Computation of forces arising from the linear Poisson–Boltzmann method in the domain-decomposition paradigm

Abhinav Jha, Michele Nottoli, Chaoyu Quan, Benjamin Stamm

Abstract

The Linearized Poisson–Boltzmann (LPB) equation is a popular and widely accepted model for accounting solvent effects in computational (bio-)chemistry. In the present article we derive the analytical forces of the domain-decomposition-based ddLPB-method. We present an efficient strategy to compute the forces and its implementation, and present numerical tests illustrating the accuracy and efficiency of the computation of the analytical forces.

1 Introduction

Most chemical processes and virtually all biochemical processes happen in condensed phase, a situation where the reacting part, or in general the studied part, is embedded in an environment which usually consists of a solvent. For this reason, solvation models, which take into account the effect of the environment on the interesting part (solute), are widely used in computational chemistry and biochemistry. These models can be broadly divided into two classes, explicit solvation models and implicit (continuum) solvation models. Explicit solvation models consider the molecular representation of both, the solute and solvent, making the method more accurate, but computationally expensive and also dependent on a large set of empirical parameters (force field). On the other hand, continuum solvation models treat the solvent as a continuum, described only by a few macroscopic properties. This approach, by its nature, cannot describe specific interactions and anisotropic environment, however it presents some large advantages, it reduces the computational cost significantly, requires fewer parameters and implicitly takes into account the sampling over the degrees of freedom of the solvent. For this reason, implicit solvation models are nowadays popular computational approaches to characterize solvent effects in the simulation of properties and processes of molecular systems in condensed phase [TP94, HN95, RS99, CT99, OL00, TMC05].

Some of the widely used continuum solvation models include the conductor-like screening model (COSMO), proposed in [KS93]; the surface and simulation of volume polarization for electrostatics (SS(V)PE) [Chi99, Chi06]; polarizable continuum model (PCM) [TMC05, MST81, CMT97, BC98, CRSB03] to name a few. In this paper, we focus on the continuum solvation model based on the Poisson–Boltzmann (PB) equations [YL90, NH91] which takes into account both the solvent relative dielectric permittivity and the ionic strength of the solvent.

We consider specifically the linearized Poisson–Boltzmann (LPB) equation which describes the electrostatic potential, $\psi$ of the solvation model in the following form

$$\nabla \cdot [\varepsilon(x) \nabla \psi(x)] + \pi(x)^2 \psi(x) = 4\pi \rho_M(x) \quad \text{in} \quad \mathbb{R}^3,$$

(1)
where $\varepsilon(x)$ is the space-dependent dielectric permittivity function, $\pi(x)$ is the modified Debye-Hückel parameter, and $\rho_M(x)$ is the solute charge distribution.

We denote the solute cavity by $\Omega$ and the solvent region by $\Omega^C = \mathbb{R}^3 \setminus \Omega$. To describe the solute-solvent region we will use the van-der Waals (vdW) surface (see Fig. 1). The solute cavity $\Omega$ is defined as a union of overlapping subdomains, $\Omega_j$, i.e.,

$$\Omega = \bigcup_{j=1}^{M} \Omega_j, \quad \Omega_j = B_{r_j}(x_j),$$

where each $\Omega_j$ is a vdW ball with radius $r_j$ and center $x_j$, and $M$ is the total number of atoms. Then $\varepsilon(x)$ has the form

$$\varepsilon(x) = \begin{cases} 
\varepsilon_1 & \text{in } \Omega, \\
\varepsilon_2 & \text{in } \Omega^C,
\end{cases}$$

where $\varepsilon_1$ and $\varepsilon_2$ are the solute and solvent’s dielectric permittivity, respectively. Furthermore, $\pi(x)$ has the form

$$\pi(x) = \begin{cases} 
0 & \text{in } \Omega, \\
\sqrt{\varepsilon_2 \kappa} & \text{in } \Omega^C,
\end{cases}$$

where $\kappa > 0$ is the Debye-Hückel screening constant of the solvent.

The classical PCM and the COSMO model can be considered as the special cases for PB solvation models. In the classical PCM, the solvent is represented as a polarizable continuous medium that is non-ionic ($\kappa = 0$) whereas the COSMO is a reduced version of the PCM, where the solvent is represented as a conductor-like continuum.

We would like to mention some of the widely used methods for solving the LPB equation such as the boundary element method (BEM), the finite difference method (FDM), and the finite element method (FEM), and we refer to [LZHM08] for a review. The main idea of the BEM is to recast the LPB equation as an integral equation defined on a two-dimensional solute-solvent interface [YL90, BFZ02, ABWT09, BCR11]. It is an efficient way to solve the LPB equation, which can be optimized using fast multipole methods [ZPH+15] and the hierarchial treecode technique [LZHM08]. The PAFMPB solver [LCHM10, ZPH+15] uses the former optimization technique, whereas the TABI-PB [GK13] uses the latter one. The PB-SAM solver developed by Head-Gordon et al. [LHG06, YHG10, YHG13] discretizes the solute-solvent interface (such as the vdW surface) with grid points on atomic spheres like a collocation method and solves the associated linear system by use of the fast multipole method. It primarily targets the interaction of disjoint molecular compounds. However, one of the limitations of all these solvers relying on integral equations and layer potentials is that it cannot be generalized to solve the nonlinear PB (NPB) equation as opposed to PDE-based methods such as the FDM or FEM.
The finite difference approach is the most popular method to solve linear or nonlinear PB equations. The main idea is to cover the region of interest with a big-box grid and choose different kinds of boundary conditions. Some of the popular software packages using the FDM include UHBD [MBW95], Delphi [LLS+12], MIBPB [CCC+10], and APBS [BSJ+01, DCL+07, JES+17]. One of the drawbacks of the FDM is that the cost can increase considerably with respect to the grid dimension.

The finite element approach, compared to FDM, provides more flexible mesh refinement and a proper convergence analysis [CHX07]. The SDPBS and SMPBS offer fast and efficient approximations of the size-modified PB equation [Xie14, YX15, JXY+15, XYX17].

In this paper, we focus on the domain decomposition (dd) framework. Recently, in [QSM19] a domain decomposition algorithm has been proposed for the LPB equation, which uses a particular Schwarz domain decomposition method to solve Eq. (1). The ideas of the ddLPB method can be traced back to the domain decomposition methods proposed for the COSMO model (ddCOSMO) [CMS13, LSC+13, LLS+14, LSL+14] and the PCM model (ddPCM) [SCLM16, GLS17, NSSL19]. These methods do not require any mesh or grid of the molecular surface, are easy to implement, and about two orders of magnitude faster than the state of the art [LLS+14]. In particular, the ddCOSMO solver can perform up to thousands of times faster than equivalent existing algorithms.

Similar to the aforementioned dd approaches the ddLPB method does also not require any mesh or grid but depends, as ddCOSMO and ddPCM, only on the Lebedev quadrature points [LL99] on a two-dimensional sphere. Hence it is convenient to be applied in molecular dynamics without re-meshing the molecular surface as in the BEM. The ddLPB solver adopts a spectral Galerkin method for discretization and benefits from high sparsity of the involved matrices for the Laplace and screening Poisson equations in Ω, which are coupled by a non-local integral equation on the boundary. The latter takes the majority of cost but can be further accelerated using for example the fast multipole method (FMM). Numerical implementations show that the ddLPB solver is very efficient even without acceleration techniques (see [QSM19] for details) and its FMM-acceleration is ongoing work in progress but not within the scope of this article. The focus of this work is to develop the framework of the computation of first derivatives and the forces for the ddLPB method, the FMM-acceleration is a subsequent step. As the spheres are centered around their nuclei, the computation of the forces becomes natural. By the nature of the problem, this is very technical task, but a necessity in order to make accessible the method to models requiring the gradient of the solvation energy with respect to the nuclear coordinates such as molecular dynamics or geometry optimization.

The paper is divided as follows: Section 2 introduces the notations and gives a summary of the domain decomposition algorithm for the LPB equation. In Section 3 we compute the forces and compute the analytical derivatives. Lastly, in Section 4 we present a comprehensive numerical study, before we conclude in Section 5.

## 2 ddLPB Method

### 2.1 Linear Poisson–Boltzmann Equations

One notes that the LPB equation (1) can be written as two equations, one defined in the solute cavity Ω, namely the Laplace equation given by

\[-\Delta \psi_r = 0 \quad \text{in} \quad \Omega, \tag{2}\]

which is obtained from transforming the Poisson equation by using the transformation \(\psi_r = \psi - \psi_0\) where \(\psi_0\) is the potential generated by \(\rho_M\) in the vacuum, i.e.,

\[-\Delta \psi_0 = \frac{4\pi}{\varepsilon_1} \rho_M \quad \text{in} \quad \mathbb{R}^3; \tag{3}\]

and a homogeneous screened Poisson (HSP) equation defined on the solvent region given by

\[-\Delta \psi_e(x) + \kappa^2 \psi_e(x) = 0 \quad \text{in} \quad \Omega^C, \tag{4}\]
with two classical jump conditions

\[
[\psi] = 0 \quad \text{on } \Gamma,
\]

\[
[\partial_n (\varepsilon \psi)] = 0 \quad \text{on } \Gamma,
\]

along the solute-solvent interface \( \Gamma = \partial \Omega \), and where \([f] = f|_\Omega - f|_{\Omega^c}\) and \(\partial_n \psi\) the normal derivative of \(\psi\). Based on the classical jump condition of \(\psi\), a coupling condition between Eq. (2) and Eq. (4) arises through a function \(h\) defined by

\[
h = S_{\kappa} \left( \partial_n \psi_e - \frac{\varepsilon_1}{\varepsilon_2} \partial_n (\psi_0 + \psi_r) \right) \quad \text{on } \Gamma,
\]

where \(S_{\kappa} : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)\) denotes a single-layer operator on \(\Gamma\) and \(H^{\pm 1/2}(\Gamma)\) denote the fractional Sobolev spaces \([Ada75]\).

