

# An implicit boundary integral method for computing electric potential of macromolecules in solvent



Yimin Zhong<sup>a,\*</sup>, Kui Ren<sup>b</sup>, Richard Tsai<sup>b,c</sup>

<sup>a</sup> Department of Mathematics, University of Texas, Austin, TX, 78712, United States

<sup>b</sup> Department of Mathematics and ICES, University of Texas, Austin, TX, 78712, United States

<sup>c</sup> KTH Royal Institute of Technology, Sweden

## ARTICLE INFO

### Article history:

Received 29 September 2017

Received in revised form 5 January 2018

Accepted 10 January 2018

### Keywords:

Poisson–Boltzmann equation

Implicit boundary integral method

Level set method

Fast multipole method

Electrostatics

Implicit solvent model

## ABSTRACT

A numerical method using implicit surface representations is proposed to solve the linearized Poisson–Boltzmann equation that arises in mathematical models for the electrostatics of molecules in solvent. The proposed method uses an implicit boundary integral formulation to derive a linear system defined on Cartesian nodes in a narrowband surrounding the closed surface that separates the molecule and the solvent. The needed implicit surface is constructed from the given atomic description of the molecules, by a sequence of standard level set algorithms. A fast multipole method is applied to accelerate the solution of the linear system. A few numerical studies involving some standard test cases are presented and compared to other existing results.

© 2018 Elsevier Inc. All rights reserved.

## 1. Introduction

The mathematical modeling and numerical simulation of electrostatics of charged macromolecule–solvent systems have been extensively studied in recent years, due to their importance in many branches of electrochemistry; see, for instance, [8,23,26,27,39,53,57,60,65,71,76,89] and references therein for recent overviews of the developments in the subject.

There are roughly two classes of mathematical models for such macromolecule–solvent systems, depending on how the effect of the solvent is modeled: explicit solvent models in which solvent molecules are treated explicitly, and implicit solvent models in which the solvent is represented as a continuous medium. While explicit solvent models are believed to be more accurate, they are computationally intractable when modeling large systems. Implicit models are therefore often an alternative for large simulations, see [6,19,22,50,90] and references therein for recent advances. The Poisson–Boltzmann model is one of the popular implicit solvent models in which the solvent is treated as a continuous high-dielectric medium [16,21,28,41,51,56,69,70,80]. This model, and many variants of it, has important applications, for instance in studying biomolecule dynamics of large proteins [4,5,10,12,33,90]. Many efficient and accurate computational schemes for the numerical solution of the model have been developed [3,7,9,18,24,30,31,36,51,54,81,88].

To introduce the Poisson–Boltzmann model, let us assume that the macromolecule has  $N_c$  atoms centered at  $\{\mathbf{z}_j\}_{j=1}^{N_c}$ , with radii  $\{r_j\}_{j=1}^{N_c}$  and charge number  $\{q_j\}_{j=1}^{N_c}$  respectively. Let  $\Gamma$  be the closed surface that separates the region occupied by the macromolecule and the rest of the space. The typical choice of  $\Gamma$  is the so-called *solvent excluded surface*, which is defined

\* Corresponding author.

E-mail addresses: yzhong@math.utexas.edu (Y. Zhong), ren@math.utexas.edu (K. Ren), ytsai@math.utexas.edu (R. Tsai).

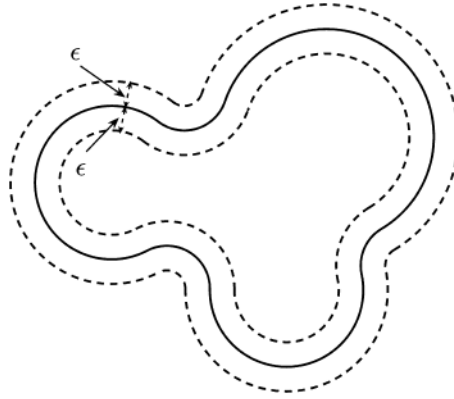


Fig. 1. A view of the “solvent excluded surface” in 2D is shown by the middle solid curve, and the narrowband  $\Gamma_\epsilon$  is shown here by the space bounded between the dashed curves.

as the boundary of the region outside the macromolecule which is accessible by a probe sphere with some small radius, say  $\rho_0$ ; see Fig. 1 for an illustration. We use  $\Omega$  to denote the region surrounded by  $\Gamma$  that includes the macromolecule.

We use a single function  $\psi$  to denote the electric potential inside and outside of  $\Omega$ . In the Poisson–Boltzmann model,  $\psi$  solves the Poisson’s equation for point charges inside  $\Omega$ , that is,

$$-\nabla \cdot (\epsilon_I \nabla \psi(\mathbf{x})) = \sum_{k=1}^{N_c} q_k \delta(\mathbf{x} - \mathbf{z}_k), \quad \text{in } \Omega$$

where  $\epsilon_I$  denotes the dielectric constant in  $\Omega$ . Outside  $\overline{\Omega}$ , that is in the solvent that excludes the interface  $\Gamma$ ,  $\psi$  solves the Poisson’s equation for a continuous distribution of charges that models the effect of the solvent, that is,

$$-\nabla \cdot (\epsilon_E \nabla \psi) = \rho_B(T, \mathbf{x}, \psi(\mathbf{x})), \quad \text{in } \mathbb{R}^3 \setminus \overline{\Omega}$$

where  $\epsilon_E$  denotes the dielectric constant of the solvent, which often has much higher value than that of the macromolecule,  $\epsilon_E \gg \epsilon_I$ . The source term  $\rho_B$  is a nonlinear function coming from the Boltzmann distribution with  $T$  denoting the temperature of the system. More precisely, for solvent containing  $m$  ionic species,

$$\rho_B(T, \mathbf{x}, \psi(\mathbf{x})) := e_c \sum_{i=1}^m c_i \bar{q}_i e^{-e_c \bar{q}_i \psi(\mathbf{x}) / k_B T}, \quad \mathbf{x} \in \mathbb{R}^3 \setminus \overline{\Omega}$$

where  $c_i, \bar{q}_i$  are the concentration and charge of the  $i$ th ionic species,  $e_c$  is the electron charge,  $k_B$  is the Boltzmann constant, and  $T$  is the absolute temperature.

The nonlinear term  $\rho_B(T, \mathbf{x}, \psi)$  in the Poisson–Boltzmann system poses significant challenges in the computational solution of the system. In many practical applications, it is replaced by the linear function  $-\bar{\kappa}_T^2 \psi(\mathbf{x})$  where the parameter  $\bar{\kappa}_T = \sqrt{\frac{2e^2 \mathbb{I}}{k_B T}}$  is called the Debye–Hückel screening parameter with  $k_B$ ,  $e$ , and  $\mathbb{I}$  being the Boltzmann constant, the unit charge, and the ionic strength respectively. This leads to the linearized Poisson–Boltzmann equation (PBE) for the electrostatic potential  $\psi$ . It takes the following form

$$\begin{aligned} -\nabla \cdot (\epsilon_I \nabla \psi(\mathbf{x})) &= \sum_{k=1}^{N_c} q_k \delta(\mathbf{x} - \mathbf{z}_k), & \text{in } \Omega, \\ -\nabla \cdot (\epsilon_E \nabla \psi(\mathbf{x})) &= -\bar{\kappa}_T^2 \psi(\mathbf{x}), & \text{in } \overline{\Omega}^c, \\ \psi(\mathbf{x})|_{\Gamma_+} &= \psi(\mathbf{x})|_{\Gamma_-}, & \text{on } \Gamma, \\ \epsilon_E \frac{\partial \psi}{\partial n}|_{\Gamma_+} &= \epsilon_I \frac{\partial \psi}{\partial n}|_{\Gamma_-}, & \text{on } \Gamma, \\ |\mathbf{x}| \psi(\mathbf{x}) &\rightarrow 0, \quad |\mathbf{x}|^2 |\nabla \psi(\mathbf{x})| \rightarrow 0, & \text{as } |\mathbf{x}| \rightarrow \infty. \end{aligned} \tag{1}$$

Here the operator  $\partial/\partial n \equiv \mathbf{n}(\mathbf{x}) \cdot \nabla$  denotes the usual partial derivative at  $\mathbf{x} \in \Gamma$  in the outward normal direction  $\mathbf{n}(\mathbf{x})$  (pointing from  $\Gamma$  outward). The usual continuity conditions, continuity of the potential and the flux across  $\Gamma$ , are assumed, and the radiation condition, which requires  $\psi$  decay to zero far away from the macromolecule, is needed to ensure the uniqueness of solutions to the linearized Poisson–Boltzmann equation. See e.g. [3,7,10,16,30,31,50,53,70,80,81].

Computational solution of the linearized Poisson–Boltzmann equation (1) in practically relevant configurations turns out to be quite challenging. Different types of numerical methods, including for instance finite difference methods [6,15,41,32,58,64], finite element methods [3,38,83–86], boundary element methods [1,2,14,44,52,54,55,87], and many more hybrid or specialized methods [13,79] have been developed; see [53] for the recent survey on the subject. Each method has its own advantages and disadvantages. Finite difference methods are easy to implement. They are the methods used in many existing software packages [15,41,32,58,64]. Finite difference methods, except the ones that are based on adaptive oct-tree structures [36,62,63], use uniform Cartesian grids and require special care for implementing the interface conditions to high order while maintaining stability. Finite element methods provide more flexibility with the geometry. However, like the finite difference methods, they often suffer from issues such as large memory storage requirement and low solution speed when dealing with large problems. Moreover, both finite difference and finite element methods need to truncate the domain in some way, therefore the radiation condition is not satisfied exactly. Boundary element methods are based on integral formulations of the Poisson–Boltzmann equation. They require only the discretization of the solvent excluded surface, i.e.  $\Gamma$ , not the macromolecule and solvent domains. The radiation condition is usually exactly, although implicitly, integrated into the integral form to be solved. However, the matrix systems resulting from boundary element formulations are often dense. Efficient acceleration, for instance preconditioning, techniques are needed to accelerate the solution of such dense systems.

In this work, we propose a fast numerical method for solving the interface/boundary value problem of the linearized Poisson–Boltzmann equation (1). The method is derived from the implicit boundary integral formulation [47] of (1) and relies on some of the classical level set algorithms [66,67] for computing the implicit interfaces and the needed geometrical information. All the involved computational procedures are defined on an underlying uniform Cartesian grid. Thus the proposed method inherits most of the flexibilities of a level set algorithm. On the other hand, since the method is derived from a boundary integral formulation of (1), it treats the interface conditions and far field conditions in a less involved fashion compared to the standard level set algorithms for similar problems. As such types of implicit boundary integral approaches are relatively new, we describe in detail how to set up a linear system and where a fast multipole method can be used for acceleration of the common matrix–vector multiplications in the resulting linear system. We demonstrate in our simulations involving non-trivial molecules defined by tens of thousands atoms that standard “kernel-independent” fast multipole methods [29] can be used easily and effectively as in a standard boundary integral method.

We conclude the introduction with the following remarks. The linearized Poisson–Boltzmann model provides a sufficiently accurate approximation in many cases, in particular when the solution’s ionic strength is relatively low; see [28] and references therein. In cases where the linearized model is not accurate enough, solving the nonlinear Poisson–Boltzmann is necessary. With appropriate far field conditions, finite difference or finite element methods provide a way to compute solutions in such case; see [63] and references therein. The computational method we develop in this work can potentially be combined with an iterative scheme for nonlinear equations, such as methods of Newton’s type [45], to solve the nonlinear Poisson–Boltzmann equation. At each iteration of the nonlinear solver, the proposed method can be adapted to solve the linearized problem as long as the coefficients involved, mainly the dielectric coefficients, are constants as currently assumed in our algorithm.

