Chapter 5: Derivatives and Integration

Chandrajit Bajaj and Andrew Gillette

October 22, 2010

Contents

1 Curvature Computations

See section in Chapter 1 of same name.

2 Numerical and Symbolic Integration

Let

$$I[f] = \int_{\Omega} w(\mathbf{x}) f(\mathbf{x}) \, d\mathbf{x}$$

where $\Omega \subset \mathbb{R}^n$. We are looking for a cubature formula which has the form

$$Q[f] = \sum_{j=1}^{N} w_j f(\mathbf{x}^{(j)}) \tag{1}$$

where the points $\mathbf{x}^{(j)}$ and the weights w_j are independent of the function f and are chosen so that $Q[f]$ gives a good approximation to $I[f]$ for some class of functions.

Because a well-behaved function can be approximated by a polynomial, thus the algebraic degree of a cubature formula is a measure of the quality of the cubature.

We discuss how to construct a high order cubature formula on an A-path, i.e. Ω is a A-patch.

2.1 Cubature Formulae Notation

Definitions

- The vector space of all algebraic polynomials in n variables of degree at most d is denoted by P_d^n.
- A polynomial $f \in P^n$ is called *d-orthogonal polynomial* if $I[fg] = 0$ whenever $fg \in P_d^n$.
- A polynomial $f \in P^n$ is called *orthogonal*(w.r.t. integral I), if $I[fg] = 0$ whenever $\deg(g) < \deg(f)$.
- A set of polynomials S is called *fundamental of degree d* if $\dim V_d^{n-1} (= \dim V_d^n - \dim V_{d-1}^n)$ linearly independent polynomials of the form $x_1^{\alpha_1} \dots x_n^{\alpha_n} + P_\alpha$, $P_\alpha \in V_{d-1}^n$, $\sum \alpha_i = d$, belong to $\text{span}(S)$.
- *Zero set* of an ideal $\mathfrak{U}$: $NG(\mathfrak{U}) = \{\mathbf{x} \in \mathbb{C}^n : \ f(\mathbf{x}) = 0 \text{ for all } f \in \mathfrak{U}\}$.

- The *Hilbert function* $\mathcal{H}$ is defined as

$$\mathcal{H}(k;\mathfrak{U}) := \begin{cases} \dim \mathcal{P}_k^n - \dim(\mathfrak{U} \bigcap \mathcal{P}_k^n), & k \in \mathbb{N}, \\ 0, & -k \in \mathbb{N}_0. \end{cases}$$

- Let $\mathfrak{U}$ be a polynomial ideal. The set $\{f_1, \ldots, f_s\} \subset \mathfrak{U}$ is an *H-basis* for $\mathfrak{U}$ if for all $f \in \mathfrak{U}$ there exist polynomials $g_1, \ldots, g_s$ such that

$$f = \sum_{j=1}^{s} g_j f_j \text{ and } \deg(g_j f_j) \leq \deg(f), \quad j = 1, \ldots, s.$$

- An ideal $\mathfrak{U}$ is a *real ideal* if all polynomials vanishing at $NG(\mathfrak{U}) \cap \mathbb{R}^n$ belong to $\mathfrak{U}$.

- *Basic orthogonal polynomials*: polynomials of the form $P^\alpha(x_1, \ldots, x_n) = x_1^{\alpha_1} \ldots x_n^{\alpha_n} + \tilde{P}$, with $\sum_{i=1}^n \alpha_i = d$ and $\tilde{P} \in P_{d-1}^n$

- A linear functional $I[\cdot]$ is *centrally symmetric* if

$$I[\mathbf{x}^\alpha] = 0 \qquad \forall \alpha \in \mathbb{N}_0^n, \ \sum_{j=1}^{n} \alpha_j \text{ odd}.$$

- Let the set of monomials $M = \{\mathbf{x}^\alpha : \alpha \in \mathbb{N}^n\}$ be ordered by $<$ such that, for any $f, f_1, f_2 \in M$, $1 \leq f$ and $f_1 \leq f_2$ imply $ff_1 \leq ff_2$. Let $f = \sum_{i=1}^m c_i f_i$ with $f_i \in M$ and $c_i \in \mathbb{R}_0$. Then the *headterm* of f =Hterm$(f) := f_m$, and the *maximal part* of $f = M(f) := c_m f_m$. For $f, g \in P^n \setminus \{0\}$ let

$$H(f,g) := \text{lcm}\{\text{Hterm(f)}, \text{Hterm(g)}\}.$$

- Let $F \subset P^n \setminus \{0\}$ be a finite set. We write $f \to_F g$ if $f, g \in P^n$ and there exist $h \in P^n, f_i \in F$ such that $f = g + hf_i$, Hterm$(g)<$Hterm(f) or $g = 0$. The map $\to_F$ is called a *reduction modulo* F. By $\to_F^+$ we donote the reflexive transitive closure of $\to_F$.

- A set $F := \{f_1, \ldots, f_l\}$ is a *Groebner basis* (G-basis) for the ideal $\mathfrak{U}$ generated by F if

$$f \in \mathfrak{U} \text{ implies } f \to_F^+ 0.$$

2.2 Constructing Cubature Formulae

2.2.1 Interpolatory Cubature Formulae

Definition: If the weights of a cubature formula of degree d are uniquely determined by the points, the cubature formula is called an *interpolatory cubature formula.*

A cubature formula that is exact for all elements of V_d^n is determined by a system of nonlinear equations

$$Q[f_i] = I[f_i], \quad i = 1, \ldots, \dim V_d^n, \tag{2}$$

where the f_i form a basis for V_d^n. If the N points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ of a cubature formula are given, then (2) is a system of $\dim V_d^n$ linear equations in the N unknown weights. Hence an interpolatory cubature formula has $N \leq \dim V_d^n$ and there exist N linearly independent polynomials $U_1, \ldots, U_N \in V_d^n$ such that

$$\det \begin{pmatrix} U_1(\mathbf{x}^{(1)}) & \ldots & U_N(\mathbf{x}^{(1)}) \\ \vdots & \vdots & \vdots \\ U_1(\mathbf{x}^{(N)}) & \ldots & U_N(\mathbf{x}^{(N)}) \end{pmatrix} \neq 0.$$

These polynomials generate a maximal, not uniquely determined, vector space of polynomials that do not vanish at all given points.

One can always find $t := \dim V_d^n - N$ polynomials $p_1, \ldots, p_t$ such that the polynomials

$$U_1, \ldots, U_N, p_1, \ldots, p_t$$

form a basis for V_d^n. Then one can solve

$$\begin{pmatrix} U_1(\mathbf{x}^{(1)}) & \ldots & U_N(\mathbf{x}^{(1)}) \\ \vdots & \vdots & \vdots \\ U_1(\mathbf{x}^{(N)}) & \ldots & U_N(\mathbf{x}^{(N)}) \end{pmatrix} \begin{pmatrix} a_{i1} \\ \vdots \\ a_{iN} \end{pmatrix} = \begin{pmatrix} p_i(\mathbf{x}^{(1)}) \\ \vdots \\ p_i(\mathbf{x}^{(N)}) \end{pmatrix}, \quad i = 1, \ldots, t,$$

and so obtain $t = \dim V_d^n - N$ linearly independent polynomials

$$R_i = p_i - \sum_{j=1}^{n} a_{ij} U_j, \quad i = 1, \ldots, t$$

that vanish at the given N points of the cubature formula. We can replace the polynomials p_i in the basis of V_d^n by the polynomials R_i.

With every cubature formula of degree d one can associate a basis of V_d^n that consists of $\dim V_d^n - N$ polynomials R_i that vanish at all the points of the cubature formula and N polynomials U_i that do not vanish at all points. A cubature formula is thus fully characterized by the polynomials R_i. The polynomials U_i give rise to a linear system that determines the weights $\omega_1, \ldots, \omega_N$:

$$Q[U_i] = \sum_{j=1}^{N} \omega_j U_i(\mathbf{x}^{(j)}) = I[U_i], \quad i = 1, \cdots, N.$$

The polynomials U_i and R_i are not uniquely determined . The direct sum of the vector spaces generated by these polynomials is

$$\text{span}\{U_i\} \oplus \text{span}\{R_i\} = V_d^n.$$

2.2.2 Ideal Theory

Theorem Let I be an integral over an $n-$dimensional region. Let $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\} \subset \mathbb{C}^n$ and $\mathfrak{U} := \{f \in V^n : f(\mathbf{x}^{(i)}) = 0,\ i = 1, \ldots, N\}$. Then the following statements are equivalent.

- $f \in \mathfrak{U} \bigcap V_d^n$ implies $I[f] = 0$.
- There exists a cubature formula $Q[f] := \sum_{j=1}^{n} \omega_j f(\mathbf{x}^{(j)})$ such that $I[f] = Q[f]$, for all $f \in V_d^n$, with at most $\mathcal{H}(d; \mathfrak{U})$ (complex) weights different from zero.

Theorem If $\{f_1, \ldots, f_s\}$ is an H-basis of a polynomial ideal $\mathfrak{U}$ and if the set of common zeros of $f_1, \ldots, f_s$ is finite and nonempty, then the following statements are equivalent.

- There is a cubature formula of degree d for the integral I which has as points the common zeros of $f_1, \ldots, f_s$. (These zeros may be multiple, leading to the use of function derivatives in the cubature formula.)
- f_i is $d-$orthogonal for $I,\ i = 1, 2, \ldots, s$.

Theorem Let $\{R_1, \ldots, R_t\} \subset P_{d+1}^n$ be a set of linearly independent d-orthogonal polynomials that is fundamental of degree $d+1$. Let $\mathfrak{U} = (R_1, \ldots, R_t)$, $V = \text{span}\{R_1, \ldots, R_t\}$. Let $N = \dim P_{d+1}^n - t$ and U be an arbitrary but fixed vector space such that $V \bigoplus U = P_{d+1}^n$. The following statements are equivalent:

(a) There exists an interpolatory cubature formula of degree d

$$Q[f] = \sum_{j=1}^{N} w_j f(\mathbf{x}^{(j)}) \tag{3}$$

with $\mathbf{x}^{(j)} \in \mathbb{R}^n$, $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\} \subset NG(\mathfrak{U})$, and $w_j > 0$.

(b)

$(i)\mathfrak{U} \cap U = \{0\}$
$(ii)I[f^2 - g] > 0$ for all $f \in U$, and $g \in \mathfrak{U}$ satisfying $f^2 - g \in P_d^n$

(c) $\mathfrak{U}$ is a real ideal and $|NG(\mathfrak{U}) \cap \mathbb{R}^n| = N$. $\mathbf{x}^{(j)}$ are elements of $NG(\mathfrak{U}) \cap \mathbb{R}^n$.

2.2.3 Bounds

Upper Bound

Tchakaloff's Theorem: Let I be an integral over an $n-$dimensional region Ω with a weight function that is nonnegative in Ω and for which the integrals of all monomials exist. then a cubature formula of degree d with $N \leq \dim V_d^n$ points exists with all points inside Ω and all weights positive.

Lower Bound

Theorem If the cubature formula $Q[f] = \sum_{j=1}^{N} \omega_j f(\mathbf{x}^{(j)}),\ \omega_j \in \mathbb{R}$ is exact for all polynomials of V_{2k}^n, then the number of points $N \geq \dim V_k^n|_\Omega$.

Theorem Let R_{2k} denote the vector space of even polynomials of $P_{2k+1}^n|_\Omega$ and R_{2k+1} denote the vector space of odd polynomials of $P_{2k+1}^n|_\Omega,\ k \in \mathbb{N}_0$. If the algebraic degree of the cubature formula $Q[f] = \sum_{j=1}^{N} \omega_j f(y^{(j)}),\ \omega_j \in \mathbb{R}$ for a centrally symmetric integral is $d = 2k+1$, then

$$\begin{array}{ll} N \geq 2\dim R_k - 1, & \text{if } k \text{ even and 0 is a point,} \\ N \geq 2\dim R_k, & \text{otherwise.} \end{array}$$

A cubature formula that attains this bound is centrally symmetric and has all weights positive.

2.2.4 The characterization of minimal formulae and the reproducing kernel

Choose the polynomials $\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots \in V^n$ such that $\phi_i(\mathbf{x})$ is orthogonal to $\phi_j(\mathbf{x})$, for all $j < i$, and $I(\phi_i \bar{\phi}_i) = 1$. This means that $\{\phi_i(\mathbf{x})\}_{i=1}^{\infty}$ is an orthogonal basis of V^n. For a given $k \in \mathbb{N}$ we set $\kappa := \dim V_k^n$ and

$$K(\mathbf{x},\mathbf{y}) := \sum_{i=1}^{\kappa} \bar{\phi}_i(\mathbf{x})\phi_i(\mathbf{y}).$$

where $\bar{\phi}_i(\mathbf{x})$ is the conjugate of $\phi_i(\mathbf{x})$. $K(\mathbf{x}, \mathbf{y})$ is reproducing kernel in the space V_k^n: if $f \in V_k^n$ then f coincides with its expansion in ϕ_i, so that for $\mathbf{a} \in \mathbb{C}^n$ fixed,

$$f(\mathbf{a}) = I[f(\mathbf{x})K(\mathbf{x},\mathbf{a})] = \sum_{i=1}^{\kappa} I[f(\mathbf{x})\bar{\phi}_i(\mathbf{x})]\phi_i(\mathbf{a}).$$

Theorem A necessary and sufficient condition for the points $\mathbf{x}^{(j)},\ j = 1, \ldots, N = \dim V_k^n$, to be the points of a cubature formula that is exact for V_{2k}^n is that

$$K(\mathbf{x}^{(r)}, \mathbf{x}^{(s)}) = b_r \delta_{rs},$$

with $b_r \neq 0$ and δ_{rs} the Kronecker symbol.

q	1	2	3	4	5
v_1	0.3333333333	0.0	0.1333333333	0.8168475729	0.05961587
v_2		0.5	0.1333333333	0.0915762135	0.47014206
v_3		0.5	0.7333333333	0.0915762135	0.47014206
v_4			0.3333333333	0.1081030181	0.79742699
v_5				0.4459484909	0.10128651
v_6				0.4459484909	0.10128651
v_7					0.33333333
w_1	0.3333333333	0.5	0.7333333333	0.0915762135	0.47014206
w_2		0.0	0.1333333333	0.8168475729	0.05961587
w_3		0.5	0.1333333333	0.0915762135	0.47014206
w_4			0.3333333333	0.4459484909	0.10128651
w_5				0.1081030181	0.79742699
w_6				0.4459484909	0.10128651
w_7					0.33333333
W_1	1.0	0.3333333333	0.5208333333	0.1099517436	0.13239415
W_2		0.3333333333	0.5208333333	0.1099517436	0.13239415
W_3		0.3333333333	0.5208333333	0.1099517436	0.13239415
W_4			-0.5625	0.2233815896	0.12593918
W_5				0.2233815896	0.12593918
W_6				0.2233815896	0.12593918
W_7					0.225

Table 1: Integration rules over triangle. $(1 - v_i - w_i, v_i, w_i)$ are barycentric coordinates of the nodes. W_i are the weights. The first row represents the algebraic precision.

2.3 Gauss Formula on an A-patch

2.3.1 Gauss Points and Weights

When we do some numerical integral calculation on an A-patch, constructing a Gauss integral formula is a good choice. While to construct the Gauss integral formula on an A-patch directly is usually difficult.

From the discussion above we have established the bijection between an A-patch and an planar triangle. The Gauss points in a triangle is already known.

We have experimented with a set of numerical integration schemes in the q-version over triangles. These include one point, three points, four points, six points and seven points rules. Table 1 summarizes these rules with coordinates, weights and algebraic precision.

Our method is to project these Gauss points onto the A-patch and use them as well as corresponded weights as the Gauss points on the A-patch.

2.3.2 Error Analysis

The function $f(x, y, z)$ defined on the A-patch S can be expressed as

$$\begin{aligned} f(x,y,z) &= f(x(\beta_1,\beta_2,t(\beta_1,\beta_2)), y(\beta_1,\beta_2,t(\beta_1,\beta_2)), z(\beta_1,\beta_2,t(\beta_1,\beta_2))) \\ &\triangleq g(\beta_1,\beta_2,t(\beta_1,\beta_2)). \end{aligned}$$

The integral on the A-patch S is

$$\begin{aligned} & \int\int_S f(x,y,z)ds \\ = \; & \int_0^1 \int_0^{1-\beta_1} g(\beta_1,\beta_2,t(\beta_1,\beta_2)) \sqrt{1 + \left(\frac{\partial t}{\partial \beta_1}\right)^2 + \left(\frac{\partial t}{\partial \beta_2}\right)^2} d\beta_2 d\beta_1 \end{aligned}$$

Let $0 \le \beta_1, \beta_2 \le 1$ and $\beta_1 + \beta_2 \le 1$, then $\beta_1 P_1 + \beta_2 P_2 + (1 - \beta_1 - \beta_2)P_3$ is a point Q in the triangle $P_1P_2P_3$. The line connecting this point and P_4 intersects the A-patch with a unique point P. The point P on this line can be written as

$$P = (1-t)\beta_1 P_1 + (1-t)\beta_2 P_2 + (1-t)(1-\beta_1-\beta_2)P_3 + tP_4 = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3 + \alpha_4 P_4 \quad (4)$$

Since this point is on the A-patch, substitute the equation above into

$$F(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = 0$$

we get a equation about t, β_1, β_2

$$F'(t, \beta_1, \beta_2) = 0. \tag{5}$$

According to equation (4)

$$\begin{aligned} f(x,y,z)|_S &= f(P)|_{P\in S} \\ &= f(\alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3 + \alpha_4 P_4) \\ &= f((1-t)\beta_1 P_1 + (1-t)\beta_2 P_2 + (1-t)(1-\beta_1-\beta_2)P_3 + tP_4) \end{aligned}$$

Here β_1, β_2, t satisfy equation (5) and $F'(\beta_1, \beta_2, t)$ is a polynomial about β_1, β_2, t. So

$$\frac{\partial t}{\partial \beta_1} = -\frac{(\partial F')/(\partial \beta_1)}{(\partial F')/(\partial t)}$$

$$\frac{\partial t}{\partial \beta_2} = -\frac{(\partial F')/(\partial \beta_2)}{(\partial F')/(\partial t)}$$

If $f(x,y,z)$ is a polynomial of degree n, then the degree of $g(\beta_1, \beta_2, t)$ should be $2n$. If the A-patch is of degree m, then $(\partial F')/(\partial \beta_i)$ and $(\partial F')/(\partial t)$ are all of degree $m-1$. So the degree of $g(\beta_1, \beta_2, t(\beta_1, \beta_2))\sqrt{1 + \left(\frac{\partial t}{\partial \beta_1}\right)^2 + \left(\frac{\partial t}{\partial \beta_2}\right)^2}$ should be of degree $2n+m$. That is to say, if we use the Gauss points of the triangle constructing a cubature formula which is exact for polynomial of degree $2n+m$ in the integral $\int_0^1 \int_0^{1-\beta_1} h(\beta_1, \beta_2) d\beta_2 d\beta_1$, it is actually exact for polynomial of degree n in the integral $\int\int_S f(x,y,z)ds$.

So we need to find a better way to calculate the integral on the A-patch.

2.4 T-method

This method was first suggested by Morrow and Patterson (1978)[21] and Schmid (1978)[47] for two-dimensional regions. It was further developed by Schmid (1980)[48]; see also Schmid (1980)[49] and Schmid (1995)[50].

Construct R_i: Consider the 2D case.

$$R_i = P^{k+1-i,i} + \sum_{j=0}^{k} \beta_{ij} P^{k-j,j} + \sum_{j=0}^{k-1} \gamma_{ij} P^{k-1-j,j}, \qquad i = 0, \ldots, k+1 \tag{6}$$

where $P^{a,b}$ are the basic orthogonal polynomials.

When the integral is centrally symmetric, β_{ij} vanish. To determine γ_{ij}, let

$$T_i = yR_i - xR_{i+1}, \qquad i = 0, \ldots, k$$

Assume two conditions:

- T_i is a polynomial of degree k and orthogonal
- xT_i, yT_i, $i = 0, \ldots, k$, are of degree $k+1$ and $xT_i, yT_i \in \text{span}\{R_0, \ldots, R_{k+1}\}$.

The above two conditions lead to linear and quadratic equations in γ_{ij}. The inequality in Theorem 6.8 leads to inequalities for γ_{ij}.

Theorem Let

$$\begin{aligned} R_i &= P^{k+1-i,i} + \textstyle\sum_{j=0}^{k-1} \gamma_{ij} P^{k-1-j,j}, & i = 0, \ldots, k+1 \\ T_i &= yR_i - xR_{i+1}, & i = 0, \ldots, k \end{aligned}$$

If the polynomials T_i are $(2k-1)-$orthogonal and if all polynomials xT_i, yT_i are elements of $\text{span}\{R_0,\ldots,R_{k+1}\}$, then $F := \{R_0,\ldots,R_{k+1},T_0,\ldots,T_k\}$ is a G-basis.

Theorem Let F be as defined in Theorem 9.1. If the common zeros of the polynomials in F are real and simple, then there exists a cubature formula of degree $2k-1$ with the elements of $NG(F)$ as points. The number of points $N \le \dfrac{k(k+3)}{2}$.

Theorem If the ideal of all polynomials that vanish at the N points of a cubature formula of degree $2k-1$ contains a fundamental set of degree $k+1$, then

$$\frac{k(k+1)}{2} + \left\lfloor \frac{k}{2} \right\rfloor \le N \le \frac{k(k+1)}{2} + \left\lfloor \frac{k}{2} \right\rfloor + 1.$$

With the N points as the ones of the cubature formula we only need to determine the weights $\omega_1,\ldots,\omega_N$ by equations

$$Q[U_i] = \sum_{j=1}^{N} \omega_j U_i(\mathbf{x}^{(j)}) = I[U_i], \quad i = 1,\ldots,N.$$

where $U_1,\ldots,U_N$ is a basis of U, such that $U \oplus (\text{span}(F) \cap P_d^n) = P_d^n$.

