DEEP NEURAL NETWORK FOR SOLVING POISSON–BOLTZMANN EQUATIONS ON PROTEIN SURFACES

Duan Chen,[1] Cuiyu He,[2,*] Xiaozhe Hu,[3] Lin Mu,[2] & Zhaiming Shen[4]

[1]*Department of Mathematics and Statistics, The University of North Carolina at Charlotte, Charlotte, North Carolina 28223, USA*

[2]*Department of Mathematics, University of Georgia, Athens, Georgia 30602, USA*

[3]*Department of Mathematics, Tufts University, Medford, Massachusetts 02155, USA*

[4]*School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332, USA*

*Address all correspondence to: Cuiyu He, Department of Mathematics, University of Georgia, Athens, Georgia 30602, USA, E-mail: cuiyu.he@uga.edu

In this paper, we propose a new deep learning method for the nonlinear Poisson–Boltzmann problems with applications in computational biology. To tackle the discontinuity of the solution, e.g., across protein surfaces, we approximate the solution by a piecewise mesh-free neural network that can capture the dramatic change in the solution across the interface. The partial differential equation problem is first reformulated as a least-squares physics-informed neural network (PINN)-type problem and then discretized to an objective function using mean squared error via sampling. The solution is obtained by minimizing the designed objective function via standard training algorithms such as the stochastic gradient descent method. Finally, the effectiveness and efficiency of the neural network are validated using complex protein interfaces on various manufactured functions with different frequencies.

KEY WORDS: *Poisson–Boltzmann equations, interface problems, least-squares method, mesh-free, piecewise neural network, deep learning*

1. INTRODUCTION

As one of the commonly used models in molecular biology, the Poisson–Boltzmann equation (PBE) provides a mathematical framework that is used to describe the distribution of electric potentials and ion concentrations in solutions containing charged molecules, such as proteins, nucleic acids, and membranes, immersed in an electrolyte solution. The model combines the Poisson equation, which relates the electric potential to the charge distribution, with the Boltzmann distribution, which describes the statistical distribution of ions based on their electrostatic interactions and thermal energy. The PBE is a mean-field model that balances biophysic details and computational efficiency, hence becoming one of the major workhorses in the community of computational electrostatics; it has remained an active research topic in the past decades.

Submitted: 11/22/24; Accepted: 12/30/24; Online: 2/3/25
https://doi/10.1615/JMachLearnModelComput.2025057332
2689–3967/25/$35.00

The importance of PBE has been well established and is discussed in Briggs and McCammon (1992), Sharp and Honi (1990), and Fogolari et al. (2002). The major challenges from a mathematical point of view of the PBE include (1) Dirac distribution sources result in a singular solution; (2) discontinuous dielectric coefficients result in interface singularity; (3) geometric complexity of molecular domains in three spatial dimensions; (4) nonlinearity. Due to these challenges, analytical solutions are usually only known for unrealistic structure geometries and/or for linearizations of the equation. In practical applications with complex biomolecular surfaces, efficient and accurate numerical methods are demanded to solve the PBE. Computationally, a variety of techniques for numerically solving the PBE, including finite difference methods (FDMs) (Honig and Nicholls, 1995; Im et al., 1998; Luo et al., 2002; Warwicker and Watson, 1982; Yu et al., 2007), finite element methods (Baker et al., 2001; Bond et al., 2010; Chen et al., 2007), boundary integral methods (BIMs) (Boschitsch and Fenley, 2004; Zauhar and Morgan, 1985), the boundary element method (Zhou et al., 2006), and finite volume methods (Baker et al., 2001), have been developed in the past few decades and incorporated into popular molecular simulation software packages, such as DelPhi (Klapper et al., 1986), ZAP (Prabhu et al., 2008), UHBD (Madura et al., 1995), MEAD (Engels et al., 1995), APBS (Holst et al., 2000), AMBER (Lu et al., 2002; Luo et al., 2002), and CHARMM (Brooks et al., 1983; Jo et al., 2008). Recently, highly accurate and efficient numerical schemes of PBE and their applications in molecular biology have attracted much interest in the computational mathematics community. Novel algorithms, such as the FDM-based matched interface and boundary (MIB) method (Chen et al., 2011; Geng et al., 2007; Zhou et al., 2008), the immersed interface method (LeVeque and Li, 1994; Li and Ito, 2001), and the BIM-based treecode acceleration technique (Geng and Krasny, 2013), are developed to handle the solute-solvent interface and singular charges of proteins appearing in the PBE. Regularization methods have been recently investigated, compared, and implemented with the MIB approach in Li et al. (2021). Other approaches have been investigated, including the adoption of the Gummel method and discontinuous bubble functions in Kwon et al. (2021b), potential field reconstruction and enrichment techniques in Borleske and Zhou (2020), coupling finite and boundary elements for the solute and solvent regions in Bosy et al. (2024), an efficient finite element iterative method in Xie (2024), and goal-oriented adaptive error estimates to identify the optimal surface mesh in Ramm et al. (2021).

The solution of a partial differential equation (PDE) using neural networks has gained increasing interest in academia and industry. Numerous methods using neural networks have been developed, including the physics-informed neural network (PINN) (Raissi et al., 2019), the deep Ritz method (Yu, 2018), the deep Galerkin method (Sirignano and Spiliopoulos, 2018), weak adversarial network (Zang et al., 2020), the deep first-order least-squares (LS) method (Cai et al., 2020), the hp-variational physical informed neural networks (Kharazmi et al., 2021), and the neural ordinary differential equation-based network named XNODE (Oliva et al., 2022), to list a few. Several neural network methods have also been designed to solve PBE. In Liu et al. (2020), a multiscale deep neural network (DNN) was proposed using the idea of radial scaling in the frequency domain. In Kwon et al. (2021a), a deep residual-based neural network (He et al., 2016) is used to solve the PBE. In Chen et al. (2024), a Poisson–Boltzmann-based machine learning model for biomolecular electrostatic analysis is trained with the second-order accurate MIBPB solver. Various neural network methods have also been invented to solve interface problems (Dwivedi et al., 2019; He et al., 2022; Wang and Zhang, 2020; Wu and Lu, 2022).

Our previous work in He et al. (2022) used deep learning methods to solve the elliptic interface problems. The second-order elliptic interface problem is rewritten as a least-squares PINN-type minimization problem. The numerical solutions are represented as DNN network functions

with parameter sets to be learned through gradient descent optimization algorithms that aim to minimize the least-squares cost functional. More specifically, we used two separate DNN models in the case of two subdomains to approximate the solution that has large jumps in derivative(s) across the intersection of subdomains (interface).

This paper continues our previous work and utilizes a piecewise machine learning-based DNN to solve the nonlinear PBE. The primary motivation is that deep neural algorithms do not require mesh fitting. It is worth noting that generating a regular mesh for a complex domain in three-dimensional (3D) space remains one of the bottleneck challenges. Furthermore, unlike the classical numerical method, no linearization is necessary using the machine learning-based method since the DNN structure is nonlinear itself.

Specifically, we investigate a deep learning scheme to solve the nonlinear PBE on the complex molecular surface. A molecular surface is defined as the combination of all atomic surfaces of a molecule's atoms. In practice, the molecular surface meshes generated by software may have small angles, redundant vertices, and low-quality elements, thus causing many problems for traditional numerical solvers. Here, we contribute to overcoming the challenges in the meshing process and developing numerical schemes. Firstly, the singularity of Dirac delta functions is tackled by a regularization approach to decompose the solution into regular and singular components. Since the singular component can be explicitly constructed, we focus on approximating the regular components from a nonlinear elliptic interface problem. For the regular problem, the computational domain is decomposed by a molecular surface into two subdomains, where separate DNNs are employed in each subdomain. By incorporating this technique, our network can effectively represent discontinuous functions across multiple domains.

As the exact integral of the least-squares problem is impossible to compute, we approximate it by discretizing it to an objective function using mean squared error via sampling points in each subdomain, on the interface surface, and the boundary. Due to the complexity of the problem, a large number of sampling points are typically required to achieve a satisfactory solution. Our approach trains the model using the ADAM-type stochastic gradient descent method. In each epoch, the objective function of mean square type is computed using a relatively small subset of data points randomly sampled from a fixed, large dataset. After a fixed number of training epochs, the subset is randomly reselected to further refine the model. This iterative process of resampling and retraining continues until a stopping criterion is met. By leveraging a small number of data points per epoch and periodically resampling, our method significantly reduces training time while maintaining high accuracy on the large dataset, in contrast to traditional approaches that rely on large, static datasets.

The remainder of the paper is organized as follows. The notation and regularization of the nonlinear PBE are discussed in Section 2. Section 3 introduces our deep least-squares method to solve the interface problem in detail. The numerical results are shown in Section 4 to demonstrate the efficiency of the proposed method. Finally, we discuss the conclusions and potential future work in Section 5.

2. THE POISSON BOLTZMANN EQUATION AND ITS REGULARIZATION

This paper employs the standard notations and definitions for the Sobolev spaces $H^s(\Omega)$ and $H^s(\partial\Omega)$. The standard associated inner products are denoted by $(\cdot,\cdot)_{s,\Omega}$ and $(\cdot,\cdot)_{s,\Gamma}$ in $\Omega \in \mathbb{R}^d$ and on $\Gamma \in \mathbb{R}^{d-1}$, respectively. The standard induced norms are denoted by $\|\cdot\|_{s,\Omega}$ and $\|\cdot\|_{s,\Gamma}$. When $s = 0$, $H^0(\Omega)$ coincides with $L^2(\Omega)$. When there is no ambiguity, the subscript Ω in the designation of norms will be suppressed.

The nonlinear PBE, which brings together the description of the electrostatic potential $u(\boldsymbol{x})$ in a protein-solvent system $\Omega \in \mathbb{R}^3$ reads

$$-\nabla \cdot (\epsilon(\boldsymbol{x})\nabla u) + I_s \sinh u = \sum_{i=1}^{N_a} z_i \delta(\boldsymbol{x} - \boldsymbol{x}_i), \tag{1}$$

with the Dirichlet boundary condition on the boundary of Ω, i.e.,

$$u(\boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x} \in \partial\Omega. \tag{2}$$

The classical PBE is developed based on a multiscale approximation strategy, and its biophysical meaning is sketched as follows:

- In this setup, the whole domain Ω is naturally divided as protein domain Ω^+ and solvent domain Ω^-. These two disjoint subdomains are separated by the molecular surface Γ, i.e., $\Omega = \Omega^+ \cup \Omega^-$ and $\Omega^+ \cap \Omega^- = \Gamma$.
- The protein is described explicitly, i.e., location ($\boldsymbol{x}_i$) and charge (z_i) of each protein atom are counted and individually contribute to the electrostatics, as shown in the right-hand side of Eq. (1). For the protein, there are total N_a atoms. Specific values of these parameters for a targeted protein can be obtained from the Protein Data Bank (PDB).
- For each protein atom, its geometry is represented by a ball with center $\boldsymbol{x}_i$ and radius r_i. The protein surface Γ is defined as the outer surface of the union of all such balls. It is assumed that the union of all atoms forms a simply connected set, which ensures that Γ separates the domain into two subdomains.
- On the other hand, the solvent's water molecules and mobile ions are modeled implicitly or as a continuum for efficiency. As result, the nonlinear term (hyperbolic sine) in Eq. (1) represents the continuum approximation of water molecules and Boltzmann distribution approximation of mobile ions corresponding to the electrostatics, with I_s being the ionic strength.

