Received: 10 May 2023 | Revised: 3 October 2023 | Accepted: 5 November 2023

DOI: 10.1002/jcc.27262

RESEARCH ARTICLE

Journal of COMPUTATIONAL CHEMISTRY WILEY

Coupling finite and boundary element methods to solve the Poisson–Boltzmann equation for electrostatics in molecular solvation

Michał Bosy[1] | Matthew W. Scroggs[2] | Timo Betcke[2] | Erik Burman[2] | Christopher D. Cooper[3]

[1]School of Computer Science and Mathematics, Kingston University London, Kingston upon Thames, UK

[2]Department of Mathematics, University College London, London, UK

[3]Department of Mechanical Engineering and Centro Científico Tecnológico de Valparaíso, Universidad Técnica Federico Santa María, Valparaíso, Chile

Correspondence
Christopher D. Cooper, Department of Mechanical Engineering and Centro Científico Tecnológico de Valparaíso, Universidad Técnica Federico Santa María, Valparaíso 2390123, Chile.
Email: christopher.cooper@usm.cl

Abstract

The Poisson–Boltzmann equation is widely used to model electrostatics in molecular systems. Available software packages solve it using finite difference, finite element, and boundary element methods, where the latter is attractive due to the accurate representation of the molecular surface and partial charges, and exact enforcement of the boundary conditions at infinity. However, the boundary element method is limited to linear equations and piecewise constant variations of the material properties. In this work, we present a scheme that couples finite and boundary elements for the linearised Poisson–Boltzmann equation, where the finite element method is applied in a confined *solute* region and the boundary element method in the external *solvent* region. As a proof-of-concept exercise, we use the simplest methods available: Johnson–Nédélec coupling with mass matrix and diagonal preconditioning, implemented using the Bempp-cl and FEniCSx libraries via their Python interfaces. We showcase our implementation by computing the polar component of the solvation free energy of a set of molecules using a constant and a Gaussian-varying permittivity. As validation, we compare against well-established finite difference solvers for an extensive binding energy data set, and with the finite difference code APBS (to 0.5%) for Gaussian permittivities. We also show scaling results from protein G B1 (955 atoms) up to immunoglobulin G (20,148 atoms). For small problems, the coupled method was efficient, outperforming a purely boundary integral approach. For Gaussian-varying permittivities, which are beyond the applicability of boundary elements alone, we were able to run medium to large-sized problems on a single workstation. The development of better preconditioning techniques and the use of distributed memory parallelism for larger systems remains an area for future work. We hope this work will serve as inspiration for future developments that consider space-varying field parameters, and mixed linear-nonlinear schemes for molecular electrostatics with implicit solvent models.

KEYWORDS

boundary element method, electrostatics, finite element method, implicit solvent model, Poisson–Boltzmann

1 | INTRODUCTION

In biologically relevant settings, the structure and function of biomolecules are largely determined by the surrounding water, which usually contains salt. To describe these systems accurately, we need to account for the solvent correctly, which has given rise to a wide range of models.[1] Highly detailed models consider every water molecule and salt ion explicitly. For solutions with large numbers of molecules, however, these models can be very computationally expensive, so implicit-solvent models—approximated models that use continuum theory to represent the ionic solution—are often used instead.[2,3] In the case of electrostatics, the implicit-solvent model is mathematically characterized by the Poisson–Boltzmann equation (PBE),[4,5] which is widely used to compute solvation free energies and mean-field potentials.

The implicit-solvent model for electrostatics describes the dissolved molecule as an infinite medium with a low-dielectric solute-shaped cavity, which contains a charge distribution from the partial charges—usually a sum of Dirac deltas at the atom's locations. The outer solvent region is represented with a high dielectric constant and considers the presence of salt. These two regions are interfaced by the molecular surface where the continuity of the electrostatic potential and electric displacement are enforced. The molecular surface can be defined in various ways.[6]