The rest of this paper is organized as follows. We first introduce in Section 2 the implicit boundary integral formulation of the linearized Poisson–Boltzmann system (1). We then present the details of the implementation of the method in Section 3. In Section 4, we present some numerical simulation results to demonstrate the performance of the algorithm. Concluding remarks are then offered in Section 5.

## 2. The implicit boundary integral formulation

The numerical method we develop in this work is based on a boundary integral formulation of the linearized Poisson–Boltzmann equation that is developed in [44].

### 2.1. Boundary integral formulation

Throughout the rest of the paper, all the coefficients involved in the equations are assumed to be constant, i.e. independent of the spatial variable. We define  $\kappa = \bar{\kappa}_T / \sqrt{\epsilon_E}$ , and introduce the fundamental solutions

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

to the Laplace equation and the one with the linear lower order term  $-\bar{\kappa}_T^2\psi$  in (1).

Following the standard way of deriving boundary integral equations, we apply Green’s theorem to the system formed by (i) the first equation in (1) and the equation for  $G_0$ , and (ii) the second equation in (1) and the equation for  $G_\kappa$ , taking into account the interface and the radiation conditions. A careful routine calculation leads to the following boundary integral equations for the potential  $\psi$  and its normal derivative  $\psi_n \equiv \partial\psi/\partial n$  on  $\Gamma$ :



$$\begin{aligned} \frac{1}{2} \psi(\mathbf{x}) + \int_{\Gamma} \left( \frac{\partial G_0(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{y})} \psi(\mathbf{y}) - G_0(\mathbf{x}, \mathbf{y}) \psi_n(\mathbf{y}) \right) d\mathbf{y} &= \sum_{k=1}^{N_c} \frac{q_k}{\epsilon_I} G_0(\mathbf{x}, \mathbf{z}_k), \\ \frac{1}{2} \psi(\mathbf{x}) - \int_{\Gamma} \left( \frac{\partial G_{\kappa}(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{y})} \psi(\mathbf{y}) - \frac{\epsilon_I}{\epsilon_E} G_{\kappa}(\mathbf{x}, \mathbf{y}) \psi_n(\mathbf{y}) \right) d\mathbf{y} &= 0. \end{aligned} \quad (2)$$

This system of boundary integral equations is the starting point of many existing numerical algorithms for the linearized Poisson–Boltzmann equation.

In our algorithm, we adopt the integral formulation proposed in [44]. This formulation reads:

$$\begin{aligned} \frac{1}{2} \left( 1 + \frac{\epsilon_E}{\epsilon_I} \right) \psi(\mathbf{x}) + \int_{\Gamma} \left( \frac{\partial G_0(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{y})} - \frac{\epsilon_E}{\epsilon_I} \frac{\partial G_{\kappa}(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{y})} \right) \psi(\mathbf{y}) d\mathbf{y} \\ - \int_{\Gamma} (G_0(\mathbf{x}, \mathbf{y}) - G_{\kappa}(\mathbf{x}, \mathbf{y})) \psi_n(\mathbf{y}) d\mathbf{y} &= \sum_{k=1}^{N_c} \frac{q_k}{\epsilon_I} G_0(\mathbf{x}, \mathbf{z}_k), \end{aligned} \quad (3)$$

$$\begin{aligned} \frac{1}{2} \left( 1 + \frac{\epsilon_I}{\epsilon_E} \right) \psi_n(\mathbf{x}) + \int_{\Gamma} \left( \frac{\partial^2 G_0(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{x}) \partial n(\mathbf{y})} - \frac{\partial^2 G_{\kappa}(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{x}) \partial n(\mathbf{y})} \right) \psi(\mathbf{y}) d\mathbf{y} \\ - \int_{\Gamma} \left( \frac{\partial G_0(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{x})} - \frac{\epsilon_I}{\epsilon_E} \frac{\partial G_{\kappa}(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{x})} \right) \psi_n(\mathbf{y}) d\mathbf{y} &= \sum_{k=1}^{N_c} \frac{q_k}{\epsilon_I} \frac{\partial G_0(\mathbf{x}, \mathbf{z}_k)}{\partial n(\mathbf{x})}. \end{aligned} \quad (4)$$

The first equation in this formulation, (3), is simply the linear combination of the two equations in (2), while the second equation in this formulation, (4), is nothing but the linear combination of the derivatives of the two equations in (2). It is shown in [44] that the potentially hypersingular integral in (4), involving the second derivatives of  $G_0$  and  $G_{\kappa}$  is actually integrable on  $\Gamma$ , thanks to the fact that

$$\frac{\partial^2 G_0(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{x}) \partial n(\mathbf{y})} - \frac{\partial^2 G_{\kappa}(\mathbf{x}, \mathbf{y})}{\partial n(\mathbf{x}) \partial n(\mathbf{y})} \sim O(|\mathbf{x} - \mathbf{y}|^{-1}), \quad |\mathbf{x} - \mathbf{y}| \rightarrow 0.$$

Moreover, when  $\kappa = 0$ , (3) is decoupled from (4), and the latter provides an explicit formula for evaluating  $\partial\psi/\partial n$  using  $\psi$ .

The main benefit of the formulation (3)–(4) is that it typically leads to, after discretization, linear systems with smaller condition numbers than the formulation in (2). The typical boundary element methods for this system (and others) require careful triangulation of the interface  $\Gamma$ ; see e.g. [1,2,14,44,52,55,87]. In the next subsection, we describe our method to discretize the boundary integral system (3) and (4) on a subset of a uniform Cartesian grid nodes in a narrowband surrounding  $\Gamma$ , without the need to parameterize  $\Gamma$ .

## 2.2. Implicit boundary integral method

Let the interface  $\Gamma$  be a closed,  $C^{1,\alpha}$  surface (in two or three dimensions) with  $\alpha > 0$  so that the distance function to  $\Gamma$  is differentiable in a neighborhood around it. Let  $d_{\Gamma}$  denote the signed distance function to  $\Gamma$  that takes the negative sign for points inside the region enclosed by  $\Gamma$ , and  $\Gamma_{\epsilon}$  denote the set of points whose distance to  $\Gamma$  is smaller than  $\epsilon$ . An implicit boundary integral formulation of a surface integral defined on  $\Gamma$  is derived by projecting points in  $\Gamma_{\epsilon}$  onto their closest points on  $\Gamma$ . With the distance function to  $\Gamma$ , the projection operator can be evaluated by

$$P_{\Gamma} \mathbf{x} := \mathbf{x} - d_{\Gamma}(\mathbf{x}) \nabla d_{\Gamma}(\mathbf{x}). \quad (5)$$

When  $\epsilon$  is smaller than the maximum principal curvatures of  $\Gamma$ , the closest point projection is well-defined in  $\Gamma_{\epsilon}$ .

An implicit boundary integral method (IBIM) [47] is built upon the following identity:

$$I_{\Gamma}[f] := \int_{\Gamma} f(\mathbf{x}) d\mathbf{s}(\mathbf{x}) = \int_{\Gamma_{\epsilon}} f(P_{\Gamma} \mathbf{x}) \delta_{\epsilon}(d_{\Gamma}(\mathbf{x})) J(\mathbf{x}) d\mathbf{x}, \quad (6)$$

which reveals the equivalence between the surface integral and its extension into a volume integral. We shall call the integral over  $\Gamma_{\epsilon}$  an implicit boundary integral. In this implicit boundary integral, one has

1. The extension of  $f(\mathbf{x})$  as a constant along the normal of  $\Gamma$  at  $\mathbf{x}$ .
2. The Jacobian  $J(\mathbf{x})$  which accounts for the change of variables between  $\Gamma$  and the level set surface of  $d_{\Gamma}$  that passes through  $\mathbf{x}$ .

3. A weight function,  $\delta_\epsilon$  compactly supported on  $[-\epsilon, \epsilon]$  satisfying

$$\int_{-\epsilon}^{\epsilon} \delta_\epsilon(\eta) d\eta = 1. \quad (7)$$

In  $\mathbb{R}^3$ , the Jacobian  $J$  takes the explicit form

$$J(\mathbf{x}) = 1 - d_\Gamma(\mathbf{x}) \Delta d_\Gamma(\mathbf{x}) + d_\Gamma(\mathbf{x})^2 \langle \nabla d, \nabla^2 d_\Gamma \nabla d_\Gamma \rangle. \quad (8)$$

It can be further related to the products of the singular values of the Jacobian matrix of  $P_\Gamma$ , which provides an alternative, and in some cases easier way, for the computation of  $J$ . See [48].

In the application of interest, the distance function to a solvent excluded surface will be twice differentiable almost everywhere in some narrowband around the surface, together with setting  $J \equiv 1$ , the proposed method is well-defined in there. In fact, the requirement on the regularity of the surface (and its signed distance function) can be further relaxed if the weight function is an even function and possesses enough vanishing moments. It is shown in [49] that with  $J \equiv 1$ , the cosine weight function (27) and  $C^1$  integrands which are not necessarily constant along surface normals,  $I_\Gamma[f]$  is approximated to second order if  $\epsilon$ . It is further shown in [49] that if the weight function has more than two vanishing moments, one may replace the Jacobian by  $J \equiv 1$  while keeping the equality in (6) valid, even for piecewise smooth surfaces containing corners and creases.

Numerically we approximate the implicit boundary integral by embedding the computational domain  $\Omega$  into the rectangle  $U = [a, b]^n$ , and subdivide  $U$  into the uniform grid  $U_h = h\mathbb{Z}^n \cap [a, b]^n$  with grid size  $h = (b - a)/N$  along each coordinate direction and  $\mathbf{x}_i$  at each grid point. We approximate the integral by

$$I_\Gamma[f] \approx S_\epsilon^h[f] := \sum_{\mathbf{x}_i \in \Gamma_\epsilon} f(\mathbf{x}_i^*) \delta_\epsilon(d_\Gamma(\mathbf{x}_i)) J(\mathbf{x}_i) h^n \quad (9)$$

where  $\mathbf{x}_i^* = \mathbf{x}_i - d_\Gamma(\mathbf{x}_i) \nabla d_\Gamma(\mathbf{x}_i)$  is the projection of  $\mathbf{x}_i$  onto  $\Gamma$ .

Thus, a typical second kind integral equation of the form

$$g(\mathbf{x}) = \lambda \beta(\mathbf{x}) + \int_{\Gamma} K(\mathbf{x}, \mathbf{y}) \beta(\mathbf{y}) ds(\mathbf{y}), \quad \mathbf{x} \in \Gamma, \quad (10)$$

can be approximated on  $U_h$  using the IBIM formulation. One would derive a linear system for the unknown function  $\bar{\beta}$  defined on the grid nodes in  $\Gamma_\epsilon$ :

$$g(P_\Gamma \mathbf{x}_i) = \lambda \bar{\beta}(\mathbf{x}_i) + h^n \sum_{\mathbf{y}_j \in \Gamma_\epsilon \cap U_h} K(P_\Gamma \mathbf{x}_i, P_\Gamma \mathbf{y}_j) \bar{\beta}(\mathbf{y}_j) \delta_\epsilon(d_\Gamma(\mathbf{y}_j)) J(\mathbf{y}_j), \quad \mathbf{x}_i \in \Gamma_\epsilon \cap U_h, \quad (11)$$

with the property that as  $h \rightarrow 0$

$$\bar{\beta}(\mathbf{x}_i) \longrightarrow \beta(P_\Gamma \mathbf{x}_i), \quad \forall \mathbf{x}_i \in \Gamma_\epsilon \cap U_h;$$

i.e. the solution to the linear system (11) converges to “the function which is the constant extension along the surface normal” of the solution of (10); see more discussions in [17,47].