Due to the multiscale physics, the dielectric function $\epsilon(\boldsymbol{x})$ in Eq. (1) is taken as a piecewise constant function on the two subdomains, i.e.,

$$\epsilon(\boldsymbol{x}) = \begin{cases} \epsilon^+, & \boldsymbol{x} \in \Omega^+ \\ \epsilon^-, & \boldsymbol{x} \in \Omega^- \end{cases}. \tag{3}$$

As a common setting, low dielectric ($\epsilon^+ = 1$) is assumed for protein domain Ω^+ and high dielectric ($\epsilon^- = 80$) for solvent domain Ω^-. Ionic strength ($I_s^+ = 0$) is assumed for protein domain Ω^+ and ($I_s^- = 0.5$) for solvent domain Ω^-. Because of the discontinuity of $\epsilon(\boldsymbol{x})$, the solution u in Ω is not smooth across the interface Γ and it is subject to the following interface conditions:

$$[\![u]\!]|_\Gamma = u^+(\boldsymbol{x}) - u^-(\boldsymbol{x}) = 0, \tag{4}$$

$$[\![\epsilon \nabla u \cdot \mathbf{n}]\!]|_\Gamma = \epsilon^+ \nabla u^+ \cdot \mathbf{n} - \epsilon^- \nabla u^- \cdot \mathbf{n} = 0, \tag{5}$$

where superscripts $\pm$ represent the restriction of a certain function on the domain $\Omega^\pm$ and $\mathbf{n}$ is the unit normal of Γ pointing from Ω^+ to Ω^-.

One obvious challenge of numerically solving the model (1) is the singularity of Dirac delta functions on the right-hand side, which could cause numerical instability. A regularization approach (Chen, 2016; Chen et al., 2011; Chen and Wei, 2010; Yu et al., 2007; Yu and Wei, 2007) is usually used to remove this singularity and the computation is then only focused on the regular part. Let

$$u = u_0 + u^*, \tag{6}$$

where u_0 and u^* denote the regular and singular parts of u, respectively. More specifically, u_0^- coincides with u^- in the solvent domain whereas u^* is only defined in the protein domain Ω^+. Note that the delta functions $\delta(\boldsymbol{x} - \boldsymbol{x}_i)$ are located in Ω^+; solution u^+ can be approached as the sum of the Green's function in free space and some correction u_0^+ for the bounded domain Ω^+, i.e., $u^+ = u^* + u_0^+$, and

$$u^*(\boldsymbol{x}) = \begin{cases} \displaystyle\sum_{i=1}^{N_a} \frac{z_i}{4\pi\epsilon^+|\boldsymbol{x} - \boldsymbol{x}_i|}, & \boldsymbol{x} \in \Omega^+ \\ 0, & \boldsymbol{x} \in \Omega^- \end{cases}. \tag{7}$$

Also note that the ionic strength parameter I_s is only defined in the solvent domain Ω^-, where $u^* = 0$ as indicated in the above equation. Hence, the original nonlinear term $\sinh(u)$ applies only to u_0 in this singularity removal scheme.

We are, therefore, left to solve the following regularized interface problem:

$$-\nabla \cdot \big(\epsilon(\boldsymbol{x})\nabla u_0(\boldsymbol{x})\big) + I_s \sinh(u_0) = 0, \qquad \boldsymbol{x} \in \Omega, \tag{8}$$

$$[\![u_0]\!]_\Gamma = -u^*, \qquad \boldsymbol{x} \in \Gamma, \tag{9}$$

$$[\![\epsilon\nabla u_0 \cdot \mathbf{n}]\!]|_\Gamma = -\epsilon^+\nabla u^* \cdot \mathbf{n}, \qquad \boldsymbol{x} \in \Gamma, \tag{10}$$

$$u_0 = g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega. \tag{11}$$

System (8)–(11) is mathematically well-posed after singularity removal. However, in the practical problem for biomolecules, the complicated molecular surface Γ poses great challenges in developing various numerical methods.

3. MESH-FREE DNN

In this section, let us discuss the mesh-free piecewise DNN structure utilized in our approach and the loss functional.

3.1 Deep Neural Network Structure

We first discuss the DNN structure used to approximate the solution $u(\boldsymbol{x})$. A DNN structure is the composition of multiple linear functions and nonlinear activation functions. Specifically, the first component of DNN is a linear transformation $\boldsymbol{T}^\ell : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_{\ell+1}}$, $\ell = 1, \cdots, L$, defined as follows,

$$\boldsymbol{T}^\ell(\boldsymbol{x}^\ell) = \boldsymbol{W}^\ell \boldsymbol{x}^\ell + \boldsymbol{b}^\ell, \quad \text{for } \boldsymbol{x}^\ell \in \mathbb{R}^{n_\ell},$$

where $\boldsymbol{W}^\ell = (w_{i,j}^\ell) \in \mathbb{R}^{n_{\ell+1} \times n_\ell}$ and $\boldsymbol{b}^\ell \in \mathbb{R}^{n_{\ell+1}}$ are parameters in the DNN. The second component is an activation function $\psi : \mathbb{R} \to \mathbb{R}$ to be chosen. Typical examples of the activation functions are tanh, Sigmoid, and ReLU. Application of ψ to a vector $\boldsymbol{x} \in \mathbb{R}^n$ is defined

component-wisely, i.e., $\psi(\boldsymbol{x}) = (\psi(x_1), \cdots, \psi(x_n))^\top$. The ℓ-th layer of the DNN can be represented as the composition of the linear transform $\boldsymbol{T}^\ell$ and the nonlinear activation function ψ, i.e.,

$$\mathcal{N}^\ell(\boldsymbol{x}^\ell) = \psi\big(\boldsymbol{T}^\ell(\boldsymbol{x}^\ell)\big), \quad \ell = 1, \cdots, L.$$

Note $\mathcal{N}^\ell : \mathbb{R}^{n_\ell} \mapsto \mathbb{R}^{n_{\ell+1}}$. A L-layer DNN is then defined as the composition of all $\mathcal{N}^\ell$, $\ell = 1, 2, \cdots, L$. In particular, for an input $\boldsymbol{x} \in \mathbb{R}^{n_1}$, a general L-layer DNN can be represented as follows,

$$\mathcal{N}(\boldsymbol{x}; \Theta) = \boldsymbol{T}^L \circ \mathcal{N}^{L-1} \circ \cdots \circ \mathcal{N}^2 \circ \mathcal{N}^1(\boldsymbol{x}), \tag{12}$$

where $\Theta \in \mathbb{R}^N$ stands for all the parameters in the DNN, i.e.,

$$\Theta = \{\boldsymbol{W}^\ell, \boldsymbol{b}^\ell, \ \ell = 1, \cdots, L\}.$$

For a fully connected DNN, we have the number of parameters $N = \sum_{\ell=1}^{L} n_{\ell+1}(n_\ell + 1)$.

In this paper, we use the DNN structure (12) to approximate the solution $u(\boldsymbol{x})$. Because of the nature of our computational domain, we use two different DNNs with parameters Θ_i, $i = 1, 2$, to approximate u_0^+ and u_0^-, respectively. The architecture of our DNN model is summarized in Fig. 1.

3.2 The Discretization in the DNN Method

For any $\boldsymbol{x} \in \Omega$, our goal is to find a $\Theta^* \in \mathbb{R}^N$ for the DNN network in Eq. (12) such that $u_{\Theta^*}(\boldsymbol{x}) \approx u_0(\boldsymbol{x})$. To find such Θ^*, we consider minimizing the PINN-type LS objective function. We note that there are various versions of LS formulations for the interface problem (8)–(10). In this work, we adopt the simple LS principle proposed in Dissanayake and Phan-Thien (1994) and propose an LS functional that incorporates the interface conditions (9) and (10) and the boundary condition (11). The LS functional is defined as follows,

$$\mathcal{J}(v; u^*, g) = \text{loss}_{\text{pde}} + \text{loss}_{\text{bc}} + \text{loss}_{\text{I}}, \tag{13}$$

for all $v \in H^1(\Omega)$, and

$$\text{loss}_{\text{pde}} = |\Omega^+|^{-1} \| - \nabla \cdot \epsilon \nabla v + I_s \sinh(v) \|^2_{0,\Omega^+} + |\Omega^-|^{-1} \| - \nabla \cdot \epsilon \nabla v + I_s \sinh(v) \|^2_{0,\Omega^-},$$

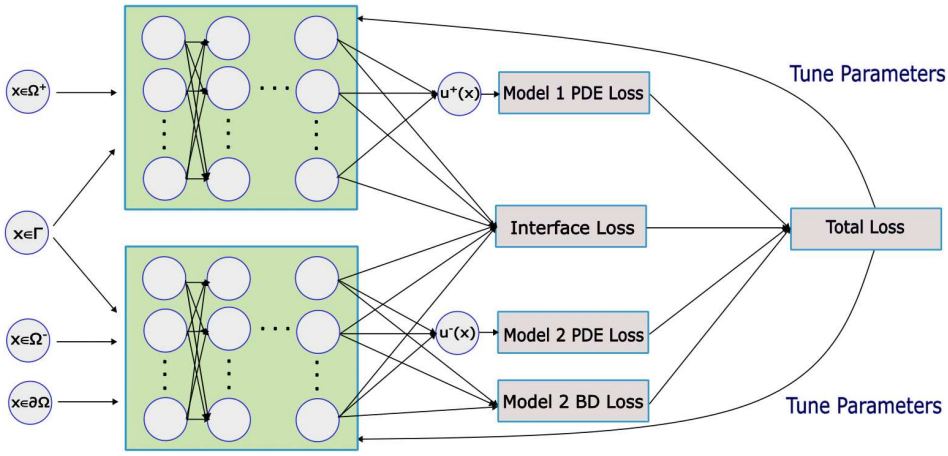


FIG. 1: Architecture of the piecewise DNN model

$$\text{loss}_{\text{bc}} = \alpha|\partial\Omega|^{-1}\|v-g\|^2_{0,\partial\Omega},$$

$$\text{loss}_{\text{I}} = |\Gamma|^{-1}\big(\beta_f\|[\![\epsilon\nabla v\cdot\vec{\boldsymbol{n}}]\!]\|^2_{0,\Gamma} + \epsilon^+\nabla u^*\cdot\vec{\mathbf{n}}\|^2_{0,\Gamma} + \beta_j\|[\![v]\!]+u^*\|^2_{0,\Gamma}\big),$$

where α and (β_j, β_f) are hyperparameters controlling the strength of constraints at the domain boundary and interface, respectively. The corresponding LS solution is then to find $u^\dagger \in H^1(\Omega)$, such that

$$\mathcal{J}(u^\dagger; u^*, g) = \inf_{v\in H^1(\Omega)} \mathcal{J}(v; u^*, g_D). \tag{14}$$

It is easy to see that when the problem (8)–(10) has a strong solution such that $-\nabla\cdot\epsilon\nabla u_0 + I_s\sinh u_0 \in L^2(\Omega)$, the solution $u^\dagger$ to Eq. (14) is equivalent to the solution u_0 to Eqs. (8)–(10). The DNN approximation to the solution $u^\dagger$ is then to find $u^\dagger_\Theta \in L^2(\Omega)$ such that

$$\mathcal{J}(u^\dagger_\Theta; u^*, g_D) = \inf_{v_\Theta\in V_\Theta} \mathcal{J}(v_\Theta; u^*, g_D), \tag{15}$$

where V_Θ represents the function class consisting of all neural networks with a fixed architecture, parameterized by the set of parameters Θ.