We call \(\psi_r\) and \(\psi_e\) the reaction potential and the extended potential, respectively. In this paper, we assume that the solute’s charge distribution \(\rho_M\) is supported in \(\Omega\) and in particular given by the sum of \(M\) point charges, i.e.,

\[
\rho_M(x) = \sum_{i=1}^{M} q_i \delta(x - x_i),
\]

(5)

where \(q_i\) denotes the (partial) charge carried on the \(i\)th atom with center \(x_i\), and \(\delta\) is the Dirac delta distribution, but the framework can easily be generalized to non-classical charges under the usual assumption \(\text{supp} (\rho_M) \subset \Omega\).

2.2 Domain Decomposition Algorithm

The domain decomposition algorithm that we will consider in this paper has been derived in \([QSM19]\). For brevity, we will not be deriving the whole method, but we will only present the main equations required for the derivation of analytical forces.

We first introduce certain notations and functions that will be used throughout the paper. We denote the characteristic function on \(\Omega_i\) by \(\chi_i\), i.e.,

\[
\chi_i(x) := \begin{cases} 
1 & \text{if } x \in \Omega_i, \\
0 & \text{else}, 
\end{cases}
\]

and then let

\[
\omega_{ij}(x) := \frac{\chi_j(x)}{\sum_{k \in N_i} \chi_k(x)} \quad \text{for } x \in \Gamma_i,
\]

(6)

where \(N_i\) denotes the set of indices of spheres intersecting \(\Omega_i\) (i not included). We make the convention that if \(|N_i| = 0\), we define \(\omega_{ij}(x) = 0\) for all \(j\). The boundary \(\Gamma_i\) of the sphere \(\Omega_i\) can either be on the solute-solvent boundary, \(\Gamma\), i.e., on the external part or inside the solute cavity, i.e., the internal part. To distinguish between the two cases we define the characteristic function, \(\chi_i^e(x)\) as

\[
\chi_i^e(x) := \begin{cases} 
1 & \text{if } x \in \Gamma_i^e, \\
0 & \text{if } x \in \Gamma_i^i,
\end{cases}
\]

where \(\Gamma_i^e\) and \(\Gamma_i^i\) denote the external and internal part of the boundary \(\Gamma_i\) respectively, see Fig. 2. With the definition of \(\omega_{ij}(x)\) from Eq. (6) we have the relation

\[
\chi_i^e(x) = 1 - \sum_{j \in N_i} \omega_{ij}(x) \quad \text{for } x \in \Gamma_i.
\]

(7)
We define the radial scaling function of order $\ell$ depending on the $i^{th}$ atom by

$$r_i^\ell(x) := \left(\frac{|x - x_i|}{r_i}\right)^\ell. \quad (8)$$

The angular dependency relative to the $i^{th}$ atom is denoted by

$$Y_{\ell m}^i(x) := Y_{\ell m}\left(x - x_i\right), \quad (9)$$

where $Y_{\ell m} : S^2 \rightarrow \mathbb{R}$ is the real-valued orthonormal spherical harmonic of degree $\ell$ and order $m$. Moreover, we define the following radial Bessel function by

$$i_{\ell}^i(x) := \frac{\imath (|x - x_i|)}{\imath (r_i)}, \quad (10)$$

where $\imath(x)$ is the modified spherical Bessel’s function of the first kind.

Finally, we have integrals over the unit sphere $S^2$ which will be numerically approximated using the Lebedev quadrature rule [LL99] with $N_{\text{leb}}$ points. The approximation over the sphere $\Omega_i$ is given by

$$\langle f, g \rangle_{n,i} := \sum_{n=1}^{N_{\text{leb}}} \omega_n f(x_n^i)g(x_n^i),$$

where $x_n^i = x_i + r_is_n$, $s_n \in S^2$, and $\omega_n$ is the quadrature weight.

The fully discretized domain decomposition algorithm for the LPB equation gives rise to the system of equations given by

$$LX = g, \quad (11)$$

where

$$L = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_1 & C_2 \end{bmatrix}, \quad X = \begin{bmatrix} X_r \\ X_e \end{bmatrix}, \quad g = \begin{bmatrix} G_0 + F_0 \\ F_0 \end{bmatrix}. \quad (11)$$

The matrices $A$, $B$, $C_1$, and $C_2$ are of the size $M (\ell_{\text{max}} + 1)^2 \times M (\ell_{\text{max}} + 1)^2$ where $\ell_{\text{max}}$ denotes the maximum degree of spherical harmonics. The vectors $G_0$ and $F_0$ on the right-hand side correspond to $\psi_0$ and $\partial_n \psi_0$, respectively, and $X_r$ and $X_e$ denote the solution vectors corresponding to the reaction potential and the extended potential, respectively. After calculating $X$, we can approximate $\psi_r$ and $\psi_e$ respectively by a linear combination of spherical harmonics as follows

$$\psi_r(x) \approx \sum_{\ell=0}^{\ell_{\text{max}}} \sum_{m=-\ell}^{\ell} [X_r]_{\ell m} r_i^\ell(x) Y_{\ell m}^i(x), \quad x \in \Omega_i, \quad (12)$$

and

$$\psi_e(x) \approx \sum_{\ell=0}^{\ell_{\text{max}}} \sum_{m=-\ell}^{\ell} [X_e]_{\ell m} i_{\ell}^i(x) Y_{\ell m}^i(x), \quad x \in \Omega_i. \quad (13)$$
We now show the specific formulas of the matrices. The \((iℓm, jℓ'\ell'')\)\(^{th}\) matrix entry for \(A\) is given by,

\[
[A]_{ij}^{mm'} := \delta_{\ell\ell'}\delta_{mm'},
\]

and the \((iℓm, jℓ'\ell'')\)\(^{th}\) matrix entry for \(B\) is given by,

\[
[B]_{ij}^{mm'} := -\sum_{n=1}^{N_{\text{lab}}} \omega_n \omega_{ij}(x_i^n) r_{\ell'\ell}(x_i^n) Y_{\ell'\ell''}(x_i^n) Y_{\ell''}(s_n), \quad i \neq j.
\]

We note that both the matrices \(A\) and \(B\) are sparse in nature, as blocks are nonzero only for interlocking vdW balls.

Next, we move to the matrices \(C_1\) and \(C_2\) where the \((iℓm, jℓ'\ell'')\)\(^{th}\) entry of \(C_1\) is given by

\[
[C_1]_{ij}^{mm'} := \varepsilon_1 \frac{\vartheta}{\varepsilon_2} \sum_{n=1}^{N_{\text{lab}}} \omega_n \chi_i^n(x_i^n) Y_{\ell m}(s_n) |Q|_{j\ell''m'}^{in} \frac{\ell'}{\ell} \frac{(\ell)}{r_j},
\]

and for \(C_2\) by,

\[
[C_2]_{ij}^{mm'} := -\sum_{n=1}^{N_{\text{lab}}} \omega_n \chi_i^n(x_i^n) Y_{\ell m}(s_n) |Q|_{j\ell''m'}^{in} \frac{1}{\ell'} \frac{r_j}{(\ell')} \frac{(\ell)}{r_j},
\]

where matrix \(Q\) is a matrix of size \(M(\ell_{\text{max}} + 1)\)\(^2\) \times \(MN_{\text{lab}}\) and the \((j\ell'\ell''m'', i\ell'\ell')\)\(^{th}\) entry is given by

\[
[Q]_{j\ell''m''}^{in} := \sum_{\ell_0m_0} C_{ij}^{\ell\ell'} [P \chi_j^{\ell}]_{\ell_0m_0} k_{\ell_0}(x_i^n) Y_{\ell_0m_0}(x_i^n),
\]

where \(k_{\ell_0}(x_i^n)\) is defined similarly to Eq. \([10]\) given by

\[
k_{\ell_0}(x_0(x - x_j)) = \frac{k_{\ell_0}(x - x_j)}{k_{\ell_0}(r_j)},
\]

\(k_{\ell_0}(x)\) is the modified spherical Bessel’s function of the second kind,

\[
C_{ij}^{\ell\ell'} = \left( \frac{1}{\ell_0(r_j)} - \frac{k_{\ell_0}(r_j)}{k_{\ell_0}(r_j)} \right)^{-1},
\]

and the notation \(\sum_{\ell m}\) denotes \(\sum_{\ell = 0}^{\ell_{\text{max}}} \sum_{m = -\ell}^{\ell}\).