Algorithm:

Step 1. Construct the basic orthogonal polynomials of degree $k-1, k, k+1$ respectively, which can be derived by solving the equations $I[P^{i,j}\mathbf{x}^{\mathbf{m}}] = 0,\ \mathbf{m} \in \mathbb{N}^n, 0 \le |\ \mathbf{m}\ | \le i+j$.

Step 2. Let

$$R_i = P^{k+1-i,i} + \sum_{j=0}^{k-1} \gamma_{ij} P^{k-1-j,j}, \qquad i = 0,\ldots,k+1$$

Step 3. Let

$$T_i = yR_i - xR_{i+1}, \qquad i = 0,\ldots,k$$

Step 4. Solve the equations

$$I[T_i\mathbf{x}^{\mathbf{m}}] = 0,\ \mathbf{m} \in \mathbb{N}^n, 0 \le |\ \mathbf{m}\ | \le k-1.$$

and

$$xT_i, yT_i \in \text{span}\{\mathrm{R}_0,\cdots,\mathrm{R}_{\mathrm{k+1}}\}$$

for γ_{ij}.

Step 5. Substitute the γ_{ij} into R_i and then find the common zeros $\mathbf{y}_s$ of $R_0,\cdots,R_{k+1}$.

Step 6. Use the zeros $\mathbf{y}_s$ as the cubature points and solve linear equations for the weights ω_s.

2.5 S-method

The S-method was suggested by Cools and Haegemans (1987)[80] in an attempt to find a method that is less dependent on the lower bound than the T-method.

Construct $R_i,\ i = 1,\ldots,k$ as (6). The polynomials R_i can be divided into two sets: $A := \{R_i : i \text{ is even. }\}$ and $B := \{R_i : i \text{ is odd. }\}$We demand that $A \subset \mathfrak{U}$ or $B \subset \mathfrak{U}$. We assign $C := A$ and $q := 0$ if we want to investigate the case $A \subset \mathfrak{U}$. We assign $C := B$ and $q := 1$ if we want to investigate the case $B \subset \mathfrak{U}$.

Let

$$S_i := y^2R_i - x^2R_{i+2},\ i = q, q+2,\ldots,k-1.$$

Assume two conditions:

- S_i be a polynomial of degree $k+1$ and be orthogonal to all polynomials of degree $k-2$.
- $S_i \in \text{span}(C)$.

The above two conditions lead to linear and quadratic equations in γ_{ij}. The inequality in Theorem 6.8 leads to inequalities for γ_{ij}.

In Cools and Haegemans (1988)[81], necessary and sufficient conditions are given for this method.

Then we use the same method to construct the cubature formulae as T-method.

3 Generalized Born Electrostatics

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 [76], 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 through a continuum representation (mean-field approximation) of the solvent [85]. 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 [102, 23], the non-polar solvation energies are of the form $G_{\text{cav}} + G_{\text{vdw}} = G^{(\text{rep})} + G^{(\text{rep})}$. In [32],$G_{(\text{rep})}$ 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 [99], the attractive van der Waals dispersion energy $G^{(\text{att})} = \sum_{i=1}^{M} G_i^{(\text{att})}$, where $G_i^{(\text{att})} = \rho_0 \int u_i^{(\text{att})}(\mathbf{x}_i, \mathbf{y})\theta(\mathbf{y})\ \mathrm{d}\mathbf{y}$, 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

$$G_{\text{cav}} + G_{\text{vdw}} = \sum_{i=1}^{M} \gamma_i A_i + pV + \rho_0 \int u_i^{(\text{att})}(\mathbf{x}_i, \mathbf{y})\theta(\mathbf{y})\ \mathrm{d}\mathbf{y}. \tag{7}$$

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

$$G_{\text{pol}} = \frac{1}{2} \int \phi_{\text{reaction}}(\mathbf{r})\rho(\mathbf{r})\ \mathrm{d}\mathbf{r}, \tag{8}$$

where $\phi_{\text{reaction}} = \phi_{\text{solvent}} - \phi_{\text{gas-phase}}$, $\phi(\mathbf{r})$ 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 $-\nabla(\varepsilon(\mathbf{x})\nabla\phi(\mathbf{x})) = \rho(\mathbf{x})$ for the electrostatic potential ϕ. Numerical

methods to solve the equation include the finite difference method [89, 63], finite element method [54, 6], and boundary element method [68]. However the PB methods are prohibitive for large molecules such as proteins due to the limited computational resources. As an alternative, (8) is approximated by a generalized Born (GB) model which is in the form of discrete sum [94]

$$G_{\text{pol}} = -\frac{\tau}{2}\sum_{i,j}\frac{q_i q_j}{\left[r_{ij}^2 + R_i R_j \exp(-\frac{r_{ij}^2}{4R_i R_j})\right]^{\frac{1}{2}}}, \tag{9}$$

where $\tau = \frac{1}{\varepsilon_p} - \frac{1}{\varepsilon_w}$, ε_p and ε_w 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

$$\mathbf{F}_\alpha^{\text{sol}} = -\frac{\partial G_{\text{sol}}}{\partial \mathbf{x}_\alpha}. \tag{10}$$

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 [72], Amber [22], Tinker [82], 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 [97] 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 [91]. 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). 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 3.1 we explain the geometric model that our energy and force computation are based on; we discuss in detail the energy computation in Section 3.2 and the force computation in Section 3.3; some implementation results are shown in Section 3.4; some details such as the fast summation algorithm and the NFFT algorithm are discussed in the appendix.

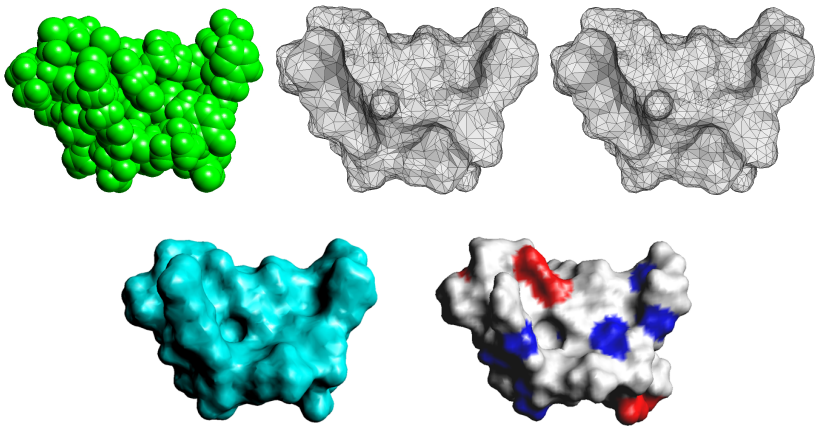


Figure 1: Top left: the discrete van der Waals surface model (436 atoms); top middle: the triangulation of the continuum Gaussian surface model with 6004 triangles; top right: the regularized triangular mesh where the quality of the elements is improved (making each as close as possible to an equilateral 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.

3.1 Geometric model

3.1.1 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: $f(\mathbf{x}) = e^{\beta(\frac{|\mathbf{x}-\mathbf{x}_i|^2}{r_i^2}-1)}$ 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:

$$f(\mathbf{x}) = \sum_{i=1}^{M} e^{\beta(\frac{|\mathbf{x}-\mathbf{x}_i|^2}{r_i^2}-1)} \tag{11}$$

where β is a parameter used to control the rate of decay of the Gaussian and known as the *blobbiness* of the Gaussian. In [84] $\beta = -2.3$, $isovalue = 1$ is indicated as a good approximation to the molecular surface.

3.1.2 Triangular mesh

The triangular mesh of the Gaussian surface is generated by using the dual contouring method [59, 109]. 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) [37]:

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

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.

3.1.3 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 4(f)) based on the triangular mesh to improve accuracy and efficiency [111]. 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).

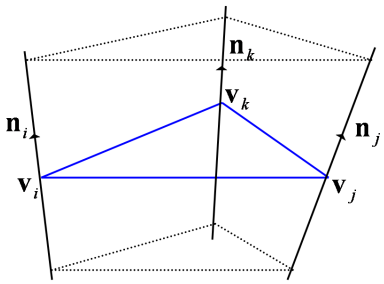


Figure 2: A prism D_{ijk} constructed with a triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$ as a basis.

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

$$F(b_1, b_2, b_3, \lambda) = \sum_{i+j+k=3} b_{ijk}(\lambda) B^3_{ijk}(b_1, b_2, b_3), \tag{12}$$

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 [111]. 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:

$$(x, \ y, \ z)^{\mathrm{T}} = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda). \tag{13}$$

We have proved in [111] 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.

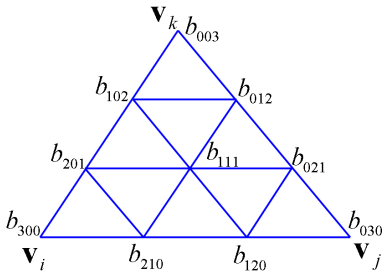


Figure 3: The control coefficients of the cubic Bernstein-Bezier basis of function F

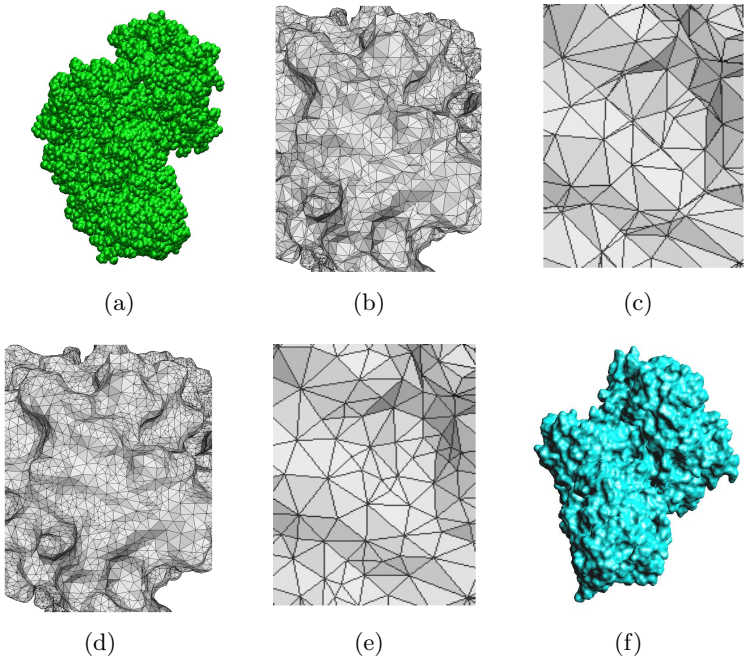


Figure 4: (a) is the discrete van der Waals model of protein 1BGX with 19,647 atoms; (b) and (c) are the zoom-in views of the the initial triangulation of the continuum surface with 85656 triangles; (d) and (e) are the zoom-in views of the quality improved mesh; (f) is the a continuum ASMS model generated based on the quality improved mesh.

3.2 Fast solvation energy computation

3.2.1 Method

Similarly to what is done for other GB models, we use (9) as the electrostatic solvation energy function. Before we compute (9), 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: 5). 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 (9) by using the fast multipole method (FMM) [44] 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 [33]. 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 [38] to compute the Born radius:

$$R_i^{-1} = \frac{1}{4\pi} \int_\Gamma \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}S \qquad i = 1, \ldots, M, \tag{14}$$

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.

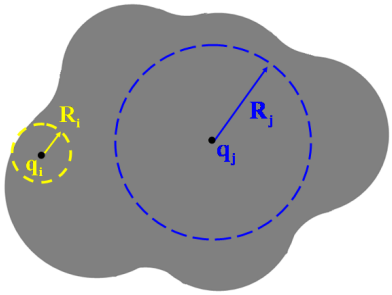
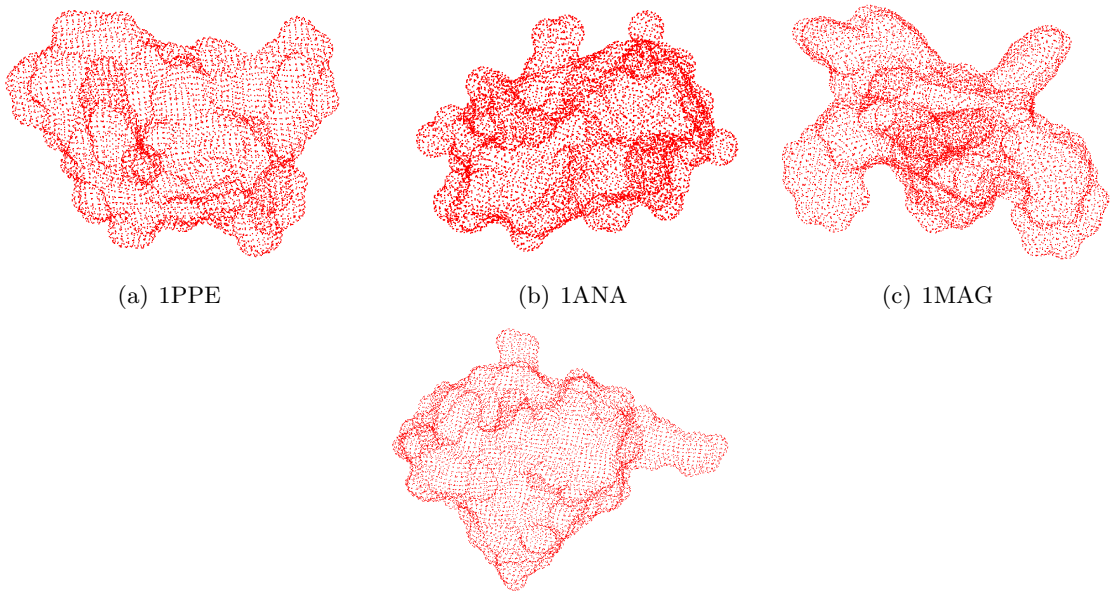


Figure 5: The effective Born radius reflects how deep a charge is buried inside the molecule. The Born radius of an atom is small if the atom is close to the surface of the molecule, otherwise the Born radius is large therefore has weaker interaction with the solvent.



(a) 1PPE (b) 1ANA (c) 1MAG

(d) 1CGI_l

Figure 6: 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.

Applying the Gaussian quadrature, We compute (14) numerically:

$$R_i^{-1} = \frac{1}{4\pi} \sum_{k=1}^{N} w_k \frac{(\mathbf{r}_k - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} \qquad i = 1, \ldots, M, \tag{15}$$

where w_k and $\mathbf{r}_k$ are the Gaussian integration weights and nodes on Γ (Figure 6). $\mathbf{r}_k$ are computed by mapping the Gaussian nodes of a master triangle to the algebraic patch via the transformation $\mathcal{T}$. 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 $\mathbf{r}_k = \mathcal{T}(\mathbf{r}_k^0)$ and $w_k = w_k^0|J(\mathcal{T})|$ where $|J(\mathcal{T})|$ is the Jacobian determinant of $\mathcal{T}$.

We formalize (15) in two steps. First we split it into two parts:

$$R_i^{-1} = \frac{1}{4\pi}\sum_{k=1}^{N}\frac{w_k\mathbf{r}_k\cdot\mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k-\mathbf{x}_i|^4} - \frac{1}{4\pi}\sum_{k=1}^{N}\frac{w_k\mathbf{x}_i\cdot\mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k-\mathbf{x}_i|^4}. \tag{16}$$

Then we split the second summation in (16) into three components:

$$\sum_{k=1}^{N}\frac{w_k\mathbf{x}_i\cdot\mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k-\mathbf{x}_i|^4} = x_i\sum_{k=1}^{N}\frac{w_kn_x^k}{|\mathbf{r}_k-\mathbf{x}_i|^4} + y_i\sum_{k=1}^{N}\frac{w_kn_y^k}{|\mathbf{r}_k-\mathbf{x}_i|^4} + z_i\sum_{k=1}^{N}\frac{w_kn_z^k}{|\mathbf{r}_k-\mathbf{x}_i|^4}. \tag{17}$$

The first summation in (16) and the three summations in (17) without the coefficients in front are of the common form:

$$G(\mathbf{x}_i) = \sum_{k=1}^{N} c_k g(\mathbf{x}_i-\mathbf{r}_k) \qquad i=1,\ldots,M, \tag{18}$$

with the kernel function $g(\mathbf{x}-\mathbf{r}_k) = \frac{1}{|\mathbf{x}-\mathbf{r}_k|^4}$ and the coefficient $c_k = w_k\mathbf{r}_k\cdot\mathbf{n}(\mathbf{r}_k)$, $w_kn_x^k$, $w_kn_y^k$, $w_kn_z^k$, respectively. (18) can be efficiently computed by using the fast summation algorithm introduced in [78] with complexity $O(M+N+n^3\log n)$, where n is a parameter used in the fast summation algorithm.

3.2.2 Fast summation

The fast summation algorithm is published in [78]. For convenience, we discuss this algorithm in this section briefly. The fast summation algorithm is often applied to compute the summations of the form

$$G(\mathbf{x}_i) = \sum_{k=1}^{N} c_k g(\mathbf{x}_i-\mathbf{r}_k), \qquad i=1,\ldots,M, \tag{19}$$

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:

$$g(\mathbf{x}-\mathbf{r}_k) = \sum_{\boldsymbol{\omega}\in I_\infty} g_{\boldsymbol{\omega}} e^{2\pi i\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{r}_k)}, \tag{20}$$

where $I_\infty := \{(\omega_1,\omega_2,\omega_3)\in\mathbb{Z}^3\}$ and $g_\omega = \int_\Pi g(\mathbf{x})e^{-2\pi\mathrm{i}\boldsymbol{\omega}\cdot\mathbf{x}}\,\mathrm{d}\mathbf{x}$. We approximate (20) by a truncated series:

$$g(\mathbf{x}-\mathbf{r}_k) \approx \sum_{\boldsymbol{\omega}\in I_n} g_{\boldsymbol{\omega}} e^{2\pi i(\mathbf{x}-\mathbf{r}_k)\cdot\boldsymbol{\omega}}, \tag{21}$$

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

$$g_{\boldsymbol{\omega}} = \frac{1}{n^3}\sum_{\mathbf{j}\in I_n} g(\frac{\mathbf{j}}{n})e^{-2\pi\mathrm{i}\boldsymbol{\omega}\cdot\mathbf{j}/n}, \qquad \boldsymbol{\omega}\in I_n. \tag{22}$$

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

Plugging (21) into (19), we get

$$\begin{aligned} G(\mathbf{x}_i) &\approx \sum_{k=1}^{N} c_k \left(\sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} e^{2\pi i (\mathbf{x}_i - \mathbf{r}_k).\boldsymbol{\omega}} \right) = \sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} \left(\sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{r}_k} \right) e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_i} \\ &= \sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} a_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_i} \end{aligned} \tag{23}$$

where

$$a_{\boldsymbol{\omega}} = \sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{r}_k}. \tag{24}$$

(23) is computed by using the NFFT algorithm with complexity $O(n^3 \log n + M)$ and (24) is computed by the NFFT$^{\mathrm{T}}$ algorithm with complexity $O(n^3 \log n + N)$. Hence the total complexity of computing (19) 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$^{\mathrm{T}}$ algorithm in Appendix 3.5 and 3.6, respectively.

3.2.3 Error analysis

The numerical analysis of the error introduced during the computation of (14) 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 (20) into finite terms, NFFT$^{\mathrm{T}}$ errors $E_{\boldsymbol{\omega}}$ when we compute the coefficients (24), and an NFFT error E_{NFFT} when we finally evaluate (23) by the NFFT algorithm.