The PBE has been solved numerically with finite difference,[7–10] finite element,[11–13] boundary element,[14–19] and analytic[20–24] methods. In particular, the boundary element method (BEM) has proven to be very efficient for high-accuracy calculations,[18,19] mainly due to the precise description of the molecular surface and point charges. However, BEM is limited to constant material properties in each region and the linear version of the PBE. Even though these limitations are acceptable in a wide range of applications, there are cases when BEM falls short: for example, if a variable permittivity is required inside the solute,[25,26] or the solute is highly charged such that the linear approximation breaks.[27]

The present article describes a methodology to overcome some of those limitations, by coupling finite and boundary element methods. This approach brings the best of both worlds—the flexibility of FEM and the efficiency of BEM—all in an accurate description of the solute molecule. FEM-BEM coupling is a popular technique in the context of mixed linear-nonlinear models,[28,29] fracture mechanics,[30] fluid-structure interaction,[31] acoustics,[32] and electromagnetics.[33–35] On the other hand, the PBE has been solved with hybrid numerical methods in the past: for example, by coupling finite differences with boundary elements[36] or finite elements[37,38] to solve the nonlinear PBE in a specific region, and to implement modifications to the PBE model (i.e., the size-modified PBE). To the best of our knowledge, this is the first time finite and boundary elements have been combined in this application. We are proposing a coupling scheme for the linear Poisson-Boltzmann equation.

In this paper, we prototype this principle with the simplest implementation possible: a Johnson–Nédélec[39] coupling, where we solve using mass-matrix and diagonal preconditioning, and without distributed memory execution. This limits the size of problems we can access currently, however, it sets the basis for future developments that use more elaborate formulations and algorithms that are readily available in open-source software libraries. In this work, we use the boundary element library Bempp-cl[40] and the finite element library FEniCSx.[41,42] These libraries are easy to use and their full functionalities may be accessed via their Python interfaces, making them ideal tools for easily implementing problems like this as well as for exploring computational efficiency and moving towards tackling large-scale problems, such as a full viral capsid.[43,44] We hope this work will inspire research along these lines.

2 | METHODOLOGY

2.1 | The implicit solvent model

The implicit solvent model[2,3] can be described mathematically as a coupled system of partial differential equations, where the Poisson equation governs in the solvent region (Ω_1 in Figure 1), and the Poisson–Boltzmann equation governs in the solute region (Ω_2 in Figure 1). These regions are interfaced by the molecular surface (Γ), where the potential (ϕ) and electric displacement ($\epsilon\partial\phi/\partial\mathbf{n}$) are continuous across the surface. The linearised Poisson-Boltzmann equation that results from approximating $\sinh(x)$ with x follows

$$\nabla^2\phi_1(\mathbf{x}) = \frac{1}{\epsilon_1}\sum_{k=1}^{N_q} q_k\delta(\mathbf{x},\mathbf{x}_k) \quad \mathbf{x}\in\Omega_1, \tag{1a}$$

$$(\nabla^2-\kappa^2)\phi_2(\mathbf{x}) = 0 \quad \mathbf{x}\in\Omega_2, \tag{1b}$$

$$\phi_1(\mathbf{x}) = \phi_2(\mathbf{x}), \quad \mathbf{x}\in\Gamma, \tag{1c}$$

$$\epsilon_1\frac{\partial\phi_1}{\partial\mathbf{n}}(\mathbf{x}) = \epsilon_2\frac{\partial\phi_2}{\partial\mathbf{n}}(\mathbf{x}) \quad \mathbf{x}\in\Gamma. \tag{1d}$$

Here, ϵ_1 and ϵ_2 are the dielectric constants in the solute and solvent, respectively; κ is the inverse of the Debye length, related to the salt concentration; and q_k are the values of the partial charges, located at $\mathbf{x}_k$.

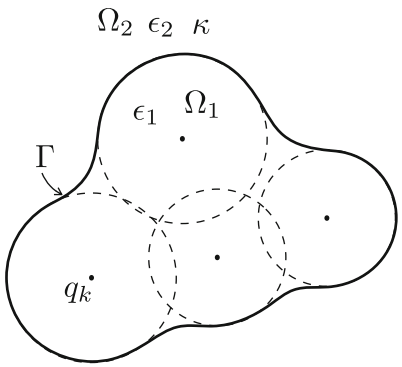

FIGURE 1 Representation of a molecule in an implicit solvent.