The matrix \(P \chi_j^{\ell}\) is of size \((\ell_{\text{max}} + 1)^2 \times (\ell_{\text{max}} + 1)^2\) whose \((\ell_0m_0, \ell'\ell'')\)\(^{th}\) entry is given by

\[
[P \chi_j^{\ell}]_{\ell_0m_0}^{\ell'\ell''} := \sum_{n=1}^{N_{\text{lab}}} \omega_n \chi_j^n(x_j^n) Y_{\ell_0m_0}(s_n) Y_{\ell'\ell''}(s_n).
\]

Finally, we have the right-hand side vectors. The \((i\ell m)\)\(^{th}\) entry of the vector \(G_0\) is given by

\[
[G_0]_{i\ell m} = -\sum_{n=1}^{N_{\text{lab}}} \omega_n \chi_i^n(x_i^n) \psi_0(x_i^n) Y_{\ell m}(s_n),
\]
where

\[
\psi_0(x) = \sum_{j=1}^{M} \frac{q_j}{\varepsilon_1 |x - x_j|},
\]

(22)
is the solution of Eq. (3) and the \((i\ell m)\)th entry of \(F_0\) is given by

\[
[F_0]_{i\ell m} = -\frac{\varepsilon_1}{\varepsilon_2} \left( \sum_{n=1}^{N_{th}} \omega_n \chi_n^\ell(x_j^n) Y_{\ell m}(s_n) \sum_{j=1}^{M} [S]_{jin} \right),
\]

(23)

where

\[
[S]_{jin} = \sum_{\ell_0 m_0} C_{\ell_0 m_0}^j C_{\ell_0 m_0}^\ell (x_j^n) Y_{\ell m}(s_n),
\]

(24)

and

\[
C_{\ell_0 m_0}^j = \sum_{n=1}^{N_{th}} \omega_n \chi_j^\ell(x_j^n) \partial_n \psi_0(x_j^n) Y_{\ell m}(s_n).
\]

(25)

### 3 Computation of Forces

The computation of the electrostatic solvation energy, \(E_s\), in [QSM19], follows the ideas of [FBM02] where the reaction potential was used to compute \(E_s\). For the computation of forces, we require the whole electrostatic potential and hence we define \(E_s\) as

\[
E_s = \frac{1}{2} \langle \psi_r, \rho_M \rangle = \frac{1}{2} \sum_{j=1}^{M} \langle X, Q \rangle_j,
\]

(26)

where \(X\) is given in Eq. (11), \(Q\) has the same size as \(X\) with

\[
[Q]_{j\ell m} = \begin{cases} q_j \delta_{\ell 0} \delta_{m 0}, & \text{if } 1 \leq j \leq M, \\ 0, & \text{if } M < j \leq 2M. \end{cases}
\]

and the inner product \(\langle \cdot, \cdot \rangle_j\) is given by

\[
\langle X, Q \rangle_j = \sum_{\ell m} [X]_{j\ell m} [Q]_{j\ell m}.
\]

The force with respect to a parameter \(\lambda\), such as the position of \(x_k\) of the \(k\)th atom, is given by,

\[
F_\lambda = \nabla_\lambda \langle E_s \rangle = \frac{1}{2} \left( \langle \nabla^\lambda X, Q \rangle + \langle X, \nabla^\lambda Q \rangle \right) = \frac{1}{2} \langle \nabla^\lambda X, Q \rangle.
\]

The ddLPB system is given by \(LX = g\). Taking the derivative with respect to \(\lambda\):

\[
\nabla^\lambda LX + L \nabla^\lambda X = \nabla^\lambda g \\
\nabla^\lambda X = L^{-1} \left( \nabla^\lambda g - \nabla^\lambda LX \right).
\]

Substituting \(\nabla^\lambda X\) in the force computation

\[
F_\lambda = \frac{1}{2} \left( L^{-1} \left( \nabla^\lambda g - \nabla^\lambda LX \right), Q \right)
\]

\[
= \frac{1}{2} \left( \langle \nabla^\lambda g - \nabla^\lambda LX \rangle, (L^{-1})^* Q \right).
\]

7
where $L^*$ is the adjoint of the matrix $L$ and $(L^{-1})^* Q$ is the solution of the system

$$L^* X_{\text{adj}} = Q.$$  

(27)

Using the definition of $X_{\text{adj}}$ we get the computation of forces as

$$F_\lambda = \frac{1}{2} \langle (\nabla^\lambda g - \nabla^\lambda LX) , X_{\text{adj}} \rangle.$$  

(28)

We note that in Eq. (28) we require the computation of the adjoint system (but only once for any number of different parameters $\lambda$) and the derivatives of the $g$ and $L$ matrix. The adjoint matrix of the system is given by

$$L^* = \begin{bmatrix} A^T & 0 \\ 0 & B^T \end{bmatrix} + \begin{bmatrix} C^T_1 & C^T_2 \\ C^T_2 & C^T_3 \end{bmatrix},$$  

(29)

where $A^T$ stands for the transpose of the matrix $A$ and respectively others.

3.1 Analytical Derivatives

We now restrict ourselves to the case where $\lambda$ denotes the central coordinate $x_k$ of the $k^{th}$ atom. We note that entries of matrix $L$ and vector $g$ have certain functions that are not smooth, namely, $\chi_i(x), \chi^e_i(x)$, and $\omega_{ij}(x)$. To define their differentiable counterparts, we follow the ideas presented in [LSC13]. We first introduce a polynomial, $p_\eta(t)$ given by

$$p_\eta(t) := \eta^{-5} (1 - t)^3 \left( 6t^2 + (15\eta - 12) t + 1 - \eta^2 - 15\eta + 6 \right),$$

where $\eta$ is a smoothness parameter. Then the regularized characteristic function is given by

$$\chi_\eta(t) = \begin{cases} 
1 & \text{if } t \leq 1 - \eta, \\
1 - \eta < t < 1, & p_\eta(t) \\
0 & \text{if } t \geq 1.
\end{cases}$$  

(30)

Using Eq. (30), the regularized version of $\omega^\eta_{ij}(x)$ defined in Eq. (6) is given by

$$\omega^\eta_{ij}(x) := d^\eta(x) \chi_\eta \left( r^j_1(x) \right), \quad \forall x \in \Gamma_i,$$  

(31)

with

$$d^\eta(x) := \min \left\{ \frac{f^i(x)}{f^i(x)}, 1 \right\},$$  

(32)

where

$$f^i(x) := \sum_{k \in N_i} \chi_\eta \left( r^k_1(x) \right)$$  

(33)

and $r^j_1$ is defined in Eq. (8). Finally, the differentiable counterpart of $\chi^e_i(x)$ is given by

$$\chi^\eta_i(x) := 1 - \sum_{j \in N_i} \omega^\eta_{ij}(x), \quad \forall x \in \Gamma_i.$$  

(34)

One thing to note is that in the definition of $d^\eta(x)$ we have a minimum which is not a smooth function. On close inspection we note that if $f^i(x) < 1$, then $d^\eta(x) = 1$, else $d^\eta(x) = 1/f^i(x)$. 

8
As noted in the previous sections, the matrices $A$ and $B$ are sparse in nature with constant diagonal entries. As we are finding derivatives with respect to the position of sphere $\Omega_i$, i.e., $x_k$, we have the following cases which gives non-zero contribution

1. $j \in N_i$ and $i = k$ (see Subfig. 3a);
2. $j \in N_i$ and $j = k$ (see Subfig. 3b);
3. $j \in N_i$ and $k \in N_i$ and $k \neq j$ (see Subfig. 3c and 3d).

Fig. 3 shows the aforementioned cases. Looking at the matrix entries for $A$ and $B$ we note that we have three terms depending on the position, namely $\omega^\eta_{ij}(x^n_k)$, $Y^\ell_{mr}(x^n_k)$, and $r^k_i(x^n_k)$ for matrix $A$; and $t^i_j(x^n_k)$ for matrix $B$.

For abbreviation, we denote $\nabla x_k$ by $\nabla^k$ in the following content. The derivative of $\omega^\eta_{ij}(x^n_k)$ is given by

$$\nabla^k (\omega^\eta_{ij}(x^n_k)) = \begin{cases} 
  d^i(x^n_k) \left[ \chi^\eta_n \left( r^i_1(x^n_k) \right) \frac{e^j(x^n_k)}{r_j} - \delta_{f_n^i > 1} \omega^\eta_{ij}(x^n_k) Z_n^i \right] & \text{if (} j \in N_i \cap (k = i) \), \\
  -d^i(x^n_k) \chi^\eta_n \left( r^i_1(x^n_k) \right) \frac{e^j(x^n_k)}{r_j} \left( 1 - \delta_{f_n^i > 1} \omega^\eta_{ij}(x^n_k) \right) & \text{if (} j \in N_i \cap (k = j) \), \\
  \omega^\eta_{ij}(x^n_k) \delta_{f_n^i > 1} d^i(x^n_k) \chi^\eta_n \left( r^i_1(x^n_k) \right) \frac{e^k(x^n_k)}{r_k} & \text{if (} j \in N_i \cap (k \in N_i \land k \neq j) \), \\
  0 & \text{else,}
\end{cases} \quad (35)$$

where

$$e^j(x) := (x - x_j)/|x - x_j|, \quad Z_n^i := \sum_{k_0 \in N_i} \chi^\eta_n \left( r^k_1(x^n_k) \right) e^{k_0}(x^n_k)/r_{k_0}, \quad f^i_n := f^i(x^n_k),$$

and

$$\delta_{f_n^i > 1} = \begin{cases} 
  1 & \text{if } f_n^i > 1, \\
  0 & \text{if } f_n^i \leq 1.
\end{cases}$$
Further, the derivative of $Y_{\ell m}^j(x_i^n)$ is given by

$$\nabla^k \left( Y_{\ell m}^j(x_i^n) \right) = (\nabla Y_{\ell m'}) \left( \frac{x_i^n - x_j}{|x_i^n - x_j|} \right) \nabla^k \frac{x_i^n - x_j}{|x_i^n - x_j|}$$

$$= \begin{cases} 
(\nabla Y_{\ell m'})^2 \left( \frac{x_i^n}{|x_i^n|} \right) \frac{1}{r_{j'}} & \text{if } (j \in N_i) \land (k = i), \\
- (\nabla Y_{\ell m'})^2 \left( \frac{x_i^n}{|x_i^n|} \right) \frac{1}{r_{j'}} & \text{if } (j \in N_i) \land (k = j), \\
0 & \text{else.}
\end{cases} \quad (36)$$

We now show the details for derivation of Eq. (36). Note that $\forall x = (x_1, x_2, x_3)$, we have

$$\nabla_x \left( \frac{x}{|x|} \right) = \frac{1}{|x|^3} \begin{pmatrix} |x|^2 - x_1^2 & -x_1x_2 & -x_1x_3 \\ -x_1x_2 & |x|^2 - x_2^2 & -x_2x_3 \\ -x_1x_3 & -x_2x_3 & |x|^2 - x_3^2 \end{pmatrix} \quad (37)$$

and

$$\nabla_x Y_{\ell m'} \left( \frac{x}{|x|} \right) \cdot x = 0, \quad (38)$$

which yield that

$$\nabla_x Y_{\ell m'} \left( \frac{x}{|x|} \right) \nabla_x \left( \frac{x}{|x|} \right) = \nabla_x Y_{\ell m'} \left( \frac{x}{|x|} \right) \frac{1}{|x|} \quad (39)$$

The equation Eq. (36) is then followed.