In the context of this paper, equations (3) and (4) will be discretized into

$$\begin{aligned} \frac{1}{2} \lambda_1 \bar{\psi}(\mathbf{x}_i) + h^3 \sum_j K_{11}(\mathbf{x}_i, \mathbf{y}_j) \omega_j \bar{\psi}(\mathbf{y}_j) - h^3 \sum_j K_{12}(\mathbf{x}_i, \mathbf{y}_j) \omega_j \bar{\psi}_n(\mathbf{y}_j) &= g_1(\mathbf{x}_i), \\ \frac{1}{2} \lambda_2 \bar{\psi}_n(\mathbf{x}_i) + h^3 \sum_j K_{21}(\mathbf{x}_i, \mathbf{y}_j) \omega_j \bar{\psi}(\mathbf{y}_j) - h^3 \sum_j K_{22}(\mathbf{x}_i, \mathbf{y}_j) \omega_j \bar{\psi}_n(\mathbf{y}_j) &= g_2(\mathbf{x}_i), \end{aligned} \quad (12)$$

where

$$\begin{aligned} \lambda_1 &= \frac{1}{2} \left( 1 + \frac{\epsilon_E}{\epsilon_I} \right), \quad \lambda_2 = \frac{1}{2} \left( 1 + \frac{\epsilon_I}{\epsilon_E} \right), \\ \omega_j &:= J(\mathbf{y}_j) \delta_\epsilon(d_\Gamma(\mathbf{y}_j)), \\ g_1(\mathbf{x}_i) &:= \sum_{k=1}^{N_c} \frac{q_k}{\epsilon_I} G_0(\mathbf{x}_i, \mathbf{z}_k), \quad g_2(\mathbf{x}_i) := \sum_{k=1}^{N_c} \frac{q_k}{\epsilon_I} \frac{\partial G_0(\mathbf{x}_i, \mathbf{z}_k)}{\partial n(\mathbf{x}_i)}, \end{aligned} \quad (13)$$

and  $K_{11}, K_{12}, K_{21}, K_{22}$  are respectively the regularized versions of the following weakly singular kernels:

$$\frac{\partial G_0(P_\Gamma \mathbf{x}, P_\Gamma \mathbf{y})}{\partial n(P_\Gamma \mathbf{y})} - \frac{\epsilon_E}{\epsilon_I} \frac{\partial G_\kappa(P_\Gamma \mathbf{x}, P_\Gamma \mathbf{y})}{\partial n(P_\Gamma \mathbf{y})}, G_0(P_\Gamma \mathbf{x}, P_\Gamma \mathbf{y}) - G_\kappa(P_\Gamma \mathbf{x}, P_\Gamma \mathbf{y}),$$

$$\frac{\partial^2 G_0(P_\Gamma \mathbf{x}, P_\Gamma \mathbf{y})}{\partial n(P_\Gamma \mathbf{x}) \partial n(P_\Gamma \mathbf{y})} - \frac{\partial^2 G_\kappa(P_\Gamma \mathbf{x}, P_\Gamma \mathbf{y})}{\partial n(P_\Gamma \mathbf{x}) \partial n(P_\Gamma \mathbf{y})}, \text{ and } \frac{\partial G_0(P_\Gamma \mathbf{x}, P_\Gamma \mathbf{y})}{\partial n(P_\Gamma \mathbf{x})} - \frac{\epsilon_I}{\epsilon_E} \frac{\partial G_\kappa(P_\Gamma \mathbf{x}, P_\Gamma \mathbf{y})}{\partial n(P_\Gamma \mathbf{x})}.$$

A simple regularization that we used in our numerical implementation is described below in the next subsection.

This formulation provides a convenient computational approach for computing boundary integrals, where the boundary is naturally defined implicitly, as a level set of a continuous function, and is difficult to parameterize. The geometrical information about the boundary is restricted to the computation of the Jacobian  $J$  and the closest point extension of the integrand  $f$  – both of which can be approximated easily by simple finite differencing applied to the distance function  $d_\Gamma(\mathbf{x})$  at grid point  $\mathbf{x}_i$  within  $\Gamma_\epsilon$ . Furthermore, the smoothness of the weight function  $\delta_\epsilon$ , along with the smoothness of the integrand will allow for higher order in  $h$  approximation of  $I[f]$  by simple Riemann sum  $S_{\Gamma_\epsilon}^h[f]$ , see for example the discussion in [48].

### 2.2.1. Regularization of the kernels

While all the kernels (the Green's functions and the particular linear combinations of them) that appear in (3)–(4) are formally integrable, an additional treatment for the singularities is needed in the numerical computation when  $\mathbf{x}$  and  $\mathbf{y}$  are close. Typically, the additional treatment corresponds to either local change of variables so that in the new variables the singularities do not exist or mesh refinement for control of numerical error amplification (particularly for Nyström methods). The proposed simple discretization of the Implicit Boundary Integral formulation on uniform Cartesian grid can be viewed as an extreme case of Nyström method, in which no mesh refinement is involved (and thus no control of numerical errors if the singularities of the kernels are left untreated). Therefore we need to regularize the kernels analytically and locally only when  $P_\Gamma \mathbf{x}$  and  $P_\Gamma \mathbf{y}$  are sufficiently close with respect to the grid spacing.

In the following, for brevity of the displayed formulas, let  $\mathbf{x}^* := P_\Gamma \mathbf{x}$ ,  $\mathbf{y}^* := P_\Gamma \mathbf{y}$  and

$$K_\theta(\mathbf{x}, \mathbf{y}) := \frac{\partial G_0(\mathbf{x}^*, \mathbf{y}^*)}{\partial n(\mathbf{y}^*)} - \theta \frac{\partial G_\kappa(\mathbf{x}^*, \mathbf{y}^*)}{\partial n(\mathbf{y}^*)}, \quad \theta \in \mathbb{R}.$$

The regularization that we will use involves a small parameter  $\tau > 0$  and is defined by

$$K_\theta^{\text{reg}}(\mathbf{x}, \mathbf{y}) = \begin{cases} \overline{K_\theta}(\mathbf{x}), & \text{if } \|\mathbf{x}^* - \mathbf{y}^*\|_p < \tau, \\ K_\theta(\mathbf{x}, \mathbf{y}), & \text{otherwise,} \end{cases} \quad (14)$$

where  $\|\mathbf{x}^* - \mathbf{y}^*\|_p$  is the distance between projections of  $\mathbf{x}^*$  and  $\mathbf{y}^*$  onto the tangent plane at  $\mathbf{x}^*$ .  $\overline{K_\theta}(\mathbf{x})$  is the average of  $K_\theta(\mathbf{x}, \cdot)$  defined as

$$\overline{K_\theta}(\mathbf{x}) = \frac{1}{V(\mathbf{x}^*; \tau)} \int_{V(\mathbf{x}^*; \tau)} K_\theta(\mathbf{x}, \mathbf{z}) ds(\mathbf{z}), \quad (15)$$

where  $V(\mathbf{x}^*; \tau)$  is the disc of radius  $\tau$  in the tangent plane of  $\Gamma$  at  $\mathbf{x}^*$ .

Thus,

$$K_{11}(\mathbf{x}, \mathbf{y}) := K_\theta^{\text{reg}}(\mathbf{x}, \mathbf{y}), \quad \text{with } \theta = \frac{\epsilon_E}{\epsilon_I}, \quad \overline{K_\theta}(\mathbf{x}) = 0, \quad (16)$$

$$K_{22}(\mathbf{x}, \mathbf{y}) := K_\theta^{\text{reg}}(\mathbf{x}, \mathbf{y}), \quad \text{with } \theta = \frac{\epsilon_I}{\epsilon_E}, \quad \overline{K_\theta}(\mathbf{x}) = 0. \quad (17)$$

Similarly, the averages of  $G_0 - G_\kappa$  and  $\frac{\partial^2 G_0}{\partial n(\mathbf{x}^*) \partial n(\mathbf{y}^*)} - \frac{\partial^2 G_\kappa}{\partial n(\mathbf{x}^*) \partial n(\mathbf{y}^*)}$  are computed and we define:

$$K_{12}(\mathbf{x}, \mathbf{y}) = \begin{cases} \frac{e^{-\kappa\tau} - 1 + \kappa\tau}{2\pi\kappa\tau^2}, & \text{if } \|\mathbf{x}^* - \mathbf{y}^*\|_p < \tau, \\ G_0(\mathbf{x}^*, \mathbf{y}^*) - G_\kappa(\mathbf{x}^*, \mathbf{y}^*), & \text{otherwise,} \end{cases} \quad (18)$$

$$K_{21}(\mathbf{x}, \mathbf{y}) = \begin{cases} 0, & \text{if } \|\mathbf{x}^* - \mathbf{y}^*\|_p < \tau, \\ \frac{\partial^2 G_0(\mathbf{x}^*, \mathbf{y}^*)}{\partial n(\mathbf{x}^*) \partial n(\mathbf{y}^*)} - \frac{\partial^2 G_\kappa(\mathbf{x}^*, \mathbf{y}^*)}{\partial n(\mathbf{x}^*) \partial n(\mathbf{y}^*)}, & \text{otherwise.} \end{cases} \quad (19)$$

Finally, we refer the readers to [17] for a recent approach for dealing with hypersingular integrals via extrapolation.

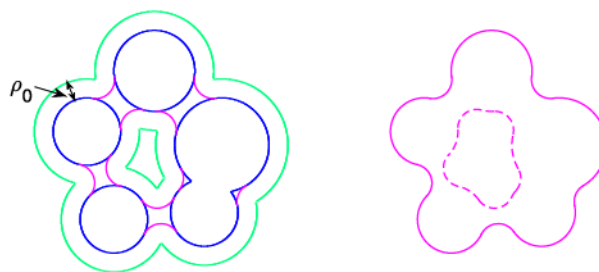


Fig. 2. The construction of the “solvent excluded surface” (SES) of a fictitious molecule defined by five atoms. The final SES is shown as the solid curve on the right plot. The van der Waals surface corresponding to the molecule is shown by the blue curve. The green curve is the “solvent accessible surface”, from which an inward eikonal flow will shrink it by a distance of  $\rho_0$  to arrive at the pink curves (solid and dashed). The dashed pink curve on the right plot shows that boundary of the cavity enclosed by the molecules. It is removed from our computation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### 3. The proposed algorithm

The proposed algorithm consists of a few stages which are outlined below:

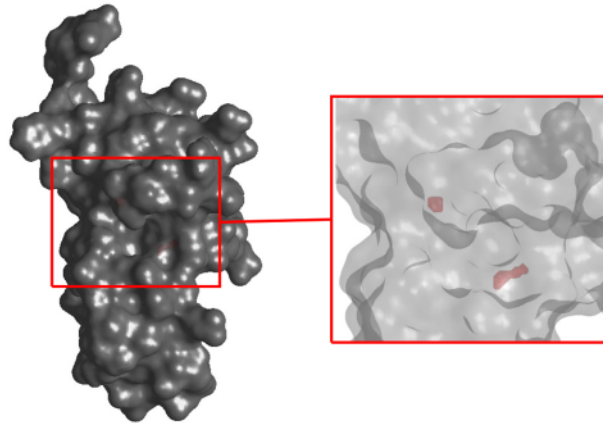
- Stage (1) **Preparation of the signed distance function** to the “solvent excluded surface” on a uniform Cartesian grid. This includes definition of an initial level set function (Section 3.1), followed by an “inward” eikonal flow of the level set function (Section 3.1.1). After the eikonal flow, we apply a step that removes from the implicit surface the interior cavities which are not accessible to solvent (Section 3.1.2). See Fig. 2 for an illustration of this process and the various surfaces involved in it. Finally we apply the reinitialization procedure (Section 3.1.3) to the level set function obtained from cavity removal. At the end of this stage, one shall obtain the signed distance function to the “solvent excluded surface” on which the linearized Poisson–Boltzmann boundary integral equation (BIE) is solved. The constructed signed distance function has the same sign as the function  $F$ , defined in (20) that is used to define the van der Waals surface, at the prescribed molecule centers.
- Stage (2) **Preparation of the linear system.** This involves computation of geometrical information, including the closest point mapping and the Jacobian (Section 3.2).
- Stage (3) **Solution of the linear system** via GMRES with a fast multipole acceleration for the matrix–vector multiplication (Section 3.2). At the end of this stage, one obtains the density  $\bar{\psi}$  defined on the grid nodes lying in  $\Gamma_\epsilon$ . This density function will be used in the evaluation of the polarization energy.
- Stage (4) **Evaluation of surface area and polarization energy** (Sections 3.4 and 3.5). The surface area is computed by using  $f \equiv 1$  in (6) and the polarization energy is computed through the density  $\bar{\psi}$  by the implicit boundary integral method.