Let I_i and $\tilde{I}_i$ denote the exact integration and the numerical output of (14) for atom i, respectively. Then We have

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

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

$$\|E\|_\infty \leq \|E_{\mathrm{Q}}\|_\infty + \|E_{\mathrm{FS}}\|_\infty + \|E_{\mathrm{NFFT}}\|_\infty + \|E_{\mathrm{NFFT^T}}\|_\infty. \tag{25}$$

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

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 (14) over Γ_e is

$$\begin{aligned} I_e &= \int_{\Gamma_e} \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}S \\ &= \iint_{\Omega_0} \frac{(\mathbf{r}(b_1, b_2) - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r}(b_1, b_2))}{|\mathbf{r}(b_1, b_2) - \mathbf{x}_i|^4} \, |J| \, \mathrm{d}b_1 \mathrm{d}b_2 \end{aligned} \tag{26}$$

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 (26). As we discuss in Appendix 3.7, $f(b_1, b_2) \in C^\infty(\Omega_0)$. Suppose we use an s-th order quadrature rule on element e, then

$$I_e = \iint_{\Omega_0} f(b_1, b_2) \, \mathrm{d}b_1 \, \mathrm{d}b_2 = \sum_{k=1}^{s_e} w_k f(b_1^k, b_2^k) + E. \tag{27}$$

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

$$f(b_1, b_2) = P_s(b_1, b_2) + R_s(b_1, b_2), \tag{28}$$

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

$$P_s(b_1, b_2) = f(b_1', b_2') + \frac{1}{s!}[(b_1 - b_1')\frac{\partial}{\partial b_1} + (b_2 - b_2')\frac{\partial}{\partial b_2}]^s f(b_1', b_2') \tag{29}$$

and the residue R_s is

$$R_s(b_1, b_2) = \frac{1}{(s+1)!}[(b_1 - b_1')\frac{\partial}{\partial b_1} + (b_2 - b_2')\frac{\partial}{\partial b_2}]^{s+1} f(b_1^*, b_2^*), \qquad (b_1^*, b_2^*) \in \Omega_0. \tag{30}$$

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

$$|R_s(b_1, b_2)| \le \frac{1}{(s+1)!}[|\frac{\partial}{\partial b_1}| + |\frac{\partial}{\partial b_2}|]^{s+1} f(b_1^*, b_2^*), \tag{31}$$

where $|\frac{\partial}{\partial b}|\cdot$ denotes $|\frac{\partial \cdot}{\partial b}|$. 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 (13), 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

$$|R_s(b_1, b_2)| \le \frac{(2h)^{s+1}}{(s+1)!}[|\frac{\partial}{\partial x}| + |\frac{\partial}{\partial y}| + |\frac{\partial}{\partial z}|]^{s+1} \tilde{f}(x^*, y^*, z^*) \le C\frac{(2h)^{s+1}}{(s+1)!} \tag{32}$$

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

$$|E| \le (\frac{1}{2} + s_e W_k)\frac{2^{s+1}}{(s+1)!} C h^{s+1}. \tag{33}$$

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 (33) 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 [30]. 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

$$|E| \le 2Ch^3. \tag{34}$$

Suppose there are N_e patches on Γ, then $|E_{\mathrm{Q}}| \le 2N_e C h^3$. So we have the same bound

$$\|E_{\mathrm{Q}}\|_\infty \le 2N_e C h^3. \tag{35}$$

Fast summation error According to the fast summation method described in Section 3.2.2, the Fourier series is truncated into a finite series

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

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

$$|E^i_{\mathrm{FS}}| \leq \|\mathbf{c}\|_\infty \sum_{k=1}^{N} |T^i_k| \tag{36}$$

where $\|\mathbf{c}\|_\infty := \max\limits_{k=1,\ldots,N} |c_k|$,

$$|T^i_k| = |\sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} g_{\boldsymbol{\omega}} e^{2\pi \mathrm{i} \boldsymbol{\omega} \cdot (\mathbf{x}_i - \mathbf{r}_k)}| \leq \sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} |g_{\boldsymbol{\omega}}| \tag{37}$$

with

$$g_{\boldsymbol{\omega}} = \int_\Pi g(\mathbf{x}) e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x} \tag{38}$$

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 3.2.2, Π is bounded and excludes 0. Let $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$. Then we rewrite $\sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} |g_{\boldsymbol{\omega}}|$ as

$$\sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} |g_{\boldsymbol{\omega}}| = \sum_{i,j,k=0,1} \sum_{\omega_1=n+1}^{\infty} \sum_{\omega_2=n+1}^{\infty} \sum_{\omega_3=n+1}^{\infty} |g_{(-1)^i \omega_1 \, (-1)^j \omega_2 \, (-1)^k \omega_3}|. \tag{39}$$

By successive integration by parts for each dimension, we get

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

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

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

Let $\mu_m = \int_\Pi |D^m g(\mathbf{x})| \, \mathrm{d}\mathbf{x}$. We obtain $|g_{\omega_1 \omega_2 \omega_3}| \leq \frac{\mu_m}{(2\pi)^m \omega_1^{m_1} \omega_2^{m_2} \omega_3^{m_3}}$. For the other terms in (39) we have the same upper bound. If we assume $m_1, m_2, m_3 \geq 2$, then

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

For $m_1 = m_2 = m_3$, we have

$$|T^i_k| \leq \frac{8\mu_6}{(2\pi)^6 n^3}. \tag{40}$$

Then for (37), we have

$$|E^i_{\mathrm{FS}}| \leq \|\mathbf{c}\|_\infty \frac{8\mu_6 N}{(2\pi)^6 n^3}. \tag{41}$$

In fact, the right hand side of (41) is independent of i. Therefore we get

$$\|E_{\mathrm{FS}}\|_\infty \leq \|\mathbf{c}\|_\infty \frac{8\mu_6 N}{(2\pi)^6 n^3}. \tag{42}$$

NFFT error The error analysis of the NFFT algorithm is thoroughly discussed at the end of Appendix 3.5. This error estimation is derived based on the analysis in [79]. In summary, the NFFT error is split into the *aliasing error* E^1_{NFFT} and the *truncation error* E^2_{NFFT} [79]:

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

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

$$\|E^1_{\mathrm{NFFT}}\|_\infty \leq \|\hat{G}\|_1 \max_{\boldsymbol{\omega}\in I_n} \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} |\frac{C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)}|, \tag{43}$$

$$\|E^2_{\mathrm{NFFT}}\|_\infty \leq \frac{1}{\sigma^3 n^3} \max_{\boldsymbol{\omega}\in I_n}(C^{-1}_{\boldsymbol{\omega}}(\xi))\|\hat{G}\|_1 \max_i \sum_{\mathbf{l}\in I_{\sigma n}} |\xi(\mathbf{x}_i - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}_i - \frac{\mathbf{l}}{\sigma n})|, \tag{44}$$

where ξ is a 1-periodic window function defined in Appendix 3.5, $C_{\boldsymbol{\omega}}(\xi)$ are the Fourier coefficients of ξ, and η is a truncated version of ξ. In the fast summation method (23), $\|\hat{G}\|_1 = \sum_{\boldsymbol{\omega}\in I_n} |g_{\boldsymbol{\omega}} a_{\boldsymbol{\omega}}|$, where $g_{\boldsymbol{\omega}}$ and $a_{\boldsymbol{\omega}}$ are defined in Section 3.2.2. Combining (43) and (44), one obtains

$$\|E_{\mathrm{NFFT}}\|_\infty \leq C(\xi, m, \sigma)\|\hat{G}\|_1. \tag{45}$$

In [78], 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 [12], $\xi(\mathbf{x}) = M_{2m}(\sigma n \mathbf{x})$, the coefficient $C(\xi, m, \sigma) = 4(\frac{1}{2\sigma-1})^{2m}$;
- powers of sinc function, $\xi(\mathbf{x}) = \frac{n(2\sigma-1)}{2m}\mathrm{sinc}^{2m}\left(\frac{(2\sigma-2)n\pi\mathbf{x}}{2m}\right)$, 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 [57]

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

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

NFFT$^{\mathrm{T}}$ error As we mentioned in Section 3.2.2, (24) is computed by the NFFT$^{\mathrm{T}}$ algorithm and then they are plugged in (23) for the following evaluation of the summation. So the NFFT error $E_{\mathrm{NFFT^T}}$ is

$$E_{\mathrm{NFFT^T}} = \sum_{\boldsymbol{\omega}\in I_n} g_{\boldsymbol{\omega}} E_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega}\cdot\mathbf{x}_i}, \tag{46}$$

where $E_{\boldsymbol{\omega}}$ denotes the error of the NFFT$^{\mathrm{T}}$ algorithm and $g_{\boldsymbol{\omega}}$ is the same as is defined in (38). Then we have

$$|E_{\mathrm{NFFT^T}}| \leq \sum_{\boldsymbol{\omega}\in I_n} |g_{\boldsymbol{\omega}} E_{\boldsymbol{\omega}}| \leq \max_{\boldsymbol{\omega}\in I_n}|E_{\boldsymbol{\omega}}| \sum_{\boldsymbol{\omega}\in I_n} |g_{\boldsymbol{\omega}}| = \|E_{\boldsymbol{\omega}}\|_\infty \|g\|_1 \tag{47}$$

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

As we discussed in Appendix 3.6, the NFFT$^{\mathrm{T}}$ error $E_{\boldsymbol{\omega}}$ is decomposed into the aliasing error ($E^1_{\boldsymbol{\omega}}$) and the truncation error ($E^2_{\boldsymbol{\omega}}$), $E_{\boldsymbol{\omega}} = E^1_{\boldsymbol{\omega}} + E^2_{\boldsymbol{\omega}}$. So

$$\|E_{\boldsymbol{\omega}}\|_\infty \leq \|E^1_{\boldsymbol{\omega}}\|_\infty + \|E^2_{\boldsymbol{\omega}}\|_\infty,$$

where $\|E^1_{\boldsymbol{\omega}}\|_\infty = \max_{\boldsymbol{\omega}\in I_n} |E^1_{\boldsymbol{\omega}}|$ and $\|E^2_{\boldsymbol{\omega}}\|_\infty = \max_{\boldsymbol{\omega}\in I_n} |E^2_{\boldsymbol{\omega}}|$. Based on the error bounds derived in Appendix 3.6,

$$\|E^1_{\boldsymbol{\omega}}\|_\infty \leq \|\mathbf{c}\|_1 \max_{\boldsymbol{\omega}\in I_n} \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \frac{C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)} \tag{48}$$

and

$$\|E^2_{\boldsymbol{\omega}}\|_\infty \leq \frac{1}{(\sigma n)^3}\|\mathbf{c}\|_1 \max_{\boldsymbol{\omega}\in I_n}(C^{-1}_{\boldsymbol{\omega}}(\xi)) \max_k \sum_{\mathbf{l}\in I_{\sigma n}} |\xi(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k) - \eta(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k)| \tag{49}$$

where $\|\mathbf{c}\|_1 = \sum_{k=1}^N |c_k|$. Comparing (48) with (43) and comparing (49) with (44) yield the error estimation of $E_{\boldsymbol{\omega}}$ which is similar to E_{NFFT}:

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

Hence

$$|E_{\mathrm{NFFT}^{\mathrm{T}}}| \leq C(\xi, m, \sigma)\|\mathbf{c}\|_1\|g\|_1. \tag{50}$$

The inequality (50) is independent of i, therefore,

$$\|E_{\mathrm{NFFT}^{\mathrm{T}}}\|_\infty \leq C(\xi, m, \sigma)\|\mathbf{c}\|_1\|g\|_1. \tag{51}$$

3.3 Fast solvation force computation

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

$$\mathbf{F}^{\mathrm{elec}}_\alpha = -\frac{\partial G_{\mathrm{sol}}}{\partial \mathbf{x}_\alpha}. \tag{52}$$

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

$$\frac{\partial G_{\mathrm{sol}}}{\partial \mathbf{x}_\alpha} = \frac{\partial}{\partial \mathbf{x}_\alpha}(G_{\mathrm{cav}} + G_{\mathrm{vdw}}) + \frac{\partial G_{\mathrm{pol}}}{\partial \mathbf{x}_\alpha} = \gamma\frac{\partial S_A}{\partial \mathbf{x}_\alpha} + \frac{\partial G_{\mathrm{pol}}}{\partial \mathbf{x}_\alpha}. \tag{53}$$

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, 56, 20]. To compute the polar force, we first define

$$G_{ij} = q_iq_j/(r^2_{ij} + R_iR_j\exp(-\frac{r^2_{ij}}{4R_iR_j}))^{1/2}. \tag{54}$$

Then

$$G_{\mathrm{pol}} = -\tau\sum_{i=1}^M\sum_{j=i+1}^M G_{ij} - \frac{\tau}{2}\sum_{i=1}^M G_{ii}. \tag{55}$$

Differentiating (55) w.r.t. $\mathbf{x}$, one gets

$$\frac{\partial G_{\mathrm{pol}}}{\partial \mathbf{x}_\alpha} = -\tau\sum_{i=1}^M\sum_{j=i+1}^M \frac{\partial G_{ij}}{\partial \mathbf{x}_\alpha} - \frac{\tau}{2}\sum_{i=1}^M \frac{\partial G_{ii}}{\partial \mathbf{x}_\alpha}, \tag{56}$$

where

$$\frac{\partial G_{ij}}{\partial \mathbf{x}_\alpha} = \frac{\partial G_{ij}}{\partial r_{ij}}\frac{\partial r_{ij}}{\mathbf{x}_\alpha} + \frac{\partial G_{ij}}{\partial R_i}\frac{\partial R_i}{\mathbf{x}_\alpha} + \frac{\partial G_{ij}}{\partial R_j}\frac{\partial R_j}{\mathbf{x}_\alpha}. \tag{57}$$

From (54), one can easily compute $\frac{\partial G_{ij}}{\partial r_{ij}}$ and $\frac{\partial G_{ij}}{\partial R_i}$, which are

$$\frac{\partial G_{ij}}{\partial r_{ij}} = q_iq_j\left(r^2_{ij} + R_iR_je^{-\frac{r^2_{ij}}{4R_iR_j}}\right)^{-\frac{3}{2}}\left(\frac{1}{4}e^{-\frac{r^2_{ij}}{4R_iR_j}} - 1\right)r_{ij},$$

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

$\frac{\partial r_{ij}}{\partial \mathbf{x}_\alpha}$ 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 (57) 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 (14). To solve this problem, we convert the integration domain back to the volume:

$$R_i^{-1} = \frac{1}{4\pi}\int_{\text{ex}} \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4}\, \mathrm{d}\mathbf{r}. \tag{58}$$

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 [42] where they first define a density function for each of the atoms

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

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

$$\varrho(\mathbf{r}) = \sum_i \chi_i - \sum_{i<j} \chi_i\chi_j + \sum_{i<j<k} \chi_i\chi_j\chi_k - \sum_{i<j<k<l} \chi_i\chi_j\chi_k\chi_l + \ldots. \tag{59}$$

There are some nice properties of this model. For example, the exterior region of the molecule is well characterized by $\varrho = 0$ 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:

$$\varrho_i(x) = \begin{cases} 1, & x \le a_i \\ \frac{2}{w^3}(x - a_i)^3 - \frac{3}{w^2}(x - a_i)^2 + 1, & a_i < x < a_i + w \\ 0, & x \ge a_i + w \end{cases} \tag{60}$$

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 $w = 1.4$ **r**A, atoms i and j are disconnected in the same sense as in the SES model iff $\varrho_i(\mathbf{r})\varrho_j(\mathbf{r}) = 0$, 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

$$\varrho(\mathbf{r}) = \sum_i \varrho_i - \sum_{i<j} \varrho_i\varrho_j + \sum_{i<j<k} \varrho_i\varrho_j\varrho_k - \sum_{i<j<k<l} \varrho_i\varrho_j\varrho_k\varrho_l. \tag{61}$$

We define the complementary function $\bar{\varrho} = 1 - \varrho$. 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 (58) can be rewritten as

$$R_i^{-1} = \frac{1}{4\pi}\int_{\mathbb{R}^3} \frac{\bar{\varrho}(\mathbf{r}, \{\mathbf{x}_j\})}{|\mathbf{r} - \mathbf{x}_i|^4}\, \mathrm{d}\mathbf{r}. \tag{62}$$

Differentiating both sides of (62), one gets

$$-\frac{1}{R_i^2}\frac{\partial R_i}{\partial \mathbf{x}_\alpha} = \frac{1}{4\pi}\int_{\mathbb{R}^3} \frac{\partial}{\partial \mathbf{x}_\alpha}\left(\frac{\bar{\varrho}(\mathbf{r}, \{\mathbf{x}_j\})}{|\mathbf{r} - \mathbf{x}_i|^4}\right)\, \mathrm{d}\mathbf{r}. \tag{63}$$

So

$$\frac{\partial R_i}{\partial \mathbf{x}_\alpha} = -\frac{R_i^2}{4\pi}\left(\int_{\mathbb{R}^3} \frac{\frac{\partial}{\partial \mathbf{x}_\alpha}\bar{\varrho}(\mathbf{r},\{\mathbf{x}_j\})}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r} + \int_{\mathrm{ex}} \frac{\partial}{\partial \mathbf{x}_\alpha}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r}\right). \tag{64}$$

For the first integral in (64),

$$\frac{\partial}{\partial \mathbf{x}_\alpha}\bar{\varrho} = -\frac{\partial}{\partial \mathbf{x}_\alpha}\varrho = -\frac{\partial \varrho_\alpha}{\partial \mathbf{x}_\alpha}\left(1 - \sum_j \varrho_j + \sum_{j<k}\varrho_j\varrho_k - \sum_{j<k<l}\varrho_j\varrho_k\varrho_l\right) = -\frac{\partial \varrho_\alpha}{\partial \mathbf{x}_\alpha} g_\alpha,$$

where j, k, l are the atoms overlapping with atom α, $g = 1 - \sum_j \varrho_j + \sum_{j<k}\varrho_j\varrho_k - \sum_{j<k<l}\varrho_j\varrho_k\varrho_l$, and

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

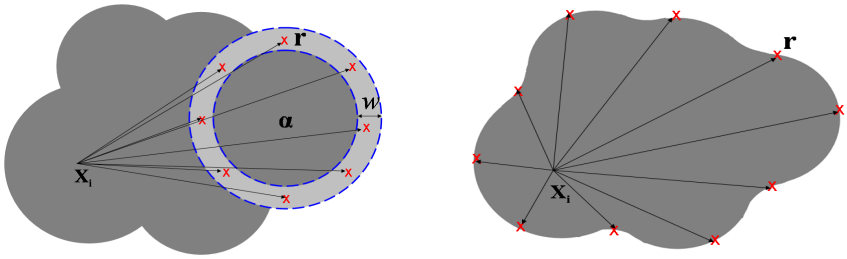
with $x = \|\mathbf{r} - \mathbf{x}_\alpha\|$. Noticing that $\frac{\partial \varrho_\alpha}{\partial \mathbf{x}_\alpha} \neq 0$ only if $a_\alpha < |\mathbf{r} - \mathbf{x}_\alpha| < a_\alpha + w$, the first integral in (64) is simplified as

$$\int_{|\mathbf{r}-\mathbf{x}_\alpha|=a_\alpha}^{|\mathbf{r}-\mathbf{x}_\alpha|=a_\alpha+w} -\frac{\partial \varrho_\alpha}{\partial \mathbf{x}_\alpha} g_\alpha(\mathbf{r}) \frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r}. \tag{65}$$

The integration domain of (65) is a regular spherical shell of the width w around atom α (Figure 7(a)). We switch to the spherical coordinate system:

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

where $(r,\theta,\phi) \in [0,w]\times[0,2\pi]\times[0,\pi]$. 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).



(a) shell (b) surface

Figure 7: When computing the derivatives of the Born radii $\frac{\partial R_i}{\partial \mathbf{x}_\alpha}$, 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.

The second integral in (64) is nonzero if $i \equiv \alpha$. In that case

$$\int_{\mathrm{ex}} \frac{\partial}{\partial \mathbf{x}_i}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r} = -\int_{\mathrm{ex}} \frac{\partial}{\partial \mathbf{r}}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r}. \tag{66}$$

We compute each component of (66) individually and convert to the surface integration (Figure: 7(b)) by the divergence theorem:

$$-\int_{\text{ex}} \frac{\partial}{\partial x}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r} = \int_{\Gamma} \frac{n_x(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}S \simeq \sum_{k=1}^{N} \frac{w_k n_x^k}{|\mathbf{r}_k-\mathbf{x}_i|^4}, \tag{67}$$

$$-\int_{\text{ex}} \frac{\partial}{\partial y}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r} = \int_{\Gamma} \frac{n_y(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}S \simeq \sum_{k=1}^{N} \frac{w_k n_y^k}{|\mathbf{r}_k-\mathbf{x}_i|^4}, \tag{68}$$

$$-\int_{\text{ex}} \frac{\partial}{\partial z}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r} = \int_{\Gamma} \frac{n_z(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}S \simeq \sum_{k=1}^{N} \frac{w_k n_z^k}{|\mathbf{r}_k-\mathbf{x}_i|^4}, \tag{69}$$

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 3.2. We compute (67), (68), and (69) 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 (66) 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 (65), 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 (65) for a fixed α for $i = 1, \ldots, M$ is $O(M)$. Moreover, since the integrand in (65) 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 (65) only if $d(i, \alpha) \le d_0$. Therefore the overall time complexity of computing (64) is $O(N + M + n^3 \log n)$.

3.4 Results

We compare the polarization energy computed based on the fast summation algorithm and the trivial summation in Table 2 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 8 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 3.2.3 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

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 2: 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 $|(G^A_{pol} - G^B_{pol})/G^B_{pol}|$.

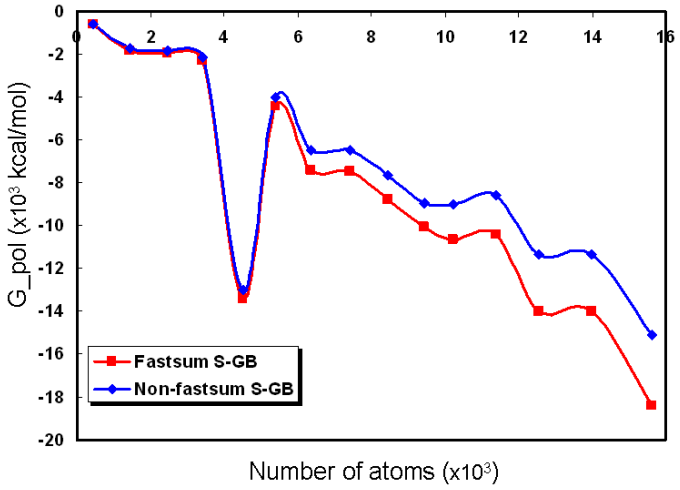
by multiplying the scaling factor, the error gets amplified. As we expect, computation time of the fastsum GB increases as M becomes large but is much faster than the traditional GB method.

In Figure 9, we compare $G_{\rm pol}$ 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.

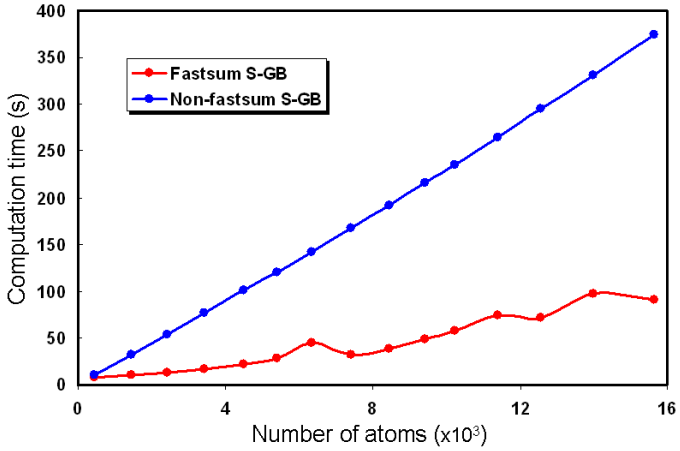
Protein ID	M	N	t_1 (s)	t_2 (s)	$T_{\rm total}$ (s)
1ANA	249	6,676	66.05	0.14	66.19
1MAG	544	7,328	69.58	0.23	69.81
1PPE_l	436	5,548	59.55	0.56	60.11
1CGI_l	852	6,792	68.71	3.27	71.98

Table 3: 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 (66) for $i = 1, \ldots, M$ and t_2 is the time for computing the rest of the terms in (57) for $i, j, \alpha = 1, \ldots, M$. $T_{\rm total}$ is the overall timing.

For the test proteins 1ANA, 1MAG, 1PPE_l, 1CGI_l, we compute the solvation force $\mathbf{F}^{\rm elec}_\alpha$, for $\alpha = 1, \ldots, M$. We show the timing results in Table 3. 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 10, 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.



