As shown in the figure, as the triangular mesh becomes denser, the fast summation result converges rapidly to the result of the trivial method but takes less computation time.

For the test proteins 1ANA, 1MAG, 1PPE_l, 1CGI_l, we compute the solvation force $\mathbf{F}_{\alpha}^{\text{elec}}$, for $\alpha = 1, \ldots, M$. We show the timing results in Table 5.2. In general, if an atom has a strong solvation force, this atom is in favor of being polarized, and hence is an active atom. On the contrary, if an atom has a weak solvation force, it is more likely to be an inactive atom. For every test protein, after we compute the solvation force for each atom, we sort the forces based on their magnitude and choose the top most active atoms and the top inactive atoms. As shown in Figure 5.3, the top 5% of the most active atoms are rendered in red and the bottom 5% of the atoms are rendered in blue. This provides a convenient and cheaper way, alternative to the experimental method, to help the biologists quickly find an active site of a protein.

6. Conclusion

We introduce a fast summation based algorithm to calculate the effective Born radii and their derivatives in the generalized Born model of implicit solvation. The algorithm relies on a variation of the formulation for the Born radii and an additional analytical volumetric density function for the derivatives. For a system of M atoms and N sampling points on the molecular surface, the trivial way of computing the Born radii requires $O(MN)$ arithmetic operations, whereas with the aids of the Fourier expansion of the kernel functions of the Born radii (and their derivatives) and the NFFT algorithm which essentially approximates the complex exponentials in the NDFT by the DFT of a fast decaying smooth window function, the Born radii as well as their derivatives can be obtained at cost of $(M + N + n^3 \log n)$ where n is the number of frequencies in the Fourier expansion. We show that the error of the algorithm decreases as the mesh gets denser, or as any of the parameters σ, m, n increase. Other than the Born model developed with a Coulomb field approximation, there has been other models for the Born radii evaluation, for example the Kirkwood-Grycuk model [35] where $R_i^{-1} = \left(\frac{3}{4\pi}\int_{\text{ex}} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^6} d\mathbf{r}\right)^{1/3}$. This model is recently applied to the GBr^{6}NL model which approximates the solvation energy of the nonlinear Poisson-Boltzmann equation [36]. It is interesting to note that we can utilize a similar quadrature point generation via ASMS and the

fast summation algorithm to speed up this GBr^{6}NL computation. In fact, by the divergence theorem, $\int_{\text{ex}} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^6} d\mathbf{r} = \frac{1}{3}\int_{\Gamma} \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^6} d\mathbf{r}$ and the rest follows similar to the methods in this paper.

Acknowledgments

This research was supported in part by NSF grant CNS-0540033 and NIH contracts R01-EB00487, R01-GM074258, R01-GM07308. We thank the reviewers, as well as Dr. Rezaul Chowdhury for all the excellent suggestions that have resulted in a considerably improved paper. We also wish to thank several members of our CVC group for developing and maintaining TexMol, our molecular modeling and visualization software tool, which was used in conjunction with our nFFTGB implementation, to produce all the pictures in our paper (http://cvcweb.ices.utexas.edu/software/).

References

1. Phillips J, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel R, Kale L, Schulten K. Scalable molecular dynamics with NAMD. J Comput Chem 2005;26:1781–1802. [PubMed: 16222654]
2. Roux B, Simonson T. Implicit solvent models. Biophys Chem 1999;78:1–20. [PubMed: 17030302]
3. Weeks J, Chandler D, Andersen H. Role of repulsive forces in determining the equilibrium structure of simple liquids. J Chemical Physics 1971;54:5237–5247.
4. Chandler D, Weeks J, Andersen H. Van der Waals picture of liquids, solids, and phase transformations. Science 1983;220:787–794. [PubMed: 17834156]
5. Eisenberg D, Mclachlan AD. Solvation energy in protein folding and binding. Nature (London) 1986;319:199–203. [PubMed: 3945310]
6. Wagoner JA, Baker NA. Assessing implicit models for nonpolar mean solvation forces: The importance of dispersion and volume terms. 2006;103:8331–8336.
7. Sharp K. Incorporating solvent and ion screening into molecular dynamics using the finite-difference Poisson-Boltzmann method. J Comput Chem 1991;12:454–468.
8. Kollman PA, Massova I, Reyes C, Kuhn B, Huo S, Chong L, Lee M, Lee T, Duan Y, Wang W, Donini O, Cieplak P, Srinivasan J, Case DA, Cheatham TE. Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models. Acc Chem Res 2000;33:889–897. [PubMed: 11123888]
9. Holst M, Baker N, Wang F. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples. J Comput Chem 2000;21:1319–1342.
10. Baker N, Holst M, Wang F. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems. J Comput Chem 2000;21:1343–1352.
11. Lu B, Zhang D, McCammon JA. Computation of electrostatic forces between solvated molecules determined by the Poisson-Boltzmann equation using a boundary element method. J Chemical Physics 2005;122:214102–214109.
12. Still WC, Tempczyk A, Hawley RC, Hendrickson T. Semianalytical treatment of solvation for molecular mechanics and dynamics. J Am Chem Soc 1990;112:6127–6129.
13. MacKerel, AD., Jr; Brooks, CL., III; Nilsson, L.; Roux, B.; Won, Y.; Karplus, M. *CHARMM: The Energy Function and Its Parameterization with an Overview of the Program*, volume 1 of *The Encyclopedia of Computational Chemistry*. John Wiley & Sons; Chichester: 1998. p. 271-277.
14. Case D, Cheatham T III, Darden T, Gohlke H, Luo R, Merz K Jr, Onufriev A, Simmerling C, Wang B, Woods R. The Amber biomolecular simulation programs. J Comput Chem 2005;26:1668–1688. [PubMed: 16200636]
15. Ren P, Ponder JW. Polarizable atomic multipole water model for molecular mechanics simulation. J Phys Chem B 2003;107:5933–5947.
16. Tsui V, Case DA. Theory and applications of the generalized Born solvation model in macromolecular simulations. Biopolymers 2001;56:275–291. [PubMed: 11754341]
17. Shih, Amy Y.; Denisov, Ilia G.; Phillips, James C.; Sligar, Stephen G.; Schulten, Klaus. Molecular dynamics simulations of discoidal bilayers assembled from truncated human lipoproteins. Biophys J 2005;88:548–556. [PubMed: 15533924]

18. Ritchie, David W. Evaluation of protein docking predictions using hex 3.1 in capri rounds 1 and 2. Proteins: Structure, Function, and Genetics July;2003 52(1):98–106.
19. Ju T, Losasso F, Schaefer S, Warren J. Dual contouring of hermite data. Proceedings of ACM SIGGRAPH 2002:339–346.
20. Zhang Y, Xu G, Bajaj C. Quality meshing of implicit solvation models of biomolecular structures. Computer Aided Geometric Design.
21. Garland M, Heckbert P. Simplifying surfaces with color and texture using quadric error metrics. IEEE Visualization 1998:263–270.
22. Zhao W, Xu G, Bajaj C. An algebraic spline model of molecular surfaces. ACM Symp Sol Phys Model 2007:297–302.
23. Greengard L, Rokhlin V. A fast algorithm for particle simulations. J Chemical Physics 1987;73:325–348.
24. Feig M, Onufriev A, Lee MS, Im W, Case DA, Brooks C III. Performance comparison of generalized Born and Poisson methods in the calculation of electrostatic solvation energies for protein structures. J Comput Chem 2004;25:265–284. [PubMed: 14648625]
25. Ghosh A, Rapp CS, Friesner RA. Generalized Born model based on a surface integral formulation. J Phys Chem B 1998;102:10983–10990.
26. Potts D, Steidl G. Fast summation at nonequispaced knots by NFFTs. SIAM J Sci Comput 2003;24:2013–2037.
27. Dunavant D. High degree efficient symmetrical Gaussian quadrature rules for the triangle. International Journal of Numerical Methods in Engineering 1985;21:1129–1148.
28. Potts, D.; Steidl, G.; Tasche, M. Modern Samplling Theory: mathematics and Applications. Birkhauser; 2001. Fast Fourier transforms for nonequispaced data: A tutorial; p. 247-270.
29. Beylkin G. On the fast Fourier transform of functions with singularities. Appl Comput Harmon Anal 1995;2:363C–381.
30. Jackson JI. Selection of a convolution function for Fourier inversion using gridding. IEEE Trans Med Imag 1991;10:473–C478.
31. Edelsbrunner H, Koehl P. The weighted-volume derivative of a space-filling diagram 2003;100:2203–2208.
32. Im W, Lee MS, Brooks C III. Generalized Born model with a simple smoothing function. J Comput Chem 2003;24:1691–1702. [PubMed: 12964188]
33. Bryant R, Edelsbrunner H, Koehl P, Levitt M. The area derivative of a space-filling diagram. Discrete Comput Geom 2004;32:293–308.
34. Grant JA, Pickup BT. A Gaussian description of molecular shape. J Phys Chem 1995;99:3503–3510.
35. Grycuk T. Deficiency of the Coulomb-field approximation in the generalized Born model: An improved formula for Born radii evaluation. J Chemical Physics 2003;119:4817–4827.
36. Tjong H, Zhou H. GBr^{6}NL: A generalized Born method for accurately reproducing solvation energy of the nonlinear Poisson-Boltzmann equation. J Chemical Physics 2007;126:195102–195106.

