Received: 23 May 2023 | Revised: 8 August 2023 | Accepted: 24 September 2023
DOI: 10.1002/jcc.27239

RESEARCH ARTICLE

Bridging Eulerian and Lagrangian Poisson–Boltzmann solvers by ESES

Sheik Ahmed Ullah[1] | Xin Yang[2] | Ben Jones[3] | Shan Zhao[4] | Weihua Geng[2] | Guo-Wei Wei[3]

[1]Model Risk, Regions Bank, Birmingham, Alabama, USA
[2]Department of Mathematics, Southern Methodist University, Dallas, Texas, USA
[3]Department of Mathematics, Michigan State University, East Lansing, Michigan, USA
[4]Department of Mathematics, University of Alabama, Tuscaloosa, Alabama, USA

Correspondence
Weihua Geng, Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA.
Email: wgeng@smu.edu

Funding information
National Institutes of Health, Grant/Award Numbers: DMS-2110914, DMS-2110922, DMS-2110869, IIS-1900473; National Science Foundation, Grant/Award Numbers: R01AI164266, R35GM148196

Abstract
The Poisson–Boltzmann (PB) model is a widely used electrostatic model for biomolecular solvation analysis. Formulated as an elliptic interface problem, the PB model can be numerically solved on either Eulerian meshes using finite difference/finite element methods or Lagrangian meshes using boundary element methods. Molecular surface generators, which produce the discretized dielectric interfaces between solutes and solvents, are critical factors in determining the accuracy and efficiency of the PB solvers. In this work, we investigate the utility of the Eulerian Solvent Excluded Surface (ESES) software for rendering conjugated Eulerian and Lagrangian surface representations, which enables us to numerically validate and compare the quality of Eulerian PB solvers, such as the MIBPB solver, and the Lagrangian PB solvers, such as the TABI-PB solver. Furthermore, with the ESES software and its associated PB solvers, we are able to numerically validate an interesting and useful but often neglected source-target symmetric property associated with the linearized PB model.

KEYWORDS
boundary integral method, conjugated Eulerian and Lagrangian molecular surfaces, electrostatics, matched interface and boundary, Poisson–Boltzmann

1 | INTRODUCTION

As one of the four components of Maxwell's equations, the Gauss's law relates the distribution of electric charges to the resulting electric field. The Gauss's law can be expressed in both differential and integral forms. The integral form describes the electric flux through a closed surface, and the differential form yields the Poisson's equation. By considering both Eulerian and Lagrangian representations of the molecular surfaces of biomolecules, this paper studies the connection between the two forms of the Gauss's law as well as the associated numerical algorithms for solving the electric potentials. In the scale of biomolecules (e.g., Å) such as protein and nucleic acids, we consider the popular Poisson–Boltzmann (PB) model with the following assumptions. (i) The biomolecules carry electrostatic charges, modeled by partial charges located at the atomic centers. These charges are obtained from force fields based on experimental and quantum chemistry computation. (ii) The biomolecules are solvated in solvent and the solvent molecules (e.g., water) cannot reach the inside, thus a dielectric surface which separates the solute and solvent will be considered. In our practice, we use the solvent excluded surface (SES),[1] also known as the molecular surface. (iii) There are dissolved electrolytes in the solvent. Due to the abundance and bulk effects of the electrolytes, they are modeled with the Boltzmann distribution, a probability function used in statistical physics to characterize state of a system of particles, with respect to temperature and energy.

With the trending of the studies and findings in biology and health care at the molecular levels, the PB model has shown its broad applications in biomolecular simulations such as protein structure,[2] chromatin packing,[3] pKa,[4-6] membrane,[7] binding energy,[8] solvation free energy,[9] ion channel profiling,[10] etc.

Mathematically, the PB model is an elliptic interface problem with several numerical difficulties such as discontinuous dielectric coefficients, singular sources, a complex interface, and an unbounded domain. Grid-based finite difference or finite volume discretizations that discretize the entire volumetric domain have been developed in, for example, References 11-17. The grid-based discretization is efficient and robust and is therefore popular. However, solvers that are based on discretizing the partial differential equation may suffer from accuracy reduction due to discontinuity of the dielectric coefficients, nonsmoothness of the solution, singularity of the sources, and truncation of the domains, unless special interface[18,19] and singularity[20-24] treatments are applied. These treatments come at the price of more complicated discretization scheme and possibly reduced convergence speed of the iterative solver for the linear system. For the linearized PB model, an alternative approach is to reformulate the linear PB equation as a boundary integral equation and use the boundary elements to discretize the molecular surface, for example References 25-33. Besides the reduction from three-dimensional space to the two-dimensional molecular boundary, this approach has the advantage that singular charges, interface conditions, and far-field condition are incorporated analytically in the formulation, and hence do not impose additional approximation errors. Furthermore the boundary integral approach can be efficiently accelerated by fast algorithms such as fast multipole method[34] and treecode.[35,36]

The underlying connection between the finite difference PB solvers and the boundary integral PB solvers is in fact the connection between the differential form and integral form of the Gauss's law. When a finite difference method is used, a rigorous treatment of interface conditions requires the location where the mesh line intersects the molecular surface as well as the normal direction of the surface at the intersection. This is called a Eulerian representation of the molecular surface as seen in Figure 1A. Alternatively the boundary integral method requires a body-fitted triangulation of the molecular surface, which is called the Lagrangian representation as seen in Figure 1B.

The molecular surface normally has three popular definitions. The van der Waals (VDW) surface, which is the union of exposed atomic surfaces, is a simple definition that can be used for both molecular modeling and visualization. However, the VDW surface gives rise to numerous geometric singularities on the surface and thus causes numerical instability when it is used as an interface for implicit solvation modeling. The other two surfaces requires a spherical solvent probe to rotate on contacting the VDW surface. The trace of the center of the probe forms the Solvent Accessible Surface (SAS) while the contact, toroidal, and reentrant surfaces of the probe sphere form the SES as introduced by Lee and Richards.[1] The SAS has relatively simple definition and is easier to be numerically described. However, even though it is smoother than VDW, there are still many sharp corners when the probe transits from one atom to the other. The SES is the smoothest among the three as an essentially C^1-continuous surface with only cusps under extreme situations.[19] Thus SES becomes the most popular surface definition in biophysics and molecular biology. In addition to its usage in biomolecular visualization, it has been widely used in implicit solvent models due to its C^1 alike smoothness. Connolly formulates the mathematical

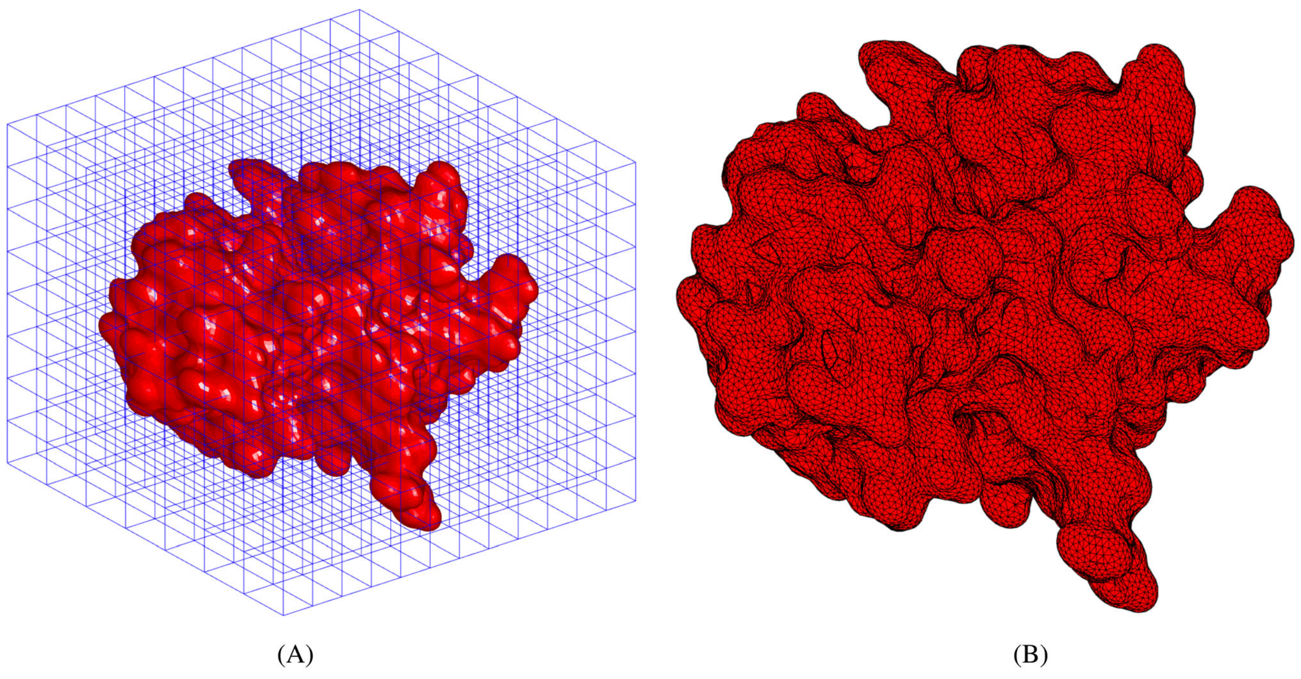

FIGURE 1 The two solvent excluded surface (SES) representation of protein 1a63: (A) Eulerian representation with location and surface normal direction of intersection between mesh lines and the SES surface; (B) Lagrangian representation with triangles and surface normal direction at the vertices.

representation of the SES for arbitrary biomolecules in terms of convex patches, saddle patches, and concave patches.[37] There are many available software generating the SES in Lagrangian representations such as MSMS,[38] NanoShaper,[39] and TMSmesh.[40]

Such representations are especially suitable for boundary integral methods, and a comparison between the popular MSMS and NanoShaper has been conducted in Reference 41. When the Eulerian representation is needed, additional work is required to find intersection of the mesh line and its normal direction by very challenging interpolation schemes.[42,43]

Previously, Liu et al. developed a software package called Eulerian solvent excluded surface (ESES)[44] for the generation of accurate SESs in Cartesian grids. ESES offers the description of the solvent and solute domains by specifying all the intersection points between the SES and the Cartesian grid lines. Additionally, the interface normal at each intersection point is evaluated. Furthermore, for a given biomolecule, the ESES software not only provides the whole surface area, but also partitions the surface area according to atomic types.

In this paper, we will focus on another important feature of ESES, the body-fitted or Lagrangian representation of the molecular surface. In the previous work, the Lagrangian representation of ESES is used only for visualization.[44] Here we will quantitatively measure the performance of ESES in triangulation and investigate its usage for boundary integral PB solver, for example, the TABI solver,[30] compared with the finite difference interface PB solver named Matched Interface and Boundary (MIB)[21,23,45] on Eulerian representation of ESES. This paper presents the first exploration in the literature for studying the conjugated Eulerian and Lagrangian surfaces of ESES, as well as the associated differential and integral forms of the Gauss's law. In addition, we study a source–target symmetric property of the linearized PB model, which has its significance in theory and application.