(a)

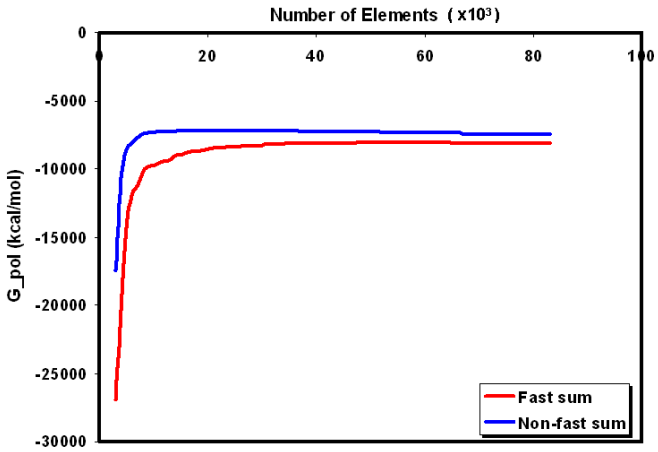


(b)

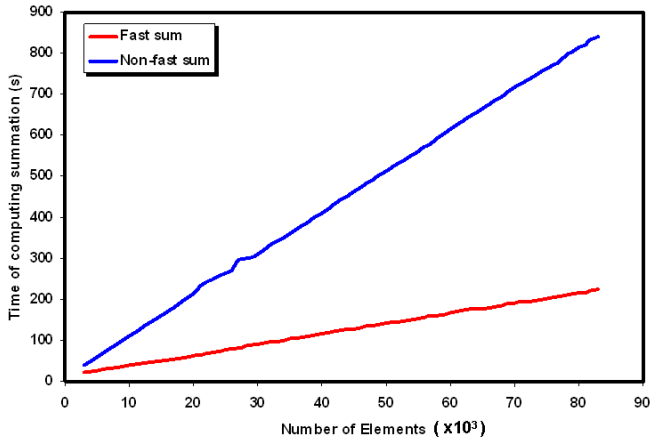
Figure 8: In (a) we compare G_{pol} computed by the fastsum GB and the non-fastsum GB for various proteins containing different number of atoms. In (b) we compare the computation time of the two methods.

3.4.1 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 [45] 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



(a)



(b)

Figure 9: 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.

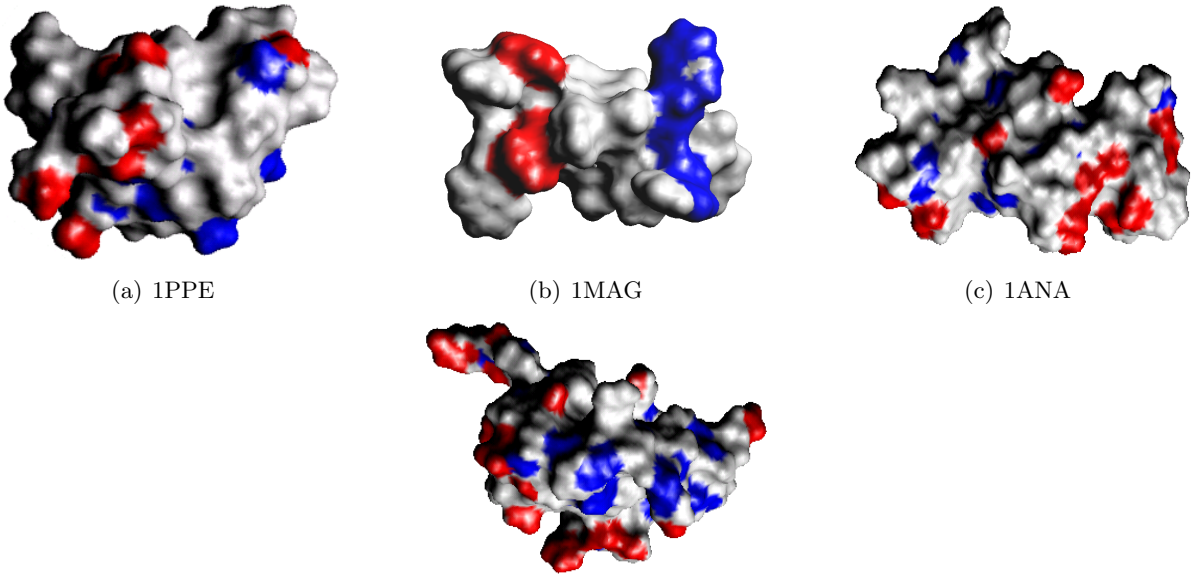
equation [96]. It is interesting to note that we can utilize a similar quadrature point generation via ASMS and the fast summation algorithm to speed up this GBr^{6}NL computation. In fact, by the divergence theorem, $\int_{\text{ex}} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^6}\, d\mathbf{r} = \frac{1}{3}\int_{\Gamma} \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^6}\, d\mathbf{r}$ and the rest follows similar to the methods in this paper.

3.5 NFFT

The NFFT [79] 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

$$G(\mathbf{x}_j) = \sum_{\boldsymbol{\omega}\in I_n} G_{\boldsymbol{\omega}} e^{2\pi \mathrm{i}\boldsymbol{\omega}\cdot\mathbf{x}_j} \qquad j = 1,\ldots,M, \tag{70}$$

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 (70) directly, one can approximate G by a function $s(\mathbf{x})$



(a) 1PPE (b) 1MAG (c) 1ANA

(d) 1CGI_l

Figure 10: 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.

which is a linear combination of the shifted 1-periodic kernel function ξ:

$$s(\mathbf{x}) := \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}}\xi(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}), \tag{71}$$

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 $\frac{\mathbf{l}}{\sigma n} := \{(\frac{l_1}{\sigma n}, \frac{l_2}{\sigma n}, \frac{l_3}{\sigma n})\}$. We have $\sigma > 1$ because of the error estimation discussed in Section 3.2.3.

The kernel function ξ is defined as

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

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

$$\xi(\mathbf{x}) = \sum_{\boldsymbol{\omega}\in\mathbb{Z}^3} C_{\boldsymbol{\omega}}(\xi)e^{2\pi i\boldsymbol{\omega}\cdot\mathbf{x}}, \tag{72}$$

with the Fourier coefficients

$$C_{\boldsymbol{\omega}}(\xi) := \int_{[-\frac{1}{2},\frac{1}{2}]^3} \xi(\mathbf{x})e^{-2\pi i\boldsymbol{\omega}\cdot\mathbf{x}}\,\mathrm{d}\mathbf{x} = \int_{\mathbb{R}^3} \xi_0(\mathbf{x})e^{-2\pi i\boldsymbol{\omega}\cdot\mathbf{x}}\,\mathrm{d}\mathbf{x} = \hat{\xi}_0(\boldsymbol{\omega}).$$

Cut off the higher frequencies in (72), one can get

$$\xi(\mathbf{x}) = \left(\sum_{\boldsymbol{\omega}\in I_{\sigma n}} + \sum_{\boldsymbol{\omega}\in\mathbb{Z}^3\setminus I_{\sigma n}}\right) C_{\boldsymbol{\omega}}(\xi)e^{2\pi i\boldsymbol{\omega}\cdot\mathbf{x}} \approx \sum_{\boldsymbol{\omega}\in I_{\sigma n}} C_{\boldsymbol{\omega}}(\xi)e^{2\pi i\boldsymbol{\omega}\cdot\mathbf{x}}. \tag{73}$$

Plug (73) into (71), we get

$$\begin{aligned} s(\mathbf{x}_j) &\approx \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}} \sum_{\boldsymbol{\omega}\in I_{\sigma n}} C_{\boldsymbol{\omega}}(\xi)e^{2\pi i\boldsymbol{\omega}\cdot(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n})} \\ &= \sum_{\boldsymbol{\omega}\in I_{\sigma n}} \tilde{G}_{\boldsymbol{\omega}} C_{\boldsymbol{\omega}}(\xi)e^{2\pi i\boldsymbol{\omega}\cdot\mathbf{x}_j}, \end{aligned} \tag{74}$$

with the coefficients

$$\tilde{G}_{\boldsymbol{\omega}} := \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}} e^{-2\pi \mathrm{i}\boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}}. \tag{75}$$

By defining

$$\tilde{G}_{\boldsymbol{\omega}} := \begin{cases} \frac{G_{\boldsymbol{\omega}}}{C_{\boldsymbol{\omega}}(\xi)} & \text{for } \boldsymbol{\omega}\in I_n, \\ 0 & \text{for } \boldsymbol{\omega}\in I_{\sigma n}\setminus I_n, \end{cases} \tag{76}$$

one can immediately get

$$s(\mathbf{x}_j) \approx \sum_{\boldsymbol{\omega}\in I_n} G_{\boldsymbol{\omega}} e^{2\pi \mathrm{i}\boldsymbol{\omega}\cdot\mathbf{x}_j} = G(\mathbf{x}_j). \tag{77}$$

The next problem is to compute $g_{\mathbf{l}}$. From (75), one can compute the coefficients $g_{\mathbf{l}}$ which are also coefficients in (77) by the discrete Fourier transform

$$g_{\mathbf{l}} = \frac{1}{\sigma^3 n^3} \sum_{\boldsymbol{\omega}\in I_{\sigma n}} \tilde{G}_{\boldsymbol{\omega}} e^{2\pi \mathrm{i}\boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}} = \frac{1}{\sigma^3 n^3} \sum_{\boldsymbol{\omega}\in I_n} \frac{G_{\boldsymbol{\omega}}}{C_{\boldsymbol{\omega}}(\xi)} e^{2\pi \mathrm{i}\boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}},\ \mathbf{l}\in I_{\sigma n}, \tag{78}$$

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 (77) by cutting off the tail of ξ. Define a function η_0:

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

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 (77), we obtain that

$$G(\mathbf{x}_j) \approx \sum_{\mathbf{l}\in I_{\sigma n,m}(\mathbf{x}_j)} g_{\mathbf{l}} \eta(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}), \tag{79}$$

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 (79). Therefore the complexity of evaluating (79) 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 3.1. *If we reorganize the above equations, it is not hard to see that, in fact,* (70) *is approximately computed by the expression*

$$G(\mathbf{x}_j) = \sum_{\boldsymbol{\omega}\in I_n} G_{\boldsymbol{\omega}} \left(\frac{1}{(\sigma n)^3 C_{\boldsymbol{\omega}}(\xi)} \sum_{\mathbf{l}\in I_{\sigma n}} \eta(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}) e^{2\pi i \boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}} \right). \tag{80}$$

From a linear algebra point of view, equation (80) *can be written as the product of a matrix and a vector. For example, for a one dimensional NFFT,* (80) *is equivalent to*

$$\mathbf{g} = \Xi\, F\, D\, \hat{\mathbf{g}} \tag{81}$$

with vectors

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

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

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

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 [79], in the first approximation (77), we see that s is equal to G after its higher frequencies in the Fourier series are cut off. Hence the error introduced in (77) which is known as the *aliasing error* is

$$\begin{aligned} E^1_{\mathrm{NFFT}} :&= \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}}\xi(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}) - G(\mathbf{x}_j) \\ &= \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \sum_{\boldsymbol{\omega}\in I_{\sigma n}} \tilde{G}_{\boldsymbol{\omega}+\mathbf{i}\sigma n} C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi) e^{2\pi\mathrm{i}(\boldsymbol{\omega}+\mathbf{i}\sigma n)\cdot\mathbf{x}_j}. \end{aligned} \tag{82}$$

Note that from (75), 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 (76), one obtains

$$|E^1_{\mathrm{NFFT}}| \le \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \sum_{\boldsymbol{\omega}\in I_n} |G_{\boldsymbol{\omega}} \frac{C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)}|. \tag{83}$$

Let $\|\hat{G}\|_1 = \sum\limits_{\boldsymbol{\omega}\in I_n} |G_{\boldsymbol{\omega}}|$. Then

$$|E^1_{\mathrm{NFFT}}| \le \|\hat{G}\|_1 \max_{\boldsymbol{\omega}\in I_n} \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} |\frac{C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)}|. \tag{84}$$

In the second approximation (79), since ξ is replaced by η, the so caused error, known as the *truncation error*, is

$$\begin{aligned} E^2_{\mathrm{NFFT}} :&= \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}}\xi(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}) - \sum_{\mathbf{l}\in I_{\sigma n,m}} g_{\mathbf{l}}\eta(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}) \\ &= \sum_{\mathbf{l}\in I_{\sigma n}\setminus I_{\sigma n,m}} g_{\mathbf{l}}[\xi(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x} - \frac{\mathbf{l}}{\sigma n})] \\ &= \sum_{\mathbf{l}\in I_{\sigma n}\setminus I_{\sigma n,m}} \frac{1}{\sigma^3 n^3} \sum_{\boldsymbol{\omega}\in I_n} \frac{G_{\boldsymbol{\omega}}}{C_{\boldsymbol{\omega}}(\xi)} e^{2\pi\mathrm{i}\boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}} [\xi(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x} - \frac{\mathbf{l}}{\sigma n})]. \end{aligned} \tag{85}$$

Thus

$$\begin{aligned} |E^2_{\mathrm{NFFT}}| &\le \frac{1}{\sigma^3 n^3} \sum_{\mathbf{l}\in I_{\sigma n}} |\frac{G_{\boldsymbol{\omega}}}{C_{\boldsymbol{\omega}}(\xi)} [\xi(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x} - \frac{\mathbf{l}}{\sigma n})]| \\ &\le \frac{1}{\sigma^3 n^3} \max_{\boldsymbol{\omega}\in I_n} (C^{-1}_{\boldsymbol{\omega}}(\xi)) \|\hat{G}\|_1 \sum_{\mathbf{l}\in I_{\sigma n}} |\xi(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x} - \frac{\mathbf{l}}{\sigma n})|. \end{aligned} \tag{86}$$

3.6 NFFT$^{\mathrm{T}}$

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

$$a(\boldsymbol{\omega}) = \sum_{k=1}^{N} c_k e^{-2\pi\mathrm{i}\boldsymbol{\omega}\cdot\mathbf{r}_k}, \qquad \boldsymbol{\omega}\in I_n. \tag{87}$$

Define a function

$$A(\mathbf{x}) := \sum_{k=1}^{N} c_k \xi(\mathbf{x} - \mathbf{r}_k), \tag{88}$$

where ξ is defined as same as in Appendix 3.5. The Fourier series of $A(\mathbf{x})$ is:

$$A(\mathbf{x}) = \sum_{\boldsymbol{\omega} \in \mathbb{Z}^3} C_{\boldsymbol{\omega}}(A) e^{2\pi \mathrm{i} \boldsymbol{\omega} \cdot \mathbf{x}}. \tag{89}$$

On the other hand,

$$\sum_{k=1}^{N} c_k \xi(\mathbf{x} - \mathbf{r}_k) = \sum_{k=1}^{N} c_k \sum_{\boldsymbol{\omega} \in \mathbb{Z}^3} C_{\boldsymbol{\omega}}(\xi) e^{2\pi i \boldsymbol{\omega} \cdot (\mathbf{x} - \mathbf{r}_k)}. \tag{90}$$

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

$$C_{\boldsymbol{\omega}}(A) = \sum_{k=1}^{N} c_k e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \mathbf{r}_k} C_{\boldsymbol{\omega}}(\xi), \qquad \boldsymbol{\omega} \in \mathbb{Z}^3. \tag{91}$$

Comparing (91) with (87) one obtains

$$a(\boldsymbol{\omega}) = \frac{C_{\boldsymbol{\omega}}(A)}{C_{\boldsymbol{\omega}}(\xi)}, \qquad \boldsymbol{\omega} \in I_n. \tag{92}$$

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

$$\begin{aligned} C_{\boldsymbol{\omega}}(A) &= \int_{[-\frac{1}{2}, \frac{1}{2}]^3} A(\mathbf{x}) e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x} \\ &= \int_{[-\frac{1}{2}, \frac{1}{2}]^3} \left(\sum_{k=1}^{N} c_k \xi(\mathbf{x} - \mathbf{r}_k) \right) e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x} \\ &= \sum_{k=1}^{N} c_k \int_{[-\frac{1}{2}, \frac{1}{2}]^3} \xi(\mathbf{x} - \mathbf{r}_k) e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x}. \end{aligned} \tag{93}$$

Discretizing the integration in (93) by the left rectangular rule leads to

$$a(\boldsymbol{\omega}) \approx \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{k=1}^{N} c_k \frac{1}{(\sigma n)^3} \sum_{\mathbf{l} \in I_{\sigma n}} \xi(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k) e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}}. \tag{94}$$

Replacing ξ with η yields

$$a(\boldsymbol{\omega}) \approx \frac{1}{(\sigma n)^3} \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{\mathbf{l} \in I_{\sigma n}} \hat{g}_{\mathbf{l}} e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}}, \tag{95}$$

where

$$\hat{g}_{\mathbf{l}} := \sum_{k=1}^{N} c_k \eta(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k), \qquad \mathbf{l} \in I_{\sigma n}. \tag{96}$$

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 $O(m^3 N)$. After computing $\hat{g}_{\mathbf{l}}$ one can easily evaluate (95) by the FFT algorithm at the complexity of $O(n^3 \log n)$. Lastly the complexity of computing (92) is $O(n^3)$. So the overall complexity of the NFFT$^{\mathrm{T}}$ algorithm is $O(m^3 N + n^3 \log n)$.

Remark 3.2. *Similar to the NFFT algorithm, we may write the one-line formula for computing* (87) *by the NFFT^T:*

$$a(\boldsymbol{\omega}) = \sum_{k=1}^{N} c_k \left(\frac{1}{(\sigma n)^3 C_{\boldsymbol{\omega}}(\xi)} \sum_{\mathbf{l} \in I_{\sigma n}} \eta(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k) e^{-2\pi i \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}} \right), \tag{97}$$

which in one dimension is equivalent to the linear system:

$$\hat{\mathbf{a}} = D^T F^* \, \Xi^T \mathbf{c} \tag{98}$$

with vectors

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

Matrix Ξ *is similar to that defined in Appendix 3.5*

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

F^* *is the conjugate transpose of the Fourier matrix* F*, and* D *is the same as that defined in Appendix 3.5. 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* $E^1_{\boldsymbol{\omega}}$ introduced in (94) and the *truncation error* $E^2_{\boldsymbol{\omega}}$ introduced in (95), $E_{\boldsymbol{\omega}} = E^1_{\boldsymbol{\omega}} + E^2_{\boldsymbol{\omega}}$, for $\boldsymbol{\omega} \in I_{\sigma n}$. By taking the Fourier expansion of ξ, we get form (94), so

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

Since

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

we have,

$$E^1_{\boldsymbol{\omega}} = a(\boldsymbol{\omega}) - \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{k=1}^{N} c_k \sum_{\mathbf{i} \in \mathbb{Z}^3} C_{\boldsymbol{\omega} + \mathbf{i} \sigma n}(\xi) e^{-2\pi \mathrm{i} \mathbf{r}_k \cdot (\boldsymbol{\omega} + \mathbf{i} \sigma n)}. \tag{99}$$

By (87),

$$E^1_{\boldsymbol{\omega}} = \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{k=1}^{N} c_k \sum_{\mathbf{i} \in \mathbb{Z}^3 \setminus \{0\}} C_{\boldsymbol{\omega} + \mathbf{i} \sigma n}(\xi) e^{-2\pi \mathrm{i} \mathbf{r}_k \cdot (\boldsymbol{\omega} + \mathbf{i} \sigma n)}. \tag{100}$$

Define $\|\mathbf{c}\|_1 = \sum_{k=1}^{N} |c_k|$. Then we have

$$|E^1_{\boldsymbol{\omega}}| \leq \|\mathbf{c}\|_1 \sum_{\mathbf{i} \in \mathbb{Z}^3 \setminus \{0\}} \frac{C_{\boldsymbol{\omega} + \mathbf{i} \sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)}. \tag{101}$$

In (95), the truncation error

$$E^2_{\boldsymbol{\omega}} = \sum_{k=1}^{N} c_k \left(\frac{1}{(\sigma n)^3 C_{\boldsymbol{\omega}}(\xi)} \sum_{\mathbf{l} \in I_{\sigma n}} [\xi(\mathbf{r}_k - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{r}_k - \frac{\mathbf{l}}{\sigma n})] e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}} \right), \tag{102}$$

which has the bound

$$|E^2_{\boldsymbol{\omega}}| \leq \frac{1}{(\sigma n)^3} \|\mathbf{c}\|_1 \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \max_k \sum_{\mathbf{l} \in I_{\sigma n}} |\xi(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k) - \eta(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k)|. \tag{103}$$

3.7 Continuity of f

As defined in Section 3.2.3,

$$f = \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4}, \tag{104}$$

where $\mathbf{r} \neq \mathbf{x}_i$ and $\mathbf{n} = \nabla F$ with F given in (12). $\mathbf{r}(b_1, b_2, \lambda)$ is simply defined in (13). In this appendix, we mainly discuss the continuity of $\mathbf{n}$. As derived in [111],

$$\nabla F = \mathcal{T}^{-1} \left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda} \right)^{\mathrm{T}} \tag{105}$$

where

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

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

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

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

$$\begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{xy} & F_{yy} & F_{yz} \\ F_{xz} & F_{yz} & F_{zz} \end{pmatrix} = \begin{pmatrix} \mathcal{T}_x & \mathcal{T}_y & \mathcal{T}_z \end{pmatrix} \begin{pmatrix} \nu & & \\ & \nu & \\ & & \nu \end{pmatrix} + \mathcal{T} M \mathcal{T}^{\mathrm{T}} \tag{106}$$

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

$$M = \begin{pmatrix} F_{b_1 b_1} & F_{b_1 b_2} & F_{b_1 \lambda} \\ F_{b_1 b_2} & F_{b_2 b_2} & F_{b_2 \lambda} \\ F_{b_1 \lambda} & F_{b_2 \lambda} & F_{\lambda\lambda} \end{pmatrix}. \tag{107}$$

To show $\mathcal{T}$ is differentiable, we take the first row of $\mathcal{T}$ and compute its derivative with respect to x, i.e. $(\frac{\partial^2 b_1}{\partial x^2} \ \frac{\partial^2 b_2}{\partial x^2} \ \frac{\partial^2 \lambda}{\partial x^2})$ as an example. We write (13) in the form of

$$\begin{cases} x = x(b_1, b_2, \lambda), \\ y = y(b_1, b_2, \lambda), \\ z = z(b_1, b_2, \lambda). \end{cases} \tag{108}$$

Taking the second derivatives of both sides of (108) with respect to x, we get

$$0 = C_f + \frac{\partial x}{\partial b_1} \frac{\partial^2 b_1}{\partial x^2} + \frac{\partial x}{\partial b_2} \frac{\partial^2 b_2}{\partial x^2} + \frac{\partial x}{\partial \lambda} \frac{\partial^2 \lambda}{\partial x^2}, \tag{109}$$

$$0 = C_g + \frac{\partial y}{\partial b_1} \frac{\partial^2 b_1}{\partial x^2} + \frac{\partial y}{\partial b_2} \frac{\partial^2 b_2}{\partial x^2} + \frac{\partial y}{\partial \lambda} \frac{\partial^2 \lambda}{\partial x^2}, \tag{110}$$

$$0 = C_h + \frac{\partial z}{\partial b_1} \frac{\partial^2 b_1}{\partial x^2} + \frac{\partial z}{\partial b_2} \frac{\partial^2 b_2}{\partial x^2} + \frac{\partial z}{\partial \lambda} \frac{\partial^2 \lambda}{\partial x^2}. \tag{111}$$

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

$$\begin{pmatrix} \frac{\partial^2 b_1}{\partial x^2} \\ \frac{\partial^2 b_2}{\partial x^2} \\ \frac{\partial^2 \lambda}{\partial x^2} \end{pmatrix} = \mathcal{T} \begin{pmatrix} -C_f \\ -C_g \\ -C_h \end{pmatrix}. \tag{112}$$

Using the same method, we can get the other rows of $\frac{\partial \mathcal{T}}{\partial x}$, matrices $\mathcal{T}_y$ and $\mathcal{T}_z$ by changing C_f, C_g, C_h in (112). Therefore $\mathcal{T}$ is differentiable. Similarly, we can compute the higher order derivatives of $\mathcal{T}$ and prove that $\mathcal{T} \in C^\infty$, thus prove $F \in C^\infty(\Omega_0)$, where Ω_0 defined in Section 3.2.3 is the canonical triangle. Therefore, as defined in (104), $f \in C^\infty(\Omega_0)$.