Appendix A. NFFT

The NFFT [28] is an algorithm for fast computation of multivariate discrete Fourier transforms for nonequispaced data in spacial domain (NDFT^1). The NDFT^1 problem is to evaluate the trigonometric polynomials

$$G(\mathbf{x}_j)=\sum_{\omega\in I_n} G_\omega e^{2\pi i\omega\cdot\mathbf{x}_j} \qquad j=1,\ldots,M, \tag{A.1}$$

where $I_n=\{(\omega_1,\omega_2,\omega_3)\in\mathbb{Z}^3: -\frac{n}{2}\le\omega_i\le\frac{n}{2}\}$. Without loss of generality, we assume $\mathbf{x}_j\in[-\frac{1}{2},\frac{1}{2}]^3$. Instead of computing the summations in (A.1) directly, one can approximate G by a function $s(\mathbf{x})$ which is a linear combination of the shifted 1-periodic kernel function ξ:

$$s(\mathbf{x}) := \sum_{\mathbf{l} \in I_{\sigma n}} g_{\mathbf{l}} \xi(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}), \tag{A.2}$$

where $I_{\sigma n} := \{(l_1, l_2, l_3) : l_i \in [-\frac{\sigma n}{2}, \frac{\sigma n}{2}] \cap \mathbb{Z}, \sigma > 1\}$ and $\frac{1}{\sigma n} := \{(\frac{l_1}{\sigma n}, \frac{l_2}{\sigma n}, \frac{l_3}{\sigma n})\}$. We have $\sigma > 1$ because of the error estimation discussed in Section 3.3.3.

The kernel function ξ is defined as

$$\xi(\mathbf{x}) := \sum_{\mathbf{i} \in \mathbb{Z}^3} \xi_0(\mathbf{x}+\mathbf{i}), \quad \text{where } \xi_0 \in L_2(\mathbb{R}^3).$$

Good candidates for ξ_0 include Gaussian, B-spline, sinc, and Kaiser-Bessel functions. Expand the periodic kernel function ξ by its Fourier series

$$\xi(\mathbf{x}) = \sum_{\omega \in \mathbb{Z}^3} C_\omega(\xi) e^{2\pi i \omega \cdot \mathbf{x}}, \tag{A.3}$$

with the Fourier coefficients

$$C_\omega(\xi) := \int_{[-\frac{1}{2}, \frac{1}{2}]^3} \xi(\mathbf{x}) e^{-2\pi i \omega \cdot \mathbf{x}} d\mathbf{x} = \int_{\mathbb{R}^3} \xi_0(\mathbf{x}) e^{-2\pi i \omega \cdot \mathbf{x}} d\mathbf{x} = \widehat{\xi_0}(\boldsymbol{\omega}).$$

Cut off the higher frequencies in (A.3), one can get

$$\xi(\mathbf{x}) = \left(\sum_{\omega \in I_{\sigma n}} + \sum_{\omega \in \mathbb{Z}^3 \setminus I_{\sigma n}} \right) C_\omega(\xi) e^{2\pi i \omega \cdot \mathbf{x}} \approx \sum_{\omega \in I_{\sigma n}} C_\omega(\xi) e^{2\pi i \omega \cdot \mathbf{x}}. \tag{A.4}$$

Plug (A.4) into (A.2), we get

$$\begin{aligned} s(\mathbf{x}_j) &\approx \sum_{\mathbf{l} \in I_{\sigma n}} g_{\mathbf{l}} \sum_{\omega \in I_{\sigma n}} C_\omega(\xi) e^{2\pi i \omega \cdot (\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n})} \\ &= \sum_{\omega \in I_{\sigma n}} \tilde{G}_\omega C_\omega(\xi) e^{2\pi i \omega \cdot \mathbf{x}_j}, \end{aligned} \tag{A.5}$$

with the coefficients

$$\tilde{G}_\omega := \sum_{\mathbf{l} \in I_{\sigma n}} g_{\mathbf{l}} e^{-2\pi i \omega \cdot \frac{\mathbf{l}}{\sigma n}}. \tag{A.6}$$

By defining

$$\tilde{G}_\omega := \begin{cases} \frac{G_\omega}{C_\omega(\xi)} & \text{for } \boldsymbol{\omega} \in I_n, \\ 0 & \text{for } \boldsymbol{\omega} \in I_{\sigma n} \backslash I_n, \end{cases} \tag{A.7}$$

one can immediately get

$$s(\mathbf{x}_j) \approx \sum_{\omega \in I_n} G_\omega e^{2\pi i \omega \cdot \mathbf{x}_j} = G(\mathbf{x}_j). \tag{A.8}$$

The next problem is to compute $g_{\mathbf{l}}$. From (A.6), one can compute the coefficients $g_{\mathbf{l}}$ which are also coefficients in (A.8) by the discrete Fourier transform

$$g_{\mathbf{l}} = \frac{1}{\sigma^3 n^3} \sum_{\omega \in I_{\sigma n}} \tilde{G}_\omega e^{2\pi i \omega \cdot \frac{\mathbf{l}}{\sigma n}} = \frac{1}{\sigma^3 n^3} \sum_{\omega \in I_n} \frac{G_\omega}{C_\omega(\xi)} e^{2\pi i \omega \cdot \frac{\mathbf{l}}{\sigma n}}, \ \mathbf{l} \in I_{\sigma n}, \tag{A.9}$$

with complexity $O(n^3 \log n)$ by the FFT algorithm.

Since the function ξ drops very fast, one can further reduce the computation complexity of (A. 8) by cutting off the tail of ξ. Define a function η_0:

$$\eta_0 := \xi_0(\mathbf{x}) \chi_{[-\frac{m}{\sigma n}, \frac{m}{\sigma n}]^3}(\mathbf{x}) \quad \text{where } m \ll \sigma n, m \in \mathbb{N}.$$

Construct the one-periodic function η the same way as ξ is constructed:

$$\eta(\mathbf{x}) = \sum_{\mathbf{i} \in \mathbb{Z}^3} \eta_0(\mathbf{x} + \mathbf{i}).$$

Replacing ξ with η in (A.8), we obtain that

$$G(\mathbf{x}_j) \approx \sum_{\mathbf{l} \in I_{\sigma n, m}(\mathbf{x}_j)} g_{\mathbf{l}} \eta(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}), \tag{A.10}$$

where $I_{\sigma n, m}(\mathbf{x}_j) = \{(l_1, l_2, l_3): \sigma n \mathbf{x}_{j,i} - m \le l_i \le \sigma n \mathbf{x}_{j,i} + m, i = 1, 2, 3\}$. There are at most $(2m + 1)^3$ nonzero terms in (A.10). Therefore the complexity of evaluating (A.10) for $j = 1, \ldots, M$ is $O(m^3 M)$. Adding the complexity of computing the coefficients $g_{\mathbf{l}}$, the overall complexity of NFFT algorithm is $O(n^3 \log n + m^3 M)$.

Remark

If we reorganize the above equations, it is not hard to see that, in fact, (A.1) is approximately computed by the expression

$$G(\mathbf{x}_j)=\sum_{\omega\in I_n} G_\omega \left(\frac{1}{(\sigma n)^3 C_\omega(\xi)} \sum_{\mathbf{l}\in I_{\sigma n}} \eta(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}) e^{2\pi \mathrm{i}\omega\cdot\frac{\mathbf{l}}{\sigma n}} \right). \tag{A.11}$$

From a linear algebra point of view, equation (A.11) can be written as the product of a matrix and a vector. For example, for a one dimensional NFFT, (A.11) is equivalent to

$$\mathbf{g}=\Xi F D \widehat{\mathbf{g}} \tag{A.12}$$

with vectors

$$\mathbf{g}:=[G(x_i)]_{i=1}^{M}, \quad \widehat{\mathbf{g}}:=[G_\omega]_{\omega=-\frac{n}{2}}^{\frac{n}{2}-1}.$$

Ξ is a sparse matrix

$$\Xi:=\left[\eta(x_i - \frac{l_j}{\sigma n})\right]_{M\times\sigma n},$$

F is the classical Fourier matrix

$$F:=\left[e^{2\pi \mathrm{i}\omega_j \frac{l_j}{\sigma n}}\right]_{\sigma n\times n},$$

and D is an $n \times n$ diagonal matrix with the iith element being $\frac{1}{\sigma n C_{\omega_i}(\xi)}$. For a multi-dimensional NFFT, it is the same as the 1D case as long as one orders the indices of the multi-dimension into one dimension.