The electrostatic potential in Ω_1 can be further decomposed into singular and regular components as $\phi_1 = \phi_c + \phi_r$, where ϕ_c is the solution to

$$\begin{aligned} \nabla^2 \phi_c(\mathbf{x}) &= \frac{1}{\epsilon} \sum_{k=1}^{N_q} q_k \delta(\mathbf{x}, \mathbf{x}_k) \quad && \mathbf{x} \in \Omega_1 \cup \Omega_2, \\ \phi_c(\mathbf{x}) &= 0 && \text{as } |\mathbf{x}| \to \infty. \end{aligned} \tag{2}$$

Physically, ϕ_c can be interpreted as the Coulomb-type potential from the point charges, whereas ϕ_r, also known as reaction potential, is originated by the polarization of the solvent and reorganization of the free ions. Usually, ϵ_1 is considered a constant value, yielding an analytic expression for ϕ_c. However, this is not the general case.

Regularized versions of the equations in (1a–d)[45,46] are widely used to numerically solve the Poisson–Boltzmann equation with finite element or finite difference methods, and have recently been extended to super-Gaussian permittivities in the solute.[47] However, here we use the standard formulation in (1a–d), as it offers more flexibility when dealing with, for example, variable permittivities, beyond super-Gaussian descriptions.

A common quantity of interest in implicit solvent models is the polar component of the solvation free energy, which is the change in Gibbs free energy as the molecule charges move from vacuum into a solute-shaped cavity inside the solvent. Considering the charge distribution ρ consists of point charges, this can be calculated as

$$\Delta G_{\text{solv}} = \frac{1}{2} \int_{\Omega_1} \rho(\mathbf{x}) \phi_r(\mathbf{x}) = \frac{1}{2} \sum_{k=1}^{N_q} q_k \phi_r(\mathbf{x}_k). \tag{3}$$

2.2 | BEM-BEM coupling

Several possible boundary integral formulations of (1a–d) exist. [48] The simplest form was presented by Yoon and Lenhoff in 1991[49] and is known as the *direct formulation*. However, the resulting system is usually ill-conditioned. Here, we use the better-conditioned formulation presented by Lu and co-workers, [15,50,51] which reads

$$\begin{aligned} &\frac{\phi_2}{2}\left(1 + \frac{\epsilon_1}{\epsilon_2}\right) - \left(K_Y^\Gamma - \frac{\epsilon_1}{\epsilon_2} K_L^\Gamma\right)\phi_2 + \left(V_Y^\Gamma - V_L^\Gamma\right)\lambda_2 \\ &\quad = \sum_{k=0}^{N_q} \frac{q_k}{4\pi\epsilon_2 |\mathbf{x}_\Gamma - \mathbf{x}_k|}, \\ &\frac{\epsilon_1}{\epsilon_2}\left(W_Y^\Gamma - W_L^\Gamma\right)\phi_2 + \frac{\lambda_2}{2}\left(1 + \frac{\epsilon_1}{\epsilon_2}\right) + \left(\frac{\epsilon_1}{\epsilon_2} K_Y'^\Gamma - K_L'^\Gamma\right)\lambda_2 \\ &\quad = \sum_{k=0}^{N_q} \frac{\partial}{\partial \mathbf{n}_\mathbf{x}} \left(\frac{q_k}{4\pi\epsilon_2 |\mathbf{x}_\Gamma - \mathbf{x}_k|}\right), \end{aligned} \tag{4}$$

where $\phi_2 = \phi_2(\mathbf{x}_\Gamma)$ is the potential on Γ as we approach from Ω_2 (i.e., the exterior Dirichlet trace), and $\lambda_2 = \frac{\partial}{\partial \mathbf{n}} \phi_2$ is the normal derivative Γ as we approach from Ω_2 (i.e., the exterior Neumann trace). Note that we can obtain ϕ_1 and λ_1 from ϕ_2 and λ_2 by applying the interface conditions (1c) and (1d). The operators V, K, K', and W that appear in Equation (4) are the single-layer, double-layer, adjoint double-layer, and hypersingular operators, respectively, for the Laplace (subscript L) and Yukawa (subscript Y, also known as modified Helmholtz) kernels. These are defined by