The rest of the paper is organized in the following ways. In the section of theory and algorithm, we briefly introduce the PB model followed by algorithms of MIB solver, TABI solver, and the SES generator ESES. In the next section, we focus on the derivation of the source-target symmetric property of the linearized PB model. In the validation section, we provide numerical results for the demonstration of ESES triangulation quality, MIB solver using its Eulerian representation and TABI solver using its Lagrangian representation. Following that, we show numerical results for the validation of symmetric properties derived in the theory and algorithm section. After a short section about software dissemination, the paper is concluded with a section of concluding remarks.

2 | METHODOLOGY

In this section, we will first briefly describe the Poisson-Boltzmann (PB) model. Then, we introduce the ESES surface generator. A summary of the MIB method PB solver and the treecode-accelerated boundary integral (TABI) PB solver will be offered next. At the end of this section, we describe the target-source symmetric property as stated in[46] and introduce its potential applications.

2.1 | The PB model

The PB model is illustrated in Figure 2A. Consider a large domain Ω in $\mathbb{R}^3$ containing the solute protein. The domain Ω is divided by the molecule surface Γ into the molecule domain Ω^- with dielectric constant ϵ^- and the solvent domain Ω^+ with dielectric constant ϵ^+, that is, $\Omega = \Omega^- \bigcup \Omega^+$. Denote the boundary of Ω as $\partial\Omega$. Charges in Ω^- are partial charges assigned to the centers of atoms by using a force field, while charges in Ω^+ are mobile ions described by the Boltzmann distribution. For $\mathbf{r} \in \mathbb{R}^3$, applying Gauss's law to the charge distribution in both Ω^- and Ω^+ leads to the nonlinear PB equation

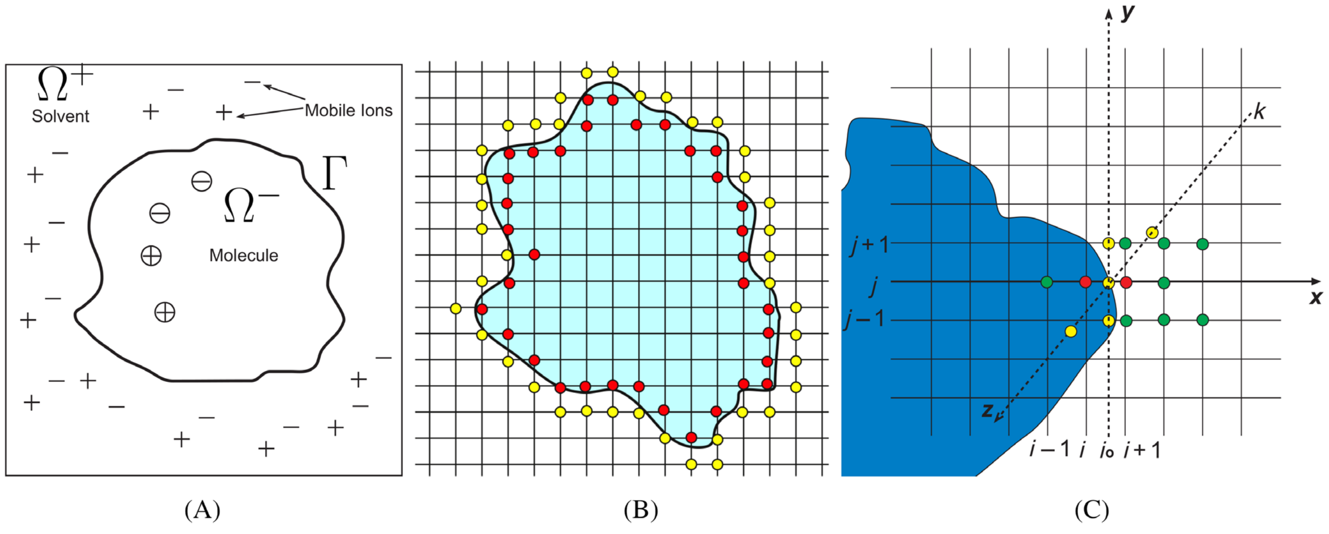

FIGURE 2 (A) The Poisson–Boltzmann (PB) model. (B) The matched interface and boundary (MIB) scheme illustrated in a two-dimensional setting with inside fictitious points in red and outside fictitious points in yellow. (C) The MIB scheme in finding the fictitious values at (i,j,k) and $(i+1,j,k)$ in the cross section of $z = z_k$.

$$-\nabla\cdot(\epsilon(\mathbf{r})\nabla\phi(\mathbf{r}))+\bar{\kappa}^2(\mathbf{r})\sinh(\phi(\mathbf{r}))=\rho(\mathbf{r}), \tag{1}$$

subject to interface continuity for the potential ϕ and flux density $\epsilon\phi_\nu$,

$$[\phi]_\Gamma=0 \text{ and } [\epsilon\phi_\nu]_\Gamma=0. \tag{2}$$

Here $\nu=(\nu_x,\nu_y,\nu_z)$ is the outer normal direction of the interface Γ, $\phi_\nu=\frac{\partial\phi}{\partial\nu}$ is the directional derivative in ν, and the notation $[f]_\Gamma=f^+-f^-$ is the difference of the function f across the interface Γ. Note ϕ is usually small for biomolecular environment, the linear PB equation in which $\sinh(\phi)$ is approximated by ϕ, is often adequately used. This work focus on the linearized PB model.

In Equations (1) and (2), ϵ is a piecewise function for the dielectric constants in Ω^- and Ω^+. Here κ is the inverse Debye screening length measuring ionic strength, and its modified version $\bar{\kappa}$ is given as $\bar{\kappa}^2=\epsilon^+\kappa^2$. The value of κ is nonzero in Ω^+ only. In our numerical implementation, unit for the length is Å and the unit of the potential ϕ is e_c/Å. After computing ϕ, one needs to multiply it with a factor 332.0716 to convert it to the unit of kcal/mol/e_c for free energy calculations. The reader can refer to References 47 and 48 for more details about definition and units of these coefficients.

The source term ρ in Equation (1) is the summation of the charge distribution in Ω^- using Delta function for N_c partial charges located at $\mathbf{r}_i$, that is, $\rho(\mathbf{r})=4\pi C\sum_{i=1}^{N_c}q_i\delta(\mathbf{r}-\mathbf{r}_i)$, where C is a constant to balance the units. Due to the singular partial charges in the source $\rho(\mathbf{r})$, the electrostatic potential $\phi(\mathbf{r})$ goes to infinity as $\mathbf{r}\to\mathbf{r}_i$. This brings considerable difficulties in numerical solution of the PB equation. We will present some regularization approaches to remove the singularity under the MIB framework.[21,23] On the outer boundary $\partial\Omega$, a Dirichlet boundary condition can be assumed[48]

$$\phi(\mathbf{r})=\phi_b(\mathbf{r}):=C\sum_{i=1}^{N_c}\frac{q_ie^{-\kappa|\mathbf{r}-\mathbf{r}_i|}}{\epsilon^+|\mathbf{r}-\mathbf{r}_i|}. \tag{3}$$

It is obvious that $\lim_{|\mathbf{r}|\to\infty}\phi(\mathbf{r})=0$, which is actually the physical boundary condition defined at the infinity. When $\partial\Omega$ is sufficiently distanced from the protein, Equation (3) provides a fairly accurate Debye–Huckel approximation[48] to the solution of the linearized PB equation.

In the next two subsections, we introduce the MIB method which solves the PB model in Cartesian grids using Eulerian surface representation and the TABI method which solves the linear PB model with boundary elements using Lagrangian surface representation.

2.2 | The MIB method

We use the linearized PB equation as in Equation (4) to explain the key ideas of the MIB method for solving the elliptic interface problem with discontinuous coefficients although nonlinear PB equation can be efficiently solved too[23,45]

$$-\nabla\cdot(\epsilon(\mathbf{r})\nabla\phi(\mathbf{r}))+\bar{\kappa}^2(\mathbf{r})\phi(\mathbf{r})=\rho(\mathbf{r}). \tag{4}$$

As shown in Figure 2A, the interface Γ divides the whole domain into two separated parts, Ω^- and Ω^+. The jump conditions across the interface are assumed to be

$$[\phi]_\Gamma=g_0(\mathbf{r}), \tag{5}$$

$$[\epsilon\phi_\nu]_\Gamma=g_1(\mathbf{r}). \tag{6}$$

Note the physical meaning of g_0 and g_1 is the jump in electrostatic potential and flux density across the interface Γ, thus g_0 and g_1 are zeros in the physical background of electrostatic potential ϕ. However, we keep the nonhomogeneous form of g_0 and g_1 here to emphasize the capability of MIB method to treat the nonhomogeneous interface jump conditions.

Consider a uniform Cartesian grid partition of the domain Ω as shown in Figure 2B, it is well known that the standard finite difference schemes lose their designed convergence near the interface and the interface jump conditions have to be used to restore the accuracy. To this end, all the grid points in Ω are classified into two types, the regular ones and the irregular ones. An *irregular* grid point is defined as a node at which the standard finite difference scheme involves grid points across the interface, that is, at least one of its four (2D) or six (3D) neighboring points is from the other side of the interface as illustrated in Figure 2B for a 2D situation. The complement set to the set of irregular grid points defines the set of *regular* grid points. At a regular point, a centered difference discretization of Equation (4) is carried out, which involves a grid node (x_i,y_j,z_k) and its six neighboring points. At each irregular point, there are two values: the true value $\phi(x_i,y_j,z_k)$ and the fictitious value $f(x_i,y_j,z_k)$. The fictitious values can be considered as the extended value from one domain to the other, whose values are obtained by interpolation schemes involving both the differential equation and the interface jump conditions. In the MIB scheme, the finite difference approximations at irregular points will be modified by using fictitious values from the other side of the interface. For example, consider a two-dimensional cross section $z=z_k$ shown in Figure 1C, and denote $\phi_{i,j,k}=\phi(x_i,y_j,z_k)$, we have the following modified finite difference approximation for the x derivative

$$\frac{\partial^2}{\partial x^2}\phi_{i+1,j,k}\approx\frac{1}{\Delta x^2}(f_{i,j,k}-2\phi_{i+1,j,k}+\phi_{i+2,j,k}), \tag{7}$$

where $f_{i,j,k}$ is a fictitious value defined at (x_i,y_j,z_k). The modified finite difference approximations at irregular points maintain the second order of accuracy, provided that the fictitious values are accurately estimated.