4 Poisson Boltzmann Electrostatics

Models of molecular potential energy are often used in biology to understand the structure-function relationships of proteins. Computation of molecular binding affinities and molecular dynamics [32, 75] involves repeated evaluation of molecular energy or forces as dynamic molecular configurations are simulated and analyzed. Electrostatic interactions of a molecule with an ionic solution are captured in the polarization term of the total potential energy. Since treating each solvent molecule discretely is extremely computationally expensive for a realistic number of molecules, a common and experimentally useful model for this polarization interaction is the Poisson-Boltzmann equation which treats the solvent as a continuous medium [34, 39].

Finite difference, finite element, and boundary element methods have all been used to solve the linearized Poisson-Boltzmann equation numerically [71]. Discretizing space with a regular lattice, the earliest solvers were based on finite difference methods [40, 74, 90]. Later finite difference approaches incorporated multigrid techniques [52, 55] and an alternate formulation [87] to improve efficiency. However, discontinuous coefficients and Dirac point charges often limit the accuracy of these methods.

Finite element methods eliminate some of these challenges by allowing the domain to be discretized with a more geometrically accurate mesh. Finite element methods have been developed and analyzed for the linearized [8, 18, 26, 27] and nonlinear Poisson-Boltzmann equation [7, 8, 24, 53]. Both finite difference and finite element methods require a discretization of three-dimensional space. If a uniform mesh of size h is used, then the number of degrees of freedom is $O(h^{-3})$. Boundary element methods provide an alternative in which all degrees of freedom lie on the molecular boundary and (for a uniform mesh) only $O(h^{-2})$ degrees of freedom are needed.

Zauhar and Morgan [106, 107, 108] formulated the linearized Poisson-Boltzmann equation as a system of boundary integral equations (the nonderivative boundary integral equations, nBIE) and solved this system numerically. The original system has been observed to exhibit poor conditioning

for iterative linear solvers [67], but an alternative formulation (the derivative boundary integral equations, dBIE) first stated by Juffer et al. [60] is well conditioned. Since the boundary element method leads to a dense linear system, and these and other [114] early methods suffer from need to compute this entire matrix.

Due to the special structure of the boundary element system, the fast multipole method [43] can be used to efficiently approximate the necessary matrix-vector products without creating the full matrix. This has been applied to several formulations of the molecular electrostatics problem: nBIE [1, 64, 69], dBIE [17, 70], models involving only Poisson's equation [13, 98] and a formulation involving only single layer densities [15]. Nearly all of these codes utilize the solvent-exposed surface produced by MSMS which is composed of spherical and toroidal patches but in some cases contains sharp corners, and some codes approximate this surface with a flat triangulation [17]. This can give hypersingular integrals which are challenging to discretize and a resulting solution error which is dominated by the geometric approximation.

For the linearized Poisson-Boltzmann equation we have designed and implemented a boundary element method and additionally studied its accuracy and efficiency on real protein structures. Our solver combines several key features which produce meaningful electrostatics calculations with modest surface mesh sizes. First, the dBIE formulation of the problem is used providing a well conditioned system for iterative methods in linear algebra. Second, by defining the molecular domain using the C^1 algebraic spline molecular surface, solutions only reflect a second order geometric error from the domain approximation and numerically problematic hypersingular integrals are avoided. Third, a general purpose fast multipole package, `KIFMM3d`, is used to efficiently approximate dense matrix computations simplifying the algorithm by separating the details of the fast multipole method from the rest of the scheme. Our freely available solver (`http://cvcweb.ices.utexas.edu/software`) is tested on a suite of actual proteins important in molecular docking. We show that our software outperforms several alternative approaches (the nonderivative boundary integral formulation and linear or nondifferentiable surface geometry) and demonstrate benefits compared to a finite difference solver. For practical examples, key parameters including singular and non-singular quadrature orders, fast multipole approximation order, and GMRES termination tolerance are tuned to greatly improve the method efficiency with minimal impact on the solution error.

Motivation for comuting the molecular polarization energy is contained in Section 4.1. In Section 4.2 the nonlinear and linearized Poisson-Boltzmann equations are stated and then the latter equation is formulated as a pair of boundary integral equations. Our numerical scheme for solving these equations is described in Section 4.3. Polarization energy is formulated as a post-processes to the Poisson-Boltzmann solution in Section 4.4. Sections 4.5 and 4.6 contain implementation details and computational experiments, respectively.

4.1 Motivation

We begin with a general outline of the molecular energetics problem including a description of the specific role of the polarization energy.

Molecular Potential Energy The total free energy of the system G is given be $G = U - TS$ where U is the potential energy, T is the temperature of the system, and S is the solute entropy. The potential energy of a molecule in solution is divided into two components: $U = E_{\text{MM}} + G_{\text{sol}}$, where E_{MM} is the molecular mechanical energy, and G_{sol} is the solvation energy. A common model for the molecular mechanical energy E_{MM} is given in [66].

For a molecule in solution, additional potential energy resulting from interaction of the solute and solvent is called the solvation energy G_{sol}. The solvation energy is often modeled by three terms:

$$G_{\text{sol}} = G_{\text{cav}} + G_{\text{vdw}} + G_{\text{pol}}, \tag{113}$$

where G_{cav} is the energy to form a cavity in the solvent, G_{vdw} is the van der Waals interaction energy between solute and solvent atoms, and the polarization energy G_{pol} is the electrostatic energy due to solvation [32, 39, 51, 89, 92].

Polarization Energy The polarization energy of a molecule occupying region Ω is the change in the electrostatic energy due to the induced polarization of the solvent,

$$G_{\text{pol}} = \frac{1}{2}\int_{\Omega} \phi_{\text{rxn}}(\mathbf{z})\rho(\mathbf{z})\ \mathbf{z}, \tag{114}$$

where $\rho(\mathbf{z})$ is the charge density at position $\mathbf{z}$ and the reaction electrostatic potential $\phi_{\text{rxn}}(\mathbf{z})$ indicates the change in electrostatic potential caused by solvation, i.e., $\phi_{\text{rxn}} = \phi_{\text{sol}} - \phi_{\text{gas}}$ where ϕ_{sol} and ϕ_{gas} are the potential of the molecular in solution and in a gas, respectively.

A number of applications involve the computation of polarization energy. For example, the binding effect of a drug (molecule 1) and its target (molecule 2) is the difference between the potential energy of the complex of the two molecules minus the sum of the potential energy of the individual molecules:

$$\Delta G_{\text{bind}} = G_{\text{complex}} - (G_{\text{molecule}_1} + G_{\text{molecule}_2}).$$

Polarization energy is an important component of each of these energy calculations.

Different theoretical approaches for computing binding solvation energy can be divided into two broad categories: explicit and implicit [32, 39, 77, 95]. Explicit solvent models adopt a atomistic treatment of both solvent and solute. Explicit approaches sample the solute-solvent space by molecular dynamics or Monte Carlo techniques which involve a large number of ions, water molecules, and molecular atoms [95]. This requires considerable computational effort and explicit solutions are often not practical especially for large domains [100].

Implicit solvent models treat the solvent as a featureless dielectric material and adopt a semi-microscopic representation of the solute. The effects of the solvent are modeled in terms of dielectric and ionic physical properties. The most widely used implicit model for molecular electrostatics is the Poisson-Boltzmann equation: it possesses a solid theoretical justification and has been explain a number of experimental observations [32, 34, 35, 86, 92, 100]. Since the solution to partial differential equations still requires substantial computational effort, several other implicit models have been developed to approximate results of the Poisson-Boltzmann model. The most common of these models is the Generalized Born formula [95, 5] which has also been used to successfully approximate polarization energy for some applications [2, 36].

4.2 The Poisson-Boltzmann Equation

A molecule is defined as a stable group of at least two atoms in a definite arrangement held together by very strong chemical covalent bonds. For a molecule embedded in an ionic solution, the domain ($\mathbb{R}^3$) is separated into open interior (Ω) and exterior regions ($\mathbb{R}^3 \setminus \Omega$) divided by the molecular surface $\Gamma = \overline{\Omega} \cap \overline{\mathbb{R}^3 \setminus \Omega}$ [65]; see Figure 11.

Two important coefficients, the dielectric coefficient $\epsilon(\mathbf{z})$ and the ion strength $\mathbb{I}(\mathbf{z})$, are assumed to be constant over Ω and $\mathbb{R}^3 \setminus \Omega$:

$$\epsilon(\mathbf{z}) = \begin{cases} \epsilon_I, & \mathbf{x} \in \Omega, \\ \epsilon_E, & \mathbf{x} \in \mathbb{R}^3 - \Omega, \end{cases} \quad \text{and} \quad \mathbb{I}(\mathbf{z}) = \begin{cases} 0, & \mathbf{z} \in \Omega, \\ \mathbb{I}, & \mathbf{z} \in \mathbb{R}^3 - \Omega. \end{cases}$$

The electrostatic potential in the interior and exterior of a molecule is governed by Poisson's equation,

$$\nabla(\epsilon(\mathbf{z})\nabla\phi(\mathbf{z})) = \rho(\mathbf{z}), \tag{115}$$

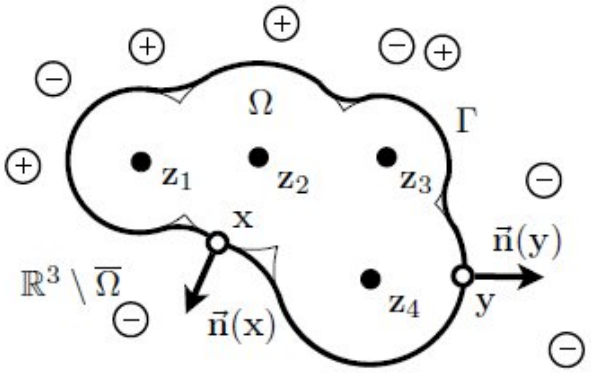


Figure 11: Molecular domain Ω for the boundary element formulation. Γ denotes the surface of molecular interior Ω. Atomic centers $\mathbf{z}_k$ are contained inside Ω while mobile ions in solution occur outside Ω. $\mathbf{x}$ and $\mathbf{y}$ are used to denote points on the molecular surface and the surface normal are denoted $\vec{\mathbf{n}}(\mathbf{x})$ and $\vec{\mathbf{n}}(\mathbf{y})$. In the discrete system, $\mathbf{x}$ is typically used to identify a collocation point while $\mathbf{y}$ usually represents a quadrature point.

where $\rho(\mathbf{z})$ is a variable charge density. This charge density contains two components: charged atoms belonging to the molecule itself and mobile ions as part of the solution. Atomic charges are assumed to be Dirac distributions while mobile ions in solution are modeled with the Boltzmann distribution,

$$\rho(\mathbf{z}) := \rho_c(\mathbf{z}) + \rho_b(\mathbf{z}) = -4\pi \sum_{k=1}^{n_c} \frac{q_k}{\epsilon_I} \delta(\mathbf{z} - \mathbf{z}_k) + \lambda(\mathbf{z}) \sum_i e_c z_i c_i e^{-e_c z_i \phi(\mathbf{z})/k_B T}. \tag{116}$$

Since $\rho(\mathbf{z})$ depends on ϕ, (115) is the nonlinear Poisson-Bolzmann equation rather than merely Poisson's equation. Definitions of each of the parameters in (116) and a few other parameters needed for the linearized version are given below.

$\epsilon(\mathbf{z})$	dielectric coefficient at $\mathbf{z}$
q_k	charge of the atom k
$\mathbf{z}_k$	location of charge q_k
n_c	number of point charges
$\lambda(\mathbf{z})$	characteristic function of the set $\mathbb{R}^3 \setminus \Omega$
e_c	charge of an electron
k_B	Boltzmann's constant
T	absolute temperature
$\mathbb{I} = \frac{1}{2}\sum_i c_i z_i^2$	ionic strength
c_i, z_i	concentration and charge of i^{th} ionic species
$\bar{\kappa}(\mathbf{z}) = \sqrt{\frac{8\pi e_c^2 \mathbb{I}(\mathbf{z})}{k_B T}}$	modified Debye-Huckel parameter

Selecting a linear approximation to the nonlinear term ρ_b produces the linearized Poisson-Boltzmann equation,

$$\nabla(\epsilon(\mathbf{z})\nabla\phi(\mathbf{z})) = \rho_c(\mathbf{z}) + \rho_b^L(\mathbf{z}), \tag{117}$$

where $\rho_b^L(\mathbf{z}) = \bar{\kappa}^2(\mathbf{z})\phi(\mathbf{z})$ is the first term of the Taylor expansion of $\rho_b(\mathbf{z})$. In many cases the linearized Poisson-Boltzmann equation provides a sufficiently accurate approximation of the nonlinear Poisson-Boltzmann equation; see [35] and references therein.

4.2.1 Boundary Integral Formulation

Potential theory [61, 93] provides the tools needed to derive a boundary integral formulation of the linearized Poisson-Boltzmann equation. We begin by separating (117) into the interior and exterior regions and explicitly stating interface conditions which must hold on molecular boundary Γ:

$$\nabla \left(\epsilon_I \nabla \phi(\mathbf{z})\right) = -\sum_{k=1}^{n_c} q_k \delta(\mathbf{z} - \mathbf{z}_k) \qquad \mathbf{z} \in \Omega, \tag{118}$$

$$\nabla \left(\epsilon_E \nabla \phi(\mathbf{w})\right) = \bar{\kappa}^2 \phi(\mathbf{w}) \qquad \mathbf{w} \in \mathbb{R}^3 \setminus \overline{\Omega}, \tag{119}$$

$$\phi(\mathbf{z})\big|_{\mathbf{z}=\mathbf{x}} = \phi(\mathbf{w})\big|_{\mathbf{w}=\mathbf{x}} \qquad \mathbf{x} \in \Gamma, \tag{120}$$

$$\frac{\partial \phi}{\partial \vec{\mathbf{n}}}(\mathbf{z})\bigg|_{\mathbf{z}=\mathbf{x}} = \frac{\epsilon_E}{\epsilon_I} \frac{\partial \phi}{\partial \vec{\mathbf{n}}}(\mathbf{w})\bigg|_{\mathbf{w}=\mathbf{x}} \qquad \mathbf{x} \in \Gamma. \tag{121}$$

Carefully applying Green's second identity to the interior and exterior regions and taking limits approaching Γ yields the boundary integral equations,

$$\frac{1}{2}\phi(\mathbf{x}) + \int_\Gamma \left[\frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})}\phi(\mathbf{y}) - G_0(\mathbf{x},\mathbf{y})\frac{\partial \phi}{\partial \vec{\mathbf{n}}}(\mathbf{y})\right] \dot{\mathbf{y}} = \sum_{k=1}^{n_c} \frac{q_k}{\epsilon_I} G_0(\mathbf{x},\mathbf{z}_k), \tag{122}$$

$$\frac{1}{2}\phi(\mathbf{x}) + \int_\Gamma \left[\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})}\phi(\mathbf{y}) - \frac{\epsilon_I}{\epsilon_E} G_\kappa(\mathbf{x},\mathbf{y})\frac{\partial \phi}{\partial \vec{\mathbf{n}}}(\mathbf{y})\right] \dot{\mathbf{y}} = 0, \tag{123}$$

where G_0 and G_κ denote the fundamental solutions of the Poisson-Boltzmann equations,

$$G_0(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi\,||\mathbf{x}-\mathbf{y}||}, \qquad \text{and} \qquad G_\kappa(\mathbf{x},\mathbf{y}) = \frac{e^{-\kappa||\mathbf{x}-\mathbf{y}||}}{4\pi\,||\mathbf{x}-\mathbf{y}||}.$$

Recall Figure 11 for an example domain including normal vectors at labeled boundary points $\mathbf{x}$ and $\mathbf{y}$.

An alternative boundary element formulation of the linearized Poisson-Boltzmann equation was proposed by Juffer et al. [60]. This system (dBIE) is produced by taking linear combinations of the original boundary integral equations and their derivatives.

$$\begin{aligned}\frac{1}{2}\left(1+\frac{\epsilon_E}{\epsilon_I}\right)\phi(\mathbf{x}) + \int_\Gamma \left(\frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} - \frac{\epsilon_E}{\epsilon_I}\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})}\right)\phi(\mathbf{y})\dot{\mathbf{y}} \\ - \int_\Gamma \left(G_0(\mathbf{x},\mathbf{y}) - G_\kappa(\mathbf{x},\mathbf{y})\right)\frac{\partial \phi(\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})}\dot{\mathbf{y}} = \sum_{k=1}^{n_c}\frac{q_k}{\epsilon_I}G_0(\mathbf{x},\mathbf{z}_k),\end{aligned} \tag{124}$$

$$\begin{aligned}\frac{1}{2}\left(1+\frac{\epsilon_I}{\epsilon_E}\right)\frac{\partial \phi(\mathbf{x})}{\partial \vec{\mathbf{n}}(\mathbf{x})} + \int_\Gamma \left(\frac{\partial^2 G_0(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{x})\partial \vec{\mathbf{n}}(\mathbf{y})} - \frac{\partial^2 G_\kappa(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{x})\partial \vec{\mathbf{n}}(\mathbf{y})}\right)\phi(\mathbf{y})\dot{\mathbf{y}} \\ - \int_\Gamma \left(\frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{x})} - \frac{\epsilon_I}{\epsilon_E}\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{x})}\right)\frac{\partial \phi(\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} = \sum_{k=1}^{n_c}\frac{q_k}{\epsilon_I}\frac{\partial G_0(\mathbf{x},\mathbf{z}_k)}{\partial \vec{\mathbf{n}}(\mathbf{x})}\dot{\mathbf{y}}.\end{aligned} \tag{125}$$

This combination of the derivatives of (122) and (123) has been selected so the kernel $\frac{\partial^2 G_0(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{x})\partial \vec{\mathbf{n}}(\mathbf{y})} - \frac{\partial^2 G_\kappa(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{x})\partial \vec{\mathbf{n}}(\mathbf{y})}$ in (125) is not hypersingular. For certain numerical schemes, this reformulation has been observed to produce a better well-conditioned linear system and fast convergence of iterative linear solvers when compared to the original boundary integral equations [67].

4.2.2 Discretization by the Collocation Method

The boundary integral equations (either dBIE or nBIE) are discretized by selecting a finite-dimensional function space and a set of collocation points. Each unknown function is required

to belong to the selected function space and the integral equations are required to hold exactly at the collocation points. The most commonly selected pairs of function spaces and collocation points are piecewise constant functions with triangle centroid collocation points and piecewise linear functions with mesh vertex collocation points. Let $\{\psi_i\}_{i=1}^{n_d}$ be a basis for the finite-dimensional function space, i.e., $\phi(\mathbf{x}) = \sum_{i=1}^{n_d} \phi_i \psi_i(\mathbf{x})$ and $\frac{\partial \phi}{\partial \vec{\mathbf{n}}}(\mathbf{x}) = \sum_{i=1}^{n_d} \partial\phi_i \psi_i(\mathbf{x})$, and let $\mathbf{x}_i$ denote the collocation points. Then the nBIE formulation becomes a linear system of equations,

$$\begin{aligned} \frac{1}{2}\sum_{j=1}^{n_d} \phi_j\psi_j(\mathbf{x}_i) + \int_\Gamma \frac{\partial G_0(\mathbf{x}_i,\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} \sum_{j=1}^{n_d} \phi_j\psi_j(\mathbf{y})\dot{\mathbf{y}} \\ - \int_\Gamma G_0(\mathbf{x}_i,\mathbf{y}) \sum_{j=1}^{n_d} \partial\phi_j\psi_j(\mathbf{y})\dot{\mathbf{y}} = \sum_{k=1}^{n_c} \frac{q_k}{\epsilon_I} G_0(\mathbf{x}_i,\mathbf{z}_k), \end{aligned} \qquad i = 1..n_d, \tag{126}$$

$$\begin{aligned} \frac{1}{2}\sum_{j=1}^{n_d} \phi_j\psi_j(\mathbf{x}_i) + \int_\Gamma \frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} \sum_{j=1}^{n_d} \phi_j\psi_j(\mathbf{y})\dot{\mathbf{y}} \\ - \int_\Gamma \frac{\epsilon_I}{\epsilon_E} G_\kappa(\mathbf{x},\mathbf{y}) \sum_{j=1}^{n_d} \partial\phi_j\psi_j(\mathbf{y})\dot{\mathbf{y}} = 0, \end{aligned} \qquad i = 1..n_d. \tag{127}$$

A similar system can be derived for the dBIE system. Solving this dense linear system (for unknowns ϕ_i and $\partial\phi_i$) involves a number of complications and simplifications. We briefly outlines the general issues here and in the next section describe our specific approaches as applied to realistic proteins.