We approximate the integrals using the Monte Carlo sampling method. More precisely, for any function v, we let

$$\begin{gathered}\frac{1}{|\Omega^\pm|}\|v\|^2_{\Omega^\pm} \approx \frac{1}{N_i^\pm}\sum_{i=1}^{N_i^\pm}|v(\boldsymbol{x}_i^\pm)|^2, \qquad \frac{1}{|\Gamma|}\|v\|^2_{\Gamma} \approx \frac{1}{N_I}\sum_{j=1}^{N_I}|v(\boldsymbol{x}_i^I)|^2,\\ \frac{1}{|\partial\Omega|}\|v\|^2_{\partial\Omega} \approx \frac{1}{N_b}\sum_{k=1}^{N_b}|v(\boldsymbol{x}_k^b)|^2,\end{gathered} \tag{16}$$

where $N_i^\pm$ is the number of points randomly sampled in the interior of $\Omega^\pm$, N_I the number of points randomly sampled on Γ, N_b the number of points randomly sampled on $\partial\Omega$; $\{\boldsymbol{x}_i^\pm\}_{i=1}^{N_i^\pm}$ is the set of points sampled inside $\Omega^\pm$, $\{\boldsymbol{x}_i^I\}_{j=1}^{N_I}$ the set of points sampled on Γ, $\{\boldsymbol{x}_k^b\}_{k=1}^{N_b}$ the set of points sampled on $\partial\Omega$.

The correspondence in Fig. 1 and losses defined above are as follows:

$$\text{Model 1 PDE loss} \approx |\Omega^+|^{-1}\|-\nabla\cdot\epsilon\nabla v + I_s\sinh(v)\|^2_{0,\Omega^+},$$

$$\text{Model 2 PDE loss} \approx |\Omega^-|^{-1}\|-\nabla\cdot\epsilon\nabla v + I_s\sinh(v)\|^2_{0,\Omega^-},$$

$$\text{Interface loss} \approx \text{loss}_{\text{I}}, \qquad \text{Model 2 BD loss} \approx \text{loss}_{\text{bc}}.$$

4. NUMERICAL EXPERIMENTS

This section presents several numerical results of our DNN method for solving the PBEs [Eq. (1)]. The computational domain Ω is modeled as a box-shaped region representing a protein-solvent system. The domain Ω is sufficiently large to encompass all the atoms defined by the molecular surface. Recall that the protein surface Γ separates the domain into two subdomains: Ω^+ (the protein domain, inside the surface) and Ω^- (the solvent domain, outside the surface). The protein surfaces considered in this paper are depicted in Fig. 2, along with their corresponding PDB IDs.

We consider a specific regularized solution,

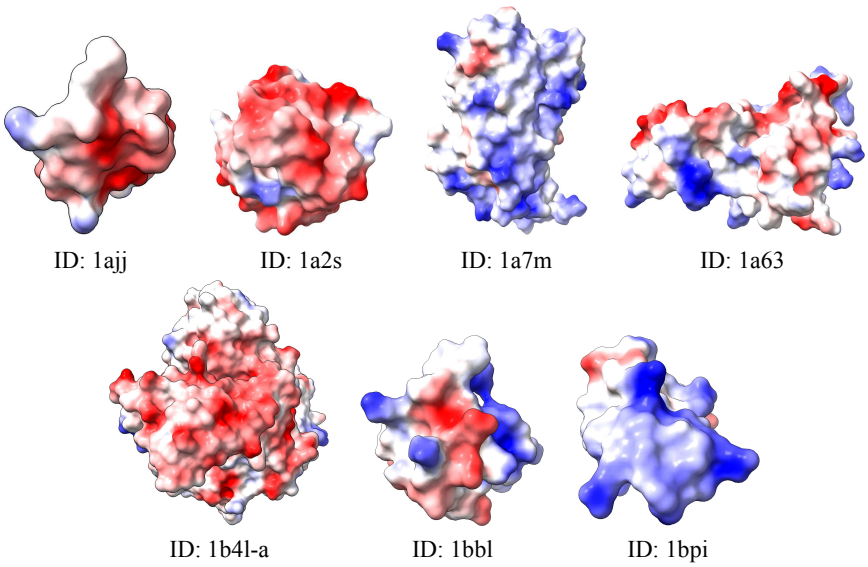


FIG. 2: Seven randomly chosen protein surfaces in our computation

$$u_0(\boldsymbol{x}) = \begin{cases} u_0^+, & \boldsymbol{x} \in \Omega^+ \\ u_0^-, & \boldsymbol{x} \in \Omega^- \end{cases}, \tag{17}$$

where $u_0^+ = \cos(ax)\cos(by)\cos(cz)$ and $u_0^- = -\cos(ax)\cos(by)\cos(cz)$, and a, b, c take the following values:

- Case 1. $a = 1, b = 1, c = 1$.
- Case 2. $a = 1, b = 2, c = 3$.
- Case 3. $a = 2, b = 3, c = 4$.

We rescale the domain Ω by multiplying a factor of 1/50 to make the computational domain close to $[-1, 1]^3$. The system (8)–(11) can be adjusted as

$$-\nabla \cdot \big(\epsilon(\boldsymbol{x})\nabla u_0\big) + I_s \sinh u_0 = f(\boldsymbol{x}), \tag{18}$$

$$[\![u_0]\!]_\Gamma = u_0^+ - u_0^- = 2\cos(ax)\cos(by)\cos(cz), \tag{19}$$

$$[\![\epsilon \nabla u_0 \cdot \vec{\mathbf{n}}]\!]|_\Gamma = (\epsilon^+ + \epsilon^-)\nabla\big(\cos(ax)\cos(by)\cos(cz)\big) \cdot \vec{\mathbf{n}}, \tag{20}$$

$$u_0|_{\partial\Omega} = -\cos(ax)\cos(by)\cos(cz), \tag{21}$$

where f is computed based on the function u_0 and the parameters are set as $\epsilon^+ = 1$, $\epsilon^- = 80$, $I_s^+ = 0$, and $I_s^- = 0.5$.

In all experiments, the neural network structure is chosen to have six hidden layers and 20 neurons in each layer. In the following tests, the penalty parameters are chosen as $\alpha = 100{,}000$, $\beta_f = 100$, $\beta_j = 100$. We begin by generating a large dataset that serves as the total pool for future random sampling or testing. The DNN model is trained by randomly selecting subsets of data from this pool. Training begins with an initial subset, and after every 100 epochs, a new

random subset is drawn from the full dataset. This process is repeated until the stopping criterion is met. The training employs the ADAM-type stochastic gradient descent method.

By minimizing the objective function over smaller subsets and frequently alternating the training data, our approach approximates the overall behavior of the full dataset while significantly reducing training costs, ultimately achieving better results. Unlike traditional methods that train the model on the entire dataset, our strategy delivers superior performance with more efficient time utilization.

During training, testing is performed concurrently to monitor the decay of the objective function. The test points, distinct from the training points but of the same size, are randomly selected from the total dataset.

The details of the total dataset profiles are summarized in Table 1, while the profiles of the smaller subsets used in each experiment will be specified individually.

Finally, the relative error for the testing subdataset is computed as

$$\text{Rel-Error} := \left(\frac{\sum_{i=1}^{N_i^-} |u_0^-(x_i^-) - u_{0,\Theta}^-(x_i^-)|^2 + \sum_{i=1}^{N_i^+} |u_0^+(x_i^+) - u_{0,\Theta}^+(x_i^+)|^2}{\sum_{i=1}^{N_i^-} |u_0(x_i^-)|^2 + \sum_{i=1}^{N_i^+} |u_0(x_i^+)|^2} \right)^{1/2}. \quad (22)$$

Here, the test points x_i^- and x_i^+ are chosen randomly from the total dataset in the interior and exterior domain, respectively.

4.1 Case 1 on Various Protein Surfaces

In the first test, the exact solution is selected as in Case 1, and the proposed scheme will be performed on various protein surfaces, as shown in Fig. 2. The parameters for the subset sizes are set as $N_i^{\pm} = 500,\ N_I = 1000,\ N_b = 2400$. The stopping criterion is defined as either the relative error on the test dataset falling below 1% or the training reaching a maximum of 5000 epochs.

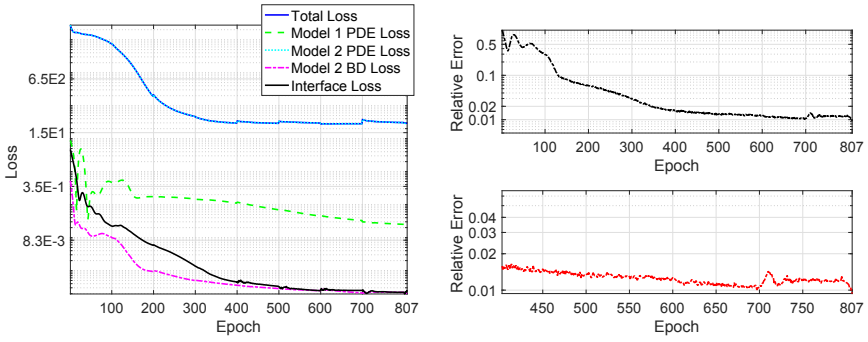
First, we apply our algorithm to the protein surface with ID: 1ajj. The convergence results for the loss functions are shown in Fig. 3(a), while the relative error, computed as in Eq. (22), is illustrated in Fig. 3(b). Note that the training and testing subsets are updated every 100 epochs.

The solution profiles of u_0^+ and u_0^- on the interface are presented in Figs. 4(a) and 4(b), respectively. Their approximations, produced by the piecewise neural network and denoted as $u_{0,\Theta}^+$ and $u_{0,\Theta}^-$, are provided in Figs. 4(c) and 4(d). Finally, the corresponding pointwise errors are shown in Figs. 4(e) and 4(f).