Lastly, we have the derivatives of the radial scaling $r_{\ell}^j(x_i^n)$ given by

$$\nabla^k \left( r_{\ell}^j(x_i^n) \right) = \begin{cases} 
\frac{e^{j}(x_i^n)\ell' r_{\ell-1}^j(x_i^n)}{r_{j}} & \text{if } (j \in N_i) \land (k = i), \\
- \frac{e^{j}(x_i^n)\ell' r_{\ell-1}^j(x_i^n)}{r_{j}} & \text{if } (j \in N_i) \land (k = j), \\
0 & \text{else},
\end{cases} \quad (40)$$

and the Bessel scaling, $\hat{r}_{\ell}^j(x_i^n)$ which is given by

$$\nabla^k \left( \hat{r}_{\ell}^j(x_i^n) \right) = \begin{cases} 
\frac{\hat{r}_{\ell}^j(|x_i^n - x_j|)e^{j}(x_i^n)}{1_{\ell'}(r_{j})} & \text{if } (j \in N_i) \land (k = i), \\
- \frac{\hat{r}_{\ell}^j(|x_i^n - x_j|)e^{j}(x_i^n)}{1_{\ell'}(r_{j})} & \text{if } (j \in N_i) \land (k = j), \\
0 & \text{else.}
\end{cases} \quad (41)$$

Collecting all the terms, we can compute the derivatives of the $(i\ell m, j\ell' m')$th element of matrix $A$ and $B$.

In the case of $i = j$,

$$\nabla^k [A_{i\ell m}]^{mm'} = 0, \quad \nabla^k [B_{i\ell m}]^{mm'} = 0. \quad (42)$$

We now consider different cases for $i \neq j$ as follows.
1. Case \( j \in N_i \) and \( k = i \) (Subfig. 3a):

\[
\nabla^k [A_{ij}]^{m'm'} = -\sum_{n=1}^{N_{nh}} \omega_n Y_{\ell m}(s_n) \left[ r_i^j(x_i^n) Y_{\ell'm'}(x_i^n) d(x_i^n) \right] \\
\times \left\{ \frac{1}{r_j} \sum_{r_j} \chi_{ij}^m \left( r_i^j(x_i^n) \right) e^j(x_i^n) - \delta_{j \neq 1} \omega_j^0(x_i^n) Z_{p_i}^n \right\} \\
+ \omega_j^0(x_i^n) \frac{r_{ij}^j}{r_j} \left\{ \ell' Y_{\ell'm'}(x_i^n) e^j(x_i^n) + (\nabla Y_{\ell'm'})^j(x_i^n) \right\} ,
\]

and

\[
\nabla^k [B_{ij}]^{m'm'} = -\sum_{n=1}^{N_{nh}} \omega_n Y_{\ell m}(s_n) \left[ \hat{r}_j^j(x_i^n) Y_{\ell'm'}(x_i^n) d(x_i^n) \right] \\
\times \left\{ \frac{1}{r_j} \sum_{r_j} \chi_{ij}^m \left( \hat{r}_j^j(x_i^n) \right) e^j(x_i^n) - \delta_{j \neq 1} \omega_j^0(x_i^n) Z_{p_i}^n \right\} \\
+ \omega_j^0(x_i^n) \frac{r_{ij}^j}{r_j} \left\{ \ell' Y_{\ell'm'}(x_i^n) e^j(x_i^n) + (\nabla Y_{\ell'm'})^j(x_i^n) \right\} .
\]

2. Case \( j \in N_i \) and \( k = j \) (Subfig. 3b):

\[
\nabla^k [A_{ij}]^{m'm'} = \sum_{n=1}^{N_{nh}} \omega_n Y_{\ell m}(s_n) \left[ r_i^j(x_i^n) Y_{\ell'm'}(x_i^n) d(x_i^n) \right] \\
\times \left\{ 1 - \delta_{j \neq 1} \omega_j^0(x_i^n) \right\} \\
+ \omega_j^0(x_i^n) \frac{r_{ij}^j}{r_j} \left\{ \ell' Y_{\ell'm'}(x_i^n) e^j(x_i^n) + (\nabla Y_{\ell'm'})^j(x_i^n) \right\} ,
\]

and

\[
\nabla^k [B_{ij}]^{m'm'} = \sum_{n=1}^{N_{nh}} \omega_n Y_{\ell m}(s_n) \left[ \hat{r}_j^j(x_i^n) Y_{\ell'm'}(x_i^n) d(x_i^n) \right] \\
\times \left\{ 1 - \delta_{j \neq 1} \omega_j^0(x_i^n) \right\} \\
+ \omega_j^0(x_i^n) \frac{r_{ij}^j}{r_j} \left\{ \ell' Y_{\ell'm'}(x_i^n) e^j(x_i^n) + (\nabla Y_{\ell'm'})^j(x_i^n) \right\} .
\]

3. Case \( (j \in N_i) \land (k \in N_i \land k \neq j) \) (Subfig. 3c and 3d):

\[
\nabla^k [A_{ij}]^{m'm'} = -\sum_{n=1}^{N_{nh}} \omega_n Y_{\ell m}(s_n) r_i^j(x_i^n) Y_{\ell'm'}(x_i^n) \omega_j^0(x_i^n) \omega_j^0(x_i^n) \\
\times \delta_{j \neq 1} d^j(x_i^n) \chi_{ij}^m \left( r_i^j(x_i^n) \right) \frac{e^j(x_i^n)}{r_k} ,
\]

11
and

\[ \nabla^k [B_{ij}]_{i'j'} = - \sum_{n=1}^{N_{nb}} \omega_n Y_{\ell m}(s_n) Y_{\ell' m'}(x^n_i) Y_{\ell' m'}(x^n_i) \omega_{ij}(x^n_i) \]

\[ \delta f_{l>1} d^k(x^n_i) \chi_0(r^k_1(x^n_i)) \frac{e^k(x^n_i)}{r_k}. \]

### 3.1.2 Dense matrices C₁ and C₂

Now, we move our attention towards the computation of derivatives for the matrices C₁ and C₂. We compute the derivative of C₁ and C₂ together, i.e., we consider

\[ [C_1 X_r + C_2 X_e]_{i'j'} = [C_1 X_r + C_2 X_e]_{i'j'}, \]

where the \((ilm)\)th entry of \([C_1 X_r + C_2 X_e]\) is given by:

\[ [C_1 X_r + C_2 X_e]_{i'j'} = \sum_{j=1}^{M} \sum_{\ell'} \sum_{m'=1}^{N_{nb}} \omega_n Y_{\ell m}(s_n) [Q]_{i'j'} \nabla^{k} \chi_i(x^n_i) [X]_{j\ell m'} \]

\[ \times [X]_{j\ell m'} \nabla^{k} \chi_i(x^n_i) + \chi_i(x^n_i) [Q]_{i'j'} \nabla^{k} \chi_i(x^n_i). \]

We note that we have two terms depending on \(x_k\), i.e., \(\chi_i(x^n_i)\) and \([Q]_{i'j'}\). Unlike for matrices A and B, we have non-trivial contributions on the diagonal as well. We divide the computation of derivative of Eq. (44) into two parts with help of the product rule as follows

\[ \nabla^k [C_1 X_r + C_2 X_e]_{i'j'} = \sum_{j=1}^{M} \sum_{\ell'} \sum_{m'=1}^{N_{nb}} \omega_n Y_{\ell m}(s_n) \left[ \frac{\nabla^{k} \chi_i(x^n_i) [X]_{j\ell m'}}{r_{j}} - \frac{1}{r_{j}} [X]_{j\ell m'} \right] \]

\[ \times \left[ [Q]_{i'j'} \nabla^{k} \chi_i(x^n_i) + \chi_i(x^n_i) \nabla^{k} [Q]_{i'j'} \right]. \]

**Derivative of \(\chi_i(x^n_i)\).** The first contribution is the derivative of \(\chi_i(x^n_i)\) when keeping \(Q\) as constant. The non zero contribution comes when \(k = i\) or \(k = N_i\). Combining (34) and (35), we have

\[ \nabla^k (\chi_i(x^n_i)) = - \sum_{j \in N_i} \nabla^k (\omega_{ij}(x^n_i)) \]

\[ \begin{cases} 
[(1 - \chi_i(x^n_i)) \delta f_{l>1} - 1] d^k(x^n_i) Z_i^n & \text{if } k = i, \\
[1 - (1 - \chi_i(x^n_i)) \delta f_{l>1}] d^k(x^n_i) \chi_i(r^k_1(x^n_i)) \frac{e^k(x^n_i)}{r_k} & \text{if } k \in N_i, \\
0 & \text{else},
\end{cases} \]

\[ \begin{cases} 
-\delta f_{l \leq 1} Z_i^n & \text{if } k = i, \\
\delta f_{l \leq 1} \chi_i(r^k_1(x^n_i)) \frac{e^k(x^n_i)}{r_k} & \text{if } k \in N_i, \\
0 & \text{else}.
\end{cases} \]

Here we use the fact that if \(f^n_n > 1\), then \(\chi_i(x^n_i) = 0\); if \(f^n_n \leq 1\), then \(d^k(x^n_i) = 1\).