$$\begin{aligned} V_i^\Gamma \varphi(\mathbf{x}) &= \oint_\Gamma g_i(\mathbf{x}, \mathbf{x}') \varphi(\mathbf{x}') d\mathbf{x}', \\ K_i^\Gamma \varphi(\mathbf{x}) &= \oint_\Gamma \frac{\partial g_i}{\partial \mathbf{n}'}(\mathbf{x}, \mathbf{x}') \varphi(\mathbf{x}') d\mathbf{x}', \\ K_i'^\Gamma \varphi(\mathbf{x}) &= \oint_\Gamma \frac{g_i}{\partial \mathbf{n}}(\mathbf{x}, \mathbf{x}') \varphi(\mathbf{x}') d\mathbf{x}', \\ W_i^\Gamma \varphi(\mathbf{x}) &= -\oint_\Gamma \frac{\partial^2 g_i}{\partial \mathbf{n}' \partial \mathbf{n}}(\mathbf{x}, \mathbf{x}') \varphi(\mathbf{x}') d\mathbf{x}', \end{aligned} \tag{5}$$

where $i \in \{L, Y\}$, $\varphi(\mathbf{x})$ can be any distribution over Γ, and

$$\begin{aligned} g_L(\mathbf{x}, \mathbf{x}') &= \frac{1}{4\pi |\mathbf{x} - \mathbf{x}'|} \\ g_Y(\mathbf{x}, \mathbf{x}') &= \frac{e^{-\kappa |\mathbf{x} - \mathbf{x}'|}}{4\pi |\mathbf{x} - \mathbf{x}'|} \end{aligned} \tag{6}$$

are the corresponding free-space Green's functions.

Having computed the electrostatic potential with Equation (4), we obtain the reaction potential (ϕ_r) in Equation (3) by subtracting the Coulombic component from ϕ_1. This gives[19]

$$\phi_r(\mathbf{r}) = -K_L^\Gamma \phi_1(\mathbf{r}) + V_L^\Gamma \lambda_1(\mathbf{r}), \tag{7}$$

where $\mathbf{r} \in \Omega_1$. This is valid as long as ϵ_1 is a constant, so we use Equation (7) for both BEM-BEM and FEM-BEM coupling simulations with constant ϵ_1.

2.3 | Novel numerical solution of the Poisson–Boltzmann equation

The boundary element method (BEM) is a standard tool for the numerical solution of the linear version of the Poisson–Boltzmann equation in molecular electrostatics. [52,53] This was implemented in numerous codes, such as AFMPB, [15] TABI, [18] PyGBe, [19,54] and more recently, with Bempp-cl.[48] One strong advantage of BEM is that it reduces the dimension of the problem by using the boundary integral formulation. Unfortunately, this advantage comes at a cost: it requires a fundamental solution to be known in order for the method to be applied. There are many linear problems for which fundamental solutions are not known: this is the case for the Poisson-Boltzmann equation with a heterogeneous permittivity inside the molecule, that is,

$$\begin{aligned} \nabla \cdot (\epsilon_1(\mathbf{x}) \nabla \phi_1(\mathbf{x})) &= \sum_{k=1}^{N_q} q_k \delta(\mathbf{x}, \mathbf{x}_k) \quad && \mathbf{x} \in \Omega_1, \\ \left(\nabla^2 - \kappa^2\right) \phi_2(\mathbf{x}) &= 0 && \mathbf{x} \in \Omega_2, \\ \phi_1(\mathbf{x}) &= \phi_2(\mathbf{x}), && \mathbf{x} \in \Gamma, \\ \epsilon_1(\mathbf{x}) \frac{\partial \phi_1}{\partial \mathbf{n}}(\mathbf{x}) &= \epsilon_2 \frac{\partial \phi_2}{\partial \mathbf{n}}(\mathbf{x}) && \mathbf{x} \in \Gamma. \end{aligned} \tag{8}$$

A finite element method (FEM) is more suitable for such a case.