All computations will be performed on functions defined on  $U_h$ . The inward eikonal flow and the reinitialization in Stage (1) are computed with commonly used routines: i.e. the third order total variation diminishing Runge–Kutta scheme (TVD RK3) [74] for time discretization, and Godunov Hamiltonian [72] for the eikonal terms  $\pm|\nabla\phi|$  with the fifth order WENO discretization [42] approximating  $\nabla\phi$ . We refer the readers to the book [66] and [77] for more detailed discussions and references. We have also arranged our codes to be openly available on GitHub.<sup>1</sup>

A quick remark is in order regarding the algorithms used to generate an implicit representation of a “solvent excluded surface”. Of course there are other approaches to generate the surfaces under the level set framework. While the general ideas appear to be similar, they are different in many details that could potentially influence performance of the algorithm that uses the prepared surfaces. We point out here that the procedure described in [63] is different from the one that proposed in Section 3.1. In particular, our method applies reinitialization *after* removing the cavities, and does not need additional numerical solution of a Dirichlet problem for Laplace equation on irregular domain as in [63]. The reinitialization after cavity removal is essential to our proposed approach which requires distance function in the narrowband surrounding the surface. If there is no cavity, our procurement does not require the reinitialization step. We refer to [63,68] for a more extensive review of other related algorithms.

<sup>1</sup> <https://github.com/GaZ3ll3/ibim-levelset>.





**Fig. 3.** Cavity in protein 1A63. The gray surface is the “solvent excluded surface” used for computing the electric potential for protein 1A63. In the right subfigure, the rendering of the gray surface is made semi-transparent in order to reveal the enclosed cavity surfaces (red). The regions enclosed by the red surfaces are the cavities to be removed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### 3.1. Creating a signed distance function to the solvent accessible surface

From molecules description the van der Waals surface,  $\Gamma^{\text{vdW}}$ , is defined as the zero level set of

$$F(\mathbf{x}) = \inf_k (\|\mathbf{x} - \mathbf{z}_k\| - r_k), \quad (20)$$

where  $\mathbf{z}_k$  and  $r_k$  denote respectively the coordinates of the molecule centers and their radii.

From the van der Waals surface, we shall define the so-called solvent excluded surface,  $\Gamma$ , as the zero level set of a continuous function  $\phi_{SES}$ .  $\phi_{SES}$  is computed by a simple inward eikonal flow, starting from an initial condition involving  $F$ , and is followed by a few iterations of the standard level set reinitialization steps. See Fig. 2 for an illustration of this procedure in two dimensions. The details are described in the following subsections.

#### 3.1.1. Inward eikonal flow

The van der Waals surface is extended outwards for a radius  $\rho_0$  to define the so-called “solvent accessible surface”, which can be conveniently defined as the zero level set of  $\phi_{SAS}$ :

$$\phi_{SAS}(\mathbf{x}) = F(\mathbf{x}) - \rho_0. \quad (21)$$

The inward eikonal flow will produce a surface with smoothed out the corners when compared to the original van der Waals surface, while keeping most of its smooth parts unchanged. For  $0 < t \leq \rho_0$ , we solve the following equation:

$$\begin{aligned} \frac{\partial \tilde{\phi}_{SAS}(\mathbf{x}, t)}{\partial t} - \|\nabla \tilde{\phi}_{SAS}\| &= 0, \quad \mathbf{x} \in U, \\ \tilde{\phi}_{SAS}(\mathbf{x}, 0) &= \phi_{SAS}(\mathbf{x}), \end{aligned} \quad (22)$$

with zero Neumann boundary conditions.

#### 3.1.2. Cavities removal

The zero level set of  $\tilde{\phi}_{SAS}$  may contain some pieces of surfaces that isolate cavities that are believed to be void of solvent. Fig. 3 provides an example of such cavities in a protein that we used for computation. The cavity removal step uses a simple sweep to remove (the boundaries of) these regions and create a level set function,  $\phi_{SES}$ , that describes only the exterior, closed and connected surface – the solvent excluded surface:

$$\tilde{\phi}_{SAS}(\mathbf{x}, \rho_0) \longrightarrow \phi_{SES}(\mathbf{x}).$$

The cavity removal consists of following steps:

1. Identify a region  $C_\epsilon$  that contains the cavities.  $C_\epsilon$  is a superset of the cavities, containing points outside of the cavities that are within  $\epsilon$  distance to the cavity surface. This can be done by a “peeling” process: by moving the set of markers initially placed on the boundary of the computational domain inwards, using a breadth-first search (BFS) algorithm. The first layer of the surfaces defined by the zero level set of  $\tilde{\phi}_{SAS}$  is defined to be the “solvent excluded surface”. We could therefore remove the remaining portion of  $\tilde{\phi}_{SAS}$ ’s zero level sets, which are regarded as corresponding to the cavities. From this process, one can easily compute a characteristic function supported on  $C_\epsilon$ .



2. Remove the cavity region by modifying the values of  $\tilde{\phi}_{SAS}$  in  $C_\epsilon$ :

$$\phi_{SES}(\mathbf{x}) := \begin{cases} \tilde{\phi}_{SAS}(\mathbf{x}), & \mathbf{x} \notin C_\epsilon, \\ -\epsilon, & \mathbf{x} \in C_\epsilon. \end{cases}$$

### 3.1.3. Reinitialization

The kinks on the solvent accessible surface (SAS) will lower the accuracy in the computation for  $\phi_{SES}(\mathbf{x}, \rho_0)$ . In addition, the cavity removal step may introduce small jump discontinuities near the removed cavity. We perform several iterations of the standard level set reinitialization to improve the equivalence of the computed  $\phi_{SES}(\mathbf{x}, \rho_0)$  and the signed distance function to  $\Gamma$  (which  $\phi_{SES}$  is supposed to be). The reinitialization equation, first appeared in [75], is defined as

$$\begin{aligned} \frac{\partial \tilde{\phi}_{SES}(\mathbf{x}, t)}{\partial t} + \text{sgn}_h(\phi_{SES}(\cdot, \rho_0))(|\nabla \tilde{\phi}_{SES}| - 1) &= 0, \\ \tilde{\phi}_{SES}(\mathbf{x}, 0) &= \phi_{SES}(\mathbf{x}, \rho_0), \end{aligned} \quad (23)$$

where the smoothed-out signum function is defined as

$$\text{sgn}_h(\phi) = \frac{\phi}{\sqrt{\phi^2 + h^2}}. \quad (24)$$

Suppose that the reinitialization equation is solved until  $t = t_n$ , i.e.  $\tilde{\phi}_{SES}(\mathbf{x}, t_n)$  is our approximation to the signed distance function  $d_\Gamma(\mathbf{x})$ , we shall compute  $\Gamma_\epsilon$  by

$$\Gamma_\epsilon := \{\mathbf{x} \in \mathbb{R}^d : -\epsilon < \tilde{\phi}_{SES}(\mathbf{x}, t_n) < \epsilon\}. \quad (25)$$

We use the standard fifth order WENO for space and third order TVD-RK scheme for time to compute the reinitialization. In general, lower order schemes will result in larger perturbation of the zero level surface, which is not supposed to be moved.

The smoothness of the signum function  $\text{sgn}_h$  may influence the efficiency and effectiveness of the reinitialization procedure. In our simulations, with the regularized signum function defined in (24), it suffices to solve (23) for  $O(\epsilon)$  amount of time, in a neighborhood close to the zero level set. With  $\Delta t = k_0 h$ , and  $\epsilon = k_1 h$ ,  $k_0, k_1 > 0$ , we run constant number of time steps for reinitialization, independent of  $h$ . Since the fifth order WENO approximation of  $\nabla \phi$  uses central differencing with seven grid nodes along each grid lines, the minimal number of time steps needed to create the signed distance in  $\Gamma_\epsilon$  is  $(k_1 + 4)/k_0$ . We refer the readers to [20] for some more detailed discussion on reinitialization of level set functions and (closest point) extension of functions from  $\Gamma$  to  $\Gamma_\epsilon$ , and an alternative higher order algorithm.

### 3.2. Projections and weights

We locate all grid points  $\mathbf{x}_i \in U_h$  satisfying that  $|\phi_{SES}(\mathbf{x}_i)| < \epsilon$  and compute projections  $\mathbf{x}_i^* \in \Gamma$  by

$$\mathbf{x}_i^* = \mathbf{x}_i - \tilde{\phi}_{SES}(\mathbf{x}_i, t_n) \nabla \tilde{\phi}_{SES}(\mathbf{x}_i, t_n). \quad (26)$$

$\nabla \tilde{\phi}_{SES}(\mathbf{x}_i, t_n)$  can be approximated either by standard central differencing or by the fifth order WENO routines. More precisely, on each grid node for each Cartesian coordinate direction, WENO returns two approximations of  $\nabla \tilde{\psi}$ , say  $p_-$  and  $p_+$ , which are generalizations of the standard forward and backward finite differences of  $\psi$ . In our numerical simulations, we use

$$\nabla \tilde{\phi}_{SES} \approx \frac{p_- + p_+}{2}.$$

For weight function  $\delta_\epsilon$ , we adopt the following cosine function with vanishing first moment,

$$\delta_\epsilon(\eta) = \begin{cases} \frac{1}{2\epsilon} \left(1 + \cos \frac{\eta\pi}{\epsilon}\right), & |\eta| < \epsilon, \\ 0, & |\eta| \geq \epsilon. \end{cases} \quad (27)$$

For general smooth nonlinear integrands and  $\epsilon \sim o(1)$  for  $h \rightarrow 0$ , the above weight function provides at most second order in  $h$  convergence. Since the chosen  $\delta_\epsilon$  is an even function of the distance to the surface, it has one vanishing moment. Therefore, the zeroth order (in distance to the surface) approximation of the Jacobian will lead to a zero order in  $\epsilon$  error. See [49] for more in depth analysis on the properties of different choices of  $\delta_\epsilon$ . In the simulations reported in this paper, we set  $J(\mathbf{x}) \equiv 1$ .