As discussed in [28], in the first approximation (A.8), we see that s is equal to G after its higher frequencies in the Fourier series are cut off. Hence the error introduced in (A.8) which is known as the *aliasing error* is

$$\begin{aligned} E^1_{\mathrm{NFFT}} &:= \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}} \xi(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}) - G(\mathbf{x}_j) \\ &= \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \sum_{\omega\in I_{\sigma n}} \tilde{G}_{\omega+\mathbf{i}\sigma n} C_{\omega+\mathbf{i}\sigma n}(\xi) e^{2\pi \mathrm{i}(\omega+\mathbf{i}\sigma n)\cdot\mathbf{x}_j}. \end{aligned} \tag{A.13}$$

Note that from (A.6), we have the condition $\tilde{G}_{\boldsymbol{\omega}+\mathbf{i}\sigma n} = \tilde{G}_{\boldsymbol{\omega}}$, for $\mathbf{i} \in \mathbb{Z}^3$ and $\boldsymbol{\omega} \in I_{\sigma n}$. By the definition (A.7), one obtains

$$|E^1_{\mathrm{NFFT}}| \le \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \sum_{\omega\in I_n} |G_\omega \frac{C_{\omega+\mathbf{i}\sigma n}(\xi)}{C_\omega(\xi)}|. \tag{A.14}$$

Let $\left\|\widehat{G}\right\|_1 = \sum_{\omega\in I_n} |G_\omega|$. Then

$$|E^1_{\text{NFFT}}| \le \left\|\widehat{G}\right\|_1 \max_{\omega\in I_n} \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} |\frac{C_{\omega+\mathbf{i}\sigma n}(\xi)}{C_\omega(\xi)}|. \tag{A.15}$$

In the second approximation (A.10), since ξ is replaced by η, the so caused error, known as the *truncation error*, is

$$\begin{aligned} E^2_{\text{NFFT}} &:= \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}}\xi(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n}) - \sum_{\mathbf{l}\in I_{\sigma n,m}} g_{\mathbf{l}}\eta(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n}) \\ &= \sum_{\mathbf{l}\in I_{\sigma n}\setminus I_{\sigma n,m}} g_{\mathbf{l}}[\,\xi(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n})] \\ = \sum_{\mathbf{l}\in I_{\sigma n}\setminus I_{\sigma n,m}} \tfrac{1}{\sigma^3 n^3} \sum_{\omega\in I_n} \tfrac{G_\omega}{C_\omega(\xi)} e^{2\pi i\omega\cdot\frac{\mathbf{l}}{\sigma n}} &[\,\xi(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n})]. \end{aligned} \tag{A.16}$$

Thus

$$\begin{aligned} |E^2_{\text{NFFT}}| &\le \tfrac{1}{\sigma^3 n^3} \sum_{\mathbf{l}\in I_{\sigma n}} |\tfrac{G_\omega}{C_\omega(\xi)}[\,\xi(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n})]| \\ &\le \tfrac{1}{\sigma^3 n^3} \max_{\omega\in I_n}(C^{-1}_\omega(\xi)) \left\|\widehat{G}\right\|_1 \sum_{\mathbf{l}\in I_{\sigma n}} |\xi(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}-\tfrac{\mathbf{l}}{\sigma n})|. \end{aligned} \tag{A.17}$$

Appendix B. NFFTT

The NFFT$^{\mathrm{T}}$ algorithm deals with the fast computation of multivariate discrete Fourier transforms for nonequispaced data in frequency domain (NDFT^2):

$$a(\boldsymbol{\omega}) = \sum_{k=1}^{N} c_k e^{-2\pi\mathrm{i}\boldsymbol{\omega}\cdot\mathbf{r}_k}, \quad \boldsymbol{\omega}\in I_n. \tag{B.1}$$

Define a function

$$A(\mathbf{x}) := \sum_{k=1}^{N} c_k \xi(\mathbf{x}-\mathbf{r}_k), \tag{B.2}$$

where ξ is defined as same as in Appendix A. The Fourier series of $A(\mathbf{x})$ is:

$$A(\mathbf{x}) = \sum_{\boldsymbol{\omega}\in\mathbb{Z}^3} C_\omega(A) e^{2\pi\mathrm{i}\boldsymbol{\omega}\cdot\mathbf{x}}. \tag{B.3}$$

On the other hand,

$$\sum_{k=1}^{N} c_k \xi(\mathbf{x}-\mathbf{r}_k)=\sum_{k=1}^{N} c_k \sum_{\omega\in\mathbb{Z}^3} C_\omega(\xi) e^{2\pi i\omega\cdot(\mathbf{x}-\mathbf{r}_k)}. \tag{B.4}$$

Hence we get the relationship of Fourier coefficients of A and ξ:

$$C_\omega(A)=\sum_{k=1}^{N} c_k e^{-2\pi i\omega\cdot\mathbf{r}_k} C_\omega(\xi), \quad \omega\in\mathbb{Z}^3. \tag{B.5}$$

Comparing (B.5) with (B.1) one obtains

$$a(\omega)=\frac{C_\omega(A)}{C_\omega(\xi)}, \quad \omega\in I_n. \tag{B.6}$$

It remains to compute $C_{\boldsymbol{\omega}}(A)$. By definition,

$$\begin{aligned} C_\omega(A)&=\int_{[-\frac{1}{2},\frac{1}{2}]^3} A(\mathbf{x}) e^{-2\pi i\omega\cdot\mathbf{x}} d\mathbf{x} \\ &=\int_{[-\frac{1}{2},\frac{1}{2}]^3} \left(\sum_{k=1}^{N} c_k \xi(\mathbf{x}-\mathbf{r}_k)\right) e^{-2\pi i\omega\cdot\mathbf{x}} d\mathbf{x} \\ &=\sum_{k=1}^{N} c_k \int_{[-\frac{1}{2},\frac{1}{2}]^3} \xi(\mathbf{x}-\mathbf{r}_k) e^{-2\pi i\omega\cdot\mathbf{x}} d\mathbf{x}. \end{aligned} \tag{B.7}$$

Discretizing the integration in (B.7) by the left rectangular rule leads to

$$a(\omega)\approx \frac{1}{C_\omega(\xi)} \sum_{k=1}^{N} c_k \frac{1}{(\sigma n)^3} \sum_{\mathbf{l}\in I_{\sigma n}} \xi\left(\frac{\mathbf{l}}{\sigma n}-\mathbf{r}_k\right) e^{-2\pi i\omega\cdot\frac{\mathbf{l}}{\sigma n}}. \tag{B.8}$$

Replacing ξ with η yields

$$a(\omega)\approx \frac{1}{(\sigma n)^3} \frac{1}{C_\omega(\xi)} \sum_{\mathbf{l}\in I_{\sigma n}} \widehat{g}_{\mathbf{l}} e^{-2\pi i\omega\cdot\frac{\mathbf{l}}{\sigma n}}, \tag{B.9}$$

where

$$\widehat{g}_{\mathbf{l}}:=\sum_{k=1}^{N} c_k \eta\left(\frac{\mathbf{l}}{\sigma n}-\mathbf{r}_k\right), \quad \mathbf{l}\in I_{\sigma n}. \tag{B.10}$$

To compute $\hat{g}_{\mathbf{l}}$, if one scans the $\mathbf{r}_k$ list, then for each $\mathbf{r}_k$ there are at most $(2m+1)^3$ grid points ($\mathbf{l}$) that contribute nonzero η. Hence, the complexity of computing $\hat{g}_{\mathbf{l}}$ is $O(m^3N)$. After computing $\hat{g}_{\mathbf{l}}$ one can easily evaluate (B.9) by the FFT algorithm at the complexity of $O(n^3 \log$

n). Lastly the complexity of computing (B.6) is $O(n^3)$. So the overall complexity of the NFFT$^{\mathrm{T}}$ algorithm is $O(m^3N + n^3 \log n)$.