The coupling of FEM and BEM is an approach that can be used to solve a wide range of multiphysics problems on unbounded domains,[55,56] as it takes advantage of both methods. On the one hand, the BEM satisfies the infinite boundary conditions exactly when they decay to zero, and only approximates boundary conditions on surfaces; hence, it is commonly used for problems involving infinite or semi-infinite domains. On the other hand, the FEM is known for its robustness and universal applicability, even for problems of inhomogeneous or non-linear nature. Here, we use the Johnson–Nédélec formulation,[39] which is the simplest formulation of FEM-BEM coupling, and we detail it next.

We start with the variational formulation of the internal problem. Applying integration by parts to the first equation of (8) we have, for every $v \in H^1_0(\Omega_1)$,

$$\langle \epsilon_1 \nabla \phi_1, \nabla v \rangle_{\Omega_1} - \langle \epsilon_1 \partial_n \phi_1, v \rangle_\Gamma = \left\langle \sum_{k=1}^{N_q} q_k \delta(\mathbf{x}, \mathbf{x}_k), v \right\rangle_{\Omega_1}, \tag{9}$$

where v is a test function, and $\langle \varphi, v \rangle_\Gamma = \int_\Gamma \varphi(\mathbf{x}) v(\mathbf{x})\, d\mathbf{x}$ and $\langle \varphi, v \rangle_{\Omega_1} = \int_{\Omega_1} \varphi(\mathbf{x}) v(\mathbf{x})\, d\mathbf{x}$ are the inner products on the surface and in the domain, respectively. Slightly abusing notation, the product $\langle \cdot, \cdot \rangle_{\Omega_1}$ on the right-hand side denotes the duality pairing. For the external problem with BEM, we define the Dirichlet trace [57]

$$\gamma : H^1(\Omega_2) \to H^{\frac{1}{2}}(\Gamma), \quad \gamma f(x) := \lim_{\Omega_2 \ni \mathbf{y} \to \mathbf{x} \in \Gamma} f(\mathbf{y}),$$

and we use the direct formulation from the second equation of (1b) to obtain

$$\frac{\phi_1}{2} - K^\Gamma_Y \gamma \phi_1 + \frac{\epsilon_1}{\epsilon_2} V^\Gamma_Y \lambda_1 = 0.$$

V and K are the single-layer and double-layer operators as defined in Equation (5).

Then, the coupling problem can be written as: *Find* $\phi_1 \in H^1(\Omega_1)$ *and* $\lambda_1 \in H^{-\frac{1}{2}}(\Gamma)$ *such that for all* $v \in H^1(\Omega_1)$ *and* $\zeta \in H^{-\frac{1}{2}}(\Gamma)$,

$$\langle \epsilon_1 \nabla \phi_1, \nabla v \rangle_{\Omega_1} - \langle \epsilon_1 \lambda_1, v \rangle_\Gamma = \left\langle \sum_{k=1}^{N_q} q_k \delta(\mathbf{x}, \mathbf{x}_k), v \right\rangle_{\Omega_1}, \tag{10a}$$

$$\left\langle \left(\tfrac{1}{2} I - K^\Gamma_Y\right) \gamma \phi_1, \zeta \right\rangle_\Gamma + \frac{1}{\epsilon_2} \left\langle V^\Gamma_Y \epsilon_1 \lambda_1, \zeta \right\rangle_\Gamma = 0. \tag{10b}$$

Note that in the case where ϵ_1 is constant, then each ϵ_1 can be moved outside of the inner product it is inside.