The integrals in (126) and (127) must be discretized by some quadrature rules, but the singular kernels prevent the use of a fixed quadrature rule over a triangulation (or similar discretization) of the boundary. For a boundary subdivided into patches $\{\Gamma_b\}_{b=1}^{n_b}$, the integral is usually broken into three parts: nonsingular, nearly singular and singular components. A different quadrature rule is used for each type of boundary patch based on which component of the integral it belongs to. The singular and non-singular integrals are usually performed only in a small neighborhood of the singulatity $\mathbf{x}_i$. The remaining integrals are evaluated using a fixed nonsingular quadrature rule and due to the rapid decay of the kernels, the simultaneous computation of these integrals for each collocation point can be accelerated with the fast multipole method [43]. For example if the first integral in (126) is discretized using a quadrature rule $\{(\mathbf{y}_q, w_w)\}_{q=1}^{n_q}$ then the resulting summations,

$$\sum_{q=1}^{n_q} \frac{\partial G_0(\mathbf{x}_i,\mathbf{y}_q)}{\partial \vec{\mathbf{n}}(\mathbf{y}_q)} w_q \sum_{j=1}^{n_d} \phi_j\psi_j(\mathbf{y}_q), \qquad i = 1..n_d, \tag{128}$$

can be accurately approximated via the fast multipole in $O(\max(n_q, n_d))$ operations assuming that the support of each basis function intersects a bounded number of boundary patches, i.e., the sum in j in (128) involves a bounded number of terms.

Following the fast multipole calculation, each of the values is corrected to include accurate singular and nearly singular quadrature rules for the appropriate boundary patches. Singular integration is usually performed with by quadrature rules tailored to the position of the singularity [29, 46, 58] while nearly singular integration usually involves (possibly adaptive) refinement of the boundary patches [16, 46, 58, 88]. In some cases singular and nearly singular integration has been studied with respect to certain specific surfaces associated with the linearized Poisson-Boltzmann equation [10, 101].

4.3 BEM for Molecular Surfaces

We describe the details of our boundary element method: how the molecular surface is defined and discretized, what basis functions are selected and how quadrature is performed.

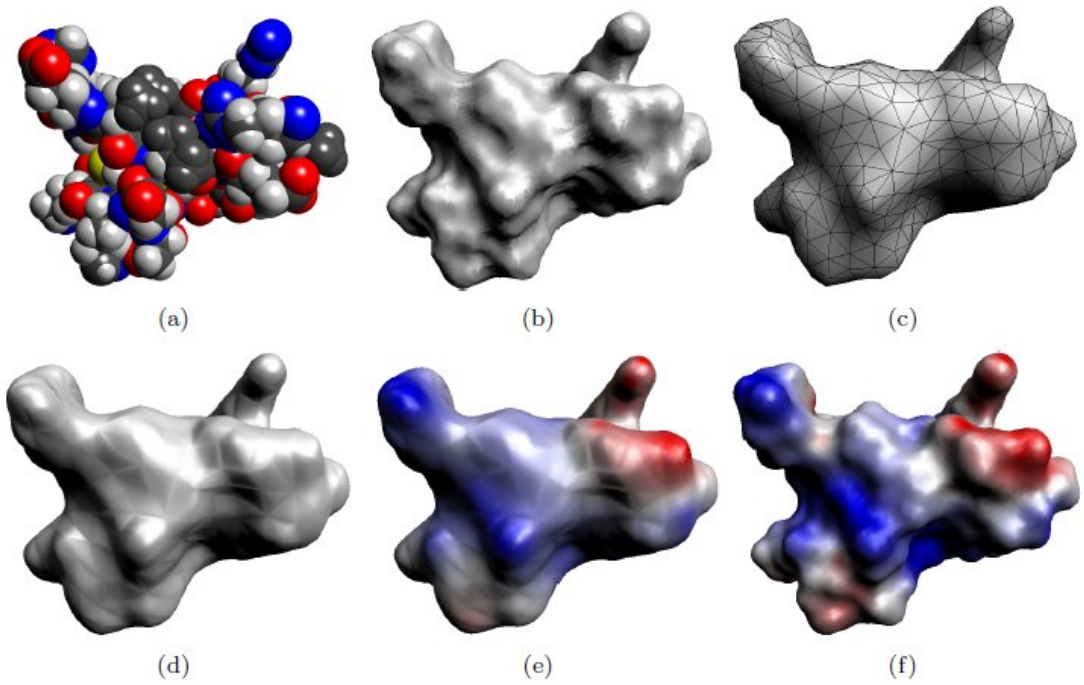


Figure 12: Molecular model of a protein (PDB id:1PPE, 436 atoms). (a) The van der Waals surface of the protein which models the molecule as a union of balls. (b) The variational molecular surface gives a smooth approximation of the van der Waals surface. (c) The variational surface is then triangulated and then decimated to produce a smaller mesh. This decimated mesh contains 1,000 triangles. (d) The algebraic spline molecular surface (ASMS) fits a smooth surface over the triangular mesh. (e) Electrostatic potential computed using the 1,000 patch ASMS. (f) Electrostatic potential using an ASMS with 74,812 patches. The surfaces in (e) and (f) are colored by the electrostatic potential, ranging from $-3.8\ k_bT/e_c$ (red) to $+3.8\ k_bT/e_c$ (blue).

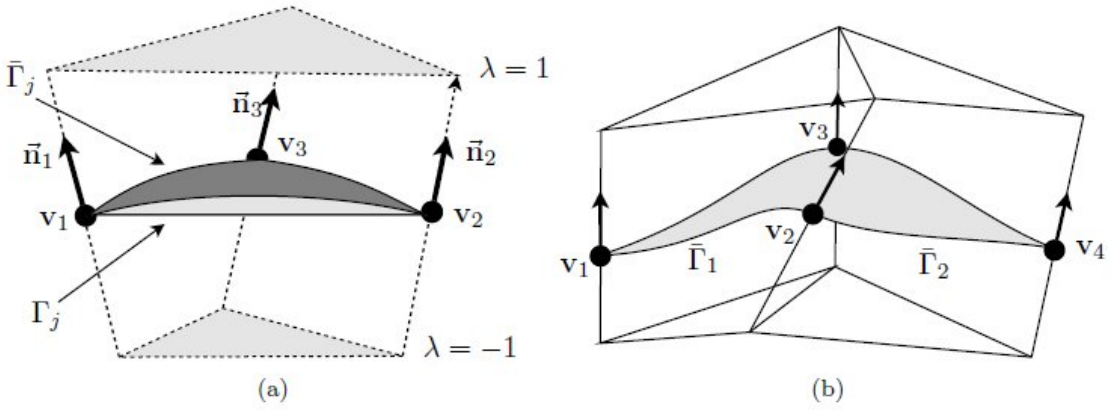


Figure 13: (a) A single prismatic scaffold region for the triangle with vertices $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ and associated surface normals $\vec{\mathbf{n}}_1$, $\vec{\mathbf{n}}_2$, and $\vec{\mathbf{n}}_3$. The surface patch $\bar{\Gamma}_i$ interpolates these normals. (b) The ASMS is smooth between two scaffold patches Γ_1 and Γ_2.

4.3.1 Construction of the Molecular Surface

To define the molecular surface Γ, we begin with an experimentally derived protein structure from the RCSB Protein Data Bank (PDB) [11], a worldwide data repository containing thousands of large bio-molecules. Each PDB structure contains of list of spacial locations for each of the atoms in a molecule. The molecular model for electrostatic calculations is obtained from a PDB file by assigning charge and radius parameters derived from a variety of force fields, e.g., AMBER [77], CHARMM [19], etc. For example, the adaptive Poisson-Boltzmann solver, `APBS`, applies the all-atom AMBER 99 force field [28].

From a configuration of atomic positions and radii a molecular surface can be defined. The simplest surfaces, the van der Waals and solvent accessible surfaces, are merely the boundary of a union of balls [65]; see Figure 12(a). Alternatively the solvent excluded surface [25, 83] is defined to be the boundary of the region outside this union of balls which is accessible by a probe sphere. The solvent excluded surface eliminates many, but not all, of the non-differentiable cusps which occur in the union-of-balls surfaces. For a smooth surface, the level-set of a sum of Gaussian functions associated with each atom is often considered; see [14, 41], for example.

We utilize the molecular surface constructed in; the surface is generated by constructing a Gaussian density function for the atom based on atomic positions and radii, evolving this function according to a variational formulation and then considering a level-set of this function; see Figure 12(b). For the resulting surface, a triangular mesh with surface normal vectors at the vertices is constructed using a dual contouring method [110]. If the surface mesh generated contains too many triangles, it is decimated following the approach in [4] and further mesh smoothing is performed as necessary; see Figure 12(c).

4.3.2 Surface Parametrization

To provide a smooth surface which interpolates mesh vertices and prescribed surface normals, we utilize the algebraic spline molecular surface (ASMS) [112]. This surface is constructed from algebraic patches or A-patches which are a kind of low degree algebraic surface with dual implicit and rational parametric representations [3]. The result is a molecular surface depicted in Figure 12(d) which can be parametrized in terms of the barycentric coordinates of the triangles allowing for easy construction of basis functions as described in the next section. We give a brief overview of this construction; complete details can be found in [112, 113].

For some triangle element Γ_j with vertices $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ and normals $\vec{\mathbf{n}}_1$, $\vec{\mathbf{n}}_2$ and $\vec{\mathbf{n}}_3$, the A-patch $\bar{\Gamma}_j$ is defined on this prism,

$$D(\Gamma_j) := \{\mathbf{y} \ : \ \mathbf{y} = b_1\mathbf{v}_1(\lambda) + b_2\mathbf{v}_2(\lambda) + b_3\mathbf{v}_3(\lambda), \ -1 \le \lambda \le 1\},$$

where $\mathbf{v}_i(\lambda) = \mathbf{v}_i + \lambda\vec{\mathbf{n}}_i$ and (b_1, b_2, b_3) are the barycentric coordinates of the triangle; see Figure 13. We define a function over the prism $D(\Gamma_j)$ in Benstein-Bezier spline form by

$$F_d(b_1, b_2, b_3, \lambda) = \sum_{i+j+k=d} b_{ijk}(\lambda) B^d_{ijk}(b_1, b_2, b_3),$$

where $B^d_{ijk}(b_1, b_2, b_3) = \frac{d!}{i!j!k!} b_1^i b_2^j b_3^k$. For $d \ge 3$ coefficients $b_{ijk}(\lambda)$ can be selected so that F_d is continuous between adjacent patches and for each vertex $F_d(\mathbf{v}_i) = 0$ and $\nabla F_d(\mathbf{v}_i) = \vec{\mathbf{n}}_i$.

The molecular surface $\bar{\Gamma}_j$ is the zero level-set of F_d,

$$\bar{\Gamma}_j = \{\mathbf{y} \ : \ \mathbf{y} = b_1\mathbf{v}_1(\lambda) + b_2\mathbf{v}_2(\lambda) + b_3\mathbf{v}_3(\lambda), \ F_d(b_1, b_2, b_3, \lambda) = 0\}. \tag{129}$$

This can be viewed as a parametric representation in two parameters b_1 and b_2. The third barycentric coordinate can be computed from the first two, $b_3 = 1 - b_1 - b_2$ and under some mild restrictions on the mesh shape and vertex normals, $F_d(b_1, b_2, b_3, \lambda) = 0$ can be solved for λ in terms of b_1 and b_2. In practice this nonlinear equation is solved numerically with Newton's method.

4.3.3 Selection of Basis Functions

We consider two different types of basis functions for the solution space and associated collocation points: piecewise constant basis functions with triangle centroids as collocation points and piecewise linear basis functions with mesh vertices as collocation points. In both cases these functions are defined based on the barycentric coordinates of an underlying triangular mesh. Since the A-patches can be parametrized by the barycentric coordinates, this construction can be directly applied to the ASMS.

4.3.4 Quadrature

Let $\{(\mathbf{b}_q, w_q)\}_{q=1}^{n_q}$ be a (generic) quadrature rule for a reference triangle T where $\mathbf{b}_q$ denote the barycentric coordinates. Using a change of variables, this rule can be transferred to an arbitrary A-patch using the parametrization (129). The resulting quadrature rule on $\bar{\Gamma}_j$ is $\{(\mathbf{y}(\mathbf{b}_q), J(\mathbf{b}_q)w_q)\}_{q=1}^{n_q}$ where $J(\mathbf{b}_q)$ denotes the Jacobian of the parametrization.

Next we outline the quadrature rules for nonsingular, singular, and nearly singular integrals to be computed. In each case, quadrature rules on a reference triangle can be transferred to the curved molecular surface using the aforementioned change of variables.

Nonsingular Quadrature and the Fast Multipole Method Nonsingular quadrature is performed using a fixed Gaussian quadrature rule. This gives a single quadrature rule for the entire surface producing integrals of the form of (128). The source density $w_q \sum_{j=1}^{n_d} \phi_j \psi_j(\mathbf{y}_q)$ must be computed at each quadrature point y_q. Since the basis functions are locally supported, the summation over j only involves a bounded number of terms for any particular quadrature point. Then for all collocation points $\mathbf{x}_i$ the summation in q can be approximated by the fast multipole method in $O(n_q \cdot n_b)$ operations.

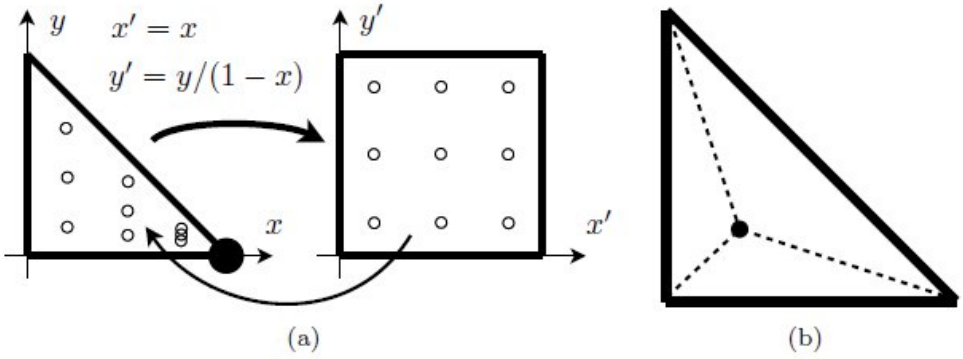


Figure 14: Singular quadrature rules. (a) Quadrature rule for a triangle with a weak singularity at a triangle vertex. (b) When singularity occurs in the triangle interior, the triangle is divided into three subtriangles at the singularity and then the scheme depicted in (a) can be applied to each subtriangle.

Singular Quadrature For smooth surfaces, the kernels in (122), (123), (124), and (125) are all integrable. By performing a change of variables to polar coordinates around the singularity, a smooth integrand is produced. For singularities occurring at a vertex of a triangle, a more computationally useful change of variables is described clearly in [29]. This coordinate change maps the a triangle into a square where a tensor-product Gaussian quadrature rule can be applied; see Figure 14.

When a triangle centroid is selected to be a collocation point, the integrand singularity occurs in the interior of the triangle. Suitable quadrature rules are formed by subdividing the triangle into three new triangles with the singularity as a new vertex; see Figure 14. Then the previous quadrature rule (which was designed for triangles with singularities at a vertex) can be applied to each of the three new triangles.

Nearly Singular Quadrature Nearly singular quadrature is performed by subdivision. On each subdivided triangle a Gaussian quadrature rule is applied. Precise convergence analysis imposes many restrictions on how this refinement should be performed and which integrals must be considered nearly singular; for examples, see [58, 104]. In Section 4.6 we demonstrate that nearly singular quadrature has limited importance for molecular structures and thus have avoided implementing a more complex (and computationally demanding) quadrature procedure.

4.4 Polarization Energy Computation

After solving for the electrostatic potential ϕ and its normal derivative $\frac{\partial \phi}{\partial \vec{\mathbf{n}}}$ the total polarization energy can be computed. Combining the expressions for the polarization energy (114) and the charge density (116) gives

$$G_{pol} = \int_{\Omega} \phi_{\text{rxn}}(\mathbf{z}) \sum_{k=1}^{n_c} q_k \delta(\mathbf{z} - \mathbf{z}_k) d\mathbf{z} = \frac{1}{2} \sum_{k=1}^{n_c} \phi_{\text{rxn}}(\mathbf{z}_k) q_k \tag{130}$$

where $\phi_{\text{rxn}}(\mathbf{x}) = \phi(\mathbf{x}) - \phi_{\text{gas}}(\mathbf{x})$ is the difference between the potential induced by the molecule in solution and the molecule in a gas.

Using Green's second identity as in the derivation of the boundary integral equations, formulas for the potential both inside and outside the molecule can be obtained; see [60] for complete details.

For a point $\mathbf{z} \in \mathbb{R}^3 \setminus \Gamma$,

$$\begin{aligned}\frac{\epsilon(\mathbf{z})}{\epsilon_I}\phi(\mathbf{z}) = &\int_\Gamma \left(\frac{\epsilon_E}{\epsilon_I} \frac{\partial G_\kappa(\mathbf{z},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} - \frac{\partial G_0(\mathbf{z},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} \right) \phi(\mathbf{y}) \dot{\mathbf{y}} \\ &+ \int_\Gamma (G_0(\mathbf{z},\mathbf{y}) - G_\kappa(\mathbf{z},\mathbf{y})) \frac{\partial \phi(\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} d\mathbf{y} + \sum_{k=1}^{n_c} \frac{q_k}{\epsilon_I} G_0(\mathbf{z},\mathbf{z}_k).\end{aligned} \tag{131}$$

The potential of the molecule in a gas is the solution to Poisson's equation (115) with constant dielectric $\epsilon(\mathbf{z}) := \epsilon_I$, and no charge density due to mobile ions $\rho(\mathbf{z}) = \rho_c(\mathbf{z})$. As the right hand side contains only a sum of Dirac functions, ϕ_{gas} is the sum of fundamental solutions to Poisson's equation,

$$\phi_{\text{gas}}(\mathbf{z}) = \sum_{k=1}^{n_c} \frac{q_k}{\epsilon_I} G_0(\mathbf{z},\mathbf{z}_k). \tag{132}$$

Subtracting (132) from (136) yields

$$\phi_{\text{rxn}}(\mathbf{z}) = \int_\Gamma \left(\frac{\epsilon_E}{\epsilon_I} \frac{\partial G_\kappa(\mathbf{z},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} - \frac{\partial G_0(\mathbf{z},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} \right) \phi(\mathbf{y}) + (G_0(\mathbf{z},\mathbf{y}) - G_\kappa(\mathbf{z},\mathbf{y})) \frac{\partial \phi(\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} d\mathbf{y},$$

for all $\mathbf{z} \in \Omega$. The fast multipole method is then used to efficiently evaluate ϕ_{rxn} at each atomic position $\mathbf{z}_k$ for the energy computation (130).

4.5 Implementation Details

Here we outline the steps in our software pipeline followed by a description of the key parameters to the algorithm.

4.5.1 Data Pipeline and Software Architecture

Given a molecular structure, a force field, and the concentrations of ions in solution, our code computes polarization energy in the following steps.

1. Molecular Structure Preparation Molecular structures contained in the Protein Data Bank [11] contain the types and positions of most of the atoms in a molecule. The software package `PDB2PQR` [28] places missing hydrogen atoms in the original structure and assigns partial charges and atomic radii based on the force field selected.

2. Molecular Surface and Triangular Surface Mesh Construction Based on the positions and radii of the atoms, a molecular surface is constructed through a level-set formulation with software. The level-set surface is approximated as a quality triangular mesh with surface normal directions specified at the vertices using a dual contouring method [110]. If necessary, this triangular mesh is decimated, and a geometric flow algorithm is applied to improve mesh quality [4].

3. Surface Parametrization The molecular surface is locally parametrized using the algebraic spline construction described in Section 4.3.2. Quadrature points are computed for each type of integral listed in Section 4.3.4.

4. Numerical Solution The linear system (equations (126) and (127) or the equivalent system for the dBIE formulation) is solved using the GMRES routine provided by `PETSc` (Portable, Extensible Toolkit for Scientific Computation) [9]. Matrix-vector products are implemented manually using `PETSc`'s shell matrix construction. Inside each matrix-vector product, `KIFMM3d` (Kernel-Independent Fast Multipole 3d Method) [103] is used to efficiently perform summations for a fixed quadrature rule and then singular and near field quadrature rules are used to provide a local correction to the least accurate portions of the integrals.

5. Energy Computation The polarization energy is computed using the formulation in Section 4.4. Numerical integration is again performed using `KIFMM3d` with local quadrature corrections to singular or nearly singular integrals.

6. User Interface and Visualization The molecular visualization and computation package `TexMol` provides a graphical interface for the algorithm parameters as well as immediate visualization of the results.

4.5.2 Algorithm Parameters

When running the algorithm, a particular formulation must be selected and a number of parameters must be set. One boundary integral formulation (nBIE or dBIE) must be selected and either piecewise constant or piecewise linear basis functions can be used. Additionally, the following parameters must be selected.

N_{g}	Number of points in triangular Gaussian quadrature rule
N_{s}	Number of points in triangular singular quadrature rule
N_{ns}	Number of subdivisions for the nearly singular quadrature rule
D_{ns}	Depth of triangles for the nearly singular quadrature rule
ϵ_{tol}	Tolerance for terminating `PETSc` GMRES routine
N_{fmm}	`KIFMM3d` accuracy parameter

The parameter N_{fmm} is the number of points used by `KIFMM3d` to represent equivalent densities and effects the accuracy of the fast multipole evaluations. `KIFMM3d` runs in $O(N_{\text{fmm}}^{\frac{3}{2}})$ time.