The following observations can be made:

TABLE 1: Number of points in total datasets for tested protein profiles

Molecular ID	N_I	N_b	N_i^-	N_i^+
1ajj	20,188	72,334	1.3123e+06	4236
1a2s	42,187	1.2139e+05	2.8618e+06	10,115
1a7m	73,580	2.2961e+05	7.3468e+06	21,106
1a63	66,006	1.9999e+05	5.8482e+06	15,792
1b4l	59,390	1.8616e+05	5.4292e+06	17,203
1bpi	60,424	1.0032e+05	2.1383e+06	7139
1bbl	24,499	86,346	1.7202e+06	4485



(a) (b)

FIG. 3: Case 1. Convergence plots of loss functions and relative error for protein ID: 1ajj. (a) Convergences of errors (Training) and (b) relative error (Test).

- Each term of the loss function decreases as expected, and the stopping criterion is met at epoch 807; see Fig. 3(a).
- By 807 epochs, the loss values have decreased significantly. For example, the total loss is reduced from 2.75×10^4 to 4.66, and the interface loss decreases from 4.6630 to 2.3563×10^{-4}, demonstrating the effectiveness of our training; see Fig. 3(a).
- For testing, we plot the relative error at each epoch. The relative error decreases significantly, following a similar pattern to the training loss, which demonstrates the accuracy of our numerical solution. Since the test datasets are chosen randomly, this further validates the accuracy of our numerical solution across the entire domain; see Fig. 3(b).
- Both the training and test convergence curves exhibit oscillations, which may be caused by the stochastic gradient method and the periodic change of training points over time.
- With the relative error being less than 1%, it is evident that the numerical solution agrees well with the exact solution in both the interior and exterior subdomains. The solution also exhibits small errors on the interface, as shown in Fig. 4.

We also conduct the research on other protein surfaces for the function u_0 in Case 1, including 1a2s (Figs. 5 and 6), 1a7m (Figs. 7 and 8), 1a63 (Figs. 9 and 10), 1b4l (Figs. 11 and 12), 1bpi (Figs. 13 and 14), and 1bbl (Figs. 15 and 16).

Lastly, we report the errors at the last epoch for all tested protein surfaces in Table 2. The data represent the relative discrete L^2-error calculated over the total dataset on the interface Γ, in the interior Ω^-, the exterior Ω^+, and on the boundary $\partial\Omega$. These numerical results validate the effectiveness of our algorithm.

4.2 Test with Different Exact Solutions

In this section, we present the results for the protein surface with ID 1ajj using different manufactured solutions. The numerical performance is compared to the exact solution across Cases 1, 2, and 3. Similar results are observed for other protein surfaces, which are omitted for brevity.

In contrast to Case 1, the solutions in Cases 2 and 3 involve higher frequency, and thus more sampling points are required to accurately resolve the exact solution. As a result, we enlarge

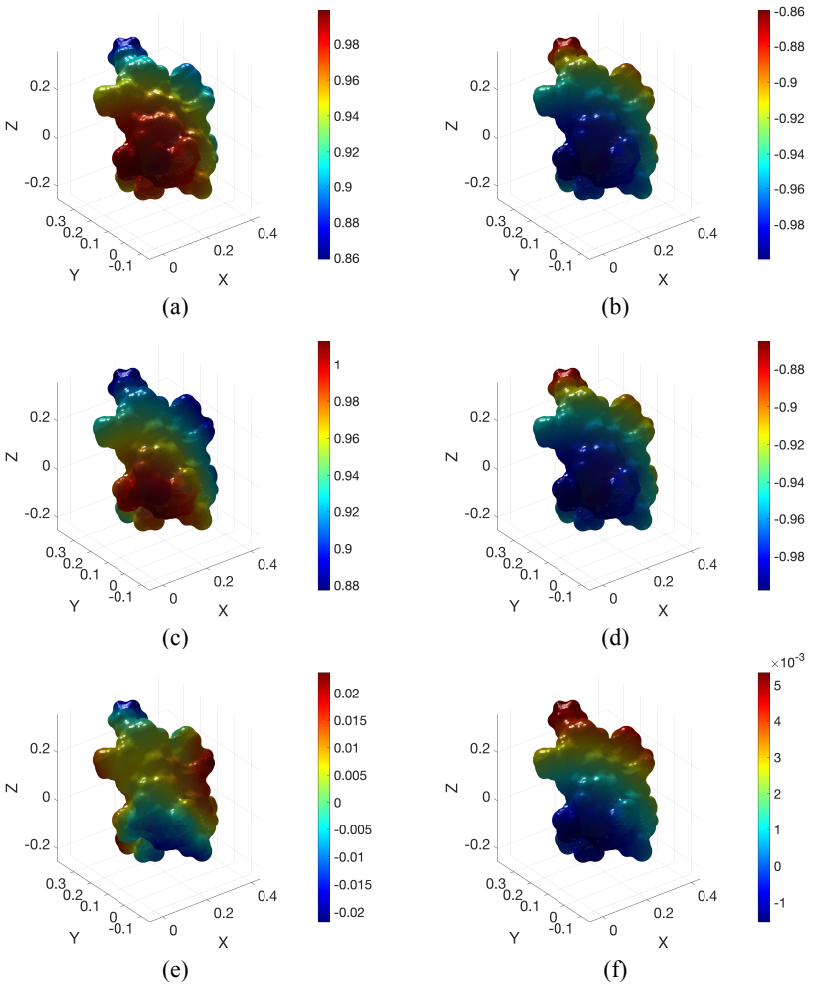


FIG. 4: Case 1. Plots for protein ID: 1ajj. (a) True solution $u_0^+(\Gamma)$, (b) true solution $u_0^-(\Gamma)$, (c) numerical solution $u_{0,\Theta}^+(\Gamma)$, (d) numerical solution $u_{0,\Theta}^-(\Gamma)$, (e) error on $u_{0,\Theta}^+(\Gamma)$, and (f) error on $u_{0,\Theta}^-(\Gamma)$.

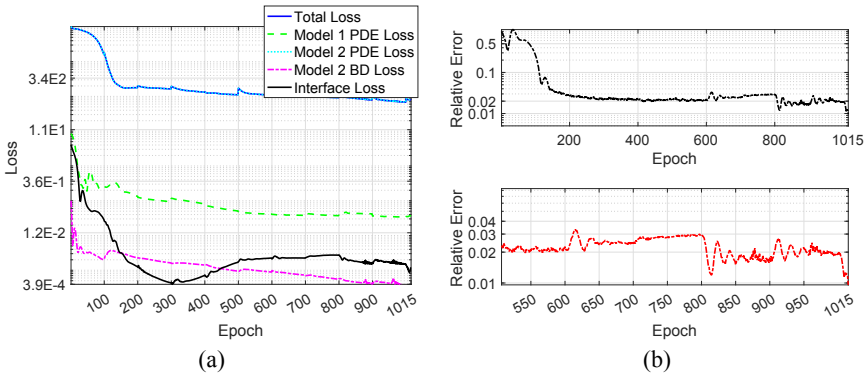


FIG. 5: Case 1. Convergence plots of loss functions and relative error for protein ID: 1a2s. (a) Convergences of errors (Training) and (b) relative error (Test).

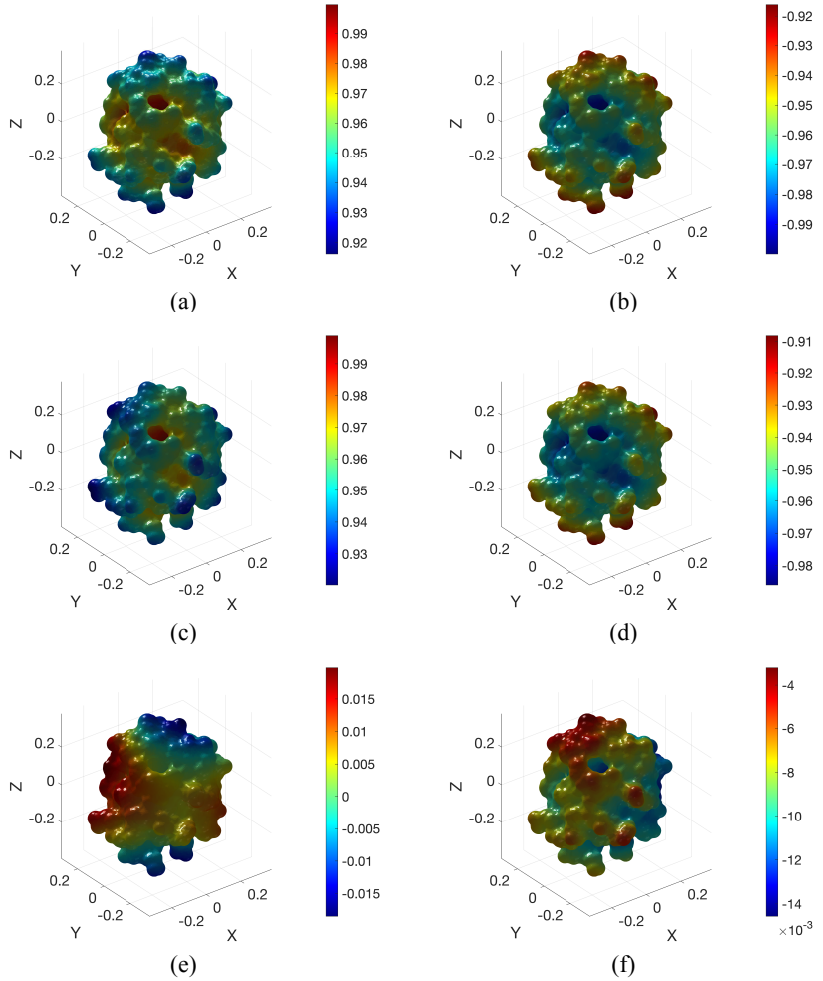


FIG. 6: Case 1. Plots for protein ID: 1a2s. (a) True solution $u_0^+(\Gamma)$, (b) true solution $u_0^-(\Gamma)$, (c) numerical solution $u_{0,\Theta}^+(\Gamma)$, (d) numerical solution $u_{0,\Theta}^-(\Gamma)$, (e) error on $u_{0,\Theta}^+(\Gamma)$, and (f) error on $u_{0,\Theta}^-(\Gamma)$.

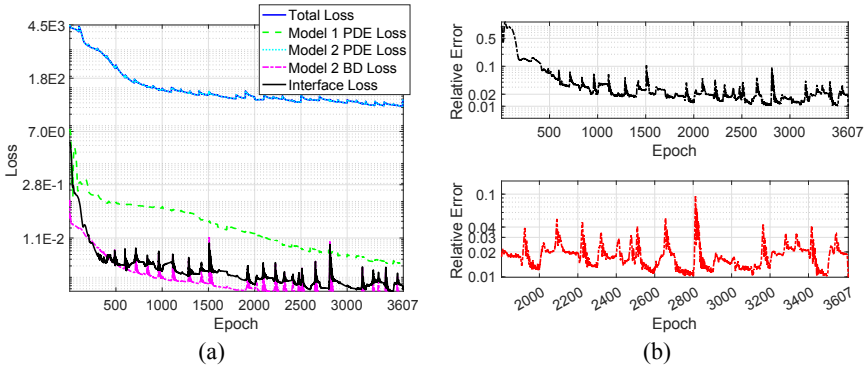


FIG. 7: Case 1. Convergence plots of loss functions and relative error for protein ID: 1a7m. (a) Convergences of errors (Training), and (b) relative error (Test).