In the MIB scheme, by applying the interface jump conditions in Equations (5) and (6), a pair of fictitious values $f_{i,j,k}$ at (x_i,y_j,z_k) and $f_{i+1,j,k}$ at (x_{i+1},y_j,z_k) will be represented as a linear combination of function values on a set of neighboring nodes $\mathbb{S}_{i,j,k}$ and jump data (g_0,g_1). For example,

$$f_{i,j,k}=\sum_{(x_I,y_J,z_K)\in\mathbb{S}_{i,j,k}}w_{I,J,K}\phi_{I,J,K}+w_0g_0+w_1g_1. \tag{8}$$

The major task of a particular MIB approximation is to determine the points set $\mathbb{S}_{i,j,k}$ and the representation weights $w_{I,J,K}$, w_0, and w_1 via discretizing Equations (5) and (6). In order to obtain the fictitious values at the two irregular points (red) located at (i,j,k) and $(i+1,j,k)$, 10 regular values of ϕ at the green and red (overlapped on green) locations in the cross section of $z=z_k$ as illustrated in Figure 1C will be involved. Similarly, in the cross sections $x=x_{i_o}$, six more regular points in Ω^+ or Ω^- are involved to interpolate the derivatives of ϕ^+ or ϕ^- in z direction respectively. Thus in general, we need totally 16 regular values of ϕ. When the interface is more irregular, the 16 points possibly cannot be located, then more complicated interpolation schemes will be needed as detailed in References 49 and 50. Finally, Equation (8) will be substituted into Equation (7) to modify the x derivative approximation. When all necessary x, y, and z derivatives are regularly or irregularly represented using ϕ or f at the grid points, a linear algebraic system of equations in discretizing Equation (4) is generated. The Eulerian representation of ESES can provide the position of (x_{i_o},y_j,z_k) where the mesh line $\{(x,y,z)|y=y_j,z=z_k\}$ intersects the molecular surface and the normal direction of the surface at (x_{i_o},y_j,z_k).

Another issue is the regularization of the source singularity as seen on the right hand side of Equation (4), where $\rho(\mathbf{r})=4\pi C\sum_{i=1}^{N_c}q_i\delta(\mathbf{r}-\mathbf{r}_i)$. Our strategy is to use the Green's function-based decomposition incorporated with the MIB schemes to treat the source singularities while maintain the second-order accuracy. Since this paper focuses on surface and interface treatment, we omit all the details for the treatment of charge singularities, and interested readers can refer to References 21 and 23. All MIBPB results reported in this work are based on the rMIB package developed in Reference 23.

2.3 | The boundary integral PB solver

In this subsection we describe the boundary integral method for computing the electrostatic surface potential and solvation energy.[30,51] We present the boundary integral form of the linearized PB model, the discretization of the boundary integral equations, the treecode algorithm for accelerating the matrix-vector product, and the preconditioning scheme to alleviate the rising condition number when the triangulation quality is deteriorated due to complex geometry. Following the tradition of boundary integral method, we use $\mathbf{x}$ and $\mathbf{y}$ to represent the spacial position as opposed to the previously used $\mathbf{r}$. We also denote $\Omega_1=\Omega^-$ and $\Omega_2=\Omega^+$. Similarly, $\epsilon_1=\epsilon^-$ and $\epsilon_2=\epsilon^+$ will also be used.

2.3.1 | The boundary integral form of the linearized PB model

This section summarizes the well-conditioned boundary integral form of the linear PB model we employ.[25,30] Applying Green's second identity and properties of fundamental solutions to Equation (1) yields the electrostatic potential in each domain,

$$\phi(\mathbf{x})=\int_\Gamma\left[G_0(\mathbf{x},\mathbf{y})\frac{\partial\phi(\mathbf{y})}{\partial\nu}-\frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{y}}\phi(\mathbf{y})\right]dS_\mathbf{y}+\sum_{k=1}^{N_c}q_kG_0(\mathbf{x},\mathbf{y}_k),\quad \mathbf{x}\in\Omega_1, \tag{9a}$$

$$\phi(\mathbf{x})=\int_\Gamma\left[-G_\kappa(\mathbf{x},\mathbf{y})\frac{\partial\phi(\mathbf{y})}{\partial\nu}+\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{y}}\phi(\mathbf{y})\right]dS_\mathbf{y},\quad \mathbf{x}\in\Omega_2, \tag{9b}$$

where $G_0(\mathbf{x},\mathbf{y})$ and $G_\kappa(\mathbf{x},\mathbf{y})$ are the Coulomb and screened Coulomb potentials,

$$G_0(\mathbf{x},\mathbf{y})=\frac{1}{4\pi|\mathbf{x}-\mathbf{y}|},\quad G_\kappa(\mathbf{x},\mathbf{y})=\frac{e^{-\kappa|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|}. \tag{10}$$

Then applying the interface conditions in Equation (2) with the differentiation of electrostatic potential in each domain yields a set of boundary integral equations relating the surface potential ϕ_1 and its normal derivative $\partial\phi_1/\partial\nu$ on Γ,

$$\frac{1}{2}(1+\varepsilon)\phi_1(\mathbf{x})=\int_\Gamma\left[K_1(\mathbf{x},\mathbf{y})\frac{\partial\phi_1(\mathbf{y})}{\partial\nu}+K_2(\mathbf{x},\mathbf{y})\phi_1(\mathbf{y})\right]dS_\mathbf{y}+S_1(\mathbf{x}),\quad \mathbf{x}\in\Gamma, \tag{11a}$$

$$\frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial\phi_1(\mathbf{x})}{\partial\nu}=\int_\Gamma\left[K_3(\mathbf{x},\mathbf{y})\frac{\partial\phi_1(\mathbf{y})}{\partial\nu}+K_4(\mathbf{x},\mathbf{y})\phi_1(\mathbf{y})\right]dS_\mathbf{y}+S_2(\mathbf{x}),\quad \mathbf{x}\in\Gamma, \tag{11b}$$

where $\varepsilon=\varepsilon_2/\varepsilon_1$. As given in Equations (12a), (12b), and (16), the kernels $K_{1,2,3,4}$ and source terms $S_{1,2}$ are linear combinations of G_0, G_k, and their first and second-order normal derivatives.[25,30]

$$\begin{aligned}K_1(\mathbf{x},\mathbf{y})&=G_0(\mathbf{x},\mathbf{y})-G_\kappa(\mathbf{x},\mathbf{y}),\\ K_2(\mathbf{x},\mathbf{y})&=\varepsilon\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{y}}-\frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{y}},\end{aligned} \tag{12a}$$

$$\begin{aligned}K_3(\mathbf{x},\mathbf{y})&=\frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{x}}-\frac{1}{\varepsilon}\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{x}},\\ K_4(\mathbf{x},\mathbf{y})&=\frac{\partial^2G_\kappa(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{x}\partial\nu_\mathbf{y}}-\frac{\partial^2G_0(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{x}\partial\nu_\mathbf{y}},\end{aligned} \tag{12b}$$

where the normal derivative with respect to $\mathbf{x}$ is given by

$$\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{x}}=-\nu(\mathbf{x})\cdot\nabla_\mathbf{x}G(\mathbf{x},\mathbf{y})=-\sum_{m=1}^{3}\nu_m(\mathbf{x})\partial_{x_m}G(\mathbf{x},\mathbf{y}), \tag{13}$$

the normal derivative with respect to $\mathbf{y}$ is given by

$$\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial\nu_\mathbf{y}}=\nu(\mathbf{y})\cdot\nabla_\mathbf{y}G(\mathbf{x},\mathbf{y})=\sum_{n=1}^{3}\nu_n(\mathbf{y})\partial_{y_n}G(\mathbf{x},\mathbf{y}), \tag{14}$$

the second normal derivative with respect to $\mathbf{x}$ and $\mathbf{y}$ is given by

$$\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{y}} \partial \nu_{\mathbf{x}}} = -\sum_{m=1}^{3}\sum_{n=1}^{3} \nu_m(\mathbf{x})\nu_n(\mathbf{y})\partial_{x_m}\partial_{y_n} G(\mathbf{x},\mathbf{y}). \tag{15}$$

and the source terms $S_{1,2}$ are

$$S_1(\mathbf{x}) = \frac{1}{\varepsilon_1}\sum_{k=1}^{N_c} q_k G_0(\mathbf{x},\mathbf{y}_k), \quad S_2(\mathbf{x}) = \frac{1}{\varepsilon_1}\sum_{k=1}^{N_c} q_k \frac{\partial G_0(\mathbf{x},\mathbf{y}_k)}{\partial \nu_{\mathbf{x}}}. \tag{16}$$

Once the potential and normal derivative of the potential from Equations (11a) and (11b) are solved, potential at any point inside the molecule can be computed via Equations (9a) and (9b) or a numerically more accurate formulation as mentioned in Reference 25:

$$\phi_1(\mathbf{x}) = \int_\Gamma \left[K_1(\mathbf{x},\mathbf{y})\frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{x},\mathbf{y})\phi_1(\mathbf{y})\right] dS_{\mathbf{y}} + S_1(\mathbf{x}), \quad \mathbf{x}\in\Omega_1. \tag{17}$$

With potential and its normal derivative on Γ, the electrostatic free energy can be obtained by

$$\begin{aligned} E^{\text{free}} &= \frac{1}{2}\sum_{k=1}^{N_c} q_k \phi_1(\mathbf{y}_k) \\ &= \frac{1}{2}\sum_{k=1}^{N_c} q_k \left(\int_\Gamma \left[K_1(\mathbf{y}_k,\mathbf{y})\frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{y}_k,\mathbf{y})\phi_1(\mathbf{y})\right] dS_{\mathbf{y}} + S_1(\mathbf{y}_k)\right). \end{aligned} \tag{18}$$

The electrostatic solvation energy can also be obtained by

$$\begin{aligned} E^{\text{sol}} &= \frac{1}{2}\sum_{k=1}^{N_c} q_k \phi^{\text{reac}}(\mathbf{y}_k) \\ &= \frac{1}{2}\sum_{k=1}^{N_c} q_k \int_\Gamma \left[K_1(\mathbf{y}_k,\mathbf{y})\frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{y}_k,\mathbf{y})\phi_1(\mathbf{y})\right] dS_{\mathbf{y}}, \end{aligned} \tag{19}$$

where $\phi^{\text{reac}}(\mathbf{x}) = \phi(\mathbf{x}) - S_1(\mathbf{x})$ is the reaction field potential.[25,30] In our numerical results, we report many results involving solving PB equation and calculating the electrostatic solvation energy.

2.3.2 | The discretization of boundary integral equations

The integrals in Equations (11a) and (11b) can be discretized by centroid collocation, which is popular due to its simplicity.[30] Alternatively, it can also be discretized using other more complicated approaches such as node-patch collocation,[52,53] curved triangles,[51] or Galerkin's method[54] with a trade-off of accuracy and efficiency. Here we provide the centroid collocation approach.