**Derivative of \([Q]_{i'j'}\).** The second contribution comes from the derivatives of matrix Q. The entries are given by Eq. (18).
In this matrix we note that three terms depend on the position namely, $[P_{X_0}^{\eta}]_{\ell,m}$, $k_i^j (x_i^n)$, and $Y_{\ell \alpha \phi}^j (x_0^n)$. To be precise, we have

$$\nabla k^k (x_0^n) = \sum_{\ell \alpha \phi} C_{i k} \left( \nabla^k \left[ P_{X_0}^{\eta}]_{\ell,m} k_i^j (x_i^n) Y_{\ell \alpha \phi}^j (x_0^n) \right] + \left[ P_{X_0}^{\eta}]_{\ell,m} \nabla^i k_i^j (x_i^n) Y_{\ell \alpha \phi}^j (x_0^n) \right] + \left[ P_{X_0}^{\eta}]_{\ell,m} k_i^j (x_i^n) \nabla^k Y_{\ell \alpha \phi}^j (x_0^n) \right].$$

The non-zero contribution of the derivative for $k_i^j (x_i^n)$ and $Y_{\ell \alpha \phi}^j (x_0^n)$ comes when $k = i$ or $k = j$. The derivative of $k_i^j (x_i^n)$ is given by:

$$\nabla^k \left( k_i^j (x_i^n) \right) = \begin{cases} k_i^j (x_i^n) - \frac{k_i^k (x_i^n) e^i (x_i^n)}{\ell_0 (r_j)} & \text{if } k = i, \\ -\frac{e^k (x_i^n) - \frac{k_i^k (x_i^n) e^j (x_i^n)}{\ell_0 (r_j)} & \text{if } k = j, \end{cases}$$

while the derivative of $Y_{\ell \alpha \phi}^j (x_0^n)$ is already given by Eq. \ref{eq36} with $\ell', m'$ replaced by $\ell_0, m_0$. The final contribution comes from the derivative of $[P_{X_0}^{\eta}]_{\ell,m}$. We have the computation of

$$\nabla^k \left[ P_{X_0}^{\eta}]_{\ell,m} = \sum_{n_0} \omega_{n_0} \nabla^k \left( \chi_j^n (x_j^n) \right) Y_{\ell \alpha \phi}^j (s_{n_0}) Y_{\ell' m'} (s_{n_0}),$$

where the derivative of $\chi_j^n (x_j^n)$ is given by Eq. \ref{eq45} with $i, n$ replaced by $j, n_0$.

### 3.1.3 Right-hand side $G_0$ and $F_0$

The final derivatives we require are those of the right-hand side $G_0$ and $F_0$. In Eq. \ref{eq21} we have two terms depending on $x_i^n$; $\chi_i^n (x_i^n)$ and $\psi_0 (x_i^n)$. The derivatives of $\chi_i^n (x_i^n)$ is given by Eq. \ref{eq45} and the derivative of $\psi_0 (x_i^n)$ is given by

$$\nabla^k \left( \psi_0 (x_i^n) \right) = \begin{cases} -\frac{1}{\varepsilon_1} \sum_{j=1}^M q_j e^j (x_i^n) \frac{e^i (x_i^n)}{|x_i^n - x_j|^2} & \text{if } k = i, \\ \frac{q_k e^k (x_i^n)}{\varepsilon_1 |x_i^n - x_k|^2} & \text{if } k = j. \end{cases}$$

Next we move towards the computation of derivative for $F_0$. We note that the entries of $F_0$ are very similar to the entries of $[C_1 X_0 + C_2 X_0]$, with only the addition of the term $\partial_n \psi_0 (x_i^n)$. The computation of other terms namely, $\chi_i^n (x_i^n)$, $k_i^j (x_i^n)$, and $Y_{\ell \alpha \phi}^j (x_0^n)$ has been taken before. The derivatives of $\partial_n \psi_0$ is given by

$$\nabla^k \left( \partial_n \psi_0 (x_i^n) \right) = \begin{cases} \sum_{j=1, j \neq i}^M q_j 3 (x_i^n - x_j) (x_i^n - x_j)^T \frac{3 (x_i^n - x_k)}{|x_i^n - x_k|^3} - \frac{3 (x_i^n - x_j)}{|x_i^n - x_j|^3} \frac{1_3}{n} & \text{if } k = i, \\ -q_k 3 (x_i^n - x_k) (x_i^n - x_k)^T \frac{3 (x_i^n - x_k)}{|x_i^n - x_k|^3} - \frac{3 (x_i^n - x_j)}{|x_i^n - x_j|^3} \frac{1_3}{n} & \text{if } k \neq i, \end{cases}$$

where $1_3$ is the identity matrix of size $3 \times 3$ and $n = s_n$ at $x_i^n$ is the unit normal derivative.

The computation of forces can be summarized as follows:

1. Solve Eq. \ref{eq11} to get the reaction potential $X_r$ and the extended potential $X_e$.
2. Solve Eq. \ref{eq29} to get the adjoint solution $X_{\text{adj}}$.
3. Compute the analytical derivatives of the matrix $L$ and the right-hand side $g$ with respect to a parameter $\lambda$.
4. Contract the analytical derivatives with the adjoint solution to get the forces.
| PDB Code | Number of Atoms (M) | Name                     | Reference  |
|----------|---------------------|--------------------------|------------|
| 1ay3     | 25                  | nodularin                | [ALM+96]   |
| 1etn     | 180                 | enterotoxin              | [OSK+96]   |
| 1du9     | 380                 | scorpion toxin           | [XWP+00]   |
| 1gzz     | 944                 | insulin-like growth factor | [HDD+02]   |
| 1jvu     | 3964                | ribonuclease A           | [VMZM01]   |
| 1qjt     | 9046                | EH1 domain               | [WTC+99]   |

Table 1: Information about the input structures.

4 Numerical Simulations

In this section, we present the numerical studies for the computation of forces. Before presenting the examples, we would like to mention some details on solving the system (1) and (27). We follow the same ideas as prescribed in [QSM19]. We transform the system (1) into a fixed point iteration technique where the $\nu$th iterative step is given by

$$
\begin{bmatrix}
A & 0 \\
0 & B
\end{bmatrix}
\begin{bmatrix}
X^{(\nu)}_r \\
X^{(\nu)}_e
\end{bmatrix} =
\begin{bmatrix}
G_0 + F_0 \\
F_0
\end{bmatrix} -
\begin{bmatrix}
C_1 & C_2 \\
C_1 & C_2
\end{bmatrix}
\begin{bmatrix}
X^{(\nu-1)}_r \\
X^{(\nu-1)}_e
\end{bmatrix}
$$

for $\nu = 1, 2, 3, \ldots$. (49)

We refer to this as outer iteration with initial conditions $\begin{bmatrix} X^{(0)}_r \\ X^{(0)}_e \end{bmatrix} = 0$. Applying the preconditioner requires as well to solve two linear systems, and we refer to them as micro-iterations as this is also performed in an iterative manner. For the two inner linear systems, we use as guess the solution of the previous macro-iterations, or zero if solving them for the first time.

For each linear system, the stopping criterion is on the relative increment of the solution, i.e.,

$$
\frac{\|X^{(\nu)} - X^{(\nu-1)}\|_{\infty}}{\|X^{(\nu)}\|_{\infty}} \leq \text{tol},
$$

where $\|\cdot\|_{\infty}$ is the $\ell^\infty$-norm of the corresponding vector. However, we use two different tolerances for the macro and micro-iterations, namely, the inner tolerance is equal to the outer tolerance divided by 100.

The code was tested on a set of input structures with different number of atoms, spanning from $10^1$ to $10^4$ atoms. We prepared the input structures using the tool PDB2PQR provided in the APBS software package [JES+17], with the AMBER force field to assign the atomic partial charges [PC03]. The radii were assigned in a subsequent step, according to a definition of a solvent accessible surface (SAS): for each atom we set its radius to its value as reported in ref. [Bon64] plus a contribution from the effective size of the solvent (1.4 Å for water). Table 1 reports detailed information about the structures.

4.1 Comparison between ddLPB and APBS

Once the structures were ready, we performed a series of calculation using both APBS and ddLPB. All the calculations were performed on a server equipped with four Intel(R) Xeon(R) Gold 6140M running at 2.30 GHz, for a total of 72 cores, and 1.2 TB of RAM.