### 3.3. Fast linear solvers

Equations (12)–(13) in Section 2.2, together with the regularization of the kernels described in Section 2.2.1, one arrives at the final linear system:

$$\Lambda \mathbf{p} + \mathbf{KWp} = \mathbf{g}, \quad (28)$$

with  $\mathbf{p}$  denoting the vector containing both  $\bar{\psi}(\mathbf{x}_j)$  and  $\bar{\psi}_n(\mathbf{x}_j)$ ,

$$\Lambda := \frac{1}{2} \begin{pmatrix} \left(1 + \frac{\epsilon_E}{\epsilon_I}\right) \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \left(1 + \frac{\epsilon_I}{\epsilon_E}\right) \mathbf{I} \end{pmatrix},$$

and  $\mathbf{W}$  is a diagonal matrix defined by the weights  $\omega_j := J(\mathbf{y}_j) \delta_\epsilon(d_\Gamma(\mathbf{y}_j))$  as defined in Section 2.2.

We solve this system by a standard GMRES algorithm. In the GMRES algorithm, we use the black-box fast multipole method (BBFMM) [29] to accelerate the multiplication of the operator  $\mathbf{K}$  to any vector. In particular, the solution of the diagonal part of (28) is used as a preconditioner. This means that the GMRES algorithm starts with the particular initial condition:

$$\mathbf{p}^{(0)} := (\Lambda + \mathbf{D})^{-1} \mathbf{g},$$

where

$$\mathbf{D} := \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

comes from the regularization of the kernels.

### 3.4. Computing surface area

In our IBIM approach, the evaluation of the surface area of  $\Gamma$  is computed by  $S_{\Gamma_\epsilon}^h[f]$  defined in (9) with  $f \equiv 1$ . See [48].

### 3.5. Computing the polarization energy

The polarization energy  $\mathcal{G}_{pol}$  of the system is given by

$$\mathcal{G}_{pol} = \frac{1}{2} \sum_{k=1}^{N_c} q_k \psi_{rxn}(\mathbf{z}_k) \quad (29)$$

where  $\psi_{rxn}(\mathbf{z}_k)$  is computed by evaluating the following boundary integral at the center of atom  $k$ ,  $\mathbf{z}_k$ :

$$\psi_{rxn}(\mathbf{z}) = \int_{\Gamma} \left( \left( \frac{\epsilon_E}{\epsilon_I} \frac{\partial G_\kappa(\mathbf{z}, \mathbf{y})}{\partial n(\mathbf{y})} - \frac{\partial G_0(\mathbf{z}, \mathbf{y})}{\partial n(\mathbf{y})} \right) \psi(\mathbf{y}) + (G_0(\mathbf{z}, \mathbf{y}) - G_\kappa(\mathbf{z}, \mathbf{y})) \frac{\partial \psi}{\partial n}(\mathbf{y}) \right) ds(\mathbf{y}).$$

In our IBIM approach, evaluation of this integral is computed by  $S_{\Gamma_\epsilon}^h[f(\mathbf{z}, \cdot)]$  defined in (9) with

$$f(\mathbf{z}, \mathbf{y}) := \left( \frac{\epsilon_E}{\epsilon_I} \frac{\partial G_\kappa(\mathbf{z}, P_\Gamma \mathbf{y})}{\partial n(\mathbf{y})} - \frac{\partial G_0(\mathbf{z}, P_\Gamma \mathbf{y})}{\partial n(\mathbf{y})} \right) \bar{\psi}(\mathbf{y}) + (G_0(\mathbf{z}, P_\Gamma \mathbf{y}) - G_\kappa(\mathbf{z}, P_\Gamma \mathbf{y})) \frac{\partial \bar{\psi}}{\partial n}(\mathbf{y}). \quad (30)$$

## 4. Numerical experiments

We now perform some numerical experiments using the computational algorithm we developed. In all the numerical simulations, we set the dielectric parameters  $\epsilon_I = 1.0$ ,  $\epsilon_E = 80.0$  and Debye–Hückel constant  $\kappa = 0.1257 \text{ \AA}^{-1}$ . We use the following parameters for the implicit boundary integral method:

- $h$  denotes the grid spacing in the uniform Cartesian grids,
- $\epsilon \equiv 2h$  denotes the width for the narrowband  $\Gamma_\epsilon$ ,
- $\tau = h$  or  $h/2$  denotes the regularization parameter used in  $K_{11}, K_{12}, K_{21}, K_{22}$ .

We set the tolerance in the GMRES algorithm to be  $10^{-5}$ , and use 4th order Chebyshev polynomials in the BBFMM preconditioner to achieve tolerance  $10^{-4}$  there. In general, smaller  $\tau$  results in more accurate approximations, if the resulting linear systems can be solved successfully. However, it cannot be too small with respect to the grid spacing, otherwise the resulting linear system becomes badly conditioned, as the current regularization approach becomes ineffective.

Most of the numerical experiments are performed on a desktop with quad-core CPU at 3.40 GHz, 16 GB RAM. The computations involving more than one million unknowns are performed on an older Linux computer with similar cache memory but sufficient RAM; for convenience in comparison, the timings presented in the tables below for simulations performed on this computer are scaled according to the clock speed and processor differences between the two computers. We think that such ad hoc scaling of timing is reasonable for the size of computations and the machines involved. We put an \* sign next to the scaled CPU timings in the tables.

In Section 4.1, we first compare the surface areas computed by our method to the ones computed by a published algorithm. In all later subsections, we present simulations of our algorithms with molecules of different sizes. In certain examples, we compare our computational results to the available published data. Particularly, we perform simulations on more realistic benchmark macromolecules taken from the RCSB Protein Data Bank (PDB) [11], and add missing heavy atoms through software PDB2PQR [25]. The atom charges and radius parameters that we will be using in our simulations are all generated through force field CHARMM [15]. The number of atoms reported in each subsection below corresponds to the number in the respective pqr file of each molecule.

Here are some general remarks on the numerical simulations using our algorithm. In the tables presented in this section, the columns titled “D.O.F.” show the total number of unknowns in the discrete systems and provides a basis to observe the rate of convergence for the computed solutions, surface areas and polarization energies. In the figures presented in this section, the electrostatics computed on two different grid resolutions are painted on top of the respective computed solvent excluded surfaces. In practice, the most important information that one would like to extract from such computations is the locations of the extrema; see e.g. [43,82]. As the figures show, the extrema of the potentials computed at the coarser grid resolutions are already at the “correct” locations on the surfaces.

As we shall see from the numerical accuracy study in Section 4.2, the foremost bottleneck of the proposed algorithm is the low order regularization for the singular integrals. However, regularization is essential, and smaller amount of it (smaller values of  $\tau$ ) leads to systems which require more GMRES iterations unless the grid spacing  $h$  is sufficiently small. See, for instance, the simulations presented in Tables 4, 5 and 6. Despite the regularization issue, the boundary integrals can be computed very accurately if wider  $\epsilon$  (with respect to the grid spacing  $h$ ) and the full expression of the Jacobian  $J$  are used. However, wider  $\epsilon$  implies a larger dense linear system needs to be solved. Most of the reported computation times are spent on the evaluation of the matrix–vector multiplications. Thus, in the simulations presented below, we choose a regime in which  $\epsilon$  is narrow but sufficient in practice for the adopted simple quadrature to resolve the surface geometry. Finally, regarding to how small  $h$  should be for a given molecule and probe size  $\rho_0$ :

$$h \approx \min_{j=1, \dots, N} \{\rho_0, r_j, 2\epsilon\} / 7 < \Delta$$

where  $\Delta$  is the minimal distance between “different parts of the surface” (think of the thin part of a dumb bell) and we define it as

$$\Delta = \inf_{\mathbf{x} \in \Gamma} \{-d_\Gamma(\mathbf{x} - s\mathbf{n}(\mathbf{x})) | \ell(\mathbf{x}) > s > 0, \text{ where } \mathbf{x} - \ell(\mathbf{x})\mathbf{n}(\mathbf{x}) \in \Gamma\}. \quad (31)$$

We shall see in the following examples, that our algorithm seems to perform well even when the discretized system is slightly outside of the above regime.

#### 4.1. Molecular SES surface area

We compare the performance of our algorithm for calculating surface areas of different proteins with that of the MSMS (Michel Sanner’s Molecular Surface) algorithm developed in [73]. For MSMS algorithm, the probe radius is set to be  $\rho_0 = 1.4$  Å and the density parameter is 1.0 for mesh generation. We use the online implementation by the High-Performance Computing at the NIH group [37] to produce the data for MSMS. In Table 1, we compare results from our method to these from MSMS for seven different proteins on a grid of size  $128^3$ . We observe that the surface areas computed by our algorithm are quite close to the MSMS’s approximate values in general. Since the MSMS results are only approximations to the true values, we did not attempt to tune algorithmic parameters (e.g. grid size, weight function) of our method to obtain results that are even closer to the MSMS results.

#### 4.2. The single ion model

We start with the single ion model developed in [46] to benchmark the solution accuracy of our numerical algorithm. We use three different relative errors, between exact and numerically represented quantities, to measure the quality of numerical solutions. They are defined as:



**Table 1**

Comparison of surface areas computed by the proposed IBIM method and by MSMS for seven different proteins from the RCSB Protein Data Bank.

Protein id	Surface area (IBIM)	Surface area (MSMS)
4INS	4732	4761
1HJE	825	801
1A2B	7540	7936
1PPE	7979	8340
2AID	8061	8304
1F15	22000	22725
1A63	6583	6659

**Table 2**

Benchmarking errors in solution of the single ion model.

Grid size	$h$ (Å)	$\tau/h$	D.O.F.	GMRES	Solution error	Area error	Energy error
$64^3$	$3.91\text{E}-1$	1	22,756	4	$1.11\text{E}-02$	$1.24\text{E}-03$	$1.21\text{E}-02$
$128^3$	$1.95\text{E}-1$	1	91,564	3	$6.90\text{E}-03$	$2.88\text{E}-04$	$5.68\text{E}-03$
$256^3$	$9.77\text{E}-2$	1	366,868	3	$3.57\text{E}-03$	$5.01\text{E}-05$	$2.90\text{E}-03$
$64^3$	$3.91\text{E}-1$	0.5	22,756	2	$5.93\text{E}-03$	$1.24\text{E}-03$	$5.98\text{E}-03$
$128^3$	$1.95\text{E}-1$	0.5	91,564	4	$3.45\text{E}-03$	$2.88\text{E}-04$	$2.49\text{E}-03$
$256^3$	$9.77\text{E}-2$	0.5	366,868	3	$2.15\text{E}-03$	$5.01\text{E}-05$	$1.31\text{E}-03$

$$\begin{aligned} \text{solution error} &= \sqrt{\frac{\int_{\Gamma} |\psi(\mathbf{x}) - \psi^*(\mathbf{x})|^2 + \left| \frac{\partial \psi(\mathbf{x})}{\partial n} - \frac{\partial \psi^*(\mathbf{x})}{\partial n} \right|^2}{\int_{\Gamma} |\psi^*(\mathbf{x})|^2 + \left| \frac{\partial \psi^*(\mathbf{x})}{\partial n} \right|^2}}, \\ \text{surface area error} &= \frac{|\mathcal{A} - \mathcal{A}^*|}{\mathcal{A}^*}, \\ \text{energy error} &= \frac{|\mathcal{G}_{pol} - \mathcal{G}_{pol}^*|}{\mathcal{G}_{pol}^*}. \end{aligned} \quad (32)$$

For a single atom with radius  $r$  and charge  $q$ , the solution to the Poisson–Boltzmann equation is given as [46]

$$\psi^*(\mathbf{x}) = \begin{cases} \frac{q}{4\pi\epsilon_I|\mathbf{x}|} + \frac{q}{4\pi r} \left( \frac{1}{\epsilon_E(1+\kappa r)} - \frac{1}{\epsilon_I} \right), & \text{if } |\mathbf{x}| < r \\ \frac{qe^{-\kappa(|\mathbf{x}|-r)}}{4\pi\epsilon(1+\kappa r)|\mathbf{x}|}, & \text{otherwise} \end{cases} \quad (33)$$

We can therefore compute the associated polarization energy

$$\mathcal{G}_{pol}^* = \frac{q^2}{8\pi r} \left( \frac{1}{\epsilon_E(1+\kappa r)} - \frac{1}{\epsilon_I} \right), \quad (34)$$

using the fact that the surface is a sphere with area  $\mathcal{A}^* = 4\pi r^2$ . We set the atom's radius to be  $r = 1$  Å and assigned charge to be  $q = 1e_c$ .