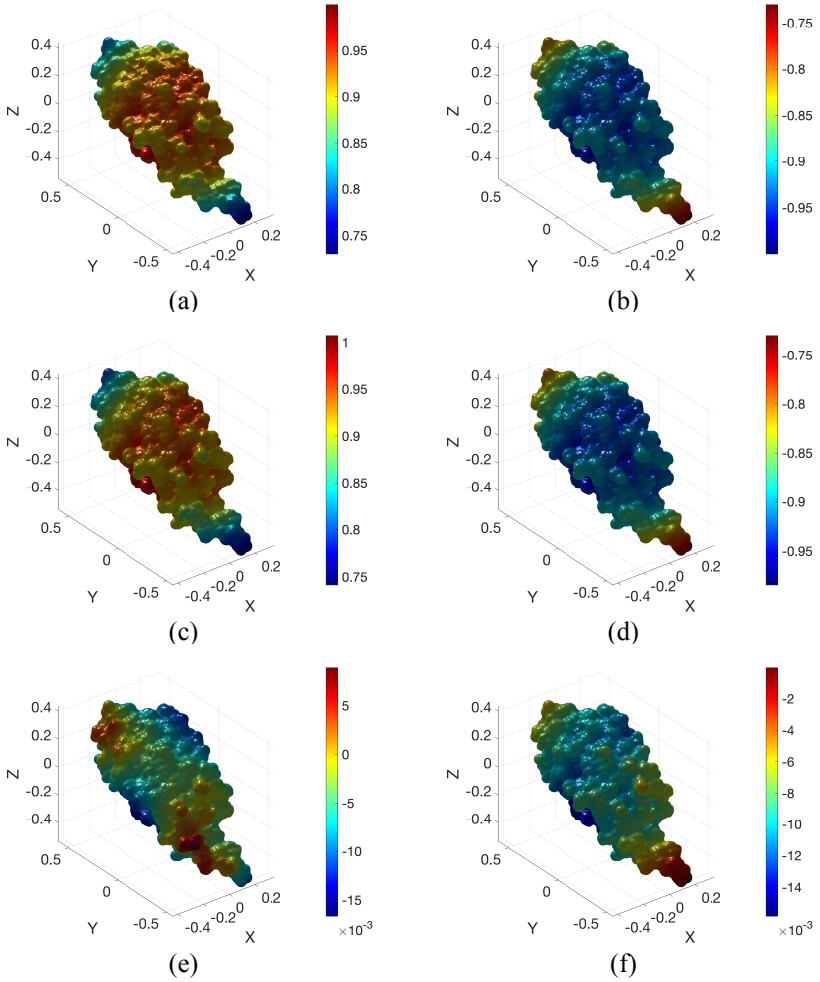


FIG. 8: Case 1. Plots for protein ID: 1a7m. (a) True solution $u_0^+(\Gamma)$, (b) true solution $u_0^-(\Gamma)$, (c) numerical solution $u_{0,\Theta}^+(\Gamma)$, (d) numerical solution $u_{0,\Theta}^-(\Gamma)$, (e) error on $u_{0,\Theta}^+(\Gamma)$, and (f) error on $u_{0,\Theta}^-(\Gamma)$.

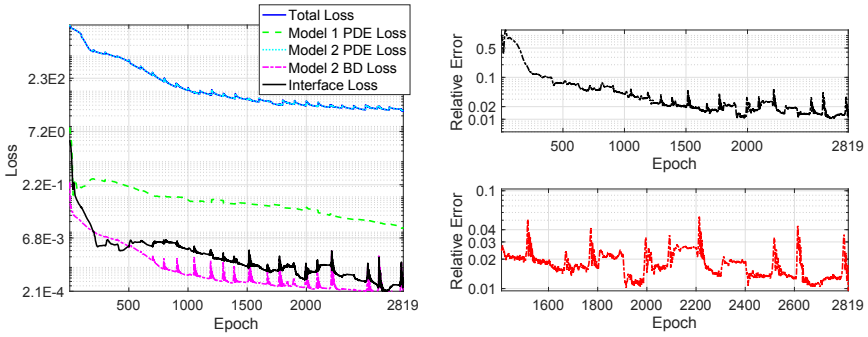


FIG. 9: Case 1. Convergence plots of loss functions and relative error for protein ID: 1a63. (a) Convergences of errors (Training), and (b) relative error (Test).

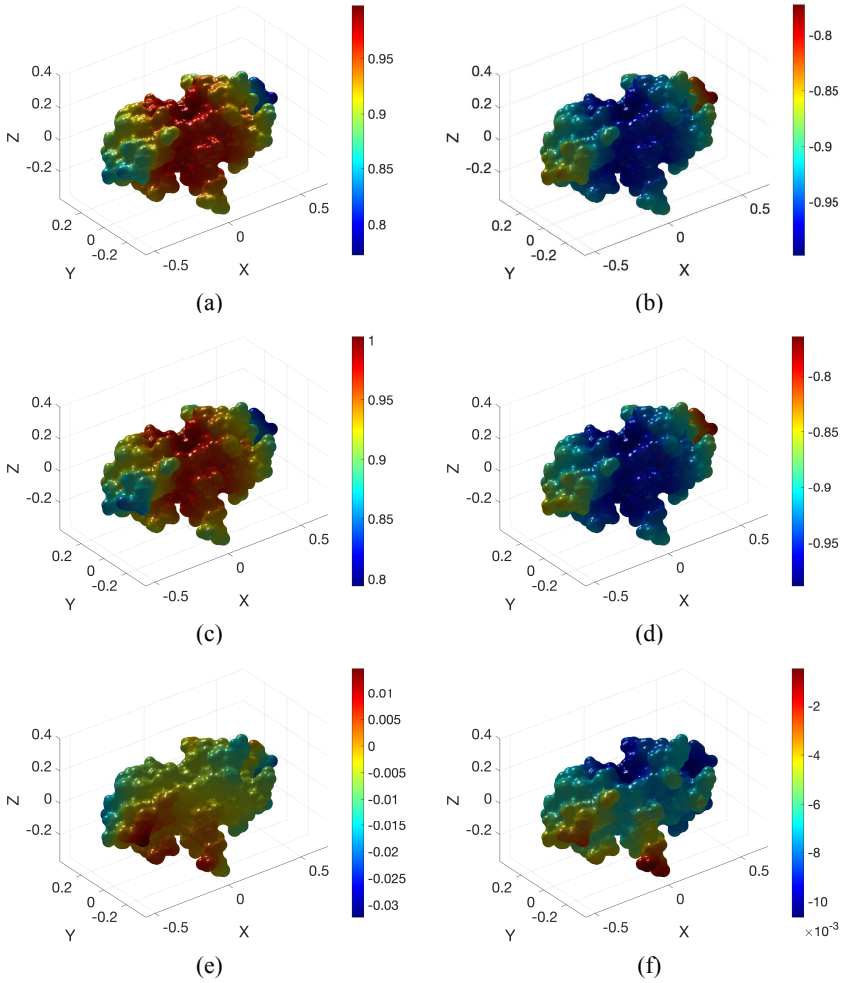


FIG. 10: Case 1. Plots for protein ID: 1a63. (a) True solution $u_0^+(\Gamma)$, (b) true solution $u_0^-(\Gamma)$, (c) numerical solution $u_{0,\Theta}^+(\Gamma)$, (d) numerical solution $u_{0,\Theta}^-(\Gamma)$, (e) error on $u_{0,\Theta}^+(\Gamma)$, and (f) error on $u_{0,\Theta}^-(\Gamma)$.

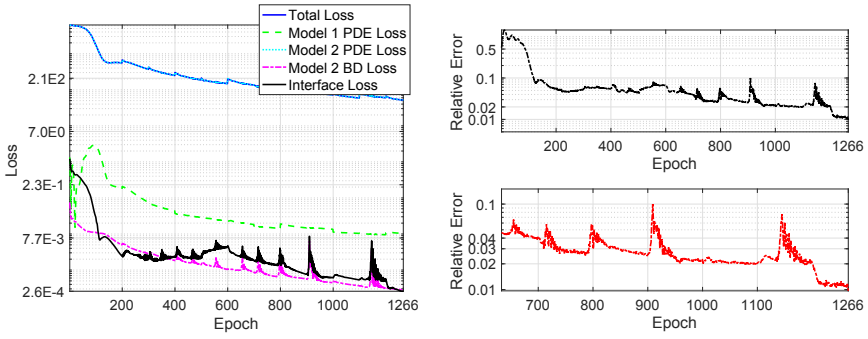


FIG. 11: Case 1. Convergence plots of loss functions and relative error for protein ID: 1b4l. (a) Convergences of errors (Training), and (b) relative error (Test).

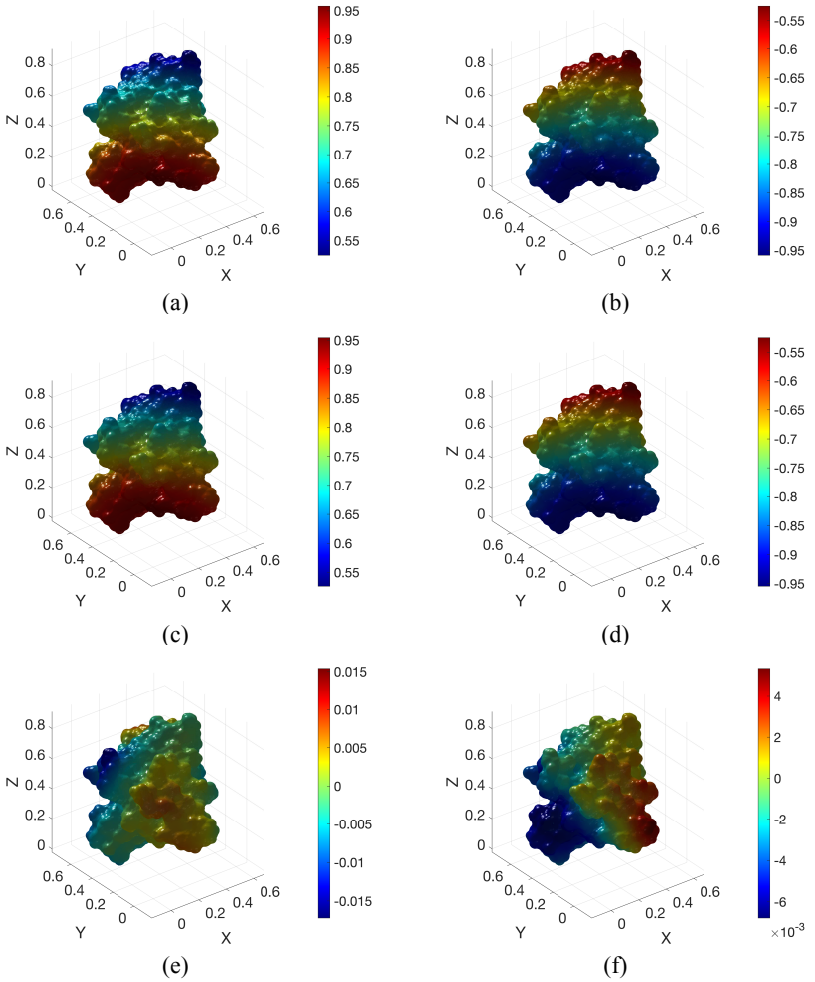


FIG. 12: Case 1. Plots for protein ID: 1b4l. (a) True solution $u_0^+(\Gamma)$, (b) true solution $u_0^-(\Gamma)$, (c) numerical solution $u_{0,\Theta}^+(\Gamma)$, (d) numerical solution $u_{0,\Theta}^-(\Gamma)$, (e) error on $u_{0,\Theta}^+(\Gamma)$, and (f) error on $u_{0,\Theta}^-(\Gamma)$.