Remark

Similar to the NFFT algorithm, we may write the one-line formula for computing (B.1) by the NFFT$^{\mathrm{T}}$:

$$a(\boldsymbol{\omega}) = \sum_{k=1}^{N} c_k \left(\frac{1}{(\sigma n)^3 C_\omega(\xi)} \sum_{\mathbf{l} \in I_{\sigma n}} \eta\left(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k\right) e^{-2\pi \mathrm{i} \omega \cdot \frac{\mathbf{l}}{\sigma n}} \right), \tag{B.11}$$

which in one dimension is equivalent to the linear system:

$$\widehat{\mathbf{a}} = D^{\mathrm{T}} F^{*} \Xi^{\mathrm{T}} \mathbf{c} \tag{B.12}$$

with vectors

$$\widehat{\mathbf{a}} := [a(\omega)]_{\omega=-\frac{n}{2}}^{\frac{n}{2}}, \quad \mathbf{c} := [c_k]_{k=1}^{N}.$$

Matrix Ξ is similar to that defined in Appendix A

$$\Xi := \left[\eta\left(\frac{l_j}{\sigma n} - r_i\right) \right]_{N \times \sigma n}.$$

F^{*} is the conjugate transpose of the Fourier matrix F, and D is the same as that defined in Appendix A. From the matrix expression, we see why the algorithm is called the “transpose” of NFFT.

Let $E_{\boldsymbol{\omega}}$ designate the error of $a(\boldsymbol{\omega})$. $E_{\boldsymbol{\omega}}$ can also be split into the *aliasing error* E_ω^1 introduced in (B.8) and the *truncation error* E_ω^2 introduced in (B.9), $E_\omega = E_\omega^1 + E_\omega^2$, for $\boldsymbol{\omega} \in I_{\sigma n}$. By taking the Fourier expansion of ξ, we get form (B.8), so

$$\begin{aligned} E_\omega^1 = a(\boldsymbol{\omega}) &= \frac{1}{C_\omega(\xi)} \sum_{k=1}^{N} c_k \frac{1}{(\sigma n)^3} \sum_{\mathbf{l} \in I_{\sigma n}} \left(\sum_{\mathbf{j} \in \mathbb{Z}^3} C_{\mathbf{j}}(\xi) e^{2\pi \mathrm{i} \left(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k\right) \cdot \mathbf{j}} \right) e^{-2\pi \mathrm{i} \omega \cdot \frac{\mathbf{l}}{\sigma n}} \\ &= a(\boldsymbol{\omega}) - \frac{1}{C_\omega(\xi)} \sum_{k=1}^{N} c_k \sum_{\mathbf{j} \in \mathbb{Z}^3} C_{\mathbf{j}}(\xi) e^{-2\pi \mathrm{i} \mathbf{r}_k \cdot \mathbf{j}} \frac{1}{(\sigma n)^3} \sum_{\mathbf{l} \in I_{\sigma n}} e^{2\pi \mathrm{i} (\mathbf{j} - \omega) \cdot \frac{\mathbf{l}}{\sigma n}}. \end{aligned}$$

Since

$$\frac{1}{(\sigma n)^3} \sum_{\mathbf{l} \in I_{\sigma n}} e^{2\pi \mathrm{i} (\mathbf{j} - \omega) \cdot \frac{\mathbf{l}}{\sigma n}} = \begin{cases} 1, & \text{if } \mathbf{j} - \boldsymbol{\omega} = \mathbf{i} \sigma n, \mathbf{i} \in \mathbb{Z}^3, \\ 0, & \text{otherwise}, \end{cases}$$

we have,

$$E_{\omega}^{1}=a(\boldsymbol{\omega})-\frac{1}{C_{\omega}(\xi)}\sum_{k=1}^{N}c_{k}\sum_{\mathbf{i}\in\mathbb{Z}^{3}}C_{\omega+\mathbf{i}\sigma n}(\xi)e^{-2\pi\mathrm{i}\mathbf{r}_{k}\cdot(\omega+\mathbf{i}\sigma n)}. \tag{B.13}$$

By (B.1),

$$E_{\omega}^{1}=\frac{1}{C_{\omega}(\xi)}\sum_{k=1}^{N}c_{k}\sum_{\mathbf{i}\in\mathbb{Z}^{3}\setminus\{0\}}C_{\omega+\mathbf{i}\sigma n}(\xi)e^{-2\pi\mathrm{i}\mathbf{r}_{k}\cdot(\omega+\mathbf{i}\sigma n)}. \tag{B.14}$$

Define $\|\mathbf{c}\|_{1}=\sum_{k=1}^{N}|c_{k}|$. Then we have

$$|E_{\omega}^{1}|\leq\|\mathbf{c}\|_{1}\sum_{\mathbf{i}\in\mathbb{Z}^{3}\setminus\{0\}}\frac{C_{\omega+\mathbf{i}\sigma n}(\xi)}{C_{\omega}(\xi)}. \tag{B.15}$$

In (B.9), the truncation error

$$E_{\omega}^{2}=\sum_{k=1}^{N}c_{k}\left(\frac{1}{(\sigma n)^{3}C_{\omega}(\xi)}\sum_{\mathbf{l}\in I_{\sigma n}}[\xi(\mathbf{r}_{k}-\frac{\mathbf{l}}{\sigma n})-\eta(\mathbf{r}_{k}-\frac{\mathbf{l}}{\sigma n})]e^{-2\pi\mathrm{i}\omega\cdot\frac{\mathbf{l}}{\sigma n}}\right), \tag{B.16}$$

which has the bound

$$|E_{\omega}^{2}|\leq\frac{1}{(\sigma n)^{3}}\|\mathbf{c}\|_{1}\frac{1}{C_{\omega}(\xi)}\max_{k}\sum_{\mathbf{l}\in I_{\sigma n}}|\xi(\frac{\mathbf{l}}{\sigma n}-\mathbf{r}_{k})-\eta(\frac{\mathbf{l}}{\sigma n}-\mathbf{r}_{k})|. \tag{B.17}$$

Appendix C. Continuity of f

As defined in Section 3.3.1,

$$f=\frac{(\mathbf{r}-\mathbf{x}_{i})\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_{i}|^{4}}, \tag{C.1}$$

where $\mathbf{r}\neq\mathbf{x}_{i}$ and $\mathbf{n}=\nabla F$ with F given in (2.2). $\mathbf{r}(b_{1}, b_{2}, \lambda)$ is simply defined in (2.3). In this appendix, we mainly discuss the continuity of $\mathbf{n}$. As derived in [22],

$$\nabla F=\mathscr{T}^{-1}\left(\frac{\partial F}{\partial b_{1}},\frac{\partial F}{\partial b_{2}},\frac{\partial F}{\partial\lambda}\right)^{\mathrm{T}}$$

(C.2)

where

$$\mathcal{J} = \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k)^{\mathrm{T}} \end{pmatrix}$$

is a nonsingular matrix. Hence **n** is well defined. Consider $\left(\frac{\partial \mathbf{n}}{\partial b_1}, \frac{\partial \mathbf{n}}{\partial b_2}\right)$:

$$\left(\frac{\partial \mathbf{n}}{\partial b_1}, \frac{\partial \mathbf{n}}{\partial b_2}\right) = \begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{xy} & F_{yy} & F_{yz} \\ F_{xz} & F_{yz} & F_{zz} \end{pmatrix} \begin{pmatrix} \frac{\partial x}{\partial b_1} & \frac{\partial x}{\partial b_2} \\ \frac{\partial y}{\partial b_1} & \frac{\partial y}{\partial b_2} \\ \frac{\partial z}{\partial b_1} & \frac{\partial z}{\partial b_2} \end{pmatrix}.$$

Let $v=\left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda}\right)^{\mathrm{T}}$. We have

$$\begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{xy} & F_{yy} & F_{yz} \\ F_{xz} & F_{yz} & F_{zz} \end{pmatrix} = \begin{pmatrix} \mathcal{J}_x & \mathcal{J}_y & \mathcal{J}_z \end{pmatrix} \begin{pmatrix} v & & \\ & v & \\ & & v \end{pmatrix} + \mathcal{J} M \mathcal{J}^{\mathrm{T}}$$

(C.3)

where $F_{xy} = \frac{\partial^2 F}{\partial x \partial y}$, $\mathcal{J}_x = \frac{\partial \mathcal{J}}{\partial x}$, and

$$M = \begin{pmatrix} F_{b_1 b_1} & F_{b_1 b_2} & F_{b_1 \lambda} \\ F_{b_1 b_2} & F_{b_2 b_2} & F_{b_2 \lambda} \\ F_{b_1 \lambda} & F_{b_2 \lambda} & F_{\lambda\lambda} \end{pmatrix}.$$

(C.4)

To show $\mathcal{J}$ is differentiable, we take the first row of $\mathcal{J}$ and compute its derivative with respect to *x*, i.e. $\left(\frac{\partial^2 b_1}{\partial x^2} \frac{\partial^2 b_2}{\partial x^2} \frac{\partial^2 \lambda}{\partial x^2}\right)$ as an example. We write (2.3) in the form of

$$\begin{cases} x = x(b_1, b_2, \lambda), \\ y = y(b_1, b_2, \lambda), \\ z = z(b_1, b_2, \lambda). \end{cases}$$