Letting $\mathbf{x}_i, i=1,\ldots,N$ denote the triangle centroids of the N triangular elements, the discretized Equations (11a) and (11b) have the following form for $i=1,\ldots,N$,

$$\begin{aligned} \frac{1}{2}(1+\varepsilon)\phi_1(\mathbf{x}_i) &= \sum_{\substack{j=1\\ j\neq i}}^{N}\left[K_1(\mathbf{x}_i,\mathbf{x}_j)\frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_2(\mathbf{x}_i,\mathbf{x}_j)\phi_1(\mathbf{x}_j)\right]\Delta s_j \\ &\quad + S_1(\mathbf{x}_i), \end{aligned} \tag{20a}$$

$$\begin{aligned} \frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial \phi_1(\mathbf{x}_i)}{\partial \nu} &= \sum_{\substack{j=1\\ j\neq i}}^{N}\left[K_3(\mathbf{x}_i,\mathbf{x}_j)\frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_4(\mathbf{x}_i,\mathbf{x}_j)\phi_1(\mathbf{x}_j)\right]\Delta s_j \\ &\quad + S_2(\mathbf{x}_i), \end{aligned} \tag{20b}$$

where Δs_j is the area of the jth boundary element for $j=1,\ldots,N$. The term $j=i$ is omitted to avoid the kernel singularity. Equations (20a) and (20b) give a linear system $Ax=b$, where x contains the surface potentials $\phi_1(\mathbf{x}_i)$ and normal derivatives $\frac{\partial \phi_1(\mathbf{x}_i)}{\partial \nu}$, weighted by the element area Δs_i, and $\mathbf{b}$ contains the source terms $S_1(\mathbf{x}_i), S_2(\mathbf{x}_i)$. The system is solved by the GMRES iterative method, which requires a matrix-vector product in each iteration.[55] Since the matrix is dense, computing the product by direct summation requires $O(N^2)$ operations, which is prohibitively expensive when N is large. These difficulties can be overcome by fast algorithms for N-body interactions such as treecode[30,56] and Fast Multipole Methods.[27,54] In the next section we describe the treecode algorithm used to accelerate the matrix-vector product.

2.3.3 | Treecode

We summarize the treecode algorithm and refer to previous work for more details.[35,57-59] The matrix-vector product Ax for Equations (20a) and (20b) has the form of N-body potentials,

$$V_i = \sum_{j=1, j\neq i}^{N} q_j G(\mathbf{x}_i,\mathbf{x}_j), \quad i=1,\ldots,N, \tag{21}$$

where G is a kernel, $\mathbf{x}_i, \mathbf{x}_j$ are centroids (also called particle locations in this context), and q_j is a charge associated with $\mathbf{x}_j$. To this end, the q_j in Equation (21) is equivalent to the $\Delta s_j \phi_1(\mathbf{x}_j)$ or $\Delta s_j \frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu}$ in Equations (20a) and (20b) and G is one of the kernels of K_{1-4}. To evaluate the potentials V_i rapidly, the particles $\mathbf{x}_i$ are divided into a hierarchy of clusters having a tree structure in a 2D illustration as in Figure 3A. The root cluster is a cube containing all the particles and subsequent levels are obtained by dividing a parent cluster into children.[57] The process continues until a cluster has fewer than N_0 particles (N_0 is a user-specified parameter representing the maximum number of particles per leaf, e.g., $N_0=3$ in Figure 3A). Then V_i is evaluated as a sum of particle–particle interactions and particle–cluster interactions, which are depicted in Figure 3B,

$$V_i \approx \sum_{c\,\in N_i}\sum_{\mathbf{x}_j\in c} q_j G(\mathbf{x}_i,\mathbf{x}_j) + \sum_{c\,\in F_i}\sum_{||\mathbf{k}||=0}^{p} a^{\mathbf{k}}(\mathbf{x}_i,\mathbf{x}_c)\, m_c^{\mathbf{k}}, \tag{22}$$

where c denotes a cluster, and N_i, F_i denote the near-field and far-field clusters of particle $\mathbf{x}_i$. The first term on the right is a direct sum for particles $\mathbf{x}_j$ near $\mathbf{x}_i$, and the second term is a pth-order Cartesian Taylor approximation about the cluster center $\mathbf{x}_c$ for clusters that are well-separated from $\mathbf{x}_i$.[35] Cartesian multi-index notation is used with $\mathbf{k}=(k_1,k_2,k_3), k_i\in\mathbb{N}, ||\mathbf{k}||=k_1+k_2+k_3, \mathbf{k}!=k_1!k_2!k_3!$. A particle $\mathbf{x}_i$ and

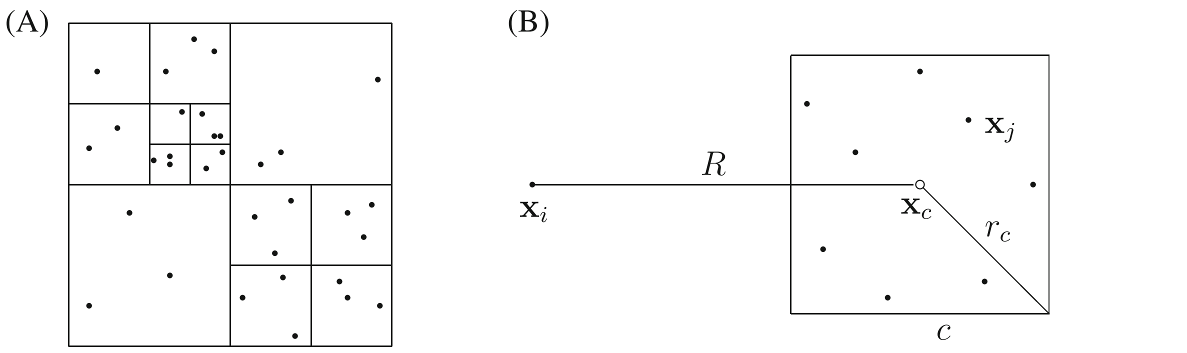

FIGURE 3 Details of treecode. (A) Tree structure of particle clusters. (B) particle–cluster interaction between particle $\mathbf{x}_i$ and cluster $c = \{\mathbf{x}_j\}$. $\mathbf{x}_c$, cluster center; R, particle-cluster distance; r_c, cluster radius.

a cluster c are defined to be well-separated if the multipole acceptance criterion (MAC) is satisfied, $r_c/R \le \theta$, where r_c is the cluster radius, $R = |\mathbf{x}_i - \mathbf{x}_c|$ is the particle–cluster distance, and θ is a user-specified parameter.[57]

The accuracy and efficiency of the treecode is controlled by the combination of order p, MAC parameter θ, and maximum particles per leaf N_0. Using the treecode, the operation count for the matrix-vector product is $O(N\log N)$, the factor N is the number of particles $\mathbf{x}_i$, and the factor $\log N$ is the number of levels in the tree.

2.3.4 | Preconditioning

In order to precondition Krylov subspace methods in solving $Ax = b$, taking the left-precondition scheme as an example, we seek a preconditioning matrix or a preconditioner M such that two conditions are satisfied:

(1) M is similar to A such that $M^{-1}A$ has improved condition compared to A thus less number of iterations are required in solving $M^{-1}Ax = M^{-1}b$ compared to solving $Ax = b$;
(2) $M^{-1}z = y$ can be efficiently computed, which is equivalent to solving y from $My = z$.

Conditions (1) and (2) cannot be improved concurrently and a tradeoff must be made.

The design of our preconditioner is motivated from the observation that in electrostatic interactions, which is also the interactions between boundary elements in solving integral equations, the short range interactions are smaller in number of interactions but more significant in strength than the long range interactions, which are large in number of interactions and computationally more expensive. Due to their large numbers, the long interactions are calculated by multipole expansions. This gives us the ideas that for a preconditioner of A, we might use short range interactions represented by the matrix M to approximate all interactions, ignoring long range interactions. Our picked short range interactions are between elements on the same leaf only. This choice has great advantages in efficiency and accuracy for solving $My = z$.

The most important advantage of this designed preconditioning matrix M is that computing $y = M^{-1}z$ or equivalently solving y from $My = z$ can be rapidly computed. As seen with details in Reference 60, by using a permutation operation, the M matrix is a block diagonal matrix such that $My = z$ can be solved using direct method, for example, LU factorization by solving each individual $M_i y_i = z_i$. Here each M_i is a square nonsingular matrix, which represents the interaction between particles/elements on the ith leaf of the tree. It is worthy to note that the efficiency is not affected even when M_i has a large condition number since direct solver is used for solving $My = z$.

2.4 | ESES for conjugated Eulerian and Lagrangian surfaces

The ESES is a software developed by Liu et al. in 2017[44] for the construction of SESs in the Eulerian representation. ESES can robustly generate meshes readily available for being used in solving the PB or PNP type of equations in the Cartesian grid. The surface information in ESES is provided in terms of intersection points between the grid lines and the interface, that is, the data for the coordinates of each intersection point and the surface normal at the point. Additionally, each grid point can be labeled as either inside or outside the interface. This information is useful for initializing the PB equation and for the enforcement of interface conditions. The analytical nature of intersection point evaluation of the present ESES enables high order PB and PNP solvers to achieve their designed order of convergence or accuracy in the L_1 norm, which is essential in evaluating the performance of these elliptic interface methods for complex biomolecular surfaces. The analytical representation of potential SES patches is first computed based on Connolly's work[37] where accurate convex patches, saddle patches and concave patches are generated and stored for fast access. Each Cartesian grid point is then classified either as inside the SES or outside the SES based on its relation with respect to these three types of patches in its neighborhood. For each Cartesian edge with one Cartesian end point inside the SES and the other outside, its analytical intersection position and corresponding normal direction with respect to the outermost surface patch is computed. There are three types of patches generated in the SES, namely, convex patches, saddle patches, and concave patches. These convex, saddle, or concave patches are determined by singletons, pairs, or triplets of atoms, respectively.

After the completeness of the Cartesian representation of the SES, the intersection of the grid points and the SES can be connected to form a triangulated surface by using the marching cubes algorithm.[61] The initial goal for the triangulation is for visualization. It turned out that this triangulation renders a conjugated Lagrangian representation of the SES, which can be used to solve the boundary integral PB equations. There are concerns that the triangles formed by ESES are not "equilateral-alike" and thus might affect the convergence of GMRES method for solving the discretized integral equations. Fortunately, as we show below, the preconditioning schemes as described above with details in[60] can resolve the issue, making the ESES Lagrangian representation another useful option as a body-fitted ESES surface.

2.5 | The source–target symmetric properties

In this section we restate an interesting symmetric property of linearized PB model for the electrostatic reaction field potential induced by a unit point charge $e_A = 1$ inside a cavity, which is immersed in solvent. The property is illustrated in Figure 4 such that the reaction field potential at location B induced by a unit point charge at location A is the same as the potential at location A induced by that point charge $e_B = 1$ at location B, that is,

$$\phi_A^{\text{reac}}(\mathbf{r}_B) = \phi_B^{\text{reac}}(\mathbf{r}_A). \tag{23}$$

2.5.1 | Derivation

This symmetric property can be derived by following the procedure as described in Reference 46, we rephrase it in a simplified way for reader's convenient access.

We try to determine the work W_A to bring a charge e_A from infinity to its position $\mathbf{r}_A$. Due to the presence of dielectric (with ϵ_2) outside of the cavity, the determination of work W_A is nontrivial. The dielectric is polarized by the point charge e_A, and this polarization leads to a potential ϕ_A^{reac} at $\mathbf{r}_A$ whose gradient is the reaction field generated by the charge e_A. As described in Reference 46, W_A can be derived from a charging process which increases the point charge at $\mathbf{r}_A$ with infinitesimally small steps from zero to its actual value e_A. At a certain stage of this process the charge will have reached the value λe_A where $\lambda \in [0,1]$. At that stage, the potential at $\mathbf{r}_A$ due to the dielectric is $\lambda \phi_A^{\text{reac}}$.