When discretized, this can also be written in matrix form. Let $\vec{\phi}_1 := [\phi^1_1, \ldots, \phi^j_1]^T$ be the vector of canonical basis functions of the finite element space $V^j_h \subset H^1(\Omega_1)$, and let $\vec{\lambda}_1 := [\lambda^1_1, \ldots, \lambda^l_1]^T$ be the vector of canonical basis functions of $\Lambda^l_h \subset H^{-\frac{1}{2}}(\Gamma)$. We define the following matrices associated with the corresponding bilinear forms

$$\begin{aligned} A_{\alpha\beta} &= \left\langle \epsilon_1 \nabla \phi^\alpha_1, \nabla \phi^\beta_1 \right\rangle_{\Omega_1}, & \tilde{M}_{\alpha\beta} &= \left\langle \epsilon_1 \lambda^\beta_1, \gamma \phi^\alpha_1 \right\rangle_\Gamma, \\ K_{\alpha\beta} &= \left\langle K^\Gamma_Y \gamma \phi^\alpha_1, \lambda^\beta_1 \right\rangle_\Gamma, & V_{\alpha\beta} &= \left\langle V^\Gamma_Y \epsilon_1 \lambda^\alpha_1, \lambda^\beta_1 \right\rangle_\Gamma, \\ M_{\alpha\beta} &= \left\langle \gamma \phi^\alpha_1, \lambda^\beta_1 \right\rangle_\Gamma, \end{aligned}$$

and vector associated with the corresponding linear form

$$\vec{f}_\beta := \left\langle \sum_{k=1}^{N_q} q_k \delta(\mathbf{x}, \mathbf{x}_k), \phi^\beta_1 \right\rangle_{\Omega_1}.$$

Using the above definitions, the discrete problem in (10a,b) can be written in the following blocked matrix form:

$$\begin{bmatrix} A & -\tilde{M}^T \\ \frac{1}{2}M - K & \frac{1}{\epsilon_2} V \end{bmatrix} \begin{bmatrix} \vec{\phi}_1 \\ \vec{\lambda}_1 \end{bmatrix} = \begin{bmatrix} \vec{f} \\ 0 \end{bmatrix}. \tag{11}$$

3 | RESULTS AND DISCUSSION

This section presents the verification and performance results of the presented FEM-BEM coupling scheme for molecules modelled as cavities with constant and varying permittivity. With a constant permittivity inside the molecule, we tested convergence against an analytic expression of the solvation energy of a sphere,[58] and then compared a more realistic geometry (arginine) with a purely BEM implementation. Then, as a more challenging test, we performed binding energy calculations from a set of 153 structures,[6] and compared them to well-established codes. We also considered a Gaussian-varying permittivity[25,26] inside the molecular cavity of arginine, and used APBS[7] to verify our results. The final tests show the scaling of the FEM-BEM coupling as the molecule size grows.

All runs were done on a Lenovo ThinkStation P620 with AMD Ryzen ThreadRipper PRO 3975WX (32-core and 3.5 GHz) and 128 GB RAM.

3.1 | Software environment

For the finite element computations, we use the software package FEniCSx [41,42] while for the boundary element computations, we use Bempp-cl [40] together with Exafmm-t.[59] FEniCSx is the successor of the widely used FEniCS finite element library.[60,61] It provides a convenient Python interface, describing problems using Unified Form Language (UFL),[62] a convenient domain-specific language specifically designed for finite element discretizations of partial differential equations. During assembly, the UFL description is transformed into efficient low-level C++ code and just-in-time compiled.[63,64] Bempp-cl is a Python package that uses low-level OpenCL kernels written in C99 to provide optimised assembly routines.[65] The built-in dense assembly routines are highly efficient for moderate discretization sizes up to a few ten thousand elements.

For very large grid sizes the user can enable fast multipole method (FMM) assembly which internally is handled in Bempp-cl through an interface to the Exafmm-t FMM library. For N surface elements this reduces the memory and computational complexity from $\mathcal{O}(N^2)$ in the dense assembly case to $\mathcal{O}(N)$ in the FMM case, making large boundary element problems tractable on standard workstations. In this work, we only use the FMM in the performance analysis for larger structures, as all other cases are small enough to run efficiently with a dense assembler.