(C.5)

Taking the second derivatives of both sides of (C.5) with respect to *x*, we get

$$0=C_f+\frac{\partial x}{\partial b_1}\frac{\partial^2 b_1}{\partial x^2}+\frac{\partial x}{\partial b_2}\frac{\partial^2 b_2}{\partial x^2}+\frac{\partial x}{\partial \lambda}\frac{\partial^2 \lambda}{\partial x^2}, \tag{C.6}$$

$$0=C_g+\frac{\partial y}{\partial b_1}\frac{\partial^2 b_1}{\partial x^2}+\frac{\partial y}{\partial b_2}\frac{\partial^2 b_2}{\partial x^2}+\frac{\partial y}{\partial \lambda}\frac{\partial^2 \lambda}{\partial x^2}, \tag{C.7}$$

$$0=C_h+\frac{\partial z}{\partial b_1}\frac{\partial^2 b_1}{\partial x^2}+\frac{\partial z}{\partial b_2}\frac{\partial^2 b_2}{\partial x^2}+\frac{\partial z}{\partial \lambda}\frac{\partial^2 \lambda}{\partial x^2}. \tag{C.8}$$

where

$$C_f=\begin{pmatrix} \frac{\partial b_1}{\partial x} \\ \frac{\partial b_2}{\partial x} \\ \frac{\partial \lambda}{\partial x} \end{pmatrix}\cdot\begin{pmatrix} \frac{\partial^2 x}{\partial b_1^2}\frac{\partial b_1}{\partial x}+\frac{\partial^2 x}{\partial b_1 \partial b_2}\frac{\partial b_2}{\partial x}+\frac{\partial^2 x}{\partial b_1 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 x}{\partial b_1 \partial b_2}\frac{\partial b_1}{\partial x}+\frac{\partial^2 x}{\partial b_2^2}\frac{\partial b_2}{\partial x}+\frac{\partial^2 x}{\partial b_2 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 x}{\partial b_1 \partial \lambda}\frac{\partial b_1}{\partial x}+\frac{\partial^2 x}{\partial b_2 \partial \lambda}\frac{\partial b_2}{\partial x}+\frac{\partial^2 x}{\partial \lambda^2}\frac{\partial \lambda}{\partial x} \end{pmatrix},$$

$$C_g=\begin{pmatrix} \frac{\partial b_1}{\partial x} \\ \frac{\partial b_2}{\partial x} \\ \frac{\partial \lambda}{\partial x} \end{pmatrix}\cdot\begin{pmatrix} \frac{\partial^2 y}{\partial b_1^2}\frac{\partial b_1}{\partial x}+\frac{\partial^2 y}{\partial b_1 \partial b_2}\frac{\partial b_2}{\partial x}+\frac{\partial^2 y}{\partial b_1 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 y}{\partial b_1 \partial b_2}\frac{\partial b_1}{\partial x}+\frac{\partial^2 y}{\partial b_2^2}\frac{\partial b_2}{\partial x}+\frac{\partial^2 y}{\partial b_2 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 y}{\partial b_1 \partial \lambda}\frac{\partial b_1}{\partial x}+\frac{\partial^2 y}{\partial b_2 \partial \lambda}\frac{\partial b_2}{\partial x}+\frac{\partial^2 y}{\partial \lambda^2}\frac{\partial \lambda}{\partial x} \end{pmatrix},$$

$$C_h=\begin{pmatrix} \frac{\partial b_1}{\partial x} \\ \frac{\partial b_2}{\partial x} \\ \frac{\partial \lambda}{\partial x} \end{pmatrix}\cdot\begin{pmatrix} \frac{\partial^2 z}{\partial b_1^2}\frac{\partial b_1}{\partial x}+\frac{\partial^2 z}{\partial b_1 \partial b_2}\frac{\partial b_2}{\partial x}+\frac{\partial^2 z}{\partial b_1 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 z}{\partial b_1 \partial b_2}\frac{\partial b_1}{\partial x}+\frac{\partial^2 z}{\partial b_2^2}\frac{\partial b_2}{\partial x}+\frac{\partial^2 z}{\partial b_2 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 z}{\partial b_1 \partial \lambda}\frac{\partial b_1}{\partial x}+\frac{\partial^2 z}{\partial b_2 \partial \lambda}\frac{\partial b_2}{\partial x}+\frac{\partial^2 z}{\partial \lambda^2}\frac{\partial \lambda}{\partial x} \end{pmatrix}.$$

So we get

$$\begin{pmatrix} \frac{\partial^2 b_1}{\partial x^2} \\ \frac{\partial^2 b_2}{\partial x^2} \\ \frac{\partial^2 \lambda}{\partial x^2} \end{pmatrix} = \mathcal{J}\begin{pmatrix} -C_f \\ -C_g \\ -C_h \end{pmatrix} \tag{C.9}$$

Using the same method, we can get the other rows of $\frac{\partial \mathcal{F}}{\partial \mathbf{x}}$, matrices $\mathcal{F}_y$ and $\mathcal{F}_z$ by changing C_f, C_g, C_h in (C.9). Therefore $\mathcal{F}$ is differentiable. Similarly, we can compute the higher order derivatives of $\mathcal{F}$ and prove that $\mathcal{F} \in C^\infty$, thus prove $F \in C^\infty(\Omega_0)$, where Ω_0 defined in Section 3.3.1 is the canonical triangle. Therefore, as defined in (C.1), $f \in C^\infty(\Omega_0)$.

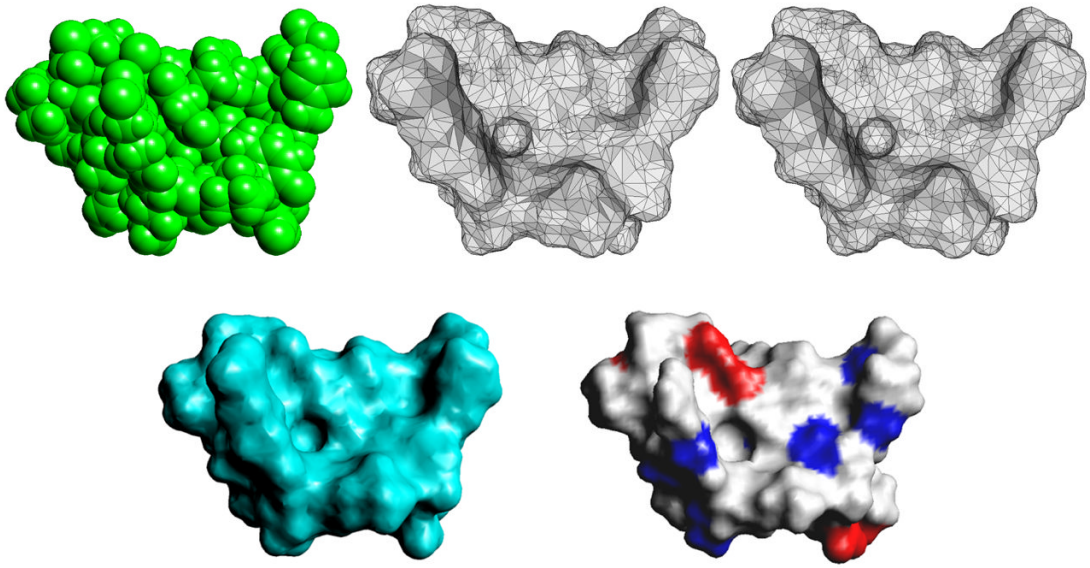


Fig. 1.1.
Top left: the discrete van der Waals surface model (436 atoms); top middle: the triangulation of the continuum Gaussian surface model with 6004 triangles; top right: the regularized triangular mesh where the quality of the elements is improved (making each as close as possible to an equilateral triangles); bottom left: the continuum ASMS model generated from the triangular mesh up right; bottom right: the molecular surface rendered according to the interaction with the solvent where red means strong and blue means weak interaction.