The energy to bring a charge element $(\delta\lambda)e_A$ at $\mathbf{r}_A$ against the reaction field of the charge λe_A, is then given by,

$$\delta W_A = \lambda \phi_A^{\text{reac}}(\mathbf{r}_A) \cdot (\delta\lambda) e_A. \tag{24}$$

Therefore the energy W_A is given by the integration of λ from 0 to 1,

$$W_A = \int_0^1 \lambda \phi_A^{\text{reac}}(\mathbf{r}_A) e_A \delta\lambda = \frac{1}{2} e_A \phi_A^{\text{reac}}(\mathbf{r}_A). \tag{25}$$

At the next step when another charge e_B is brought to $\mathbf{r}_B$, the corresponding energy consists of three parts:

- $e_B \phi_A^{\text{coul}}(\mathbf{r}_B)$: the energy to move e_B against the field due to the charge e_A. This part is generated by the Coulombic interaction between the charges e_B and e_A.
- $e_B \phi_A^{\text{reac}}(\mathbf{r}_B)$: the energy to move e_B against the reaction field at $\mathbf{r}_B$ due to the charge e_A.
- $\frac{1}{2} e_B \phi_B^{\text{reac}}(\mathbf{r}_B)$: the energy to move e_B against its own reaction field, which can be similarly derived as in Equation (25).

Therefore the work to bring the charge e_B to its location $\mathbf{r}_B$ would be,

$$W_B = e_B \phi_A^{\text{coul}}(\mathbf{r}_B) + e_B \phi_A^{\text{reac}}(\mathbf{r}_B) + \frac{1}{2} e_B \phi_B^{\text{reac}}(\mathbf{r}_B). \tag{26}$$

Altogether the total energy W to bring two charges e_A and e_B into the cavity becomes,

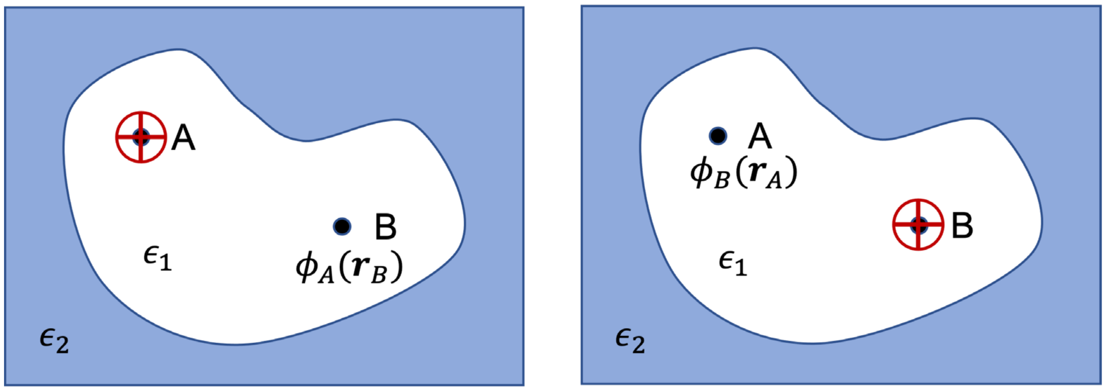

FIGURE 4 The symmetric property inside a solvated cavity.

$$W=W_A+W_B=\frac{1}{2}e_A\phi_A^{\text{reac}}(\mathbf{r}_A)+e_B\phi_A^{\text{coul}}(\mathbf{r}_B)+e_B\phi_A^{\text{reac}}(\mathbf{r}_B)+\frac{1}{2}e_B\phi_B^{\text{reac}}(\mathbf{r}_B). \tag{27}$$

Similarly, the energy necessary to bring these two charges into the cavity at the same locations but in the reverse sequence is,

$$W=\frac{1}{2}e_B\phi_B^{\text{reac}}(\mathbf{r}_B)+e_A\phi_B^{\text{coul}}(\mathbf{r}_A)+e_A\phi_B^{\text{reac}}(\mathbf{r}_A)+\frac{1}{2}e_A\phi_A^{\text{reac}}(\mathbf{r}_A). \tag{28}$$

Since the energy does not depend on the sequence in which the charges are brought into the cavity, we can equate Equations (27) and (28) to find,

$$e_B\phi_A^{\text{coul}}(\mathbf{r}_B)+e_B\phi_A^{\text{reac}}(\mathbf{r}_B)=e_A\phi_B^{\text{coul}}(\mathbf{r}_A)+e_A\phi_B^{\text{reac}}(\mathbf{r}_A). \tag{29}$$

The Coulomb part in Equation (29) is proven to be equal in section 12 of Reference 46 as,

$$e_B\phi_A^{\text{coul}}(\mathbf{r}_B)=e_A\phi_B^{\text{coul}}(\mathbf{r}_A), \tag{30}$$

which results in the equality,

$$e_B\phi_A^{\text{reac}}(\mathbf{r}_B)=e_A\phi_B^{\text{reac}}(\mathbf{r}_A). \tag{31}$$

Hence, the energy of a point charge in the reaction field of another charge is equal to the energy of the second charge in the reaction field of the first one. For a particular case when the the point charges $e_A=e_B=1$ we have Equation (23). Similarly, when there are mobile ions appeared in solvent described by the Boltzmann distribution, the same equality could be derived for a linearized PB model.

2.5.2 | Application

This symmetric property is physically sound and can have potential applications in assisting the algorithm development and numerical analysis. Here we elaborate one of these applications by connecting to the boundary integral PB solver TABI in the previous section. This application is inspired by work from Juffer et al.[62] and has potential application in computing protein pKa.[5,62,63]

Consider the discretized forms of Equations (20a) and (20b) for governing potential and its normal derivatives as

$$Sx=Bq, \tag{32}$$

where $x\in\mathbb{R}^{2N}$ is the vector of potential and its normal derivative at the collocation location, $q\in\mathbb{R}^{N_c}$ is the vector of charges at atomic centers, $S\in\mathbb{R}^{2N\times 2N}$ and $B\in\mathbb{R}^{2N\times N_c}$ are the corresponding matrices to make Equations (20a) and (20b) valid. Similarly, the discretized form of Equation (17) for $\phi\in\mathbb{R}^{N_c}$, $R\in\mathbb{R}^{N_c\times 2N}$, and $D\in\mathbb{R}^{N_c\times N_c}$ is

$$\phi=\phi_{\text{reac}}+\phi_{\text{coul}}=Rx+Dq, \tag{33}$$

and that of Equations (18) and (19) is

$$E_{\text{free}}=\frac{1}{2}q^T\phi=\frac{1}{2}q^T(RS^{-1}B+D)q=\frac{1}{2}q^TWq, \tag{34}$$

and

$$E_{\text{sol}}=\frac{1}{2}q^T\phi_{\text{reac}}=\frac{1}{2}q^TRS^{-1}Bq=\frac{1}{2}q^TAq, \tag{35}$$

where $A=RS^{-1}B$ is symmetric in the sense that

$$A_{ij}=e_i^TAe_j=e_j^TAe_i=A_{ji}, \tag{36}$$

where e_i is the unit vector with ith entry nonzero. Note the symmetry of A is not straightforward to prove, however the middle equality in Equation (36) is valid since it implies exactly the symmetric property we have stated. i.e., the equivalence of the reaction field potential energy for unit charge at r_i under the potential induced by a unit charge at r_j and the reaction field potential energy for unit charge at r_j under the potential induced by a unit charge at r_i. With the symmetry of A, which indicates the symmetry of W since D is symmetric too, we can calculate the site–site interaction in Reference 5 as

$$\triangle G_{ij}=t_i^TWt_j=\frac{1}{2}(t_i+t_j)^TW(t_i+t_j)-\frac{1}{2}t_i^TWt_i-\frac{1}{2}t_j^TWt_j. \tag{37}$$

reducing from solving N_t^2 systems to $N_t(N_t-1)/2+N_t$ systems, where N_t is the number of titration sites whose proton will change under different pH environment. Note $t_i\in\mathbb{R}^{N_c}$ is the charge vector with all entries zero except q_i at the ith position.

3 | RESULTS AND DISCUSSIONS

In this section we will numerically demonstrate that ESES can generate high fidelity SESs with both Eulerian and Lagrangian representations. Following that, we use ESES together with TABI and MIB PB solvers to verify the symmetric properties as derived in the section of theory and algorithms. All calculations use solute dielectric constant $\epsilon_1=1$ and solvent dielectric constant $\epsilon_2=80$, and we set the ionic concentration I_s as zero by focusing only on the Poisson's equation or Gauss's law. Similar results can be achieved for nonzero ionic concentrations. For test cases on spherical cavities with analytical solution, the tolerance for bi-conjugate gradient solver used by MIB solver and the tolerance for GMRES used by TABI and HOBI solver are all set as 10^{-10}. Other than that, the tolerance for MIB solver is set as 10^{-6} and the tolerance for TABI solver is set as 10^{-4} as the default values. In TABI solver,[30] the treecode parameters are set as (Maximum Acceptance Criterion) MAC parameter $\theta=0.8$, Taylor expansion order $p=3$, and maximum number of particles per leaf $N_0=100$.

All code used are run in serial. When large problems are involved, parallel computing can be partially implemented as specified below.

The ESES code[44] (https://github.com/WeilabMSU/ESES) can be run with OpenMP protocol. The HOBI code[51] (https://sourceforge.net/projects/hobipb/) can be implemented with MPI. The C++ TABI code[53] (https://github.com/Treecodes/TABI-PB) can be implemented with MPI and GPU in a hybrid parallelization. The Fortran TABI code[30] (https://github.com/gengwh/TABI-PB) can be implemented in MPI (in preparation[64]) utilizing the parallel algorithm developed recently.[65] The parallelization of MIB solver is within our future research plans.

3.1 | ESES performance

In this paper, we will only lightly touch ESES's Eulerian representation.[44] Our main focus is on its Lagrangian representation. These results are computed using a 13inch MacBook Pro with intel core-i5 processor and 16 GB of RAM. We first discuss the associated triangulation quality.

3.1.1 | Triangulation quality

With ESES, the triangles forming the Lagrangian representation of the SES are generated by connecting the vertices using marching cubic algorithms.[61] These vertices are intersections between surface and the mesh lines. In the earlier work,[44] these triangles are only used to visualize the molecular surface. We here extend their usage being applied by the boundary integral PB solver. These triangles have good fidelity to the molecular surface. But the *triangulation quality* as termed from the finite element or boundary element community is crucial for numerical simulations. In particular, triangles that are *quasi-uniform* (essentially the same size) and similar to equilateral triangles are preferred. As shown in Figure 5, we compare the triangulation quality of the triangulated SES on protein 1AJJ generated by ESES[44] and NanoShaper.[39] When similar number of triangles are generated by ESES and NanoShaper, we can see the distribution of areas of triangles from NanoShaper is more centered around 0.1Å^2 than that from ESES while the distribution of maximum angles of triangles from NanoShaper is closer to 60° than that from ESES.