4.6 Experimental Results

Two types of experiments are considered: a simple example with a known solution and realistic protein complexes from the `ZDOCK` benchmark [73]. Results are compared to solutions of the linearized Poisson-Boltzmann equations produced by the multigrid finite difference method provided in `APBS` version 1.2.1 [8, 55].

4.6.1 Single Ion Model

We begin by studying the simplest molecule: a single atom with radius r and charge q. In this case an explicit solution to the linearized Poisson-Boltzmann equation is known [62]:

$$\phi^*(\mathbf{x}) = \begin{cases} \frac{q}{4\pi\epsilon_I|\mathbf{x}|} + \frac{q}{4\pi r}\left[\frac{1}{\epsilon_E(1+\kappa)} - \frac{1}{\epsilon_I}\right] & \mathbf{x} \in \Omega, \\ \frac{qe^{-\kappa(|\mathbf{x}|-r)}}{\epsilon_E(1+\kappa r)|\mathbf{x}|} & \mathbf{x} \notin \Omega. \end{cases} \tag{133}$$

The resulting polarization energy is

$$G^*_{\text{pol}} = \frac{q^2}{8\pi r}\left[\frac{1}{\epsilon_E(1+\kappa)} - \frac{1}{\epsilon_I}\right]. \tag{134}$$

Table 4: Comparison of solution error under several quadrature procedures on the single ion example. Each quadrature scheme is listed as $N_g/N_s/D_{ns}$. In all cases $N_{ns} = 6$ and $\epsilon_{tol} = 10^{-7}$.

	Mesh	Quadrature Scheme				
h	Vertices	3/9/3	3/9/0	3/4/0	1/9/0	1/4/0
4.9e-1	4^2	5.88e-2	5.91e-2	6.05e-2	6.15e-2	6.29e-2
2.5e-1	8^2	1.93e-2	1.94e-2	1.98e-2	2.13e-2	2.18e-2
1.2e-1	16^2	5.36e-3	5.38e-3	5.55e-3	6.47e-3	6.64e-3
6.2e-2	32^2	1.41e-3	1.41e-3	1.49e-3	1.98e-3	2.06e-3
3.1e-2	64^2	3.62e-4	3.65e-4	4.06e-4	6.50e-4	6.90e-4
1.6e-2	128^2	9.54e-5	9.70e-5	1.17e-4	2.38e-4	2.58e-4

Table 5: Comparison of the nBIE and dBIE formulations for the single ion model.

ϵ_{tol}	10^{-5}				10^{-8}			
	nBIE		dBIE		nBIE		dBIE	
h	Error	It.	Error	It.	Error	It.	Error	It.
4.9e-1	1.63e-1	5	5.91e-2	6	1.63e-1	10	5.91e-2	8
2.5e-1	4.47e-2	11	1.93e-2	9	4.47e-2	26	1.93e-2	15
1.2e-1	1.15e-2	6	5.38e-3	6	1.15e-2	28	5.38e-3	13
6.2e-2	2.95e-3	4	1.41e-3	6	2.95e-3	34	1.41e-3	13
3.1e-2	7.49e-4	3	3.65e-4	6	7.58e-4	43	3.65e-4	12

This example is used to test the various parameter settings for `PB-CFMM`. Relative error between the exact solution ϕ^* and the numerical solution ϕ is measured in the L^2-norm,

$$\text{Error} = \sqrt{\frac{\int_\gamma \left(\phi^*(\mathbf{y}) - \phi(\mathbf{y})\right)^2 + \left(\frac{\partial \phi^*(\mathbf{y})}{\partial \vec{\mathbf{n}}} - \frac{\partial \phi(\mathbf{y})}{\partial \vec{\mathbf{n}}}\right)^2 \dot{\mathbf{y}}}{\int_\gamma \left(\phi^*(\mathbf{y})\right)^2 + \left(\frac{\partial \phi^*(\mathbf{y})}{\partial \vec{\mathbf{n}}}\right)^2 \dot{\mathbf{y}}}}. \tag{135}$$

While derivatives of ϕ suggest that this expression is more closely related to the H^1-norm, ϕ and $\frac{\partial \phi}{\partial \vec{\mathbf{n}}}$ are independent unknowns in the boundary integral formulation. So (135) is the L^2-norm of the unknown vector $\left(\phi, \frac{\partial \phi}{\partial \vec{\mathbf{n}}}\right)$.

We begin by selecting acceptable quadrature rules. Table 4 contains a comparison of the solution error under different quadrature configurations. The simplified nature of our nearly singular quadrature scheme means that eventually (i.e., when the size of the triangles in the surface mesh becomes small enough) the quadratic convergence rate of the method will be lost. The table demonstrates that if the nonsingular and singular quadrature rules are of high enough order, nearly singular quadrature can be avoided. In practice we see that the three-point Gaussian quadrature rule for nonsingular integrals and the nine-point (i.e., three by three) rule for singular integrals are sufficient to preserve the convergence rate in the typical ranges that we consider. Higher degree integration rules do not reduce the solution error for the mesh sizes listed.

Table 5 contains a comparison of nonderivative ((122) and (123)) and derivative ((124) and ((125)) boundary integral formulations. In [67] it is reported that matrices corresponding to the derivative boundary integral equations are better conditioned for iterative solvers. We observe this when performing the computation for small ϵ_{tol}. However for modest ϵ_{tol} values we find that both formulations terminate in many fewer iterations without an impact on the solution error. Since the dBIE formulation requires four times as many fast multipole calls as the nBIE formulation (and thus typically four times the runtime), it can be desirable to use the nBIE formulation in certain situations. This likely explains how the nBIE formulation has been used successfully by

Table 6: Comparing the performance of the algebraic spline molecular surface (ASMS) to a linear approximation of the domain. The exact energy value is -81.450 kcal/mol.

	A-Spline			Linear		
h	L^2 Error	Energy	It.	L^2 Error	Energy	It.
4.9e-1	5.90e-2	-75.56	6	5.81e-1	-137.67	4
2.5e-1	1.94-2	-80.08	9	3.60e-1	-100.73	9
1.2e-1	5.38e-3	-81.61	6	1.92e-1	-89.37	10
6.2e-2	1.41e-3	-81.43	6	9.80e-2	-85.01	9
3.1e-2	3.65-4	-81.44	4	1.65e-3	-83.14	9

Table 7: Comparison of `APBS` and `PB-CFMM` for the single ion example. The exact polarization energy is -81.450 kcal/mol with the interior and exterior dielectric constants 2 and 80.

solver name	h	# degrees of freedom	G_{pol} (kcal/mol)	memory (mb)	time (seconds)
`APBS`	4.0e-1	17^3	-87.663	1.4	0.56
	2.0e-1	33^3	-84.476	8.2	1.18
	1.0e-1	65^3	-82.178	59	8.83
	5.0e-2	129^3	-81.831	448	57.73
	2.5e-2	257^3	-81.594	3510	426.30
`PB-CFMM`	2.5e-1	8^2	-80.077	38	5.90
	1.2e-1	16^2	-81.358	68	13.60
	6.2e-2	32^2	-81.428	125	46.56
	3.1e-2	64^2	-81.444	275	203.60
	1.6e-2	128^2	-81.449	995	830.09

some research groups; e.g. [1].

Table 6 contains a comparison of a curved A-spline molecular surface and a linear approximation of the geometry. Polarization energy converges at the expected quadratic rate for the curved geometry and at a linear rate for the linear geometry. Even for very coarse meshes (i.e., before the faster convergence rate has taken effect) the curved geometry performs much better. This is likely due to the hypersingular integrals associated with corners of the polygonal domain. Since the A-spline molecular surface is differentiable, it produces no hypersingular integrals and thus no associated numerical problems.

Table 7 contains a comparison of our solver with `APBS` for the single ion example. While the computational time for each method is linear in the number of degrees of freedom, the number of degrees of freedom grows at $O(h^{-3})$ for the finite difference solver compared to $O(h^{-2})$ for the boundary element solver. The finite difference solver is much more efficient per degree of freedom: this is expected because the linearity of the fast multipole method involves a larger constant then the local finite difference computations. The boundary element method gives a more accurate result when compared to finite difference grids with the same length scale.

4.6.2 Protein Binding Examples

We focus our experiments on a set of 212 ligand-receptor protein complexes from the `ZDOCK` benchmark [73]. Based on our experiments on the single ion model, we choose a conservative parameter set for `PB-CFMM`: $N_{\mathrm{g}} = 3$, $N_{\mathrm{s}} = 9$, $D_{\mathrm{ns}} = 0$, $\epsilon_{\mathrm{tol}} = 10^{-5}$, and $N_{\mathrm{fmm}} = 6$. These parameter settings were seen to preserve the expected convergence rates for the single ion model at small length scales. Atomic charge and radius information is generated using the AMBER 99 force field.

Table 8 contains a summary of the results of running `PB-CFMM` and `APBS` on the set of test

Table 8: Comparison of PB-CFMM and APBS on 212 molecules from the ZDOCK Benchmark. Error in the energy value is computed with respect to the finest mesh using the same solver and reported as a percentage.

solver	PB-CFMM			APBS		
# of DOF	2000	8000	32000	65^3	129^3	257^3
median energy error %	2.72	0.44	-	5.36	3.94	-
max energy error %	32.68	3.58	-	44.06	7.4	-
median # of iterations	19	22	24	-	-	-
max # of iterations	78	53	46	-	-	-
median compute time	37.14	173.45	801.94	13.84	80.29	524.64
median time per iter	1.92	8.12	32.08	-	-	-
median memory usage	65	150	469	126	535	3577

molecules. The runtime of both solvers is observed to be linear in the number of degrees of freedom as expected. Error in the energy values is computed with respect to the energy computed at the finest level. We see that PB-CFMM-computed energy values are more consistent than those computed with APBS.

Note that the median difference between the finest scale PB-CFMM and APBS results is 3.15%. This appears to be much higher than the error in the PB-CFMM computations. Some of this discrepancy is due to the differences in the molecular surfaces used by the two solvers since the surfaces given to PB-CFMM involve some pre-processing; recall Section 4.3.1. Figure 15 contains plots of the energy values computed under the different solvers and mesh sizes.

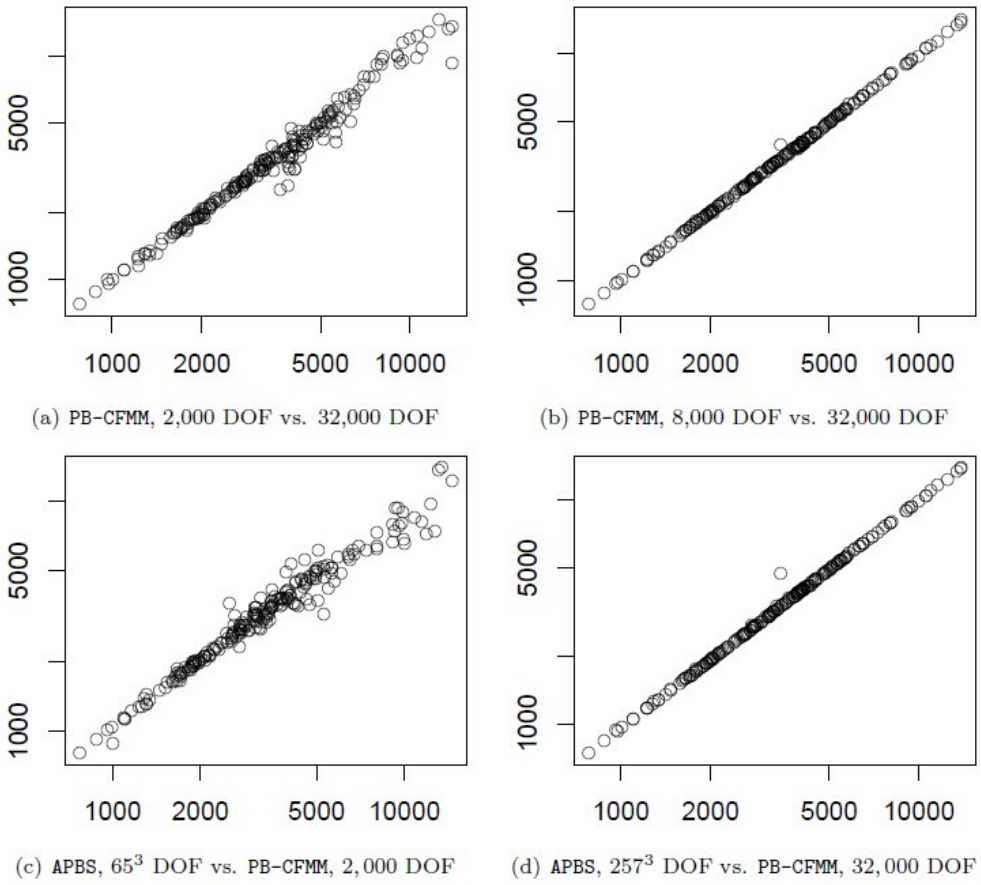
For a more detailed look at the results, we consider the per-atom energy values (i.e. individual terms in the summation (130)) for a particular molecule, nuclear transport factor 2 (PDB id: 1A2K). Figure 16 contains plots of the per-atom energy values for different mesh resolutions. The per-atom energies are consistent, especially between the highest resolution meshes. The median error over all atoms is 0.03 kcal/mol while the maximum error is 3.29 kcal/mol. Of the $3,179$ atoms, 46 have errors larger than 1 kcal/mol, and only two atoms have error larger than 2 kcal/mol. Figure 17 contains comparisons of per-atom energies resulting from the APBS solver.

Figure 18 demonstrates an electric potential computation for a typical protein complex. Figures 18(a-c) depict electric potential a molecule using different resolution surfaces meshes. These results can be compared to those produces by APBS shown in Figure 18(d). Figures 18(e-f) depict the potential computed separately for the two components. Finally Figure 18(g) contains the surface potential for the entire complex.

We demonstrate the need for the derivative boundary formulation and a curved approximation of the geometry by comparison to simpler alternatives. For this task we considered a set of 20 proteins from the ZDOCK benchmark. The derivative boundary integral formulation requires fewer iterations to terminate and for a fixed tolerance ϵ_{tol}, gives a more accurate solution. Specifically we compared the dBIE and nBIE formulations for a very modest GMRES tolerance, $\epsilon_{\text{tol}} = 10^{-3}$. The results are tabulated in Table 9. The dBIE formulation requires noticeably fewer GMRES iterations, but the dBIE formulation requires more computation time because it requires 16 fast multipole calls per iterations compared to only four required in the nBIE formulation. However, the advantage is of the dBIE formulation is seen by looking at the error in the energy value computed after termination: the dBIE energy values are very near the final value for a large ϵ_{tol} while the nBIE energy values contain substantial error.

The curved representation of the geometry yields a similar, yet more dramatic, impact on the energy computation: the computation requires more GMRES iterations while yielding much poorer energy values.

Table 10 contains results of our algorithm on the 20 protein test set when varying the fast



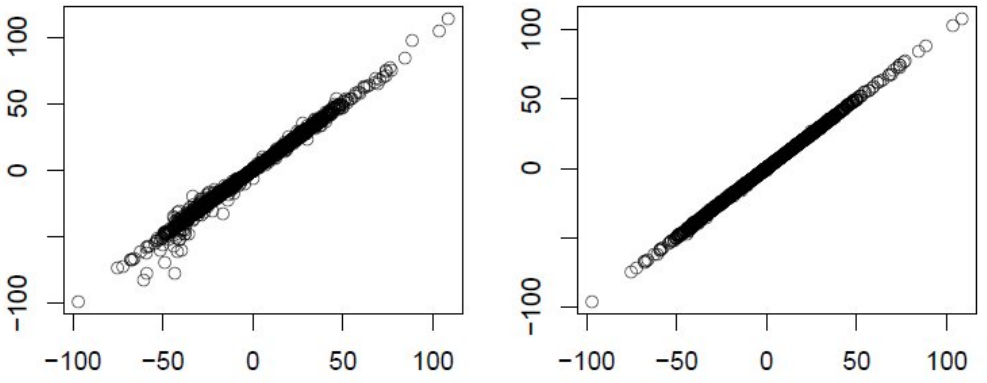
(a) PB-CFMM, 2,000 DOF vs. 32,000 DOF

(b) PB-CFMM, 8,000 DOF vs. 32,000 DOF

(c) APBS, 65^3 DOF vs. PB-CFMM, 2,000 DOF

(d) APBS, 257^3 DOF vs. PB-CFMM, 32,000 DOF

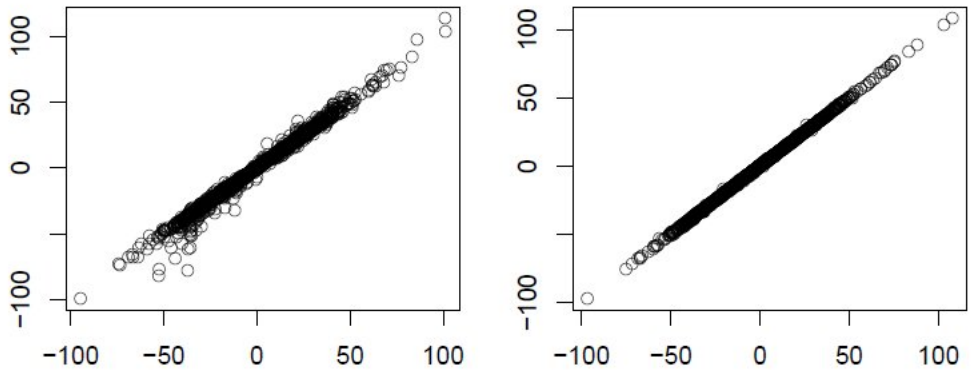
Figure 15: Scatter plots of polarization energy values computed for 212 proteins using different solvers and mesh sizes.



(a) Comparison of per-atom energy with 2,000 and 32,000 vertex meshes.

(b) Comparison of per-atom energy with 8,000 and 32,000 vertex meshes.

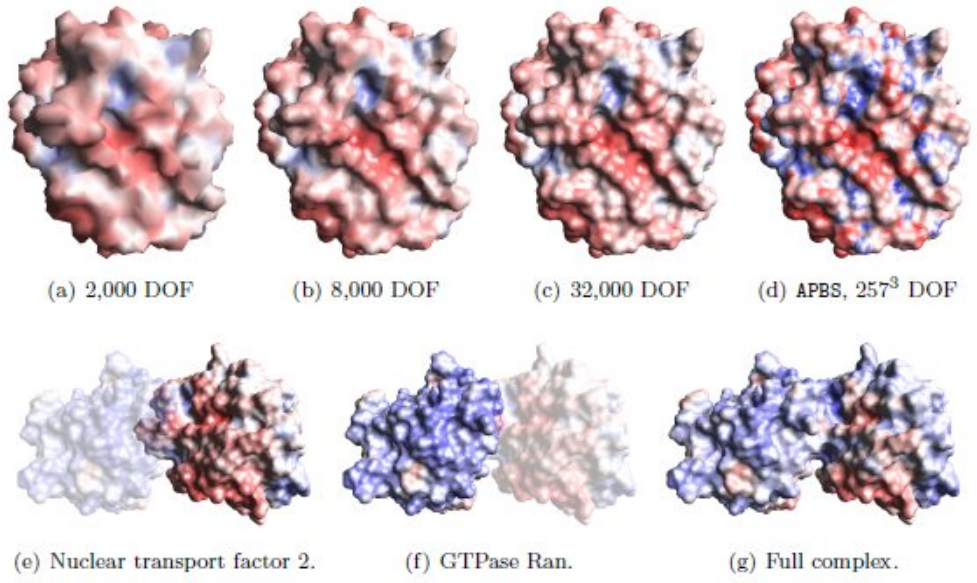
Figure 16: Per-atom polarization energy values are compared for nuclear transport factor 2 (PDB id: 1A2K). Polarization energy is computed using surface meshes with 2,000, 8,000, and 32,000 mesh vertices. The energy values for the high-resolution (32,000 vertex) mesh are given on the horizontal axis.



(a) 33^3-grid cell APBS computation (horizontal axis) vs 2,000 vertex PB-CFMM computation (vertical axis).

(b) 257^3-grid cell APBS computation (horizontal axis) vs 32,000 vertex PB-CFMM computation (vertical axis).

Figure 17: Per-atom polarization energy values are compared for nuclear transport factor 2 (PDB id: 1A2K). Polarization energy is compared between PB-CFMM and APBS.



(a) 2,000 DOF (b) 8,000 DOF (c) 32,000 DOF (d) APBS, 257^3 DOF

(e) Nuclear transport factor 2. (f) GTPase Ran. (g) Full complex.

Figure 18: The electrostatic potential on molecular surface for the complex between nuclear transport factor 2 and GTPase Ran (PDB id: 1A2K). In all cases, the potential is between $-3.8\ k_bT/e_c$ (red) and $+3.8\ k_bT/e_c$ (blue). (a-c) Electric potential of the nuclear transport factor 2 molecule using surface meshes containing 2,000, 8,000, and 32.000 triangles. (d) The surface potential computed by APBS. (e-f) Electric potential of the two component molecules. (g) Electric potential of the molecular complex.

Table 9: Comparison of nBIE and dBIE formulations on 20 example proteins. Error is computed with respect to the numerical solutions on the same mesh using a much lower GMRES tolerance (10^-7). *Computation was halted after 100 GMRES iterations: each computation involving linear geometry reached 100 iterations.

geometry formulation	A-Spline nBIE	A-Spline dBIE	Linear dBIE
median # iterations	40	17	*
max # iterations	47	26	*
median energy error	11.28	0.12	50.65
max energy error	17.55	0.46	61.92

Table 10: Results of polarization energy computation on 20 example proteins when varying N_{fmm}. Error in the energy computation is reported as a percentage.

N_{fmm}	2	4	6	8
median energy error	0.77	4.21×10^{-3}	1.24×10^{-4}	-
median compute time	316	493	737	1151

multipole accuracy parameter N_{fmm}. Meaningful differences in the final energy computation are only apparent for the lowest N_{fmm} value and even then these differences are small. In practice, we observe that polarization energy computations are not very sensitive to the fast multipole accuracy, especially when compared to the effects of the problem formulation and the molecular surface selection.