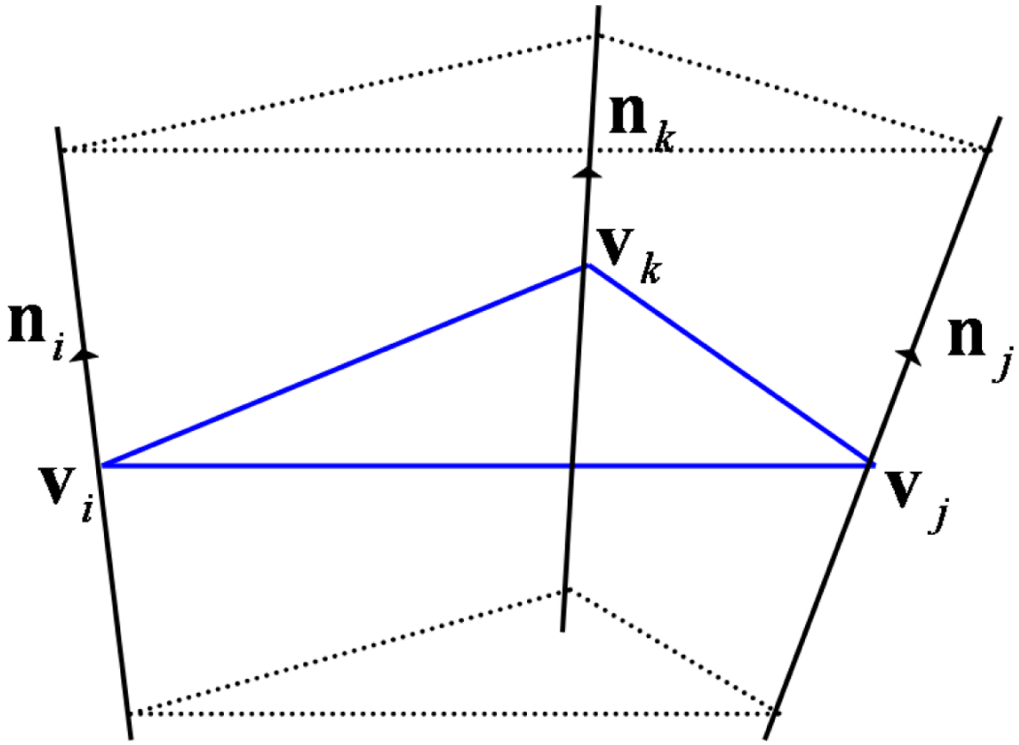


Fig. 2.1 .
A prism D_{ijk} constructed with a triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$ as a basis.

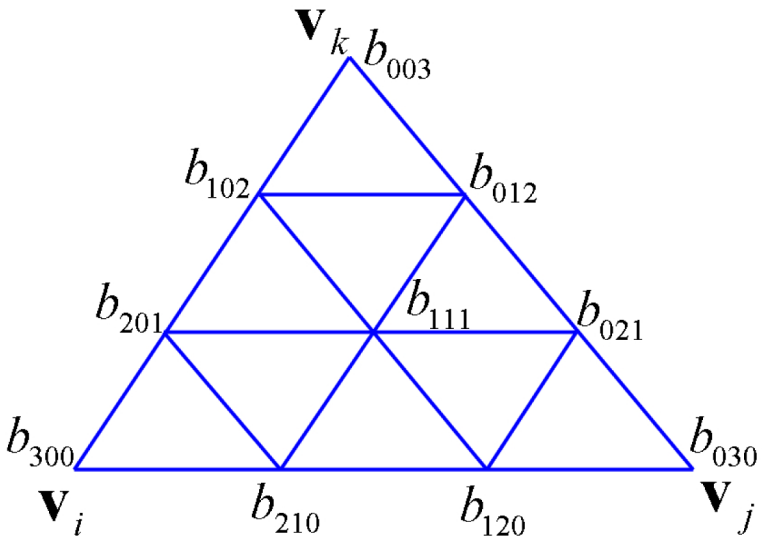


Fig. 2.2 .
The control coefficients of the cubic Bernstein-Bezier basis of function F

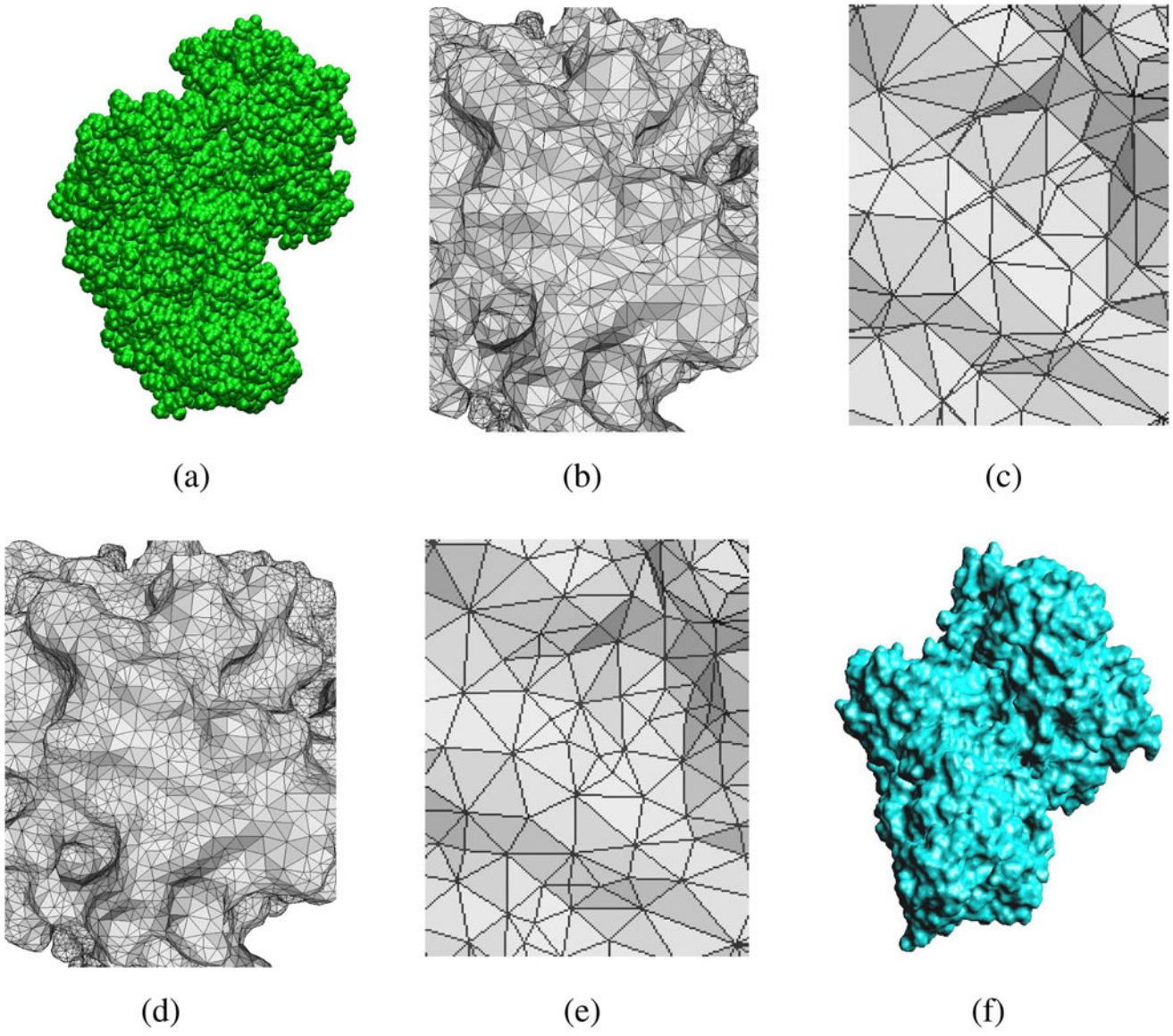


Fig. 2.3 .
(a) is the discrete van der Waals model of protein 1BGX with 19,647 atoms; (b) and (c) are the zoom-in views of the the initial triangulation of the continuum surface with 85656 triangles; (d) and (e) are the zoom-in views of the quality improved mesh; (f) is the a continuum ASMS model generated based on the quality improved mesh.