Fortunately, our work on preconditioning scheme[60] as briefly explained in the theory and algorithm section previously can alleviate this pain and guarantees TABI-PB solver's rapid convergence when ESES triangulation is used. This will be shown with examples in the next subsection.

3.1.2 | TABI using Lagrangian ESES versus TABI using NanoShaper

There are extensive studies on the Eulerian SES representation from ESES as opposed to the Lagrangian representation from MSMS[38] in Reference 44. We provide an example below for solving the PB model on triangulated SES generated by ESES and NanoShaper[39] using the TABI solver as shown in Table 1. When we refine the mesh, we adjust the parameters in ESES (Column 1) and NanoShaper (Column 2) to generate similar number of triangles N as seen in Columns 4 and 5. Note at the same time, ESES has the number of Cartesian grid points as shown in Column 3. From this example, we can see as the mesh is refined the electrostatic solvation energy computed by TABI solver using ESES in Column 6 is approaching to that using NanoShaper in Column 7, resulting in smaller difference ΔE_{sol} as seen in Column 8. The total CPU time is similar when different surfaces are used as shown in Columns 9 and 10. This owes to the preconditioning scheme,[60] similar number of iterations are used when different surfaces from ESES and NanoShaper are used. We had more tests on some other proteins and the results showed the similar pattern thus we only report this example here. It shows that when its Lagrangian surface representation is used for boundary integral PB solver, ESES works as well as popular surface generators such as NanoShaper and MSMS.

3.1.3 | MIB using Eulerian ESES versus MIB using Eulerian MSMS

Following by the validation of Lagrangian representation of ESES using TABI solver, we next move to the validation of Eulerian

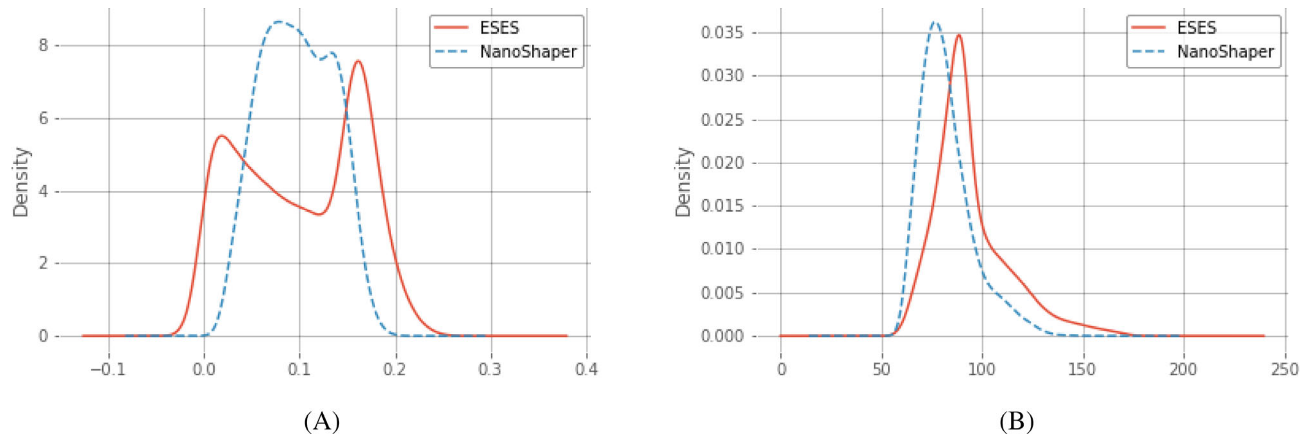

FIGURE 5 Triangulation quality comparison between Eulerian solvent excluded surface (ESES) and NanoShaper using SES of protein 1AJJ: (A) distribution in terms of kernel density estimate (KDE) of triangles' areas in Å^2; (B) distribution of triangles' maximum angles in degrees.

representation of ESES using MIB solver. Just like we need the well-established NanoShaper for comparison, here we use MSMS instead.[38] Note MSMS and NanoShaper had a direct generation of a Lagrangian SES, and Zhou in his thesis[43] developed a Fortran wrapper of MSMS to convert the Lagrangian representation to the Eulerian representation, which is the reason why we use MSMS here.

We calculate the solvation energies for three different molecules: one spherical cavity of radius 2Å with a centered unit charge, and proteins 2pde and 1aho. For the spherical cavity, we reported the result in Table 2. From this table, we can see when MSMS surface with a fixed density is used, the solvation energy converges to the case using the finest mesh (from top to bottom with h changes from 1 to 0.125). When ESES surface is used, the same pattern can be observed. The most significant pattern from this table is that MSMS results with refined density converges to ESES's with sufficiently fine mesh ($h = 0.125$).

For the two proteins, we report the results in Figure 6. Depending on the size of the protein, we use MSMS density 20, 30, 100 for protein 1PDE, and use MSMS density 10, 20, 30 for protein 1AHO. Results in the figure show that increasing MSMS density yields a solvation energy approaching that of ESES.

3.1.4 | MIB using Eulerian ESES versus TABI using Lagrangian ESES

We next provide an example showing the connection between the Eulerian representation and Lagrangian representation of ESES. These two representations are used by solving the PB model using finite difference methods with interface treatment, that is, the MIB solver[23,45] and using the boundary integral method that is, the TABI solver.[30] The results on Protein 1AJJ are shown in Table 3. Here we have no control about the number of the triangles with the Lagrangian representation. Once the mesh size h in Column 1 is determined for the Eulerian representation, the number of grid points for MIB in Column 2 and the number of triangles in Column 3 are determined as well. We can see the solvation energy from MIB in Column 4 and that from TABI in Column 5 are approaching to each other as mesh refined. From the CPU time in Columns 7 and 8, the TABI solver is slower at the very beginning but shows advantage in convergence speed when the mesh is refined due to the $O(N\log N)$ treecode acceleration.

3.2 | Validation of the symmetric properties

After the tests of ESES with both Eulerian and Lagrangian representations, we further validate the source–target symmetric property described previously for different types of molecular structures. In the subsections below, we consider two hypothetical molecules with simple geometry and a protein 1AJJ. All of these molecules have two locations $\mathbf{r}_A$ and $\mathbf{r}_B$ inside the molecular surface as the interchangeable source and target locations. We calculated the reaction field potential $\phi_A^R(\mathbf{r}_B)$ at $\mathbf{r}_B$ induced by a unit charge at $\mathbf{r}_A$ and the reaction field potential $\phi_B^R(\mathbf{r}_A)$ at $\mathbf{r}_A$ induced by a unit charge at $\mathbf{r}_B$. These two quantities are theoretically equal as derived previously. Our numerical methods validate this source–target symmetric property while the advantage and limitation of the methods can also be seen from these

TABLE 1 Computing electrostatic solvation energy of a protein (PDB 1AJJ) with TABI solver using two molecular surface triangulation software Eulerian solvent excluded surface (ESES) and NanoShaper (NS): Densities are used as a parameter to control the number of triangular faces N; Dimension (for ESES only) is the number of grid points in each direction; ΔE_{sol} is the difference of solvation energy calculated using TABI solver with ESES and NanoShaper surface generators.

Densities		Dimension	N		E_{sol} (kcal/mol)			CPU time (s)	
ESES	NS	ESES	ESES	NS	ESES	NS	ΔE	ESES	NS
1.12	0.90	28*30*33	5084	5096	−1222.20	−1261.92	39.72	2.18	2.14
0.80	1.26	39*42*46	10124	10168	−1180.16	−1188.74	8.58	5.70	5.55
0.56	1.80	56*60*65	20640	20884	−1157.24	−1160.53	3.29	15.16	15.17
0.40	2.50	78*84*91	40752	40772	−1146.93	−1148.53	1.60	34.89	34.05
0.28	3.58	111*119*130	83232	83640	−1141.76	−1142.54	0.77	76.33	76.84
0.20	5.00	155*167*182	163276	163292	−1138.94	−1139.26	0.32	196.14	193.70

h	MSMS (den = 10)	MSMS (den = 20)	MSMS (den = 100)	ESES
1	−82.48219907	−82.27048894	−82.16175924	−79.58699949
0.5	−82.52123400	−82.32504614	−82.19843970	−82.01643791
0.25	−82.53823124	−82.33111065	−82.21280699	−82.28426183
0.125	−82.53534892	−82.34891499	−82.21596355	−82.26497101

TABLE 2 Computing electrostatic solvation energy of a spherical cavity with a centered charge using MIB solver on the Lagrangian representation from Eulerian solvent excluded surface (ESES) and from MSMS.

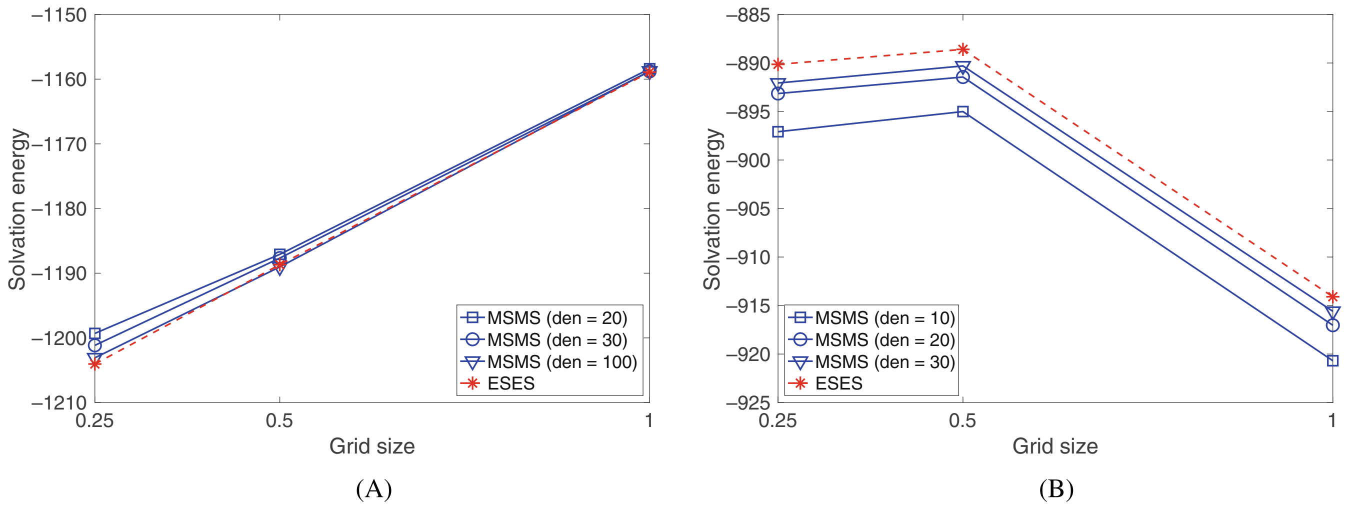

FIGURE 6 Computing electrostatic solvation energy for the proteins 2PDE (left) and 1AHO (right) using matched interface and boundary solver on the Lagrangian representation from Eulerian solvent excluded surface and from MSMS.