We performed simulations under different mesh and IBIM parameters. The results are summarized in Table 2. Our method converges in very small numbers (usually 3 ~ 4) of iterations. This benchmark calculation shows that our numerical algorithms can indeed achieve similar solution accuracies to those achieved by other algorithms developed recently [2,14].

#### 4.3. Protein 1A63

In this numerical example, we compute the polarization energy for protein 1A63, the E. Coli Rho factor, of the Protein Data Bank. The protein has 2065 atoms with different radii. The information on the locations and radii of the atoms are all available in [11].

In Fig. 4 we plot the potential  $\bar{\psi}$  on the constructed “solvent excluded surface”, computed on two different grids,  $128^3$  (left) and  $256^3$  (right). Further computational results are tabulated in Table 3. The computed values of the polarization energy  $\mathcal{G}_{pol}$  can be compared to the existing estimations,  $\mathcal{G}_{pol}^{\text{TABI}} = -2374.64$  kcal/mol from the treecode-based boundary integral solver TABI [31] and  $\mathcal{G}_{pol}^{\text{APBS}} = -2350.58$  kcal/mol from the finite difference solver APBS [78]. The computational results are tabulated in Table 3.

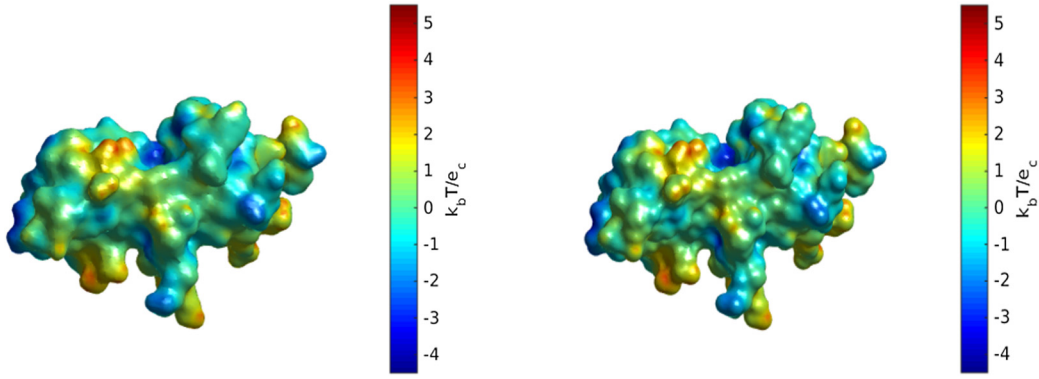


Fig. 4. The electrostatic potential on the surface of the PDB-1A63 protein. Left: on grid  $128^3$ . Right: on grid  $256^3$ .

Table 3

Numerical results on protein 1A63 under different algorithmic parameters.

Grid size	$h$ (Å)	$\tau/h$	D.O.F.	GMRES	$\mathcal{G}_{pol}$ (kcal/mol)	CPU (s)	Area (Å <sup>2</sup> )
$128^3$	$6.11\text{E}-1$	1	71,597	12	−2392.22	606.8	6583
$256^3$	$3.05\text{E}-1$	1	293,627	13	−2366.40	3041	6801
$128^3$	$6.11\text{E}-1$	0.5	71,597	13	−2345.41	772.3	6583
$256^3$	$3.05\text{E}-1$	0.5	293,627	14	−2347.74	3808	6801

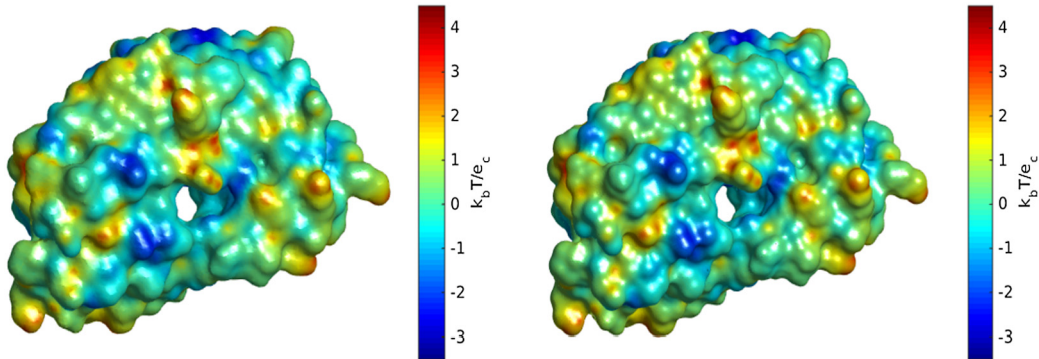


Fig. 5. The electrostatic potential on the surface of protein 2AID for two grids of size  $128^3$  (left) and  $256^3$  (right).

Table 4

Numerical results on protein 2AID under different algorithmic parameters.

Grid size	$h$ (Å)	$\tau/h$	D.O.F.	GMRES	$\mathcal{G}_{pol}$ (kcal/mol)	CPU (s)	Area (Å <sup>2</sup> )
$128^3$	$5.80\text{E}-1$	1	97,108	13	−2318.69	940.7	8061
$256^3$	$2.90\text{E}-1$	1	397,930	14	−2321.72	4521	8335
$128^3$	$5.80\text{E}-1$	0.5	97,108	24	−2282.14	1745	8601
$256^3$	$2.90\text{E}-1$	0.5	397,930	15	−2306.70	4906	8335

#### 4.4. Protein 2AID

Here we compute the polarization energy for protein 2AID, a non-peptide inhibitor complexed with HIV-1 protease. This protein has 3130 atoms. In Fig. 5 we plot the potential  $\bar{\psi}$  on the constructed “solvent excluded surface” of this protein, computed on two different grids. Further computational results are tabulated in Table 4.

#### 4.5. Protein 1F15

In this example, we compute the polarization energy for protein 1F15, the cucumber mosaic virus. The protein has 8494 atoms. In Fig. 6 we plot the potential  $\bar{\psi}$  on the constructed “solvent excluded surface”, computed on two different grids. Further computational results are tabulated in Table 5.

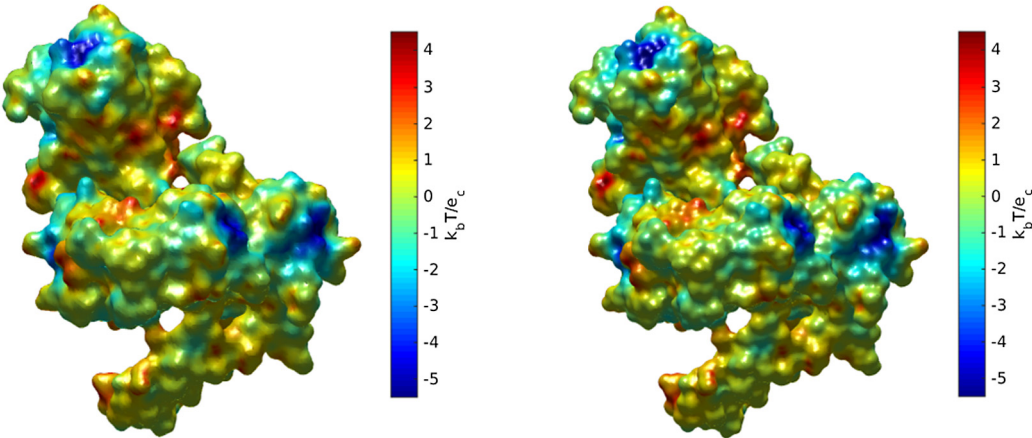


Fig. 6. The electrostatic potential on the surface of protein 1F15 for two different grids of sizes  $128^3$  (left) and  $256^3$  (right).

**Table 5**  
Numerical results on protein 1F15 under different algorithmic parameters.

Grid size	$h$ (Å)	$\tau/h$	D.O.F.	GMRES	$\mathcal{G}_{pol}$ (kcal/mol)	CPU (s)	Area (Å <sup>2</sup> )
$128^3$	$7.72\text{E}-1$	1	147,463	17	$-7770.00$	2586	22000
$256^3$	$3.86\text{E}-1$	1	613,726	21	$-7818.83$	12357	22847
$512^3$	$1.93\text{E}-1$	1	2,497,309	25	$-7891.05$	45681*	23238
$128^3$	$7.72\text{E}-1$	0.5	147,463	31	$-7682.67$	4667	22000
$256^3$	$3.86\text{E}-1$	0.5	613,726	26	$-7774.76$	14129	22847
$512^3$	$1.93\text{E}-1$	0.5	2,497,309	29	$-7875.86$	52687*	23238

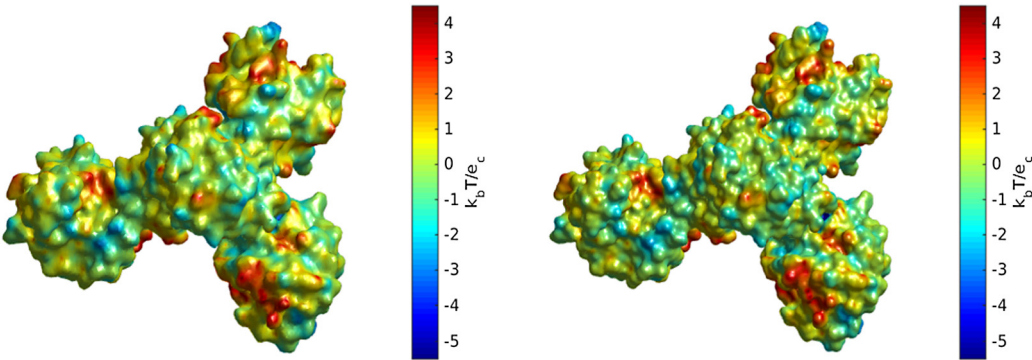


Fig. 7. The electrostatic potential on the surface of protein 1A2K for two different grids of sizes  $128^3$  (left) and  $256^3$  (right).

4.6. Protein 1A2K

In this numerical example, we compute the polarization energy for protein 1A2K, the GTPase RAN-NTF2 complex. The protein has 13627 atoms.

In Fig. 7 we plot the potential  $\bar{\psi}$  on the constructed “solvent excluded surface”, computed on two different grids. Further computational results are tabulated in Table 6.

4.7. Protein: 1PMA

In this example, we compute the polarization energy for proteasome from thermoplasma acidophilum (PDB id: 1PMA) with 93017 atoms. In Fig. 8 we plot the potential  $\bar{\psi}$  on the constructed “solvent excluded surface”, computed on two different grids. Further computational results are tabulated in Table 7.