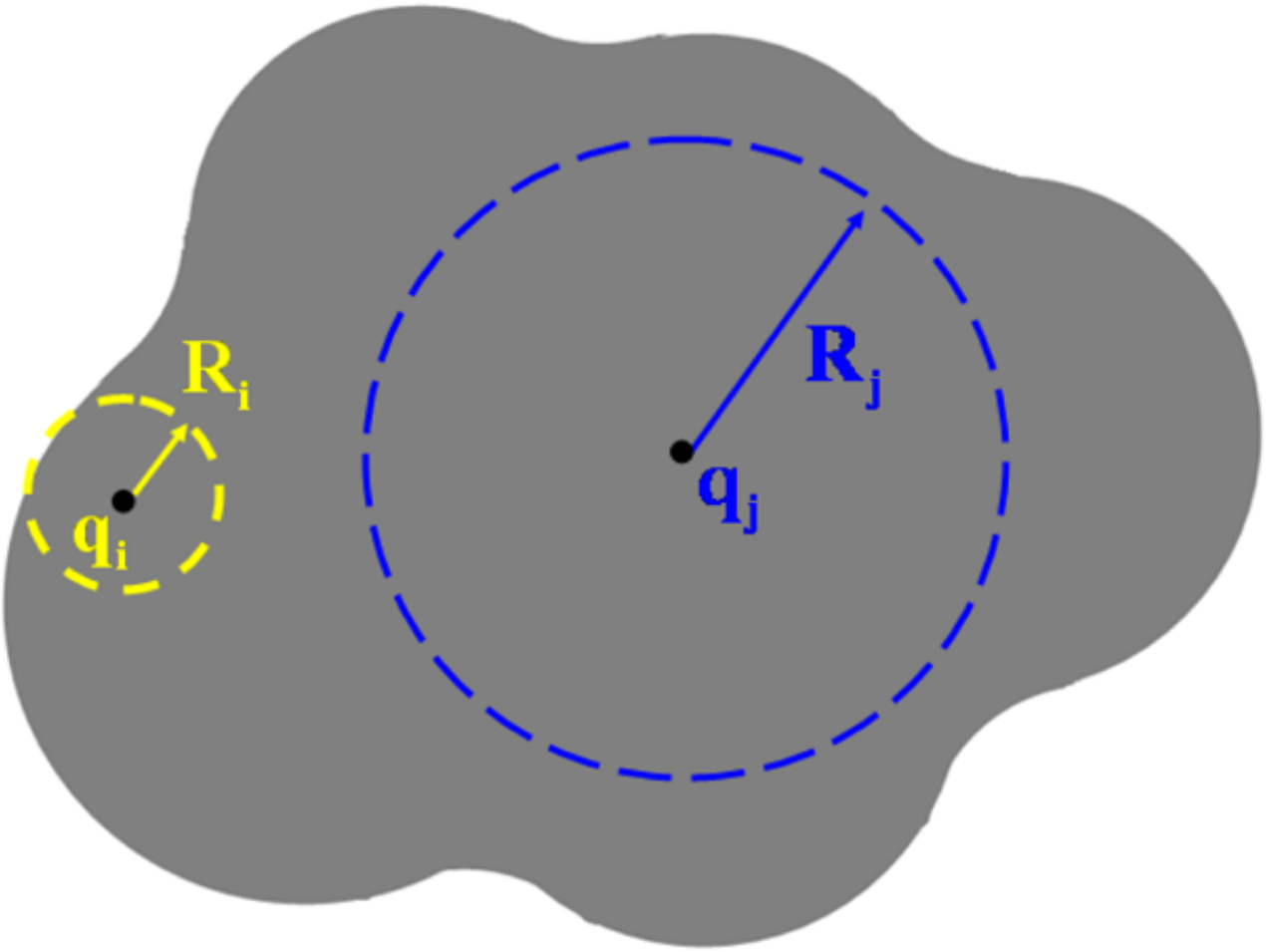


Fig. 3.1 .
The effective Born radius reflects how deep a charge is buried inside the molecule. The Born radius of an atom is small if the atom is close to the surface of the molecule, otherwise the Born radius is large therefore has weaker interaction with the solvent.

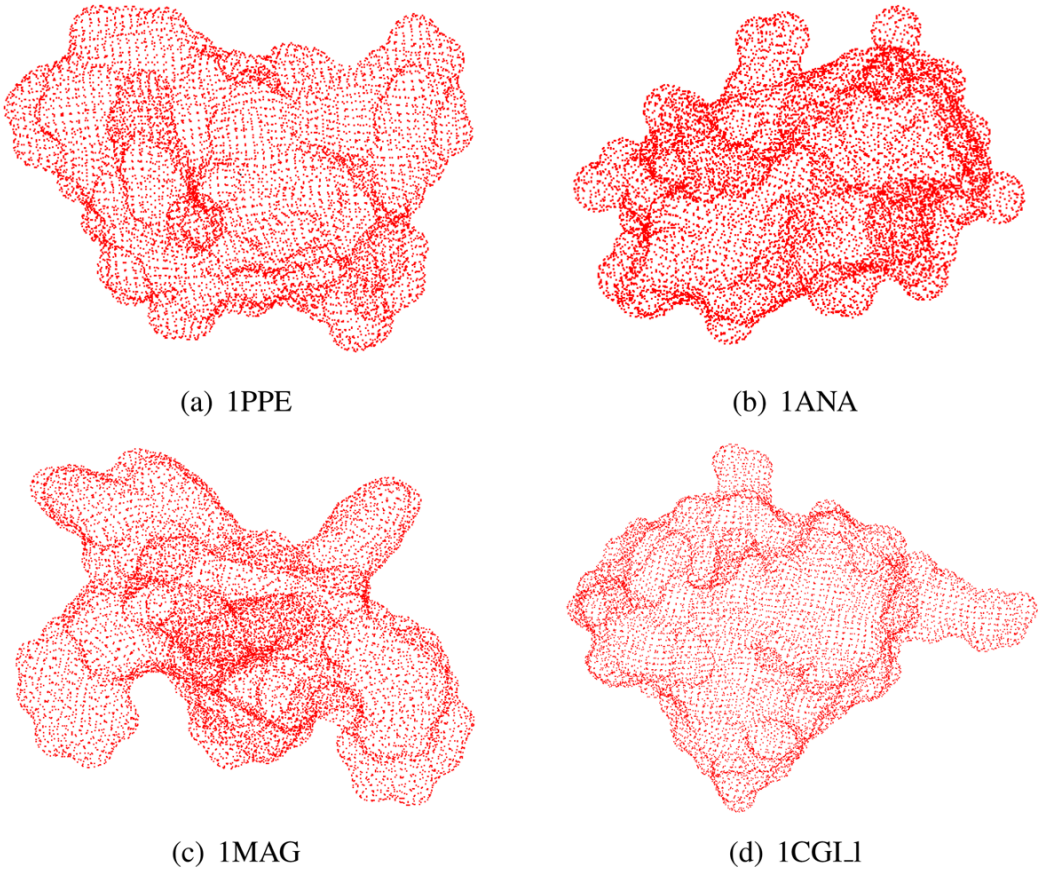
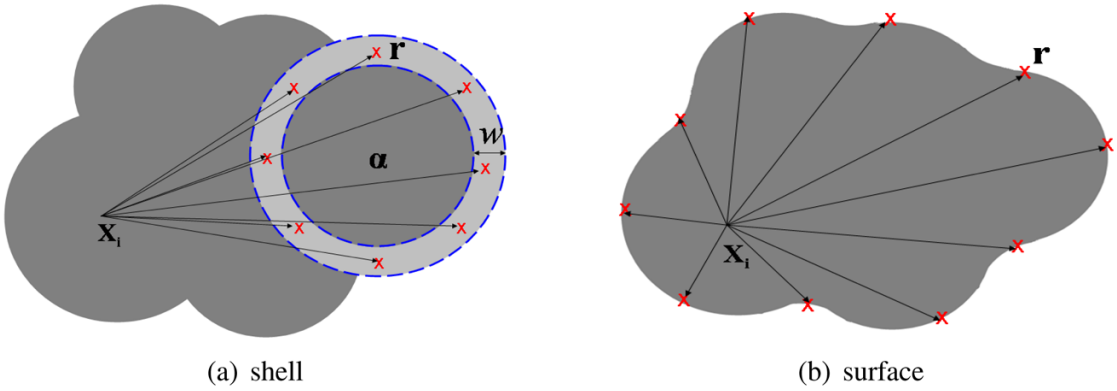


Fig. 3.2 .
Gaussian integration points on the surface of protein (a) 1PPE, (b) 1ANA, (c) 1MAG, and (d) 1CGI l. The surfaces are partitioned into 24244 triangular patches for (a), 28620 triangular patches for (b), 30624 triangular patches for (c), and 29108 triangular patches for (d). There are three Gaussian quadrature nodes per triangle. The nodes are then mapped onto the ASMS to form the red point cloud.



(a) shell (b) surface

Fig. 4.1 .

When computing the derivatives of the Born radii $\frac{\partial R_i}{\partial x_\alpha}$, the quadrature points of the first integral are points within a spherical shell around atom α, as shown in (a), whereas the second integral is necessary when $i \equiv \alpha$ and the quadrature points are points on the surface, as shown in (b). The dark region represents the molecule, the light grey region is the shell of width w around atom α.