To couple FEniCSx with Bempp-cl we load a volume mesh with FEniCSx. We then export the corresponding boundary mesh into Bempp-cl and assemble the boundary spaces there. Bempp-cl provides numerical trace operators that can translate from the degree of freedom (DOF) representation in FEniCSx to the DOF representation in Bempp-cl. The corresponding translation work is handled opaquely and the user only needs to deal with high-level interfaces of FEniCSx operators, Bempp-cl operators, and trace operators. FMM assembly fits automatically into this framework and can be enabled or disabled as a simple configuration option. Once the discrete block matrices in Equation (11) are built with Bempp-cl and FEniCSx, we assemble them into one matrix and solve the linear system with Scipy's[66] GMRES, in this case, with a tolerance of 10^{-5}.

Docker images containing FEniCSx, Bempp-cl, and Exafmm-t are publicly available (https://bempp.com/installation.html), and all codes used to generate the results in this section are available as Jupyter Notebooks that can be reproducibly executed in an appropriate environment. The results in this section were obtained using version 0.6.0 of FEniCSx and version 0.3.0 of Bempp-cl. All codes to reproduce the results of this manuscript can be found in the GitHub repository https://github.com/MichalBosy/FEM_BEM_coupling/.

3.2 | Results with constant permitivitty

In implicit-solvent models, the molecule is usually considered as a region with constant permittivity: in our computations, we use $\epsilon_1 = 2$. In the solvent region, we used the permittivity of water ($\epsilon_2 = 80$) and an inverse of the Debye length of $\kappa = 0.125\,\text{Å}^{-1}$. In this case, there is a known analytic solution for ϕ_c in Equation (2), so it is enough to compute ϕ_r. We do this using Equation (7) with both BEM-BEM and FEM-BEM coupling approaches. For FEM-BEM, the integral over Γ in Equation (7) corresponds to the trace of the solution vector from Equation (11).

3.3 | Convergence of a spherical cavity

The Kirkwood sphere[58] is a standard benchmark test for the Poisson–Boltzmann equation in molecular electrostatics. In this case, we considered a spherical cavity of radius $R = 2\,\text{Å}$, with three charges ($q_1 = 1$, $q_2 = 1$, and $q_3 = 0.75$) placed at $\mathbf{x}_1 = (1,0,0)$, $\mathbf{x}_2 = (0.7,0.7,0)$, and $\mathbf{x}_3 = (-0.5,-0.5,0)$. Figure 2 shows the percentage error of the FEM-BEM approach, and a reference BEM-BEM implementation, compared to the analytic solution ($\Delta G_{\text{solv}} = -336.0396\,\text{kcal/mol}$). In these runs, the FEM mesh was generated using GMSH [67] with 2, 6, 21 and 83 vertices per Å^2 on the SES. The triangular mesh on the boundary surface of the FEM mesh was used for the BEM runs. The error in Figure 2 decays linearly with the number of vertices on the surface. This agrees with the expected convergence for P1 elements, indicating a correct implementation of the numerical scheme.

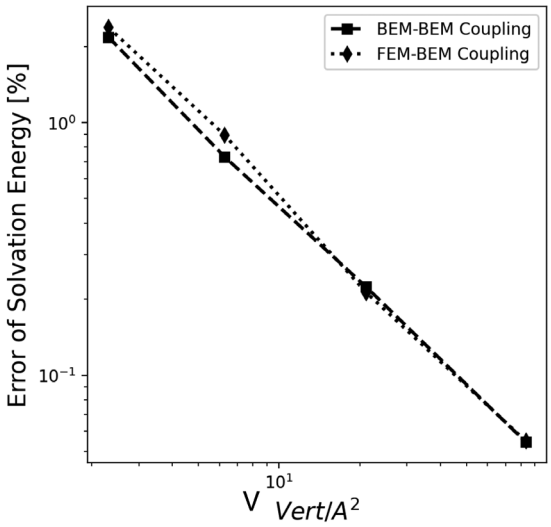

FIGURE 2 Error for the Kirkwood sphere.