We set the (relative) dielectric constant of the solute’s region to 1 (vacuum) and the dielectric constant of the environment to 78.54 (water). We included two ions of charge $+1$ and $-1$, both in concentration 0.1 M, which combined with a temperature of 298.15 K, correspond to $\kappa = 0.104$. For what concerns APBS, the calculations were performed using the box provided by PDB2PQR (keyword grid), which is enough to contain the structures, and a number of grid points (keyword grid) suitable for the multigrid algorithm, hence calculated using

$$
n = e^{2^{\ell+1}} + 1,
$$

(51)
where \( n \) is the number of grid points along a given dimension, \( \ell \) is the depth of the multilevel solver (keyword \texttt{nlev}), and \( c \) is an arbitrary integer. We choose \( c \) such that, with \( \ell = -4 \), a certain target density of points is achieved. In the following discussion, we report the actual density of points computed with \( \ell = -4 \) and as an average over the three dimensions. Finally, the remaining relevant keywords are \texttt{chgm = spl4, bcfl = mdh, srad = 0.0, and swin = 0.3}.

For what concerns ddLPB, we set the tolerance \( \texttt{tol} = 10^{-6} \), the number of Lebedev grid points to 302, and the smooth-switching window using \( \eta = 0.1 \). The maximum degree of spherical harmonics is set to values between 2 and 12, to study the convergence of the results.

We compare the energies obtained from APBS and ddLPB for the molecules presented in Table 1. Since there is no exact energy, we compute the reference energy of APBS from linear extrapolation and the one of ddLPB from exponential fitting respectively (see \[QSM19\] for details). The energy and memory w.r.t. different discretization parameters are illustrated in Figure 4 and Tables 2-3. It can be observed that for each molecule, the difference between two reference energies is less than 1\%, which actually validates the energy computation of ddLPB. Table 2 and 3 give the numerical values for the solvation energy and memory presented in Figure 4, as well as the number of macro-iterations. Solving the linear system requires a relatively low number of macro-iterations, thus making the method particularly efficient. Furthermore, the number of macro-iterations is stable regardless of the input structure, suggesting that the method retains its efficiency even on systems different than those presented in this benchmark. The memory usage is quadratic in \( \ell \text{max} \) and linear in the number of atoms, the latter is a requirement for applying the method to very large systems. On the other hand, APBS has memory requirements linear with respect to the system’s volume and cubic in the grid point density. For both the codes, increasing the accuracy results in a high memory usage, however the lower scaling of ddLPB memory requirements makes it possible to achieve higher accuracies. As an example, for the intermediate-size system 1du9, by using ddLPB it is possible to achieve an accuracy of \( \approx 0.2\% \) with a memory usage of 7.7 GB, whereas the same accuracy cannot be achieved using APBS due to a too high memory usage. The same finding holds also for the larger systems.

### 4.2 Numerical validation of analytical forces

The analytical forces computed by Eq. \((52)\) have been tested against numerical forces done through finite differences. The numerical forces are evaluated using the following definition

\[
D_h[E_s](\lambda) := \frac{E_s(\lambda + h) - E_s(\lambda)}{h}.
\]

(52)

Here, \( \lambda \) is a generic parameter, for instance one component of a nuclear coordinate, and \( 0 < h \ll 1 \) is a small step size.

For the numerical test, we selected the two smallest structure (1ay3, 1etn) and we computed all the numerical derivatives with respect to the nuclear coordinates using Eq. \((52)\), for various finite step sizes. Due to high computational cost related to the repeated number of calculations, we used a coarser discretization: maximum degree of spherical harmonics 2, 110 Lebedev grid points, convergence set to \( 10^{-6} \).

Due to the finite difference approximation of the analytical derivative we expect a first-order convergence of

\[
\text{Err}_{j,\alpha}(h) := D_h[E_s](x_{j,\alpha}) - \frac{\partial E_s}{\partial x_{j,\alpha}}, \quad \text{with} \quad x_j = (x_{j,1}, x_{j,2}, x_{j,3})^T,
\]

with respect to \( h \). Note that the force acting on the component \( \alpha = 1, 2, 3 \) of nuclei \( j \) due to the solvation model is given by \( F_{j,\alpha} = -\frac{\partial E_s}{\partial x_{j,\alpha}} \).

Figure 5 illustrates the convergence of the maximum (\( \ell^\infty \)-error) and the root-mean-squared deviation (RMSD), or equivalently the \( \ell^2 \)-error, of the error vector \( \text{Err} \) as a function of \( h \) and first-order convergence is indeed observed. However, beyond \( h = 10^{-5} \), the finite precision of the algorithms interferes with the convergence of the numerical forces.

Computing the finite-difference approximation \( D_h[E_s](x_{j,\alpha}) \) with respect to every coordinate becomes very expensive for large structures, which is the reason we restrict this benchmark and verification to the
Figure 4: Solvation energies computed using ddLPB and using APBS with different discretizations. The gray lines report the converged energy and a ±1% interval from it.
| PDB Code | ddLPB | | | APBS |
|----------|-------|-------|-------|-------|
| 1ay3     |       |       |       |       |
| 1etn     |       |       |       |       |
| 1du9     |       |       |       |       |
| \( \ell_{\text{max}} \) | Energy (kJ/mol) | Rel. En. | Mem. (GB) | Iter. | Energy (kJ/mol) | Rel. En. | Mem. (GB) |
| 2        | -126.591 | 0.0323 | 0.0347 | 5 | 0.4353 | -134.7127 | 0.0302 | 0.0215 |
| 3        | -128.4347 | 0.0182 | 0.0463 | 6 | 0.0127 | -134.0656 | 0.0253 | 0.0218 |
| 4        | -129.5154 | 0.0127 | 0.0416 | 6 | 0.2213 | -132.6796 | 0.0147 | 0.0638 |
| 5        | -129.6667 | 0.0088 | 0.0473 | 6 | 0.1697 | -132.3780 | 0.0124 | 0.1244 |
| 6        | -129.9653 | 0.0065 | 0.0569 | 6 | 0.1333 | -131.9791 | 0.0093 | 0.2448 |
| 7        | -130.1668 | 0.0050 | 0.0752 | 6 | 0.0900 | -131.5849 | 0.0063 | 0.7906 |
| 8        | -130.3308 | 0.0037 | 0.0922 | 6 |       |           |       |       |
| 9        | -130.4356 | 0.0029 | 0.1249 | 6 |       |           |       |       |
| 10       | -130.5462 | 0.0021 | 0.1525 | 6 |       |           |       |       |
| 12       | -130.6886 | 0.0010 | 0.2873 | 6 |       |           |       |       |
| fit      | -130.8167 |       |       |       | fit | -130.7595 |       |       |
| 2        | -513.7158 | 0.0275 | 0.0506 | 5 | 0.4647 | -534.1626 | 0.0144 | 0.0651 |
| 3        | -519.2851 | 0.0170 | 0.0741 | 6 | 0.4070 | -534.1284 | 0.0143 | 0.0971 |
| 4        | -521.7994 | 0.0122 | 0.1759 | 6 | 0.2820 | -531.2851 | 0.0089 | 0.2921 |
| 5        | -523.3340 | 0.0093 | 0.2157 | 6 | 0.1757 | -529.5239 | 0.0056 | 0.9963 |
| 6        | -524.3216 | 0.0075 | 0.3652 | 7 | 0.1417 | -529.0144 | 0.0046 | 1.8298 |
| 7        | -525.1337 | 0.0059 | 0.5984 | 7 | 0.0973 | -528.2927 | 0.0032 | 6.1708 |
| 8        | -525.7536 | 0.0048 | 0.9038 | 7 |       |           |       |       |
| 9        | -526.2919 | 0.0037 | 1.3659 | 7 |       |           |       |       |
| 10       | -526.6967 | 0.0030 | 1.9110 | 7 |       |           |       |       |
| 12       | -527.2242 | 0.0020 | 3.6499 | 8 |       |           |       |       |
| fit      | -528.2632 |       |       |       | fit | -526.5857 |       |       |
| 2        | -1193.2787 | 0.0413 | 0.0713 | 6 | 0.4450 | -1264.5724 | 0.0206 | 0.2115 |
| 3        | -1214.4131 | 0.0243 | 0.1243 | 6 | 0.2680 | -1254.2918 | 0.0123 | 0.8418 |
| 4        | -1224.2947 | 0.0164 | 0.2358 | 6 | 0.1837 | -1249.6227 | 0.0086 | 2.6162 |
| 5        | -1230.0184 | 0.0118 | 0.4473 | 7 | 0.1440 | -1247.4550 | 0.0068 | 5.5273 |
| 6        | -1233.5552 | 0.0089 | 0.7642 | 7 | 0.0960 | -1244.4098 | 0.0043 | 20.4569 |
| 7        | -1236.2257 | 0.0068 | 1.2134 | 7 |       |           |       |       |
| 8        | -1238.2298 | 0.0052 | 1.8665 | 8 |       |           |       |       |
| 9        | -1239.6524 | 0.0040 | 2.8115 | 8 |       |           |       |       |
| 10       | -1240.8053 | 0.0031 | 4.0202 | 8 |       |           |       |       |
| 12       | -1242.3783 | 0.0019 | 7.0228 | 8 |       |           |       |       |
| fit      | -1244.6815 |       |       |       | fit | -1239.0272 |       |       |