TABLE 3 Computing electrostatic solvation energy of protein 1AJJ using TABI solver[30] on the Lagrangian representation and using matched interface and boundary (MIB) solver[23] on the Eulerian representation of the molecular surface, both generated by Eulerian solvent excluded surface (ESES); zero ionic strength.

			E_{sol} (kcal/mol)			CPU time (s)	
h	Dimension	*N*	MIB	TABI	Δ*E*	MIB	TABI
1	31*34*37	6464	−1137.35	−1208.98	71.63	0.95	2.97
0.5	62*67*73	25,956	−1138.04	−1153.46	15.42	8.80	20.45
0.25	124*134*146	104,412	−1139.25	−1140.60	1.35	107.04	116.53
0.125	248*267*292	418,544	−1139.37	−1137.48	−1.89	1312.65	522.24

test cases. We focus on the Poisson model ($\kappa = 0$) while the linearized Poisson-Boltzannn model ($\kappa \neq 0$) can be verified in a similar manner. We mainly used the MIB solver but in one case the high order boundary integral (HOBI) PB solver[51] is used for the pursuit of higher accuracy. These results are computed using a 15inch MacBook Pro with intel core-i7 processor and 16 GB of RAM.

3.2.1 | A spherical cavity

This example considers a spherical atom centered at the origin $\mathbf{r_0} = (0,0,0)$ and with radius 2Å. Our source and target locations are at $\mathbf{r}_A = (1,0,0)$ and $\mathbf{r}_B = (1,1,0)$. We have used the MIB PB solver[23] to calculate the reaction field potentials $\phi^R_A(\mathbf{r}_B)$ and $\phi^R_B(\mathbf{r}_A)$ as reported in Table 4. Meanwhile, we also applied the HOBI PB solver for comparison as reported in Table 5. The exact reaction field potential equals to −207.124201675656 kcal/mol/e_c, which can be found using the scheme described in References 47 and 66. With this reference value, we can calculate the errors and orders as shown in both Tables 4 and 5. Here the reported errors are absolute errors compared with exact values.

Form Table 4, we can see that $\phi^R_A(\mathbf{r}_B)$, $\phi^R_B(\mathbf{r}_A)$, and $|\phi^R_A(\mathbf{r}_B) - \phi^R_B(\mathbf{r}_A)|$ all converge approximately in second order. This can be further verified by result shown in Table 5 for the same case solved by the high-order boundary integral PB solver.[51] In Table 5, the number of the triangles is quadrupled each step, thus the length is halved. We observed second order convergence pattern in average.

3.2.2 | Molecule with two overlapping spherical atoms

In this test case, we consider a molecule of two atoms. In particular we have the radius of the sphere A as $R_A = 3.0001$ with its center on the x-axis at $\mathbf{r}_A = (-2,0,0)$ and the radius of the sphere B as $R_B = 2.0001$ with its center on the x-axis at $\mathbf{r}_B = (2,0,0)$. The reaction field potential ϕ^R_A is induced by the charge $e_A = 1$ located at $\mathbf{r}_A$ and the reaction field potential ϕ^R_B is induced by the charge $e_B = 1$ located at $\mathbf{r}_B$. Table 6 shows the reaction field potentials for different grid sizes from the MIB PB solver. Since the actual potentials are not available, errors of the reaction field potentials for different grid sizes are calculated by comparing with the reaction field potential for $h = 0.125$, the finest grid. From this case, we can see approximately second order convergence for $\phi^R_A(\mathbf{r}_B)$, $\phi^R_B(\mathbf{r}_A)$ and $|\phi^R_A(\mathbf{r}_B) - \phi^R_B(\mathbf{r}_A)|$.

3.2.3 | Protein 1AJJ

Finally in validating the symmetric properties we considered the protein 1AJJ whose molecular surface is generated with ESES.

TABLE 4 Reaction field potential for two locations inside a spherical atom located at $\mathbf{r}_A=(1,0,0)$ and $\mathbf{r}_B=(1,1,0)$ by the MIB solver with $\kappa=0$.

h	$\phi_A^R(\mathbf{r}_B)$	Error	Order	$\phi_B^R(\mathbf{r}_A)$	Error	Order	$\lvert\phi_A^R(\mathbf{r}_B)-\phi_B^R(\mathbf{r}_A)\rvert$	Order
1	−206.9180	2.07e−1		−207.3678	2.43e−1		4.50e−1	
1/2	−207.1797	5.51e−2	1.91	−207.2954	1.71e−1	0.51	1.16e−1	1.96
1/4	−207.0967	2.79e−2	0.98	−207.1909	6.63e−2	1.37	9.42e−2	0.30
1/8	−207.1201	4.48e−3	2.64	−207.1425	1.79e−2	1.89	2.24e−2	2.07
1/16	−207.1229	1.70e−3	1.40	−207.1290	4.40e−3	2.02	6.10e−3	1.88

Notes: The potential ϕ_A^R is induced by the charge $e_A=1$ located at $\mathbf{r}_A$ and the potential ϕ_B^R is induced by the charge $e_B=1$ located at $\mathbf{r}_B$. These reaction field potentials are compared to the actual Kirkwood potential $=-207.124201675656$ kcal/mol/e_c, where e_c is the fundamental charge of an electron, to find the errors and the convergence orders of errors; h represents the grid size.

TABLE 5 The same case as in Table 4 solved by the HOBI solver; N represents the number of triangular elements.

N	$\phi_A^R(\mathbf{r}_B)$	Error	Order	$\phi_B^R(\mathbf{r}_A)$	Error	Order	$\lvert\phi_A^R(\mathbf{r}_B)-\phi_B^R(\mathbf{r}_A)\rvert$	Order
80	−202.7689	4.36		−205.2846	1.84		2.52	
320	−206.3661	7.58e−1	2.52	−207.1341	9.87e−3	7.54	7.68e−1	1.71
1280	−207.0331	9.11e−2	3.06	−207.1869	6.27e−2	−2.67	1.54e−1	2.32
5120	−207.1192	5.02e−3	4.18	−207.1386	1.44e−2	2.12	1.94e−2	2.99
20,480	−207.1263	2.09e−3	1.26	−207.1268	2.62e−3	2.46	5.24e−4	5.21
81,920	−207.1254	1.19e−3	0.81	−207.1246	4.13e−4	2.66	7.79e−4	−0.57

TABLE 6 Reaction field potential for two locations inside a hypothetical molecule with two overlapping spherical atoms by MIB Solver; $R_A=3.0001$ with its center on the x-axis at $\mathbf{r}_A=(-2,0,0)$ and the radius of the sphere B as $R_B=2.0001$ with its center the x-axis at $\mathbf{r}_B=(2,0,0)$.

h	$\phi_A^R(\mathbf{r}_B)$	Error	Order	$\phi_B^R(\mathbf{r}_A)$	Error	Order	$\lvert\phi_A^R(\mathbf{r}_B)-\phi_B^R(\mathbf{r}_A)\rvert$	Order
1	6.92374	3.40e−1		6.7924	2.10e−1		1.31e−1	
0.5	6.66279	7.86e−2	2.11	6.6370	5.48e−2	1.94	2.58e−2	2.35
0.25	6.59671	1.25e−2	2.65	6.5881	5.80e−3	3.24	8.64e−3	1.58
0.125	6.58417	0		6.5823	0		1.89e−3	2.19

We report the result using the Eulerian representation thus the PB equation was solved using the Cartesian grid-based MIB method. In this simulation, we first removed all the atomic charges and then put one unit charge at one of the two locations $\mathbf{r}_A=(-0.169,7.698,13.415)$ or $\mathbf{r}_B=(4.942,6.666,12.698)$ such that $|\mathbf{r}_A-\mathbf{r}_B|=5$Å, and the location $\mathbf{r}_A$ or $\mathbf{r}_B$ will be the location we can calculate the induced reaction field potential $\phi_A^R(\mathbf{r}_B)$ or $\phi_B^R(\mathbf{r}_A)$. Table 7 shows the reaction field potentials, errors, and orders for different grid sizes. Since the actual potentials are not available, errors of the reaction field potentials for different grid sizes are calculated by comparing with the reaction field potential for $h=0.125$, the finest grid. From this case, we can see approximately second-order convergence for $\phi_A^R(\mathbf{r}_B)$, $\phi_B^R(\mathbf{r}_A)$. The result of $|\phi_A^R(\mathbf{r}_B)-\phi_B^R(\mathbf{r}_A)|$ is saturated at the finest grid $h=0.125$, which could be explained by the reason that the boundary condition is given as $1/|\mathbf{r}-\mathbf{r}_A|$ or $1/|\mathbf{r}-\mathbf{r}_B|$ on the boundary. This is the unavoidable numerical artifact for finite difference PB solvers.

4 | SOFTWARE DISSEMINATION

All the algorithms ESES, MIB, TABI, and HOBI as mentioned in this paper has their corresponding software available for the potential users. The following provide the software dissemination information.

(1) The ESES source code is available to download from the Github link: https://github.com/WeilabMSU/ESES, which is currently maintained by Dr. Wei's group at Michigan State University.
(2) The MIBPB package is maintained by Dr. Geng's group at Southern Methodist University accessible from the link https://github.com/gengwh/rMIB-PB. Note this package was originated at Dr. Wei's group and collaboratively developed with Dr. Zhao's group at University of Alabama.
(3) The TABI solver is originated from Dr. Robert Krasny's group at University of Michigan. It has its C++ version maintained by Dr. Wilson with Github link https://github.com/Treecodes/TABI-PB

TABLE 7 Reaction field potential for two point charges inside the molecular surface of protein 1AJJ by matched interface and boundary (MIB) solver.

h	$\phi_A^R(\mathbf{r}_B)$	Error	Order	$\phi_B^R(\mathbf{r}_A)$	Error	Order	$\lvert\phi_A^R(\mathbf{r}_B)-\phi_B^R(\mathbf{r}_A)\rvert$	order
1	3.3169	2.18e−1	–	3.2052	1.10e−1	–	1.12e−1	–
0.5	3.1268	2.75e−2	2.98	3.1135	1.83e−2	2.58	1.33e−2	3.07
0.25	3.1026	3.32e−3	3.05	3.1000	4.82e−3	1.93	2.58e−3	2.37
0.125	3.0993	–	–	3.0952	–	–	4.08e−3	−0.66

Note: This result is produced on a High Performance Computer in Alabama Super Computer system (https://www.asc.edu).

and Fortran version with Github link https://github.com/gengwh/TABI-PB maintained Dr. Geng's group.

(4) The Fortran version HOBI package is available at https://sourceforge.net/projects/hobipb/, maintained by Dr. Geng's group. In addition, a C-version Cartesian FMM-accelerated Galerkin Boundary Integral (FAGBI) PB solver[54] is also available at https://github.com/gengwh/CFMM-PB maintained by Dr. Geng's and Dr. Tausch's group at Southern Methodist University, and Dr. Jiahui Chen's group at University of Arkansas.