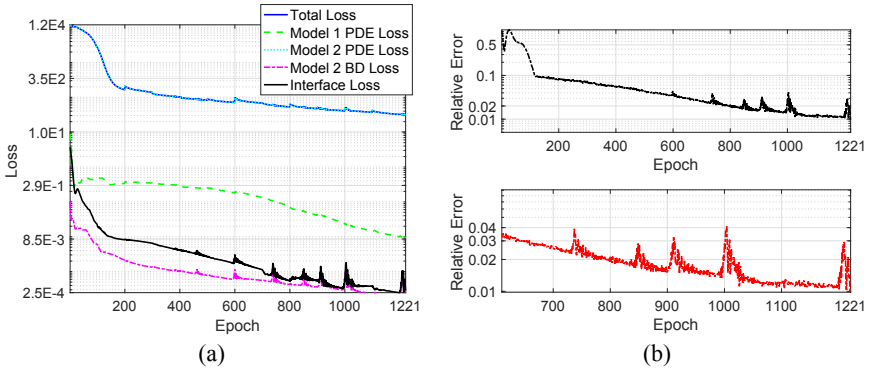


FIG. 13: Case 1. Convergence plots of loss functions and relative error for protein ID: 1bpi. (a) Convergences of errors (Training), and (b) relative error (Test).

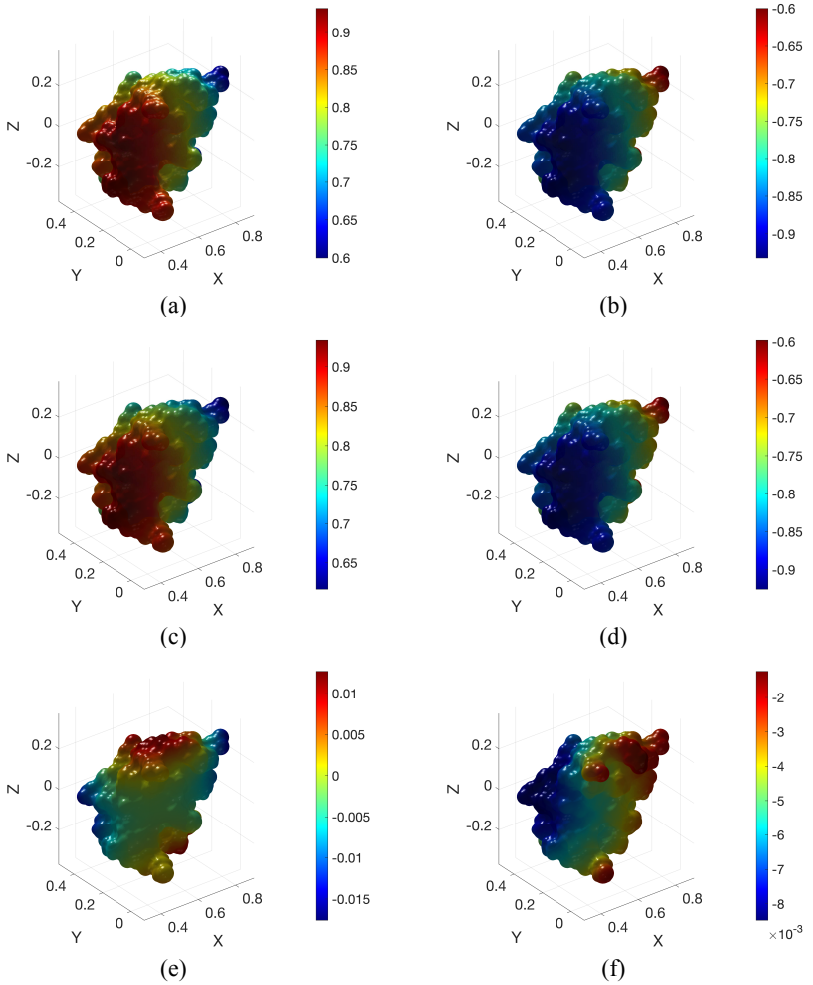


FIG. 14: Case 1. Plots for protein ID: 1bpi. (a) True solution $u_0^+(\Gamma)$, (b) true solution $u_0^-(\Gamma)$, (c) numerical solution $u_{0,\Theta}^+(\Gamma)$, (d) numerical solution $u_{0,\Theta}^-(\Gamma)$, (e) error on $u_{0,\Theta}^+(\Gamma)$, and (f) error on $u_{0,\Theta}^-(\Gamma)$.

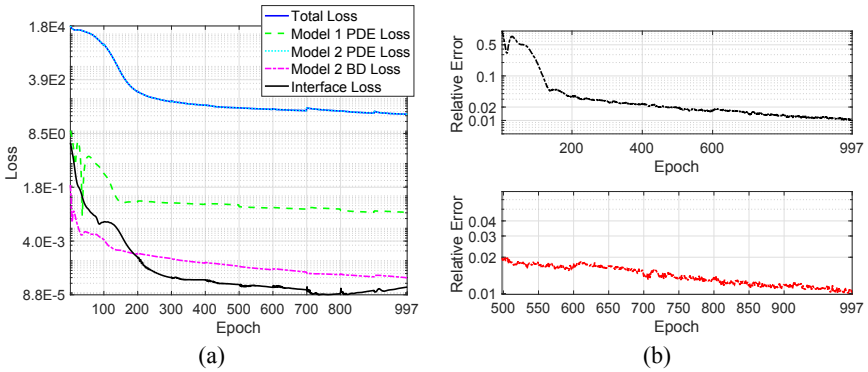


FIG. 15: Case 1. Convergence plots of loss functions and relative error for protein ID: 1bbl. (a) Convergences of errors (Training), and (b) relative error (Test).

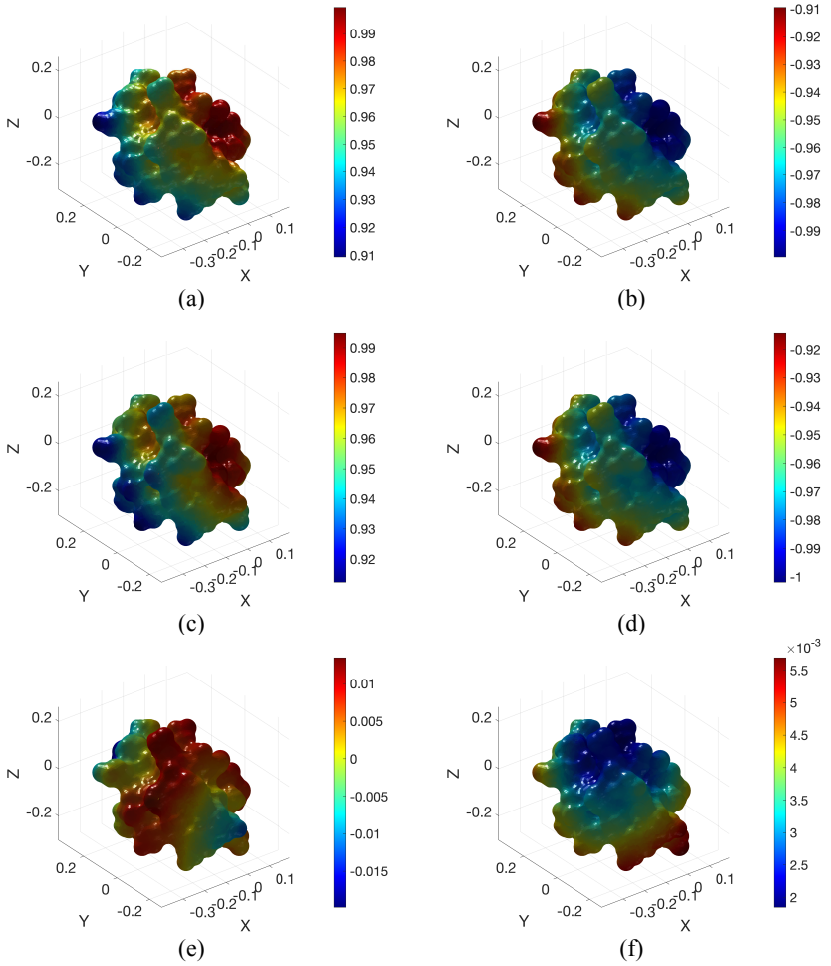


FIG. 16: Case 1. Plots for protein ID: 1bbl. (a) True solution $u_0^+(\Gamma)$, (b) true solution $u_0^-(\Gamma)$, (c) numerical solution $u_{0,\Theta}^+(\Gamma)$, (d) numerical solution $u_{0,\Theta}^-(\Gamma)$, (e) error on $u_{0,\Theta}^+(\Gamma)$, and (f) error on $u_{0,\Theta}^-(\Gamma)$.

TABLE 2: Case 1. Relative error for different protein surface

ID	Rel-Error on Γ	Rel-Error in Ω^-	Rel-Error in Ω^+	Rel-Error on $\partial\Omega$
1ajj	6.9458e–03	7.1164e–03	1.4358e–02	4.3751e–02
1a2s	9.2936e–03	6.2110e–03	2.2096e–02	1.8781e–01
1a7m	9.8668e–03	7.2240e–03	2.4976e–02	8.3051e–02
1a63	8.4063e–03	6.9132e–03	2.2616e–02	7.2910e–02
1b4l	5.7055e–03	5.6519e–03	2.5829e–02	9.8939e–02
1bpi	7.1478e–03	5.6880e–03	1.7951e–02	6.1308e–02
1bbl	5.6032e–03	5.8930e–03	1.8938e–02	6.9235e–02

the training dataset by increasing the number of sampling points. For the problem in Case 1, we use $N_I = 1000$, $N_b = 2400$, $N_i^- = N_+ = 500$; for Case 2, we set $N_I = 1500$, $N_b =$

3500, $N_i^- = N_+ = 500$; and for Case 3, we choose $N_I = 2500$, $N_b = 5000$, $N_i^- = 1000$, $N_+ = 500$. The stopping criteria for Case 2 is set the same as Case 1, i.e., either the relative error falling below 1% or reaching a maximum of 5000 epochs. For Case 3, the maximal epoch number is adjusted to 20,000.