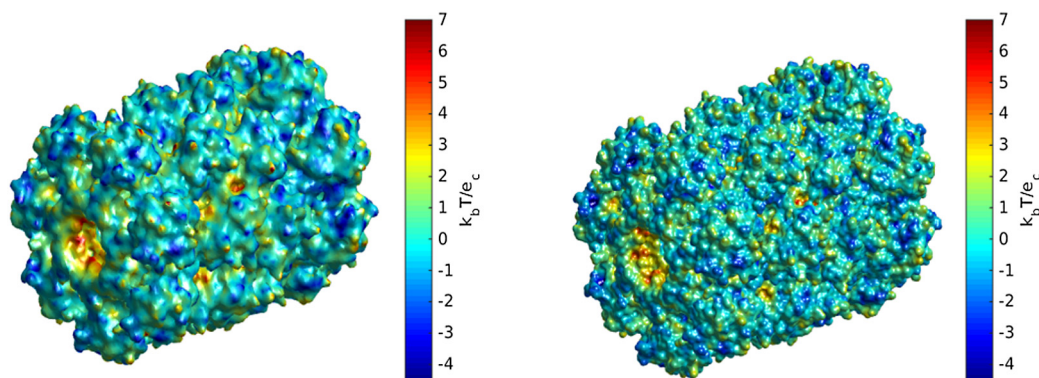
5. Concluding remarks

We present in this paper a new numerical method for solving the boundary value problem of the linearized Poisson–Boltzmann equation, which is widely used to model the electric potential for macromolecules in solvent. Our new method



**Table 6**  
Numerical results on protein 1A2K under different algorithmic parameters.

Grid size	$h$ (Å)	$\tau/h$	D.O.F.	GMRES	$\mathcal{G}_{pol}$ (kcal/mol)	CPU (s)	Area (Å <sup>2</sup> )
128 <sup>3</sup>	1.08E+0	1	99,783	12	−7190.38	1032	29497
256 <sup>3</sup>	5.42E−1	1	429,451	17	−8902.62	6120	31501
384 <sup>3</sup>	3.61E−1	1	983,418	17	−8920.93	11948	32047
512 <sup>3</sup>	2.71E−1	1	1,765,673	18	−8963.12	25496*	32364
128 <sup>3</sup>	1.08E+0	0.5	99,783	21	−9004.68	2309	29497
256 <sup>3</sup>	5.42E−1	0.5	429,451	43	−8789.45	15528	31501
384 <sup>3</sup>	3.61E−1	0.5	983,418	40	−8859.47	26030	32047
512 <sup>3</sup>	2.71E−1	0.5	1,765,673	23	−8921.64	33558*	32364



**Fig. 8.** The electrostatic potential on molecular surface for the proteasome from thermoplasma acidophilum (PDB id: 1PMA). Left: the potential computed on a 128<sup>3</sup> grid. Right: the potential computed on a 512<sup>3</sup> grid.

**Table 7**  
Protein 1PMA. IBIM's result of relative error in polarization energy and total run time w.r.t. different grid sizes.

Grid size	$h$ (Å)	$\tau/h$	D.O.F.	GMRES	$\mathcal{G}_{pol}$ (kcal/mol)	CPU (s)	Area (Å <sup>2</sup> )
128 <sup>3</sup>	1.67E+0	1	222,478	15	−15544.35	3360	1.6014E+5
256 <sup>3</sup>	8.35E−1	1	1,026,938	18	−46865.78	14141*	1.8034E+5
384 <sup>3</sup>	5.56E−1	1	2,429,367	20	−50071.90	42927*	1.8851E+5
512 <sup>3</sup>	4.17E−1	1	4,418,314	22	−50144.63	70880*	1.9262E+5

relies on the standard level set method [66,67] for preparing the distance function to the “molecular surface”. Contrary to the typical level set method, in which some partial differential equations are discretized with some suitable boundary conditions, ours involves the solution of an integral equation which is derived from an implicit boundary integral formulation [47]. Similar to the typical level set methods, and contrary to the typical boundary integral methods, the proposed method involves computation only with functions defined on uniform Cartesian grids. Our numerical simulations show that in addition to the flexibility that comes from the level set methods, the proposed method can be as computationally efficient as other boundary integral based algorithms. We show by our numerical simulations that the solutions of the resulting linear systems can be accelerated easily by some existing fast multipole methods. Furthermore, the eikonal flow and reinitialization in Stage (1) of the proposed algorithm rely on widely available explicit solvers and can be trivially parallelized.

There are several possible improvements that could be investigated in the future. First of all, the quadrature for the implicit boundary integral formulation can be improved to increase the order of accuracy. This includes improvement of the regularization of the kernel singularities and the use of full expression of the Jacobian  $J$ . One may also consider different grid geometries, as the underlying mathematical formulation does not require uniform Cartesian grids. For example, the adaptive oct-tree structure used in [36] or radial basis functions may be considered.

As all the presented simulations were computed on two moderate desktop computers, the reported results show the potential of the proposed method for molecular dynamics simulations involving very large molecules.

Finally, let us mention that the numerical method we proposed here can be generalized to solve many similar model problems for electrostatics in related areas of electrochemistry [40,59,61,34,35].

## Acknowledgements

Zhong and Ren are partially supported by the National Science Foundation through grants DMS-1321018 and DMS-1620473. Tsai is partially supported by the National Science Foundation through grants DMS-1318975 and

DMS-1720171, and Army Research Office Grant No. W911NF-12-1-0519. We would like to thank Professor Chandrajit Bajaj for motivating conversations on topics related to this paper.

## References

- [1] M.D. Altman, J.P. Bardhan, J.K. White, 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 (2009) 132–153.
- [2] C. Bajaj, S.-C. Chen, A. Rand, An efficient higher-order fast multipole boundary element solution for Poisson–Boltzmann-based molecular electrostatics, *SIAM J. Sci. Comput.* 33 (2011) 826–848.
- [3] N. Baker, M. Holst, F. Wang, Adaptive multilevel finite element solution of the Poisson–Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems, *J. Comput. Chem.* 22 (2000) 1343–1352.
- [4] N. Baker, D. Sept, S. Joseph, M.J. Holst, J.A. McCammon, Electrostatics of nanosystems: application to microtubules and the ribosome, *Proc. Natl. Acad. Sci.* 98 (2001) 10037–10041.
- [5] N.A. Baker, Poisson–Boltzmann methods for biomolecular electrostatics, *Methods Enzymol.* 383 (2004) 94–118.
- [6] N.A. Baker, Improving implicit solvent simulations: a Poisson-centric view, *Curr. Opin. Struct. Biol.* 15 (2005) 137–143.
- [7] M. Baptista, R. Schmitz, B. Dünweg, Simple and robust solver for the Poisson–Boltzmann equation, *Phys. Rev. E* 80 (2009) 016705.
- [8] A.J. Bard, L.R. Faulkner, *Electrochemical Methods: Fundamentals and Applications*, 2nd ed., Wiley, New York, 2000.
- [9] J.P. Bardhan, Numerical solution of boundary-integral equations for molecular electrostatics, *J. Chem. Phys.* 130 (2009) 094102.
- [10] J.P. Bardhan, Nonlocal continuum electrostatic theory predicts surprisingly small energetic penalties for charge burial in proteins, *J. Chem. Phys.* 135 (2011) 104113.
- [11] H.M. Berman, T. Battistuz, T. Bhat, W.F. Bluhm, P.E. Bourne, K. Burkhardt, Z. Feng, G.L. Gilliland, L. Iype, S. Jain, et al., The protein data bank, *Acta Crystallogr. Sect. D, Biol. Crystallogr.* 58 (2002) 899–907.
- [12] I. Borukhov, D. Andelman, H. Orland, Steric effects in electrolytes: a modified Poisson–Boltzmann equation, *Phys. Rev. Lett.* 79 (1997) 435–437.
- [13] A.H. Boschitsch, M.O. Fenley, Hybrid boundary element and finite difference method for solving the nonlinear Poisson–Boltzmann equation, *J. Comput. Chem.* 25 (2004) 935–955.
- [14] A.H. Boschitsch, M.O. Fenley, H.-X. Zhou, Fast boundary element method for the linear Poisson–Boltzmann equation, *J. Phys. Chem. B* 106 (2002) 2741–2754.
- [15] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, M. Karplus, CHARMM: a program for macromolecular energy, minimization, and dynamics calculations, *J. Comput. Chem.* 4 (1983) 187–217.
- [16] Q. Cai, J. Wang, H.-K. Zhao, R. Luo, On removal of charge singularity in Poisson–Boltzmann equation, *J. Chem. Phys.* 130 (2009) 145101.
- [17] C. Chen, R. Tsai, Implicit boundary integral methods for the Helmholtz equation in exterior domains, *Res. Math. Sci.* (2017), <https://doi.org/10.1186/s40687-017-0108-y>, in press.
- [18] D. Chen, Z. Chen, C.J. Chen, W.H. Geng, G.W. Wei, MIBPB: a software package for electrostatic analysis, *J. Comput. Chem.* 32 (2011) 756–770.
- [19] J.H. Chen, C.L. Brooks III, J. Khandogin, Recent advances in implicit solvent-based methods for biomolecular simulations, *Curr. Opin. Struct. Biol.* 18 (2008) 140–148.
- [20] L. Cheng, R. Tsai, Redistancing by flow of time dependent Eikonal equation, *J. Comput. Phys.* 227 (2008).
- [21] D.M. Chipman, Solution of the linearized Poisson–Boltzmann equation, *J. Chem. Phys.* 120 (2004) 5566–5575.
- [22] C.J. Cramer, D.G. Truhlar, Implicit solvation models: equilibria, structure, spectra, and dynamics, *Chem. Rev.* 99 (1999) 2161–2200.
- [23] M.E. Davis, J.A. McCammon, Electrostatics in biomolecular structure and dynamics, *Chem. Rev.* 90 (1990) 509–521.
- [24] D.A. Di Pietro, R. Specogna, An a posteriori-driven adaptive mixed high-order method with application to electrostatics, *J. Comput. Phys.* 326 (2016) 35–55.
- [25] T.J. Dolinsky, J.E. Nielsen, J.A. McCammon, N.A. Baker, PDB2PQR: an automated pipeline for the setup of Poisson–Boltzmann electrostatics calculations, *Nucleic Acids Res.* 32 (2004) W665–W667.
- [26] M. Feig, C.L. Brooks III, Recent advances in the development and application of implicit solvent models in biomolecule simulations, *Curr. Opin. Struct. Biol.* 14 (2004) 217–224.
- [27] M. Feig, A. Onufriev, M.S. Lee, W. Im, D.A. Case, C.L. Brooks III, Performance comparison of generalized Born and Poisson methods in the calculation of electrostatic solvation energies for protein structures, *J. Comput. Chem.* 25 (2004) 265–284.
- [28] F. Fogolari, A. Brigo, H. Molinari, The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology, *J. Mol. Recognit.* 15 (2002) 377–392.
- [29] W. Fong, E. Darve, The black-box fast multipole method, *J. Comput. Phys.* 228 (2009) 8712–8725.
- [30] W. Geng, Parallel higher-order boundary integral electrostatics computation on molecular surfaces with curved triangulation, *J. Comput. Phys.* 241 (2013) 253–265.
- [31] W. Geng, R. Krasny, A treecode-accelerated boundary integral Poisson–Boltzmann solver for electrostatics of solvated biomolecules, *J. Comput. Phys.* 247 (2013) 62–78.
- [32] W. Geng, S. Yu, G. Wei, Treatment of charge singularities in implicit solvent models, *J. Chem. Phys.* 127 (2007) 114106.
- [33] S. Grandison, R. Penfold, J.-M. Vanden-Broeck, A rapid boundary integral equation technique for protein electrostatics, *J. Comput. Phys.* 224 (2007) 663–680.
- [34] M. Harmon, I.M. Gamba, K. Ren, Numerical algorithms based on Galerkin methods for the modeling of reactive interfaces in photoelectrochemical (PEC) solar cells, *J. Comput. Phys.* 327 (2016) 140–167.
- [35] Y. He, I.M. Gamba, H.C. Lee, K. Ren, On the modeling and simulation of reaction-transfer dynamics in semiconductor-electrolyte solar cells, *SIAM J. Appl. Math.* 75 (2015) 2515–2539.
- [36] A. Helgadóttir, F. Gibou, A Poisson–Boltzmann solver on irregular domains with Neumann or Robin boundary conditions on non-graded adaptive grid, *J. Comput. Phys.* 230 (2011) 3830–3848.
- [37] High-performance computing at the NIH, <https://hpcwebapps.cit.nih.gov/structbio/basic.html>.
- [38] M.J. Holst, F. Said, Numerical solution of the nonlinear Poisson–Boltzmann equation: developing more robust and efficient methods, *J. Comput. Chem.* 16 (1995) 336–364.
- [39] B. Honig, A. Nicholls, Classical electrostatics in biology and chemistry, *Science* 268 (1995) 1144–1149.
- [40] T.L. Horng, T.C. Lin, C. Liu, B. Eisenberg, PNP equations with steric effects: a model of ion flow through channels, *J. Phys. Chem. B* 116 (2012) 11422–11441.
- [41] W. Im, D. Beglov, B. Roux, Continuum solvation model: computation of electrostatic forces from numerical solutions to the Poisson–Boltzmann equation, *Comput. Phys. Commun.* 111 (1998) 59–75.
- [42] G.-S. Jiang, D. Peng, Weighted ENO schemes for Hamilton–Jacobi equations, *SIAM J. Sci. Comput.* 21 (2000) 2126–2143.
- [43] S. Jones, H.P. Shanahan, H.M. Berman, J.M. Thornton, Using electrostatic potentials to predict DNA-binding sites on DNA-binding proteins, *Nucleic Acids Res.* 31 (2003) 7189–7198.