5 | CONCLUSIONS

In this paper, we investigate the distinguished features of the ESES software[44] in generating conjugated Eulerian and Lagrangian surfaces. We numerically validate the quality of surface discretization under both frames by using two recently developed PB solvers: a Cartesian based MIBPB solver and a Lagrangian TABI-PB solver. The numerical results shows that, thanks to the ESES, the desired convergence of both solvers using the Eulerian and Lagrangian representations of the molecular surface generated by ESES. Furthermore, with ESES software and its associated PB solvers, an interesting and useful but often ignored source-target symmetric property associated with the linearized PB model is numerically validated.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (NSF) grants DMS-2110914, DMS-2110922, DMS-2110869, and IIS-1900473 and NIH grants R01AI164266 and R35GM148196.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in ESES at https://github.com/WeilabMSU/ESES.

ORCID

Shan Zhao https://orcid.org/0000-0002-3023-2107
Weihua Geng https://orcid.org/0000-0001-9911-6588
Guo-Wei Wei https://orcid.org/0000-0002-5781-2937

REFERENCES

[1] B. Lee, F. M. Richards, *J. Mol. Biol.* **1971**, *55*, 379.
[2] N. Huang, Y. Chelliah, Y. Shan, C. A. Taylor, S.-H. Yoo, C. Partch, C. B. Green, H. Zhang, J. S. Takahashi, *Science* **2012**, *337*, 189. http://www.sciencemag.org/content/337/6091/189.full.pdf, http://www.sciencemag.org/content/337/6091/189.abstract
[3] D. A. Beard, T. Schlick, *Biopolymers* **2001**, *58*, 106.
[4] E. Alexov, E. L. Mehler, N. Baker, A. M. Baptista, Y. Huang, F. Milletti, J. E. Nielsen, D. Farrell, T. Carstensen, M. H. M. Olsson, et al., *Proteins* **2011**, *79*, 3260 ISSN 1097-0134 (Electronic); 0887-3585 (Linking).
[5] J. Hu, S. Zhao, W. Geng, *Commun. Comput. Phys.* **2018**, *23*, 520.
[6] J. Chen, W. Geng, D. Reynolds, *Comput. Phys. Commun* **2021a**, *260*, 107742.
[7] Y. C. Zhou, B. Lu, A. A. Gorfe, *Phys. Rev. E* **2010**, *82*, 041923.
[8] D. D. Nguyen, B. Wang, G.-W. Wei, *J. Comput. Chem.* **2017**, *38*, 941.
[9] J. A. Wagoner, N. A. Baker, *Proc. Natl. Acad. Sci.* **2006**, *103*, 8331. http://www.pnas.org/content/103/22/8331.full.pdf, http://www.pnas.org/content/103/22/8331.abstract
[10] N. Unwin, *J. Mol. Biol.* **2005**, *346*, 967. http://www.sciencedirect.com/science/article/pii/S0022283604016018
[11] N. A. Baker, *Methods Enzymol.* **2004**, *383*, 94. http://www.sciencedirect.com/science/article/B7CV2-4C47J19-6/2/5c077798e7534bffa0a9b9e2c6959e22
[12] W. Im, D. Beglov, B. Roux, *Comput. Phys. Commun.* **1998**, *111*, 59.
[13] B. Honig, A. Nicholls, *Science* **1995**, *268*, 1144.
[14] L. Li, C. Li, S. Sarkar, J. Zhang, S. Witham, Z. Zhang, L. Wang, N. Smith, M. Petukh, E. Alexov, *BMC Biophys.* **2012**, *5*, 9. https://www.ncbi.nlm.nih.gov/pubmed/22583952
[15] M. Mirzadeh, M. Theillard, A. Helgadöttir, D. Boy, F. Gibou, *Commun. Comput. Phys.* **2013**, *13*, 150.
[16] R. Luo, L. David, M. K. Gilson, *J. Comput. Chem.* **2002**, *23*, 1244 http://www3.interscience.wiley.com/cgi-bin/abstract/96516852/ ABSTRACT
[17] J. Ying, D. Xie, *J. Comput. Phys.* **2015**, *298*, 636 http://www.sciencedirect.com/science/article/pii/S0021999115004027.
[18] Z. Qiao, Z. Li, T. Tang, *J. Comput. Math.* **2006**, *24*, 252.
[19] S. Yu, W. Geng, G. W. Wei, *J. Chem. Phys.* **2007a**, *126*, 244108.
[20] R. Egan, F. Gibou, *J. Comput. Phys.* **2017**, *346*, 71.
[21] W. Geng, S. Yu, G. W. Wei, *J. Chem. Phys.* **2007**, *127*, 114106.
[22] Q. Cai, J. Wang, H.-K. Zhao, R. Luo, *J. Chem. Phys.* **2009**, *130*, 145101. http://scitation.aip.org/content/aip/journal/jcp/130/14/10.1063/1.3099708
[23] W. Geng, S. Zhao, *J. Comput. Phys.* **2017**, *351*, 25.
[24] A. Lee, W. Geng, S. Zhao, *J. Comput. Phys.* **2020**, *426*, 109958.
[25] A. Juffer, B. van Keulen, A. van der Ploeg, H. Berendsen, *J. Comput. Phys.* **1991**, *97*, 144.
[26] A. H. Boschitsch, M. O. Fenley, H.-X. Zhou, *J. Phys. Chem. B* **2002**, *106*, 2741.
[27] B. Lu, X. Cheng, J. A. McCammon, *J. Comput. Phys.* **2007**, *226*, 1348. http://www.sciencedirect.com/science/article/pii/S0021999107002379
[28] C. Bajaj, S.-C. Chen, A. Rand, *SIAM J. Sci. Comput.* **2011**, *33*, 826.
[29] B. Zhang, B. Lu, X. Cheng, J. Huang, N. P. Pitsianis, X. Sun, J. A. McCammon, *Commun. Comput. Phys.* **2013**, *13*, 107.

[30] W. Geng, R. Krasny, *J. Comput. Phys.* **2013**, *247*, 62. http://www.sciencedirect.com/science/article/pii/S0021999113002404
[31] Y. Zhong, K. Ren, R. Tsai, *J. Comput. Phys.* **2018**, *359*, 199. http://www.sciencedirect.com/science/article/pii/S0021999118300317
[32] E. Cancès, Y. Maday, B. Stamm, *J. Chem. Phys.* **2013**, *139*, 054111.
[33] C. Quan, B. Stamm, Y. Maday, *SIAM J. Sci. Comput.* **2019**, *41*, B320.
[34] L. F. Greengard, J. Huang, *J. Comput. Phys.* **2002**, *180*, 642. http://www.sciencedirect.com/science/article/pii/S002199910297110X
[35] P. Li, H. Johnston, R. Krasny, *J. Comput. Phys.* **2009**, *228*, 3858.
[36] L. Wang, R. Krasny, S. Tlupova, *Commun. Comput. Phys.* **2020**, *28*, 1415.
[37] M. L. Connolly, *J. Mol. Graphics* **1985**, *3*, 19.
[38] M. F. Sanner, A. J. Olson, J. C. Spehner, *Biopolymers* **1996**, *38*, 305.
[39] S. Decherchi, W. Rocchia, *PLoS One* **2013**, *8*, 1.
[40] M. Chen, B. Lu, *J. Chem. Theory Comput.* **2011**, *7*, 203.
[41] L. Wilson, R. Krasny, *J. Comput. Chem.* **2021**, *42*, 1552.
[42] W. Rocchia, S. Sridharan, A. Nicholls, E. Alexov, A. Chiabrera, B. Honig, *J. Comput. Chem.* **2002**, *23*, 128.
[43] Y. Zhou, *Ph.D. Thesis*, Michigan State University. **2006**.
[44] B. Liu, B. Wang, R. Zhao, Y. Tong, G.-W. Wei, *J. Comput. Chem.* **2017**, *38*, 446.
[45] D. Chen, Z. Chen, C. Chen, W. Geng, G. W. Wei, *J. Comput. Chem.* **2011**, *32*, 657.
[46] C. Bottcher, *Theory of Electric Polarization. Dielectrics in Static Fields*, Vol. I, Elsevier; AMSTERDAM, **1973**.
[47] W. Geng, *Comput. Math. Biophys.* **2015**, *3*, 43.
[48] M. J. Holst, *Ph.D. Thesis*, UIUC. **1994**.
[49] S. Yu, G. W. Wei, *J. Comput. Phys.* **2007**, *227*, 602. http://www.sciencedirect.com/science/article/B6WHY-4PF1WN4-1/2/a142bb0596d6f227a0537b4790d00bcf
[50] S. Yu, Y. Zhou, G. Wei, *J. Comput. Phys.* **2007b**, *224*, 729. http://www.sciencedirect.com/science/article/pii/S002199910600547X
[51] W. Geng, *J. Comput. Phys.* **2013**, *241*, 253. http://www.sciencedirect.com/science/article/pii/S0021999113000739
[52] B. Lu, J. A. McCammon, *J. Chem. Theory Comput.* **2007**, *3*, 1134.
[53] L. Wilson, W. Geng, R. Krasny, *J. Phys. Chem. B* **2022**, *126*, 7104.
[54] J. Chen, J. Tausch, W. Geng, *J. Comput. Phys.* **2023**, *478*, 111981.
[55] Y. Saad, M. Schultz, *SIAM J. Sci. Stat. Comput.* **1986**, *7*, 856.
[56] L. Wilson, N. Vaughn, R. Krasny, *Comput. Phys. Commun.* **2021**, *265*, 108017.
[57] J. Barnes, P. Hut, *Nature* **1986**, *324*, 446.
[58] Z.-H. Duan, R. Krasny, *J. Comput. Chem.* **2001**, *22*, 184.
[59] K. Lindsay, R. Krasny, *J. Comput. Phys.* **2001**, *172*, 879. http://www.sciencedirect.com/science/article/pii/S0021999101968627
[60] J. Chen, W. Geng, *J. Comput. Phys.* **2018a**, *373*, 750.
[61] W. E. Lorensen, H. E. Cline, *SIGGRAPH Comput. Graph.* **1987**, *21*, 163.
[62] A. H. Juffer, P. Argos, H. J. Vogel, *J. Phys. Chem. B* **1997**, *101*, 7664.
[63] J. Chen, J. Hu, Y. Xu, R. Krasny, W. Geng, *J. Comput. Biophys. Chem.* **2021b**, *20*, 175.
[64] X. Yang, E. Sliheet, R. Iriye, D. Reynolds, W. Geng, In preparation. **2023**.
[65] J. Chen, W. Geng, G.-W. Wei, *Chin. J. Chem. Phys.* **2021c**, *34*, 683.
[66] J. G. Kirkwood, *J. Comput. Phys.* **1934**, *7*, 351.

How to cite this article: S. A. Ullah, X. Yang, B. Jones, S. Zhao, W. Geng, G.-W. Wei, *J. Comput. Chem.* **2024**, *45*(6), 306. https://doi.org/10.1002/jcc.27239