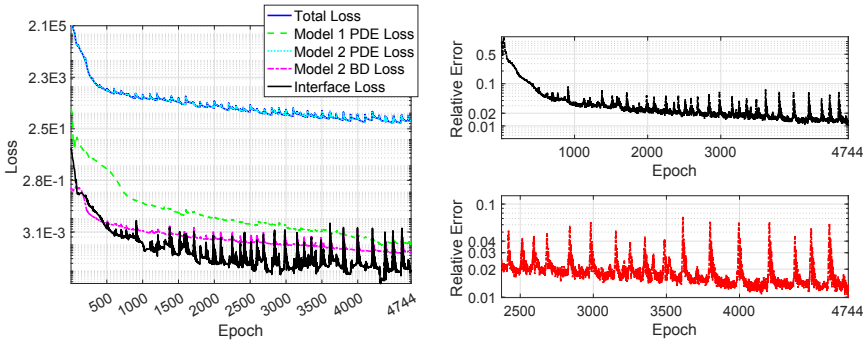
The convergence profiles for Cases 2 and 3 are shown in Figs. 17 and 18, respectively. As expected, more severe oscillations are observed as the frequency of the solutions increases. Nevertheless, the numerical solution is able to converge to the true solution with additional epochs.

The numerical solutions and errors on the protein surface are presented in Figs. 19 and 20. Once again, we observe strong agreement between the true solutions and the approximations, both inside and outside the interfaces.

Finally, the L^2 relative errors at the final epoch on the total dataset are summarized in Table 3.

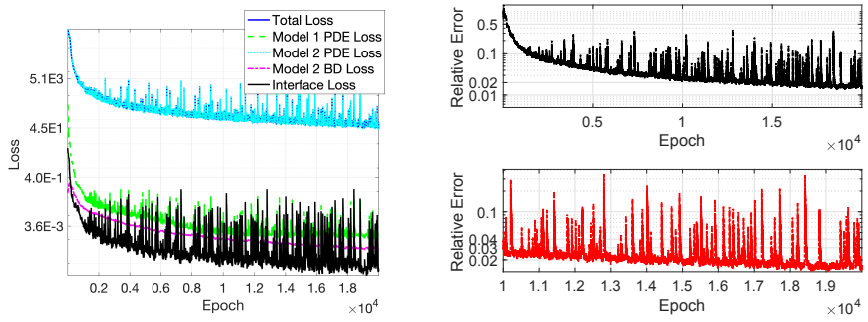
5. CONCLUSIONS

The PBE is a crucial model for studying electrostatics in molecular biology. Mathematically, it is a nonlinear elliptic PDE with discontinuous coefficients and delta singularities. Additionally, the PBE involves complex interface and interface conditions due to the multiscale modeling strategies employed for the protein-solvent system. With the advent of machine learning algorithms,



(a) (b)

FIG. 17: Case 2. Convergence plots of loss functions and the relative error for protein ID: 1ajj. (a) Convergences of errors (Training), and (b) relative error (Test).



(a) (b)

FIG. 18: Case 3. Convergence plot for protein ID: 1ajj. (a) Convergences of errors (Training), and (b) relative error (Test).

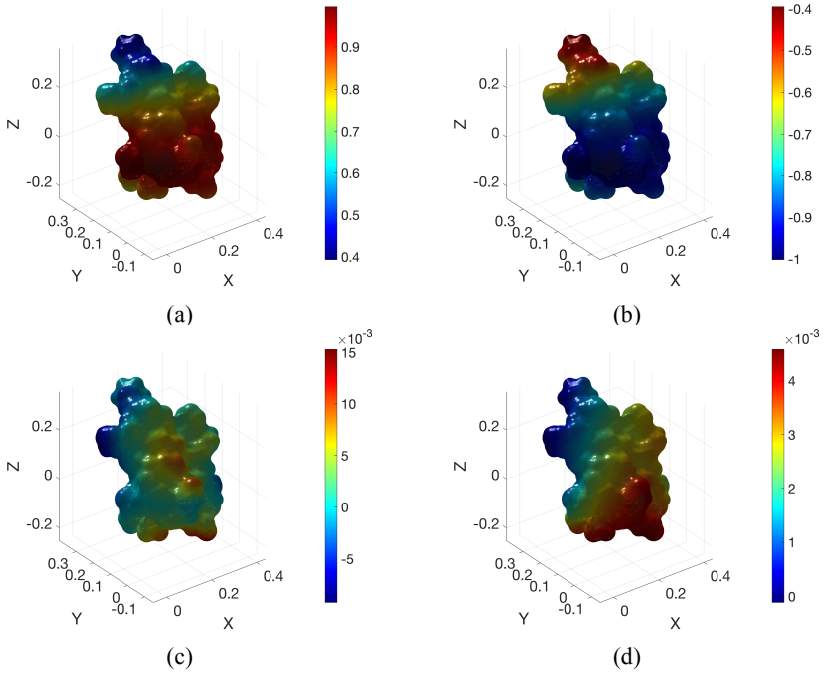


FIG. 19: Case 2. Convergence plots of loss functions and the relative error for protein ID: 1ajj. (a) Numerical solution $u^{+}_{0,\Theta}(\Gamma)$, (b) numerical solution $u^{-}_{0,\Theta}(\Gamma)$, (c) error on $u^{+}_{0,\Theta}(\Gamma)$, and (d) error on $u^{-}_{0,\Theta}(\Gamma)$.

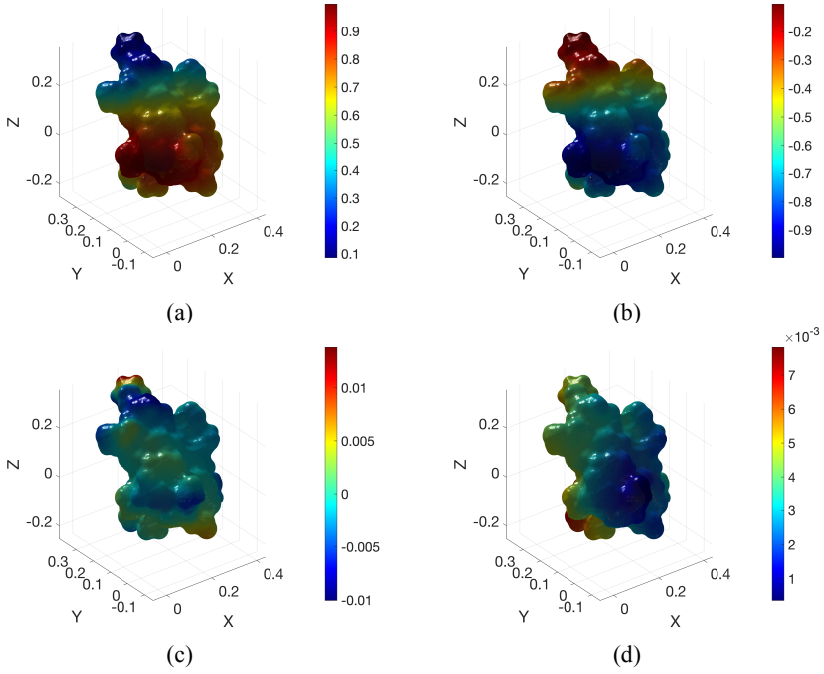


FIG. 20: Case 3. Plots for protein ID: 1ajj. (a) Numerical solution $u^{+}_{0,\Theta}(\Gamma)$, (b) numerical solution $u^{-}_{0,\Theta}(\Gamma)$, (c) error on $u^{+}_{0,\Theta}(\Gamma)$, and (d) error on $u^{-}_{0,\Theta}(\Gamma)$.

TABLE 3: Relative error for all tested cases on protein ID: 1ajj

	Epochs	Rel-Err on Γ	Rel-Err in Ω^-	Rel-Err in Ω^+	Rel-Err on $\partial\Omega^-$
Case 1	807	6.9458e–03	7.1164e–03	1.4358e–02	4.3751e–02
Case 2	4744	3.8763e–03	3.2960e–03	2.7126e–02	6.0950e–02
Case 3	20,000	6.1436e–03	3.3356e–03	2.9388e–02	6.8934e–02

the challenges of nonlinearity, singularity, and, most notably, the intricate geometry of the interface in solving the PBE numerically can be effectively addressed using neural network methods.

In this paper, we proposed a deep learning approach to compute numerical solutions for the PBE with biologically meaningful interfaces.

Future research can explore several directions. First, the fixed penalty constants for individual loss terms may be suboptimal; investigating adaptive weights for these losses could enhance training accuracy. Second, both the sampling and training strategies can be improved. Adaptive sampling, focusing on regions with higher loss while reducing sampling in regions with lower loss may increase training efficiency. For training, exploring optimizers like Levenberg–Marquardt could potentially yield better accuracy. Finally, as the spatial position of biological interfaces may depend on time in practice, extending the current framework to handle moving interface problems is a natural direction for future work.

ACKNOWLEDGMENTS

Duan Chen is partially supported by the National Institutes of Health under the Grant R01GM148971. Lin Mu is partially supported by the National Science Foundation under the Grant DMS-2309557.

REFERENCES

Baker, N.A., Sept, D., Joseph, S., Holst, M.J., and McCammon, J.A., Electrostatics of Nanosystems: Application to Microtubules and the Ribosome, *Proc. Natl. Acad. Sci. USA*, vol. **98**, no. 18, pp. 10037–10041, 2001.

Bond, S.D., Chaudhry, J.H., Cyr, E.C., and Olson, L.N., A First-Order System Least-Squares Finite Element Method for the Poisson–Boltzmann Equation, *J. Comput. Chem.*, vol. **31**, no. 8, pp. 1625–1635, 2010.

Borleske, G. and Zhou, Y., Enriched Gradient Recovery for Interface Solutions of the Poisson–Boltzmann Equation, *J. Comput. Phys.*, vol. **421**, p. 109725, 2020.

Boschitsch, A.H. and Fenley, M.O., Hybrid Boundary Element and Finite Difference Method for Solving the Nonlinear Poisson–Boltzmann Equation, *J. Comput. Chem.*, vol. **25**, no. 7, pp. 935–955, 2004.

Bosy, M., Scroggs, M.W., Betcke, T., Burman, E., and Cooper, C.D., Coupling Finite and Boundary Element Methods to Solve the Poisson–Boltzmann Equation for Electrostatics in Molecular Solvation, *J. Comput. Chem.*, vol. **45**, no. 11, pp. 787–797, 2024.