We have described a complete software pipeline for computing the electrostatic potential and polarization energy of biomolecules based on atomic descriptions. Our software is based on general purpose scientific computing codes `PETSc` and `KIFMM3d`, and performs favorably against a specialized linearized Poisson-Boltzmann solver. Our experiments demonstrate the benefits of the dBIE formulation of the Poisson-Boltzmann equation and a smooth representation of the molecular surface when simulating actual proteins.

In a similar fashion to the polarization energy, interior and exterior electrostatic potential and per-atom forces can also be computed as a post-process to the Poisson-Boltzmann solver. Integral formulations of the interior and exterior electrostatic potential are given in [60] while a derivation of the atomic forces can be found in [39]. Also worth consideration are more detailed models of molecular electrostatics including an ion exclusion layer surrounding the molecule and regions of differing dielectric constant. Altman et al. [1] formulate a system including these features with respect to the nBIEs and a similar extension should apply to the dBIE system. Moreover, the construction of the ASMS [113] should be useful in generating parallel surfaces required by the ion exclusion layer by picking different level sets a single function over the prismatic scaffold region.

Both `PETSc` and `KIFMM3d` are designed for parallel computation [105] and can be applied to our solution approach. However, for a number of problems the Poisson-Boltzmann equation must be solved many times; for example, the molecular docking problem requires polarization energy to be computed over many potential docked configurations. In such cases it is often more natural to find separate Poisson-Boltzmann solutions in parallel rather than parallelize the individual computations.

4.7 Interior and exterior electrostatic potential

Using Green's second identity as in the derivation of the boundary integral equations, formulas for the potential both inside and outside the molecule can be obtained; see [60] for complete details.

For a point $\mathbf{z} \in \mathbb{R}^3 \setminus \Gamma$,

$$\frac{\epsilon(\mathbf{z})}{\epsilon_I}\phi(\mathbf{z}) = \int_\Gamma \left(\frac{\epsilon_E}{\epsilon_I} \frac{\partial G_\kappa(\mathbf{z},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} - \frac{\partial G_0(\mathbf{z},\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} \right) \phi(\mathbf{y}) \dot{\mathbf{y}} + \int_\Gamma (G_0(\mathbf{z},\mathbf{y}) - G_\kappa(\mathbf{z},\mathbf{y})) \frac{\partial \phi(\mathbf{y})}{\partial \vec{\mathbf{n}}(\mathbf{y})} d\mathbf{y} + \sum_{k=1}^{n_c} \frac{q_k}{\epsilon_I} G_0(\mathbf{z},\mathbf{z}_k). \tag{136}$$

When multiple evaluations of the potential are required, the integrals in (136) are discretized and evaluated using the techniques in Section 4.3.4. In cases where all evaluation points are sufficiently far from the molecular surface, a fixed Gaussian quadrature rule and the fast multipole method are sufficient; no correction for nearly singular integrals in needed.

References

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

[2] C. Bajaj, R. Chowdhury, and V. Siddavinahalli. F^{2}dock: A fast Fourier based error-bounded approach to protein-protein docking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2009. Accepted for publication.

[3] C. Bajaj and G. Xu. A-Splines: Local interpolation and approximation using G_k- continuous piecewise real algebraic curves. *Comput. Aided Geom. Des.*, 16:557–578, 1999.

[4] C. Bajaj and G. Xu. Smooth shell construction with mixed prism fat surfaces. *Geom. Model. Comput. Suppl.*, 14:19–35, 2001.

[5] C. Bajaj and W. Zhao. Fast molecular solvation energetics and force computation. *SIAM J. Sci. Comput.*, 31(6):4524, 2010.

[6] N. Baker, M. Holst, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems. *J. Comput Chem.*, 21:1343–1352, 2000.

[7] N. Baker, M. Holst, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II: refinement at solvent accessible surfaces in biomolecular systems. *J. Comput. Chem.*, 21:1343–1352, 2000.

[8] N. Baker, D. Sept, S. Joseph, M. Holst, and J. McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proc. Natl. Acad. Sci*, pages 10037–10041, 1998.

[9] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[10] J. P. Bardhan, M. D. Altman, D. J. Willis, S. M. Lippow, B. Tidor, and J. K. White. Numerical integration techniques for curved-element discretizations of molecule-solvent interfaces. *J. Chem. Phys.*, 127:014701, 2007.

[11] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The Protein Data Bank. *Nucleic Acids Research*, pages 235–242, 2000.

[12] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2:363C–381, 1995.

[13] R. Bharadwaj, A. Windemuth, S. Sridharan, B. Honig, and A. Nicholls. The fast multipole boundary element method for molecular electrostatics: An optimal approach for large systems. *J. Comput. Chem.*, 16:898, 1995.

[14] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graphics*, 1(3):235–256, 1982.

[15] A. J. Bordner and 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.*, 24:353–367, 2003.

[16] S. Börm and W. Hackbusch. Hierarchical quadrature for singular integrals. *Computing*, 74(2):75–100, 2005.

[17] A. H. Boschitsch, M. O. Fenley, and H.-X. Zhou. Fast boundary element method for the linear Poisson-Boltzmann equation. *J. Phys. Chem.*, 106:2741–2754, 2002.

[18] W. R. Bowen and A. O. Sharif. Adaptive finite element solution of the nonlinear Poisson-Boltzmann equation: A charged spherical particle at various distances from a charge cylindrical pore in a charged planar surface. *J. Colloid Interface Sci.*, 187:363–374, 1997.

[19] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.*, 4:187–217, 1983.

[20] R. Bryant, H. Edelsbrunner, P. Koehl, and M. Levitt. The area derivative of a space-filling diagram. *Discrete Comput. Geom.*, 32:293–308, 2004.

[21] C. R. Morrow and T. N. L. Patterson. Construction of algebraic cubature rules using polynomial ideal theory. *SIAM J. Numer. Anal.*, 15:953–976, 1978.

[22] D. Case, T. Cheatham, III, T. Darden, H. Gohlke, R. Luo, K. Merz, Jr., A. Onufriev, C. Simmerling, B. Wang, and R. Woods. The Amber biomolecular simulation programs. *J. Comput Chem.*, 26:1668–1688, 2005.

[23] D. Chandler, J. Weeks, and H. Andersen. Van der Waals picture of liquids, solids, and phase transformations. *Science*, 220:787–794, 1983.

[24] L. Chen, M. Holst, and J. Xu. The finite element approximation of the nonlinear Poisson-Boltzmann equation. *SIAM J. Numer. Anal.*, 45(6):2298–2320, 2007.

[25] M. L. Connolly. Analytical molecular surface calculation. *J. Appl. Cryst.*, 16(5):548–558, 1983.

[26] C. Cortis and R. Friesner. An automatic three-dimensional finite element mesh generation system for the Poisson-Boltzmann equation. *J. Comput. Chem.*, 18:1570–1590, 1997.

[27] C. Cortis and R. Friesner. Numerical solution of the Poisson-Boltzmann equation using tetrahedral finite element methods. *J. Comput. Chem.*, 18:1591–1608, 1997.

[28] T. Dolinsky, J. Nielsen, A. McCammon, and N. Baker. Pdb2pqr: an automated pipeline for the setup of Poisson Boltzmann electrostatics calculations. *Nucleic Acids Research*, 32:665–667, 2004.

[29] M. G. Duffy. Quadrature over a pyramid or cube of integrands with a singularity at a vertex. *SIAM J. Numer. Anal.*, 19(6):1260–1262, 1982.

[30] D. Dunavant. High degree efficient symmetrical Gaussian quadrature rules for the triangle. *International Journal of Numerical Methods in Engineering*, 21:1129–1148, 1985.

[31] H. Edelsbrunner and P. Koehl. The weighted-volume derivative of a space-filling diagram. *PNAS*, 100:2203–2208, 2003.

[32] D. Eisenberg and A. D. Mclachlan. Solvation energy in protein folding and binding. *Nature (London)*, 319:199–203, 1986.

[33] M. Feig, A. Onufriev, M. S. Lee, W. Im, D. A. Case, and C. Brooks, III. Performance comparison of generalized Born and Poisson methods in the calculation of electrostatic solvation energies for protein structures. *J. Comput Chem.*, 25:265–284, 2004.

[34] F. Fogolari, A. Brigo, and H. Molinari. The Poisson-Boltzmann equation for biomolecular electrostatics: a tool for structural biology. *J. Mol. Recognit.*, 15(6):377–392, 2002.

[35] F. Fogolari, P. Zuccato, G. Esposito, and P. Viglino. Biomolecular electrostatics with the linearized Poisson-Boltzmann equation. *Biophys. J.*, 76(1):1–16, 1999.

[36] H. A. Gabb, R. M. Jackson, and M. J. E. Sternberg. Modelling protein docking using shape complementarity, electrostatics and biochemical information. *J. Mol. Biol.*, 272:106–120, 1997.

[37] M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. *IEEE Visualization*, pages 263–270, 1998.

[38] A. Ghosh, C. S. Rapp, and R. A. Friesner. Generalized Born model based on a surface integral formulation. *J. Phys. Chem. B*, 102:10983–10990, 1998.

[39] M. K. Gilson, M. E. Davis, B. A. Luty, and J. McCammon. Computation of electrostatic forces on solvated molecules using the Poisson-Boltzmann equation. *J. Phys. Chem.*, 97:3591–3600, 1993.

[40] M. K. Gilson, K. A. Sharp, and B. H. Honig. Calculating the electrostatic potential of molecules in solution: Method and error assessment. *J. Comput Chem.*, 9:327–335, 1987.

[41] J. Grant and B. Pickup. A Gaussian description of molecular shape. *J. Phys. Chem.*, 99:3503–3510, 1995.

[42] J. A. Grant and B. T. Pickup. A gaussian description of molecular shape. *J. Phys. Chem.*, 99:3503–3510, 1995.

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

[44] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J Chemical Physics*, 73:325–348, 1987.

[45] T. Grycuk. Deficiency of the Coulomb-field approximation in the generalized Born model: An improved formula for Born radii evaluation. *J Chemical Physics*, 119:4817–4827, 2003.

[46] J. L. Guermond. Numerical quadratures for layer potentials over curved domains in R^3. *SIAM J. Numer. Anal.*, 29(5):1347–1369, 1992.

[47] Hans Joachim Schmid. On cubature formulae with a minimal number of knots. *Numer. Math.*, 31:281–297, 1978.

[48] Hans Joachim Schmid. Interpolatorische Kubaturformeln und reelle Ideale. *Math. Z.*, 170:267–282, 1980.

[49] Hans Joachim Schmid. Interpolatory cubature formulae and real ideals. In R. D. Vore and K. Scherer, editor, *Quantitative Approximation*, pages 245–254. Academic, New York, 1980.

[50] Hans Joachim Schmid. Two-dimensional Minimal Cubature Formulas and Matrix Equations. *SIAM J. Matrix Anal. Appl.*, 16(3):898–921, 1995.

[51] R. B. Hermann. Theory of hydrophobic bonding. ii. correlation of hydrocarbon solubility in water with solvent cavity surface area. *J. Phys. Chem.*, 76:2754–2759, 1972.

[52] M. Holst. *Multilevel Methods for the Poisson-Boltzmann Equation.* PhD thesis, University of Illinois at Urbana-Champaign, 1993.

[53] M. Holst, N. Baker, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I: algorithm and examples. *J. Comput. Chem.*, 21(15):1319–1342, 2000.

[54] M. Holst, N. Baker, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples. *J. Comput Chem.*, 21:1319–1342, 2000.

[55] M. Holst and F. Saied. Multigrid solution of the Poisson-Boltzmann equation. *J. Comput. Chem.*, 14(1):105–113, 1993.

[56] W. Im, M. S. Lee, and C. L. Brooks. Generalized born model with a simple smoothing function. *J. Comput Chem.*, 24:1691–1702, 2003.

[57] J. I. Jackson. Selection of a convolution function for Fourier inversion using gridding. *IEEE Trans. Med. Imag.*, 10:473–C478, 1991.

[58] C. G. L. Johnson and L. R. Scott. An analysis of quadrature errors in second-kind boundary integral methods. *SIAM J. Numer. Anal.*, 26:1356, 1989.

[59] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *Proceedings of ACM SIGGRAPH*, pages 339–346, 2002.

[60] A. H. Juffer, E. F. F. Botta, B. A. M. van Keulen, A. van der Ploeg, and H. J. C. Berensen. The electric potential of a macromolecule in a solvent: a fundamental approach. *J. Chem. Phys.*, 97:144–171, 1991.

[61] O. D. Kellogg. *Foundations of Potential Theory.* Frederick Ungar Publishing Company, 1929.

[62] J. G. Kirkwood. Theory of solutions of molecules containing widely separated charges with special application to zwitterions. *J. Chem. Phys.*, 2:351, 1934.

[63] P. Kollman, I. Massova, C. Reyes, B. Kuhn, S. Huo, L. Chong, M. Lee, T. Lee, Y. Duan, W. Wang, O. Donini, P. Cieplak, J. Srinivasan, D. A. Case, and T. E. Cheatham. Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models. *Acc. Chem. Res.*, 33:889–897, 2000.

[64] S. S. Kuo, M. D. Altman, J. P. Bardhan, B. Tidor, and J. K. White. Fast methods for simulation of biomolecule electrostatics. *IEEE/ACM Int. Conf. Comput.-Aided Des.*, pages 466–473, 2002.

[65] B. Lee and F. M. Richards. The interpretation of protein structures: estimation of static accessibility. *J. Mol. Biol.*, 55(3):379–400, 1971.

[66] M. Levitt, M. Hirshberg, R. Sharon, and V. Daggett. Potential energy function and parameters for simulations of the molecular dynamics of proteins and nucleic acids in solution. *Comp. Phys. Comm.*, 91:215–231, 1995.

[67] J. Liang and S. Subramaniam. Computation of molecular electrostatics with boundary element methods. *Biophys. J.*, 73:1830–1841, 1997.

[68] B. Lu, D. Zhang, and J. McCammon. Computation of electrostatic forces between solvated molecules determined by the Poisson-Boltzmann equation using a boundary element method. *J Chemical Physics*, 122:214102–214109, 2005.

[69] B. Z. Lu, X. Cheng, J. Huang, and J. A. McCammon. Order N algorithm for computation of electrostatic interactions in biomolecular systems. *Proc. Natl. Acad. Sci.*, 103:19314–19319, 2006.

[70] B. Z. Lu, D. Q. Zhang, and 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):214102, 2005.

[71] B. Z. Lu, Y. C. Zhou, M. J. Holst, and J. A. McCammon. Recent progress in numerical methods for the Poisson-Boltzmann equation in biophysical applications. *Comm. Comput. Phys.*, 3(5):973–1009, 2008.

[72] A. MacKerel Jr., C. Brooks III, L. Nilsson, B. Roux, Y. Won, and M. Karplus. Charmm: The energy function and its parameterization with an overview of the program. In P. v. R. Schleyer et al., editor, *The Encyclopedia of Computational Chemistry*, volume 1, pages 271–277. John Wiley & Sons: Chichester, 1998.

[73] J. Mintseris, K. Wiehe, B. Pierce, R. Anderson, R. Chen, J. Janin, and Z. Weng. Protein-protein docking benchmark 2.0: an update. *Proteins Struct. Funct. Bioinf.*, 60(2):214–216, 2005.

[74] A. Nicholls and B. Honig. A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. *J. Comput. Chem.*, 12:435–445, 1991.

[75] T. J. Perun and C. L. Propst. *Computer-Aided Drug Design: Methods and Applications.* Informa Healthcare, 1989.

[76] J. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. Skeel, L. Kale, and K. Schulten. Scalable molecular dynamics with NAMD. *J. Comput Chem.*, 26:1781–1802, 2005.

[77] J. W. Ponder and D. A. Case. Force fields for protein simulations. *Adv. Protein Chem.*, 66:27–85, 2003.

[78] D. Potts and G. Steidl. Fast summation at nonequispaced knots by NFFTs. *SIAM J. Sci. Comput.*, 24:2013–2037, 2003.

[79] D. Potts, G. Steidl, and M. Tasche. Fast fourier transforms for nonequispaced data: A tutorial. In J. Benedetto and P. Ferreira, editors, *Modern Samplling Theory: mathematics and Applications*, pages 247–270. Birkhauser, 2001.

[80] R. Cools and A. Haegemans. Construction of fully symmetric cubature formulae of degree 4k-3 for fully symmetric planar regions. *J. comput. Appl. Math.*, 17:173–180, 1987.

[81] R. Cools and A. Haegemans. Construction of symmetric cubature formulae with the number of knots (almost) equal to Möller's lower bound. In H. Brass and G. Hämmerlin, editor, *Numerical Integration III*, pages 25–36. Birkhäuser, Basel, 1988.

[82] P. Ren and J. W. Ponder. Polarizable atomic multipole water model for molecular mechanics simulation. *J. Phys. Chem. B*, 107:5933–5947, 2003.

[83] F. M. Richards. Areas, volumes, packing and protein structure. *Annu. Rev. Biophys. Bioeng.*, 6:151–176, 1977.

[84] D. W. Ritchie. Evaluation of protein docking predictions using hex 3.1 in capri rounds 1 and 2. *Proteins: Structure, Function, and Genetics*, 52(1):98–106, July 2003.

[85] B. Roux and T. Simonson. Implicit solvent models. *Biophysical Chemistry*, 78:1–20, 1999.

[86] B. Roux and T. Simonson. Implicit solvent models. *Biophys. Chem.*, 78(1-2):1–20, 1999.

[87] A. Sayyed-Ahmad, K. Tuncay, and P. J. Ortoleva. Efficient solution technique for solving the Poisson-Boltzmann equation. *J. Comput. Chem.*, 25:1068–1074, 2004.

[88] C. Schwab. Variable order composite quadrature of singular and nearly singular integrals. *Computing*, 53(2):173–194, 1994.

[89] K. Sharp. Incorporating solvent and ion screening into molecular dynamics using the finite-difference Poisson-Boltzmann method. *J. Comput Chem.*, 12:454–468, 1991.

[90] K. Sharp and B. Honig. Applications of the finite defference Poisson-Boltzmann method to proteins and nucleic acids. *Struct. Methods: DNA Protein Complexes Proteins*, 2:211–214, 1990.

[91] A. Y. Shih, I. G. Denisov, J. C. Phillips, S. G. Sligar, and K. Schulten. Molecular dynamics simulations of discoidal bilayers assembled from truncated human lipoproteins. *Biophys. J.*, 88:548–556, 2005.

[92] T. Simonson and A. T. Bruenger. Solvation free energies estimated from macroscopic continuum theory: An accuracy assessment. *J. Phys. Chem.*, 98:4683 – 4694, 1994.

[93] I. Stakgold. *Boundary Value Problems of Mathematical Physics*, volume II. SIAM, 2000.

[94] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc*, 112:6127–6129, 1990.

[95] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc.*, 112:6127–6129, 1990.

[96] H. Tjong and H. Zhou. GBr^{6}NL: A generalized Born method for accurately reproducing solvation energy of the nonlinear Poisson-Boltzmann equation. *J Chemical Physics*, 126:195102–195106, 2007.

[97] V. Tsui and D. A. Case. Theory and applications of the generalized Born solvation model in macromolecular simulations. *Biopolymers*, 56:275–291, 2001.

[98] Y. N. Vorobjev and H. A. Scheraga. A fast adaptive multigrid boundary element method for macromolecular electrostatic computations in a solvent. *J. Comput. Chem.*, 18:569–583, 1996.

[99] J. Wagoner and N. Baker. Assessing implicit models for nonpolar mean solvation forces: The importance of dispersion and volume terms. *PNAS*, 103:8331–8336, 2006.

[100] J. Wagoner and N. A. Baker. Solvation forces on biomolecular structures: A comparison of explicit solvent and Poisson-Boltzmann models. *J. Comput. Chem.*, 25:1623–1629, 2004.

[101] X. Wang, J. N. Newman, and J. White. Robust algorithms for boundary-element integrals on curved surfaces. *Model. Simul. of Microsys.*, pages 473–476, 2000.

[102] J. Weeks, D. Chandler, and H. Andersen. Role of repulsive forces in determining the equilibrium structure of simple liquids. *JCP*, 54:5237–5247, 1971.

[103] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole method in two and three dimensions. *J. Comput. Phys.*, 196(2):591–626, 2004.

[104] L. Ying, G. Biros, and D. Zorin. A high-order 3D boundary integral equation solver for elliptic PDEs in smooth domains. *J. Chem. Phys.*, 219(1):247–275, 2006.

[105] L. Ying, G. Biros, D. Zorin, and H. Langston. A new parallel kernel-independent fast multipole method. In *ACM/IEEE Conf. Supercomput.*, 2003.

[106] R. J. Zauhar and R. S. Morgan. A new method for computing the macromolecular electric potential. *J. Mol. Biol.*, 186:815–820, 1985.

[107] R. J. Zauhar and R. S. Morgan. The rigorous computation of the molecular electric potential. *J. Comput. Chem.*, 9(2):171–187, 1988.

[108] R. J. Zauhar and R. S. Morgan. Computing the electric potential of biomolecules: application of a new method of molecular surface triangulation. *J. Comput. Chem.*, 11:603–622, 1990.

[109] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Cagd*, 23:510–530, 2006.

[110] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Comput. Aided Geom. Des.*, 23:510–530, 2006.

[111] W. Zhao, G. Xu, and C. Bajaj. An algebraic spline model of molecular surfaces. *ACM Symp. Sol. Phys. Model.*, 2007:297–302, 2007.

[112] W. Zhao, G. Xu, and C. Bajaj. An algebraic spline model of molecular surfaces. *Proc. ACM Symp. Solid Phys. Model.*, pages 297–302, 2007.

[113] W. Zhao, G. Xu, and C. Bajaj. An algebraic spline model of molecular surfaces for energetic computations. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 2009. Accepted for publication.

[114] H. Zhou. Boundary element solution of macromolecular electrostatics: interaction energy between two proteins. *Biophys. J.*, 65:955–963, 1993.