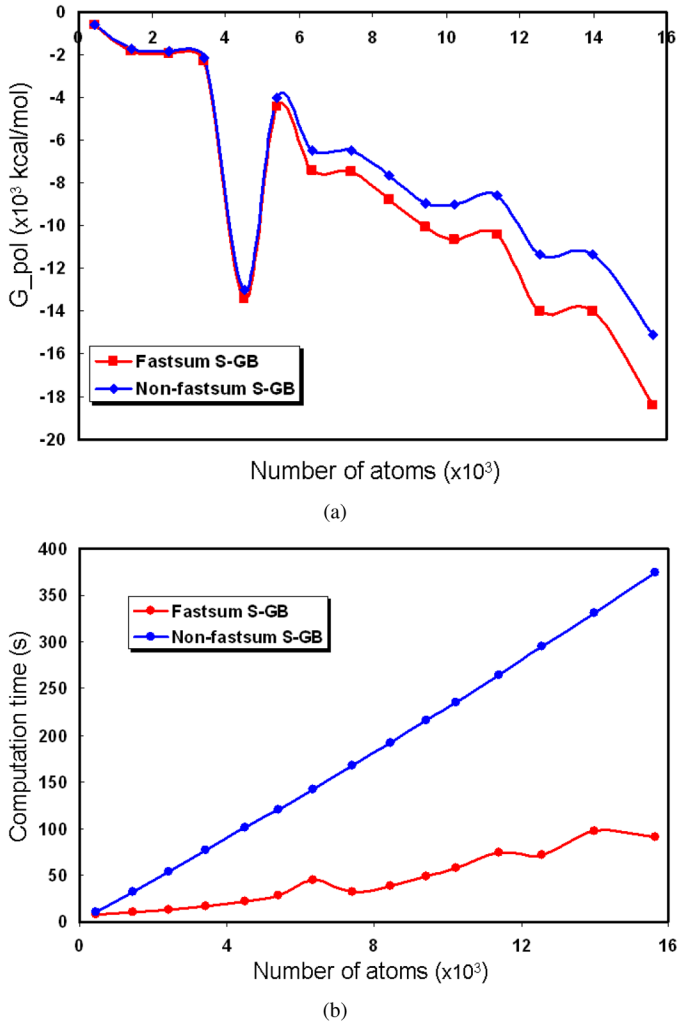
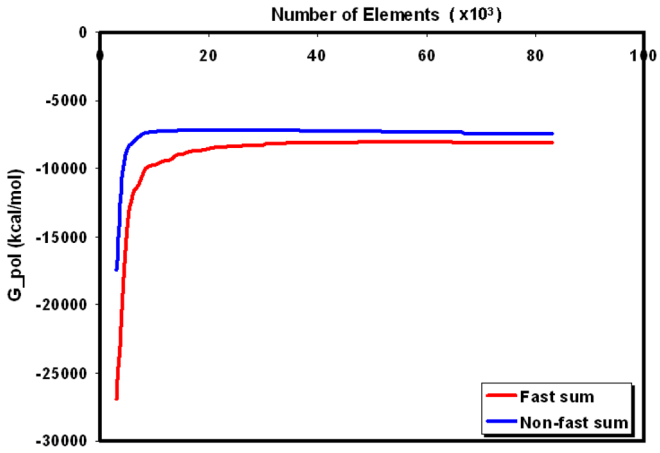
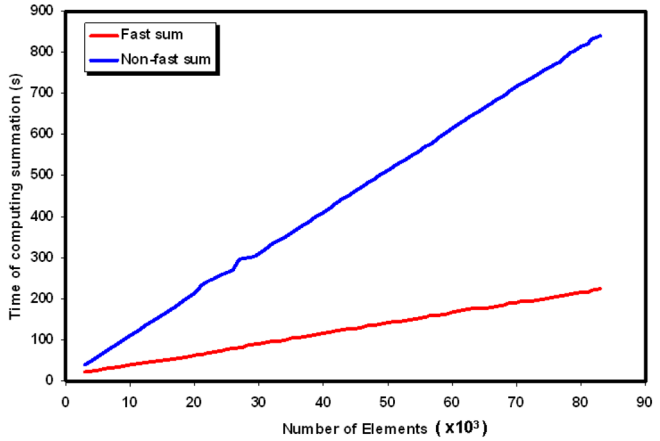


Fig. 5.1 .
In (a) we compare G_{pol} computed by the fastsum GB and the non-fastsum GB for various proteins containing different number of atoms. In (b) we compare the computation time of the two methods.



(a)



(b)

Fig. 5.2 .

For protein 1JPS, in (a) we compare G_{pol} computed by the fastsum GB and the non-fastsum GB with various number of surface elements. In (b) we compare the computation time of the two methods.

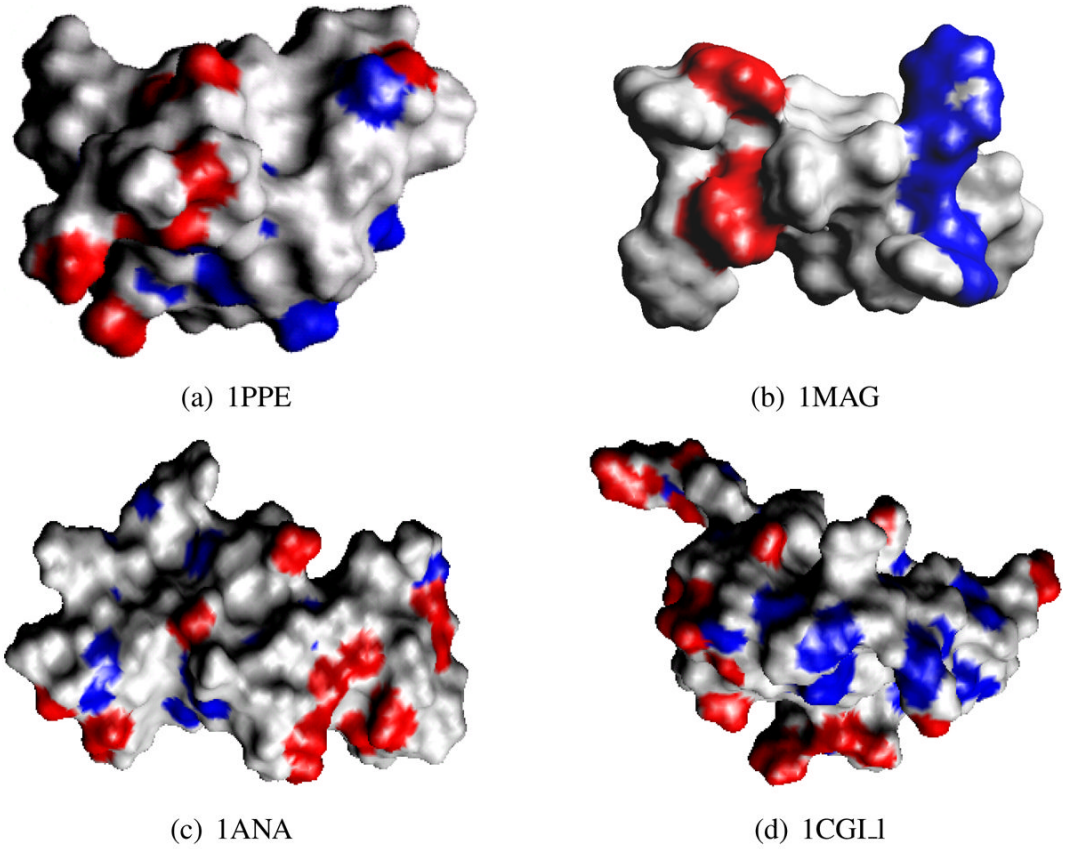


Fig. 5.3 .
Atoms that have the greatest electrostatic solvation force (top 5%) are colored in red; atoms that have the weakest electrostatic solvation force (bottom 5%) are colored in blue.

Table 5.1

Comparison of the electrostatic solvation energy G_{pol} (kcal/mol) and computation time (second) of the fast summation method (A) and the trivial summation method (B). M is the number of atoms. N_e is the number of patches, N is the number of integration points. n, σ, and m are parameters in the fast summation method. ε is the relative percentage error $|(G^A_{pol} - G^B_{pol})/G^B_{pol}|$.

Protein ID		**1CGI_l**	**1BGX**	**1DE4**	**1N2C**
M		852	19,647	26,003	39,946
N_e		29,108	112,636	105,288	83,528
N		116,432	450,544	421,152	334,112
n		100	100	100	100
σ		2	2	2	2
m		4	4	4	4
A	G_{pol}	−1380.988	−19734.848	−25754.552	−41408.959
	timing	86	358	863	631
B	G_{pol}	−1343.150	−19297.528	−25388.455	−40675.383
	timing	49	4327	5368	9925
ε		2.8%	2.3%	1.4%	1.8%

Table 5.2

Force calculation timing: M is the number of atoms, N is the number of triangles in the surface triangular mesh, t_1 is the time (in seconds) for computing (4.15) for i = 1, …, M and t_2 is the time for computing the rest of the terms in (4.6) for i, j, α = 1,…, M. T_{total} is the overall timing.

Protein ID	*M*	*N*	t_1 (s)	t_2 (s)	T_{total} (s)
1ANA	249	6,676	66.05	0.14	66.19
1MAG	544	7,328	69.58	0.23	69.81
1PPE_l	436	5,548	59.55	0.56	60.11
1CGI_l	852	6,792	68.71	3.27	71.98