Briggs, J.M. and McCammon, J.A., Computation Unravels Mysteries of Molecular Biophysics: The Methods of Computational Biophysics Are Helping Us to Elucidate Biomolecular Mechanisms and to Design New Materials and Devices, *Comput. Phys.*, vol. **6**, no. 3, pp. 238–243, 1992.

Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S.A., and Karplus, M., CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, *J. Comput. Chem.*, vol. **4**, no. 2, pp. 187–217, 1983.

Cai, Z., Chen, J., Liu, M., and Liu, X., Deep Least-Squares Methods: An Unsupervised Learning-Based Numerical Method for Solving Elliptic PDEs, *J. Comput. Phys.*, vol. **420**, p. 109707, 2020.

Chen, D., A New Poisson-Nernst-Planck Model with Ion Water Interactions for Charge Transport in Ion Channels, *Bull. Math. Biol.*, vol. **78**, pp. 1703–1726, 2016.

Chen, D., Chen, Z., Chen, C., Geng, W., and Wei, G.W., MIBPB: A Software Package for Electrostatic Analysis, *J. Comput. Chem.*, vol. **32**, no. 4, pp. 756–770, 2011.

Chen, D. and Wei, G.W., Modeling and Simulation of Electronic Structure, Material Interface and Random Doping in Nano-Electronic Devices, *J. Comput. Phys.*, vol. **229**, pp. 4431–4460, 2010.

Chen, J., Xu, Y., Yang, X., Cang, Z., Geng, W., and Wei, G.W., Poisson–Boltzmann-Based Machine Learning Model for Electrostatic Analysis, *Biophys. J.*, vol. **123**, pp. 2807–2814, 2024.

Chen, L., Holst, M.J., and Xu, J., The Finite Element Approximation of the Nonlinear Poisson–Boltzmann Equation, *SIAM J. Numer. Anal.*, vol. **45**, no. 6, pp. 2298–2320, 2007.

Dissanayake, M.G. and Phan-Thien, N., Neural-Network-Based Approximations for Solving Partial Differential Equations, *Commun. Numer. Methods Eng.*, vol. **10**, no. 3, pp. 195–201, 1994.

Dwivedi, V., Parashar, N., and Srinivasan, B., Distributed Physics Informed Neural Network for Data-Efficient Solution to Partial Differential Equations, arXiv preprint arXiv:1907.08967, 2019.

Engels, M., Gerwert, K., and Bashford, D., Computational Studies of the Early Intermediates of the Bacteriorhodopsin Photocycle, *Biophys. Chem.*, vol. **56**, nos. 1-2, pp. 95–104, 1995.

Fogolari, F., Brigo, A., and Molinari, H., The Poisson–Boltzmann Equation for Biomolecular Electrostatics: A Tool for Structural Biology, *J. Mol. Recognit.*, vol. **15**, no. 6, pp. 377–392, 2002.

Geng, W. and Krasny, R., A Treecode-Accelerated Boundary Integral Poisson–Boltzmann Solver for Electrostatics of Solvated Biomolecules, *J. Comput. Phys.*, vol. **247**, pp. 62–78, 2013.

Geng, W., Yu, S., and Wei, G., Treatment of Charge Singularities in Implicit Solvent Models, *J. Chem. Phys.*, vol. **127**, no. 11, p. 114106, 2007.

He, C., Hu, X., and Mu, L., A Mesh-Free Method Using Piecewise Deep Neural Network for Elliptic Interface Problems, *J. Comput. Appl. Math.*, vol. **412**, p. 114358, 2022.

He, K., Zhang, X., Ren, S., and Sun, J., Deep Residual Learning for Image Recognition, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, pp. 770–778, 2016.

Holst, M., Baker, N., and Wang, F., Adaptive Multilevel Finite Element Solution of the Poisson–Boltzmann Equation I. Algorithms and Examples, *J. Comput. Chem.*, vol. **21**, no. 15, pp. 1319–1342, 2000.

Honig, B. and Nicholls, A., Classical Electrostatics in Biology and Chemistry, *Science*, vol. **268**, no. 5214, pp. 1144–1149, 1995.

Im, W., Beglov, D., and Roux, B., Continuum Solvation Model: Computation of Electrostatic Forces from Numerical Solutions to the Poisson–Boltzmann Equation, *Comput. Phys. Commun.*, vol. **111**, nos. 1-3, pp. 59–75, 1998.

Jo, S., Vargyas, M., Vasko-Szedlar, J., Roux, B., and Im, W., PBEQ-Solver for Online Visualization of Electrostatic Potential of Biomolecules, *Nucleic Acids Res.*, vol. **36**, no. suppl 2, pp. W270–W275, 2008.

Kharazmi, E., Zhang, Z., and Karniadakis, G.E., Hp-VPINNs: Variational Physics-Informed Neural Networks with Domain Decomposition, *Comput. Methods Appl. Mech. Eng.*, vol. **374**, p. 113547, 2021.

Klapper, I., Hagstrom, R., Fine, R., Sharp, K., and Honig, B., Focusing of Electric Fields in the Active Site of Cu–Zn Superoxide Dismutase: Effects of Ionic Strength and Amino-Acid Modification, *Proteins: Struct. Funct. Bioinf.*, vol. **1**, no. 1, pp. 47–59, 1986.

Kwon, I., Jo, G., and Shin, K.S., A Deep Neural Network Based on ResNet for Predicting Solutions of Poisson–Boltzmann Equation, *Electronics*, vol. **10**, no. 21, p. 2627, 2021a.

Kwon, I., Kwak, D.Y., and Jo, G., Discontinuous Bubble Immersed Finite Element Method for Poisson–Boltzmann–Nernst–Planck Model, *J. Comput. Phys.*, vol. **438**, p. 110370, 2021b.

Lee, A., Geng, W., and Zhao, S., Regularization Methods for the Poisson–Boltzmann Equation: Comparison and Accuracy Recovery, *J. Comput. Phys.*, vol. **426**, p. 109958, 2021.

LeVeque, R.J. and Li, Z., The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources, *SIAM J. Numer. Anal.*, vol. **31**, no. 4, pp. 1019–1044, 1994.

Li, Z. and Ito, K., Maximum Principle Preserving Schemes for Interface Problems with Discontinuous Coefficients, *SIAM J. Sci. Comput.*, vol. **23**, no. 1, pp. 339–361, 2001.

Liu, Z., Cai, W., and Xu, Z.Q.J., Multi-Scale Deep Neural Network (MscaleDNN) for Solving Poisson–Boltzmann Equation in Complex Domains, arXiv preprint arXiv:2007.11207, 2020.

Lu, B.Z., Chen, W.Z., Wang, C.X., and Xu, X.J., Protein Molecular Dynamics with Electrostatic Force Entirely Determined by a Single Poisson–Boltzmann Calculation, *Proteins: Struct. Funct. Bioinf.*, vol. **48**, no. 3, pp. 497–504, 2002.

Luo, R., David, L., and Gilson, M.K., Accelerated Poisson–Boltzmann Calculations for Static and Dynamic Systems, *J. Comput. Chem.*, vol. **23**, no. 13, pp. 1244–1253, 2002.

Madura, J.D., Briggs, J.M., Wade, R.C., Davis, M.E., Luty, B.A., Ilin, A., Antosiewicz, J., Gilson, M.K., Bagheri, B., and Scott, L.R., Electrostatics and Diffusion of Molecules in Solution: Simulations with the University of Houston Brownian Dynamics Program, *Comput. Phys. Commun.*, vol. **91**, nos. 1-3, pp. 57–95, 1995.

Oliva, P.V., Wu, Y., He, C., and Ni, H., Towards Fast Weak Adversarial Training to Solve High Dimensional Parabolic Partial Differential Equations Using XNODE-WAN, *J. Comput. Phys.*, vol. **463**, p. 111233, 2022.

Prabhu, N.V., Panda, M., Yang, Q., and Sharp, K.A., Explicit Ion, Implicit Water Solvation for Molecular Dynamics of Nucleic Acids and Highly Charged Molecules, *J. Comput. Chem.*, vol. **29**, no. 7, pp. 1113–1130, 2008.

Raissi, M., Perdikaris, P., and Karniadakis, G.E., Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations, *J. Comput. Phys.*, vol. **378**, pp. 686–707, 2019.

Ramm, V., Chaudhry, J.H., and Cooper, C.D., Efficient Mesh Refinement for the Poisson–Boltzmann Equation with Boundary Elements, *J. Comput. Chem.*, vol. **42**, no. 12, pp. 855–869, 2021.

Sharp, K.A. and Honi, B., Interactions in Macromolecules: Theory and Applications, *Annu. Rev. Biophys. Biophys. Chem.*, vol. **19**, no. 301, p. 32, 1990.

Sirignano, J. and Spiliopoulos, K., DGM: A Deep Learning Algorithm for Solving Partial Differential Equations, *J. Comput. Phys.*, vol. **375**, pp. 1339–1364, 2018.

Wang, Z. and Zhang, Z., A Mesh-Free Method for Interface Problems Using the Deep Learning Approach, *J. Comput. Phys.*, vol. **400**, p. 108963, 2020.

Warwicker, J. and Watson, H., Calculation of the Electric Potential in the Active Site Cleft Due to α-Helix Dipoles, *J. Mol. Biol.*, vol. **157**, no. 4, pp. 671–679, 1982.

Wu, S. and Lu, B., INN: Interfaced Neural Networks as an Accessible Meshless Approach for Solving Interface PDE Problems, *J. Comput. Phys.*, vol. **470**, p. 111588, 2022.

Xie, D., An Efficient Finite Element Solver for a Nonuniform Size-Modified Poisson-Nernst-Planck Ion Channel Model, arXiv preprint arXiv:2405.01619, 2024.

Yu, B., The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems, *Commun. Math. Stat.*, vol. **6**, no. 1, pp. 1–12, 2018.

Yu, S., Geng, W., and Wei, G.W., Treatment of Geometric Singularities in Implicit Solvent Models, *J. Chem. Phys.*, vol. **126**, no. 24, 2007.

Yu, S.N. and Wei, G.W., Three-Dimensional Matched Interface and Boundary (MIB) Method for Treating Geometric Singularities, *J. Comput. Phys.*, vol. **227**, pp. 602–632, 2007.

Zang, Y., Bao, G., Ye, X., and Zhou, H., Weak Adversarial Networks for High-Dimensional Partial Differential Equations, *J. Comput. Phys.*, vol. **411**, p. 109409, 2020.

Zauhar, R. and Morgan, R., A New Method for Computing the Macromolecular Electric Potential, *J. Mol. Biol.*, vol. **186**, no. 4, pp. 815–820, 1985.

Zhou, Y., Feig, M., and Wei, G.W., Highly Accurate Biomolecular Electrostatics in Continuum Dielectric Environments, *J. Comput. Chem.*, vol. **29**, no. 1, pp. 87–97, 2008.

Zhou, Y., Zhao, S., Feig, M., and Wei, G.W., High Order Matched Interface and Boundary Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources, *J. Comput. Phys.*, vol. **213**, no. 1, pp. 1–30, 2006.