Table 2: Solvation energy, memory, and number of iterations for ddLPB and APBS. "Rel. En." stands for the relative energy, "Mem." stands for the memory, "Iter." stands for the number of macro-iterations, and "h" stands for the grid spacing (APBS).
| PDB Code | ddLPB | APBS |
|----------|-------|------|
| ℓ<sub>max</sub> | Energy (kJ/mol) | Rel. En. | Mem. (GB) | Iter. | h (Å) | Energy (kJ/mol) | Rel. En. | Mem. (GB) |
| lgzz (940) | 2 | 1519.2354 | 0.0476 | 0.1456 | 6 | 0.5377 | -1628.9437 | 0.0272 | 0.3676 |
| | 3 | 1549.8266 | 0.0284 | 0.3078 | 7 | 0.4213 | -1617.3532 | 0.0199 | 0.7047 |
| | 4 | 1564.6469 | 0.0191 | 0.6345 | 7 | 0.2820 | -1607.4435 | 0.0136 | 2.2714 |
| | 5 | 1572.8942 | 0.0140 | 1.2059 | 8 | 0.1910 | -1601.1297 | 0.0096 | 7.5833 |
| | 6 | 1578.0656 | 0.0107 | 2.1281 | 8 | 0.1447 | -1597.5460 | 0.0074 | 18.8807 |
| | 7 | 1582.1489 | 0.0082 | 3.5272 | 8 |  |  |  |
| | 8 | 1585.2221 | 0.0062 | 5.5394 | 8 |  |  |  |
| | 9 | 1587.3698 | 0.0049 | 8.3408 | 8 |  |  |  |
| | 10 | 1589.1164 | 0.0038 | 12.0967 | 8 |  |  |  |
| | 12 | 1591.3968 | 0.0024 | 23.3188 | 8 |  |  |  |
| fit | -1595.1571 | fit | -1585.8625 | |
| ljvu (3964) | 2 | 6273.8845 | 0.0402 | 0.5364 | 6 | 0.5480 | -6635.4317 | 0.0187 | 1.2473 |
| | 3 | 6381.7231 | 0.0237 | 1.3233 | 8 | 0.4450 | -6619.2091 | 0.0162 | 2.2848 |
| | 4 | 6438.8201 | 0.0149 | 2.9063 | 8 | 0.2840 | -6577.4270 | 0.0098 | 9.1260 |
| | 5 | 6469.8886 | 0.0102 | 5.7337 | 9 | 0.1953 | -6557.6254 | 0.0067 | 31.0542 |
| | 6 | 6490.4115 | 0.0070 | 10.2830 | 9 |  |  |  |
| | 7 | 6505.7527 | 0.0047 | 17.1820 | 9 |  |  |  |
| | 8 | 6511.1894 | 0.0030 | 27.1599 | 9 |  |  |  |
| | 9 | 6525.6799 | 0.0017 | 41.0288 | 10 |  |  |  |
| | 10 | 6532.4230 | 0.0006 | 59.6610 | 10 |  |  |  |
| | 12 | 6541.7062 | 0.0008 | 114.8609 | 12 |  |  |  |
| fit | 6536.4728 | fit | 6513.8665 | |
| lggt (9046) | 2 | 18411.4422 | 0.0244 | 1.2203 | 6 | 0.5690 | -19075.6126 | 0.0122 | 2.9358 |
| | 3 | 18603.4737 | 0.0142 | 3.1045 | 8 | 0.4840 | -19041.8281 | 0.0104 | 4.7910 |
| | 4 | 18701.0899 | 0.0090 | 6.9332 | 9 | 0.2900 | -18962.9862 | 0.0062 | 24.8033 |
| | 5 | 18757.4013 | 0.0060 | 13.7357 | 10 |  |  |  |
| | 6 | 18793.3707 | 0.0041 | 24.7674 | 10 |  |  |  |
| | 7 | 18819.8066 | 0.0027 | 41.5118 | 11 |  |  |  |
| | 8 | 18839.1510 | 0.0017 | 65.6922 | 11 |  |  |  |
| | 9 | 18853.5883 | 0.0009 | 99.2852 | 11 |  |  |  |
| | 10 | 18864.8898 | 0.0003 | 144.5051 | 11 |  |  |  |
| | 12 | 18880.1518 | 0.0005 | 279.6300 | 11 |  |  |  |
| fit | 18871.2924 | fit | 18845.8852 | |

Table 3: Solvation energy, memory, and number of iterations for ddLPB and APBS. "Rel. En." stands for the relative energy, "Mem." stands for the memory, "Iter." stands for the number of macro-iterations, and "h" stands for the grid spacing (APBS).
5 Conclusion

In this work, we provide the detailed derivation of analytical forces for the ddLPB numerical method which efficiently approximates solutions to the linearized Poisson-Boltzmann equation that is a frequent model used in computational (bio-) chemistry. The derivation is technical but mandatory and is based on an adjoint method to compute analytical derivatives of the energy with respect to (possibly many) external parameters such as the nuclear coordinates which result in the computation of the forces. The implementation of the energy and forces have been validated by a series of benchmark problems and by comparing the results with those of the APBS-package. The current implementation scales quadratically w.r.t. the number of atoms and it is work in progress to accelerate the quadratic bottlenecks with the fast multipole method (FMM) to achieve a fully linearly scaling ddLPB-implementation for energy and forces.

Acknowledgements

CQ is supported by NSFC Grant 11901281, the Guangdong Basic and Applied Basic Research Foundation (2020A1515010336), and the Stable Support Plan Program of Shenzhen Natural Science Fund (Program Contract No. 20200925160747003). AJ and BS are acknowledging support by the German Research Foundation (DFG) under project 440641818. MN gratefully acknowledges funding from Gaussian, Inc.

References

[ABWT09] Michael D. Altman, Jaydeep P. Bardhan, Jacob K. White, and Bruce Tidor. Accurate solution of multi-region continuum biomolecule electrostatic problems using the linearized Poisson-
Boltzmann equation with curved boundary elements. *Journal of Computational Chemistry*, 30(1):132–153, January 2009.

[Ada75] R. A. Adams. *Sobolev spaces*. Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York-London, 1975. Pure and Applied Mathematics, Vol. 65.

[ALM+96] Arto Annila, Jaana Lehtimäki, Kimmo Mattila, John E. Eriksson, Kaarina Sivonen, Tapio T. Rantala, and Torbjörn Drakenberg. Solution structure of nodularin. *Journal of Biological Chemistry*, 271(28):16695–16702, July 1996.

[BC98] Vincenzo Barone and Maurizio Cossi. Quantum calculation of molecular energies and energy gradients in solution by a conductor solvent model. *The Journal of Physical Chemistry A*, 102(11):1995–2001, January 1998.

[BCR11] Chandrajit Bajaj, Shun-Chuan Chen, and Alexander Rand. An efficient higher-order fast multipole boundary element solution for Poisson–Boltzmann-based molecular electrostatics. *SIAM Journal on Scientific Computing*, 33(2):826–848, January 2011.

[BDD+02] Andrzej M. Brzozowski, Eleanor J. Dodson, Guy G. Dodson, Garib N. Murshudov, Chandra Verma, Johan P. Turkenburg, Frederik M. de Bree, and Zbigniew Dauter. Structural origins of the functional divergence of human insulin-like growth factor-i and insulin. *Biochemistry*, 41(30):9389–9307, July 2002.

[BFZ02] Alexander H. Boschitsch, Marcia O. Fenley, and Huan-Xiang Zhou. Fast boundary element method for the linear Poisson-Boltzmann equation. *The Journal of Physical Chemistry B*, 106(10):2741–2754, February 2002.

[Bon64] A. Bondi. van der Waals volumes and radii. *The Journal of Physical Chemistry*, 68(3):441–451, March 1964.

[BSJ+01] N. A. Baker, D. Sept, S. Joseph, M. J. Holst, and J. A. McCammon. Electrostatics of nanosystems: Application to microtubules and the ribosome. *Proceedings of the National Academy of Sciences*, 98(18):10037–10041, August 2001.

[CCC+10] Duan Chen, Zhan Chen, Changjun Chen, Weihua Geng, and Guo-Wei Wei. MIBPB: A software package for electrostatic analysis. *Journal of Computational Chemistry*, 32(4):756–770, September 2010.

[Chi99] Daniel M. Chipman. Simulation of volume polarization in reaction field theory. *The Journal of Chemical Physics*, 110(16):8012–8018, April 1999.

[Chi06] Daniel M. Chipman. New formulation and implementation for volume polarization in dielectric continuum theory. *The Journal of Chemical Physics*, 124(22):224111, June 2006.

[CHX07] Long Chen, Michael J. Holst, and Jinchao Xu. The finite element approximation of the nonlinear Poisson–Boltzmann equation. *SIAM Journal on Numerical Analysis*, 45(6):2298–2320, January 2007.

[CMS13] Eric Cancès, Yvon Maday, and Benjamin Stamm. Domain decomposition for implicit solvation models. *The Journal of Chemical Physics*, 139(5):054111, August 2013.

[CMT97] E. Cancès, B. Mennucci, and J. Tomasi. A new integral equation formalism for the polarizable continuum model: Theoretical background and applications to isotropic and anisotropic dielectrics. *The Journal of Chemical Physics*, 107(8):3032–3041, August 1997.
Maurizio Cossi, Nadia Rega, Giovanni Scalmani, and Vincenzo Barone. Energies, structures, and electronic properties of molecules in solution with the c-PCM solvation model. *Journal of Computational Chemistry*, 24(6):669–681, April 2003.

Christopher J Cramer and Donald G Truhlar. Implicit solvation models: equilibria, structure, spectra, and dynamics. *Chemical Reviews*, 99(8):2161–2200, 1999.

T. J. Dolinsky, P. Czodrowski, H. Li, J. E. Nielsen, J. H. Jensen, G. Klebe, and N. A. Baker. PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucleic Acids Research*, 35(Web Server):W522–W525, May 2007.