- [44] A. Juffer, E. Botta, B. van Keulen, A. van der Ploeg, H. Berendsen, The electric potential of a macromolecule in a solvent: a fundamental approach, *J. Comput. Phys.* 97 (1991) 144–171.
- [45] C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.
- [46] J.G. Kirkwood, Theory of solutions of molecules containing widely separated charges with special application to zwitterions, *J. Chem. Phys.* 2 (1934) 351–361.
- [47] C. Kublik, N. Tanushev, R. Tsai, An implicit interface boundary integral method for Poisson's equation on arbitrary domains, *J. Comput. Phys.* 247 (2013) 279–311.
- [48] C. Kublik, R. Tsai, Integration over curves and surfaces defined by the closest point mapping, *Res. Math. Sci.* 3 (2016).
- [49] C. Kublik, R. Tsai, An extrapolative approach to integration over hypersurfaces in the level set framework, *Math. Comp.* (2018), <https://doi.org/10.1090/mcom/3282>, in press.
- [50] A.W. Lange, J.M. Herbert, A simple polarizable continuum solvation model for electrolyte solutions, *J. Chem. Phys.* 134 (2011) 204110.
- [51] J. Li, D. Xie, A new linear Poisson–Boltzmann equation and finite element solver by solution decomposition approach, *Commun. Math. Sci.* 13 (2015) 315–325.
- [52] J. Liang, S. Subramaniam, Computation of molecular electrostatics with boundary element methods, *Biophys. J.* 73 (1997) 1830–1841.
- [53] B. Lu, Y. Zhou, M. Holst, J. McCammon, Recent progress in numerical methods for the Poisson–Boltzmann equation in biophysical applications, *Commun. Comput. Phys.* 3 (2008) 973–1009.
- [54] B.Z. Lu, X. Cheng, J. Huang, J.A. McCammon, Order  $n$  algorithm for computation of electrostatic interactions in biomolecular systems, *Proc. Natl. Acad. Sci.* 103 (2006) 19314–19319.
- [55] B.Z. Lu, X. Cheng, J.A. McCammon, New-version-fast-multipole-method accelerated electrostatic calculations in biomolecular systems, *J. Comput. Phys.* 226 (2007) 1348–1366.
- [56] R. Luo, L. David, M.K. Gilson, Accelerated Poisson–Boltzmann calculations for static and dynamic systems, *J. Comput. Chem.* 23 (2002) 1244–1253.
- [57] A.D. Mackerell Jr., Empirical force fields for biological macromolecules: overview and issues, *J. Comput. Chem.* 25 (2004) 1584–1604.
- [58] J.D. Madura, J.M. Briggs, R.C. Wade, M.E. Davis, B.A. Luty, A. Ilin, J. Antosiewicz, M.K. Gilson, B. Bagheri, L.R. Scott, J.A. McCammon, Electrostatics and diffusion of molecules in solution – simulations with the University-of-Houston Brownian dynamics program, *Comput. Phys. Commun.* 91 (1995) 57–95.
- [59] S. Mafé, J. Pellicer, V.M. Aguilera, A numerical approach to ionic transport through charged membranes, *J. Comput. Phys.* 75 (1988) 1–14.
- [60] A.V. Marenich, C.J. Cramer, D.G. Truhlar, Universal solvation model based on solute electron density and on a continuum model of the solvent defined by the bulk dielectric constant and atomic surface tensions, *J. Phys. Chem. B* 113 (2009) 6378–6396.
- [61] S.R. Mathur, J.Y. Murthy, A multigrid method for the Poisson–Nernst–Planck equations, *Int. J. Heat Mass Transf.* 52 (2009) 4031–4039.
- [62] M. Mirzadeh, M. Theillard, F. Gibou, A second-order discretization of the nonlinear Poisson–Boltzmann equation over irregular geometries using non-graded adaptive Cartesian grids, *J. Comput. Phys.* 230 (2011) 2125–2140.
- [63] M. Mirzadeh, M. Theillard, A. Helgadóttir, D. Boy, F. Gibou, An adaptive, finite difference solver for the nonlinear Poisson–Boltzmann equation with applications to biomolecular computations, *Commun. Comput. Phys.* 13 (2013) 150–173.
- [64] A. Nicholls, R. Bharadwaj, B. Honig, GRASP – graphical representation and analysis of surface-properties, *Biophys. J.* 64 (1993) A166.
- [65] M. Orozco, F.J. Luque, Theoretical methods for the description of the solvent effect in biomolecular systems, *Chem. Rev.* 100 (2000) 4187–4225.
- [66] S. Osher, R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, 2000.
- [67] S. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, *J. Comput. Phys.* 79 (1988) 12–49.
- [68] Q. Pan, X.-C. Tai, Model the solvent-excluded surface of 3D protein molecular structures using geometric PDE-based level-set method, *Commun. Comput. Phys.* 6 (2009) 777–792.
- [69] P. Ren, J. Chun, D.G. Thomas, M.J. Schnieders, M. Marucho, J. Zhang, N.A. Baker, Biomolecular electrostatics and solvation: a computational perspective, *Q. Rev. Biophys.* 45 (2012) 427–491.
- [70] W. Rocchia, E. Alexov, B. Honig, Extending the applicability of the nonlinear Poisson–Boltzmann equation: multiple dielectric constants and multivalent ions, *J. Phys. Chem. B* 105 (2001) 6507–6514.
- [71] B. Roux, T. Simonson, Implicit solvent models, *Biophys. Chem.* 78 (1999) 1–20.
- [72] E. Rouy, A. Tourin, A viscosity solutions approach to shape-from-shading, *SIAM J. Numer. Anal.* 29 (1992) 867–884.
- [73] M.F. Sanner, A.J. Olson, J.-C. Spehner, Reduced surface: an efficient way to compute molecular surfaces, *Biopolymers* 38 (1996) 305–320.
- [74] C.-W. Shu, S. Osher, Efficient implementation of essentially nonoscillatory shock-capturing schemes, *J. Comput. Phys.* 77 (1988) 439–471.
- [75] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* 114 (1994) 146–159.
- [76] J. Tomasi, B. Mennucci, R. Cammi, Quantum mechanical continuum solvation models, *Chem. Rev.* 105 (2005) 2999–3094.
- [77] Y.-H.R. Tsai, S. Osher, Total variation and level set methods in image science, *Acta Numer.* 14 (2005) 509–573.
- [78] S. Unni, Y. Huang, R.M. Hanson, M. Tobias, S. Krishnan, W.W. Li, J.E. Nielsen, N.A. Baker, Web servers and services for electrostatics calculations with APBS and PDB2PQR, *J. Comput. Chem.* 32 (2011) 1488–1491.
- [79] Y.N. Vorobjev, J.A. Grant, H.A. Scheraga, A combined iterative and boundary element approach for solution of the nonlinear Poisson–Boltzmann equation, *J. Am. Chem. Soc.* 114 (1992) 3189–3196.
- [80] J. Wang, C. Tan, Y.-H. Tan, Q. Lu, R. Luo, Poisson–Boltzmann solvents in molecular dynamics simulations, *Commun. Comput. Phys.* 3 (2008) 1010–1031.
- [81] S. Weggler, V. Rutka, A. Hildebrandt, A new numerical method for nonlocal electrostatics in biomolecular simulations, *J. Comput. Phys.* (2010) 4059–4074.
- [82] P.K. Weiner, R. Langridge, J.M. Blaney, R. Schaefer, P.A. Kollman, Electrostatic potential molecular surfaces, *Proc. Natl. Acad. Sci.* 79 (1982) 3754–3758.
- [83] D. Xie, Y. Jiang, A nonlocal modified Poisson–Boltzmann equation and finite element solver for computing electrostatics of biomolecules, *J. Comput. Phys.* 322 (2016) 1–20.
- [84] D. Xie, Y. Jiang, P. Brune, L.R. Scott, A fast solver for a nonlocal dielectric continuum model, *SIAM J. Sci. Comput.* 34 (2012) B107–B126.
- [85] Y. Xie, J. Ying, D. Xie, SMPBS: web server for computing biomolecular electrostatics using finite element solvers of size modified Poisson–Boltzmann equation, *J. Comput. Chem.* 38 (2017) 541–552.
- [86] J. Ying, D. Xie, A new finite element and finite difference hybrid method for computing electrostatics of ionic solvated biomolecule, *J. Comput. Phys.* 298 (2015) 636–651.
- [87] R.J. Zauhar, R.S. Morgan, Computing the electric potential of biomolecules: application of a new method of molecular surface triangulation, *J. Comput. Chem.* 11 (1990) 603–622.
- [88] B. Zhang, B.Z. Lu, X. Cheng, J.F. Huang, N.P. Pitsianis, X. Sun, J.A. McCammon, Mathematical and numerical aspects of the adaptive fast multipole Poisson–Boltzmann solver, *Commun. Comput. Phys.* 13 (2013) 107–128.
- [89] L.Y. Zhang, E. Gallicchio, R.A. Friesner, R.M. Levy, Solvent models for protein–ligand binding: comparison of implicit solvent Poisson and surface generalized Born models with explicit solvent simulations, *J. Comput. Chem.* 22 (2001) 591–607.
- [90] Z. Zhang, S. Witham, E. Alexov, On the role of electrostatics in protein–protein interactions, *Phys. Biol.* 8 (2011) 035001.