F. Fogolari, A. Brigo, and H. Molinari. The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology. *Journal of Molecular Recognition*, 15, 2002.

Weihua Geng and Robert Krasny. A treecode-accelerated boundary integral Poisson–Boltzmann solver for electrostatics of solvated biomolecules. *Journal of Computational Physics*, 247:62–78, August 2013.

Paolo Gatto, Filippo Lipparini, and Benjamin Stamm. Computation of forces arising from the polarizable continuum model within the domain-decomposition paradigm. *The Journal of Chemical Physics*, 147(22):224108, December 2017.

Barry Honig and Anthony Nicholls. Classical electrostatics in biology and chemistry. *Science*, 268(5214):1144–1149, 1995.

Elizabeth Jurrus, Dave Engel, Keith Star, Kyle Monson, Juan Brandi, Lisa E. Felberg, David H. Brookes, Leighton Wilson, Jiahui Chen, Karina Liles, Minju Chun, Peter Li, David W. Gohara, Todd Dolinsky, Robert Konesy, David R. Koes, Jens Erik Nielsen, Teresa Head-Gordon, Weihua Geng, Robert Krasny, Guo-Wei Wei, Michael J. Holst, J. Andrew McCammon, and Nathan A. Baker. Improvements to the APBS biomolecular solvation software suite. *Protein Science*, 27(1):112–128, October 2017.

Yi Jiang, Yang Xie, Jinyong Ying, Dexuan Xie, and Zeyun Yu. SDPBS web server for calculation of electrostatics of ionic solvated biomolecules. *Computational and Mathematical Biophysics*, 3(1), November 2015.

A. Klamt and G. Schüürmann. COSMO: a new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradient. *J. Chem. Soc., Perkin Trans. 2*, (5):799–805, 1993.

Benzhuo Lu, Xiaolin Cheng, Jingfang Huang, and J. Andrew McCammon. AFMPB: An adaptive fast multipole poisson–boltzmann solver for calculating electrostatics in biomolecular systems. *Computer Physics Communications*, 181(6):1150–1160, June 2010.

Itay Lotan and Teresa Head-Gordon. An analytical electrostatic model for salt screened interactions between multiple proteins. *Journal of Chemical Theory and Computation*, 2(3):541–555, 2006.

V. I. Lebedev and D. N. Laikov. A quadrature formula for a sphere of the 131st algebraic order of accuracy. *Dokl. Akad. Nauk*, 366(6):741–745, 1999.

Lin Li, Chuan Li, Subhra Sarkar, Jie Zhang, Shawn Witham, Zhe Zhang, Lin Wang, Nicholas Smith, Marharyta Petukh, and Emil Alexov. DelPhi: a comprehensive suite for DelPhi software and associated resources. *BMC Biophysics*, 5(1), May 2012.
[LLS+14] Filippo Lipparini, Louis Lagardère, Giovanni Scalmani, Benjamin Stamm, Eric Cancès, Yvon Maday, Jean-Philip Piquemal, Michael J. Frisch, and Benedetta Mennucci. Quantum calculations in solution for large to very large molecules: A new linear scaling QM/continuum approach. The Journal of Physical Chemistry Letters, 5(6):953–958, February 2014.

[LSC+13] F. Lipparini, B. Stamm, E. Cancès, Y. Maday, and B. Mennucci. Fast domain decomposition algorithm for continuum solvation models: Energy and first derivatives. Journal of Chemical Theory and Computation, 9(8):3637–3648, 2013. PMID: 26584117.

[LSL+14] Filippo Lipparini, Giovanni Scalmani, Louis Lagardère, Benjamin Stamm, Eric Cancès, Yvon Maday, Jean-Philip Piquemal, Michael J. Frisch, and Benedetta Mennucci. Quantum, classical, and hybrid QM/MM calculations in solution: General implementation of the ddCOSMO linear scaling strategy. The Journal of Chemical Physics, 141(18):184108, November 2014.

[LZHM08] Benzhuo Lu, Yongcheng Zhou, Michael Holst, and J Mccammon. Recent progress in numerical methods for the Poisson-Boltzmann equation in biophysical applications. Communications in Computational Physics, 37060:973–1009, 04 2008.

[MBW+95] Jeffry D. Madura, James M. Briggs, Rebecca C. Wade, Malcolm E. Davis, Brock A. Luty, Andrew Ilin, Jan Antosiewicz, Michael K. Gilson, Babak Bagheri, L.Ridgway Scott, and J. Andrew McCammon. Electrostatics and diffusion of molecules in solution: simulations with the university of houston brownian dynamics program. Computer Physics Communications, 91(1-3):57–95, September 1995.

[MST81] S. Miertuš, E. Scrocco, and J. Tomasi. Electrostatic interaction of a solute with a continuum. a direct utilization of AB initio molecular potentials for the prevision of solvent effects. Chemical Physics, 55(1):117–129, February 1981.

[NH91] Anthony Nicholls and Barry Honig. A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. Journal of Computational Chemistry, 12(4):435–445, May 1991.

[NSSL19] Michele Nottoli, Benjamin Stamm, Giovanni Scalmani, and Filippo Lipparini. Quantum Calculations in Solution of Energies, Structures, and Properties with a Domain Decomposition Polarizable Continuum Model. J. Chem. Theory Comput., 15(11):6061–6073, November 2019.

[OL00] Modesto Orozco and F Javier Luque. Theoretical methods for the description of the solvent effect in biomolecular systems. Chemical Reviews, 100(11):4187–4226, 2000.

[OSK+91] H Ozaki, T Sato, H Kubota, Y Hata, Y Katsube, and Y Shimonishi. Molecular structure of the toxin domain of heat-stable enterotoxin produced by a pathogenic strain of Escherichia coli. a putative binding site for a binding protein on rat intestinal epithelial cell membranes. Journal of Biological Chemistry, 266(9):5934–5941, March 1991.

[PC03] Jay W. Ponder and David A. Case. Force fields for protein simulations. In Protein Simulations, pages 27–85. Elsevier, 2003.

[QSM19] C. Quan, B. Stamm, and Y. Maday. A domain decomposition method for the Poisson-Boltzmann solvation models. SIAM J. Sci. Comput., 41(2):B320–B350, 2019.

[RS99] Benoit Roux and Thomas Simonson. Implicit solvent models. Biophysical chemistry, 78(1-2):1–20, 1999.

[SCLM16] Benjamin Stamm, Eric Cancès, Filippo Lipparini, and Yvon Maday. A new discretization for the polarizable continuum model within the domain decomposition paradigm. The Journal of Chemical Physics, 144(5):054101, February 2016.
[TMC05] Jacopo Tomasi, Benedetta Mennucci, and Roberto Cammi. Quantum mechanical continuum solvation models. *Chemical Reviews*, 105(8):2999–3094, July 2005.

[TP94] Jacopo Tomasi and Maurizio Persico. Molecular interactions in solution: an overview of methods based on continuous distributions of the solvent. *Chemical Reviews*, 94(7):2027–2094, 1994.

[VMZM01] Luigi Vitagliano, Antonello Merlino, Adriana Zagari, and Lelio Mazzarella. Reversible substrate-induced domain motions in ribonuclease A. *Proteins: Structure, Function, and Genetics*, 46(1):97–104, December 2001.

[WTC+99] Brian Whitehead, Marco Tessari, Alfonso Carotenuto, Paul M. P. van Bergen en Henegouwen, and Geerten W. Vuister. The EH1 domain of eps15 is structurally classified as a member of the s100 subclass of EF-hand-containing proteins. *Biochemistry*, 38(35):11271–11277, August 1999.

[Xie14] Dexuan Xie. New solution decomposition and minimization schemes for Poisson–Boltzmann equation in calculation of biomolecular electrostatics. *Journal of Computational Physics*, 275:294–309, 2014.

[XWP+00] Yingqi Xu, Jihui Wu, Jiming Pei, Yunyu Shi, Yonghua Ji, and Qingchun Tong. Solution structure of BnP02, a new potassium channel blocker from the venom of the Chinese scorpion Buthus martensi Karsch. *Biochemistry*, 39(45):13669–13675, October 2000.

[XYX17] Yang Xie, Jinyong Ying, and Dexuan Xie. SMPBS: Web server for computing biomolecular electrostatics using finite element solvers of size modified Poisson-Boltzmann equation. *Journal of Computational Chemistry*, 38(8):541–552, January 2017.

[YHG10] Eng-Hui Yap and Teresa Head-Gordon. New and efficient Poisson-Boltzmann solver for interaction of multiple proteins. *Journal of Chemical Theory and Computation*, 6(7):2214–2224, 2010.

[YHG13] Eng-Hui Yap and Teresa Head-Gordon. Calculating the bimolecular rate of protein–protein association with interacting crowders. *Journal of Chemical Theory and Computation*, 9(5):2481–2489, 2013.

[YL90] Byung Jun Yoon and A. M. Lenhoff. A boundary element method for molecular electrostatics with electrolyte effects. *Journal of Computational Chemistry*, 11(9):1080–1086, October 1990.

[YX15] Jinyong Ying and Dexuan Xie. A new finite element and finite difference hybrid method for computing electrostatics of ionic solvated biomolecule. *Journal of Computational Physics*, 298:636–651, October 2015.

[ZPH+15] Bo Zhang, Bo Peng, Jingfang Huang, Nikos P. Pitsianis, Xiaobai Sun, and Benzhuo Lu. Parallel AFMPB solver with automatic surface meshing for calculation of molecular solvation free energy. *Computer Physics Communications*, 190:173–181, May 2015.