3.4 | Performance with arginine

As a more realistic test, we assessed the performance of the FEM-BEM coupling technique against a BEM-BEM implementation for arginine. The structure of arginine was taken from the protein data bank, and parameterized with the Amber[68] force field. We generated surface meshes containing 4.1, 6.7, 8.6, 17, and 24.5 vertices per Å^2 with Nanoshaper.[69] These densities correspond to a grid-scale parameter in Nanoshaper equal to 1.6, 2.0, 2.4, 3.4, and 4.0, respectively, where the grid scale is the reciprocal of the average characteristic length of the triangles. For our BEM-BEM solver, we used these meshes directly. For the FEM-BEM solver, we created volume meshes from these using pyGAMer, [70] which invoked TetGen[71] with a quality parameter (radius-edge ratio) of 1.0.

The solvation energy computed with the two schemes is presented in Figure 3, which, as expected, converges to a similar answer as the mesh is refined. Figure 4 compares the iteration count and time-to-solution. The left plot shows that using only BEM outperforms the coupled FEM-BEM approach in terms of the iteration count. However, if we look at the total time that solvers take to obtain the solution, we can see the advantage of using the FEM-BEM coupling. The higher computational cost is caused by the need to use a hypersingular operator in the BEM formulation, and the fact that we are not using any acceleration method (i.e., FMM). The timings for the FEM-BEM coupling scale are at a greater rate than the pure BEM

counterpart, indicating that work on preconditioners and other acceleration methods will be required in order to make the FEM-BEM approach viable for large problems.

3.5 | Results with variable permittivity

3.5.1 | Note on modeling differences between finite difference and finite/boundary element methods

To facilitate verification, we conducted a comparative analysis between our FEM-BEM approach and established Poisson-Boltzmann solvers, including APBS,[7] MIBPB,[72,73] CPB,[74] PBSA,[75,76] and Delphi.[77] These benchmarks primarily employ finite difference approximations, setting the stage for inherent modelling disparities with respect to our FEM-BEM method, and making this comparison a difficult task. Firstly, the boundary integral formulation in the external domain naturally enforces the decay of the electrostatic potential to zero at infinity, while finite difference-based solvers necessitate the imposition of effective boundary conditions along the mesh periphery. This boundary condition critically influences the electrostatic potential and may deviate from the true solution if the size of the mesh box is insufficient. Secondly, the definition of the molecular surface differs slightly across software, resulting in different domain geometries. Our FEM-BEM approach employs a triangulation of the solvent-excluded surface (based on NanoShaper[69]), while finite difference meshes produce a staircase representation of this surface. This discrepancy is addressed by the matched interface technique in MIBPB,[72,73] however, the resulting surface does not exactly match NanoShaper's triangulation.

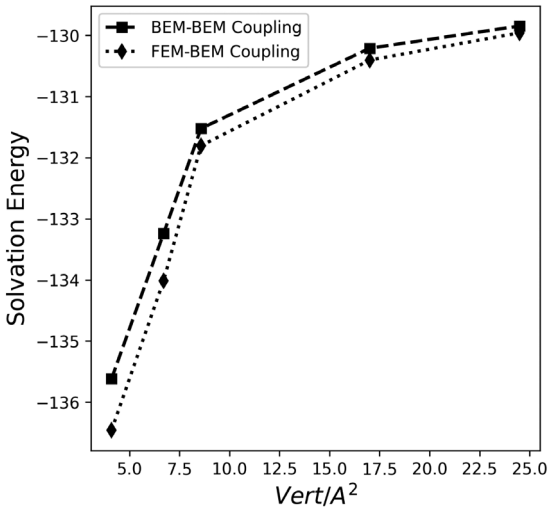

FIGURE 3 Solvation energy for arginine with a constant permittivity.

3.5.2 | Binding energy calculations for a large data set

The binding energy of two molecules in a complex is

$$\Delta G_{bind} = \Delta G_{solv,complex} - \Delta G_{solv,1} - \Delta G_{solv,2} + \Delta G_{coul} \quad (12)$$

where $\Delta G_{solv,complex}$ is the solvation energy of the complex, $\Delta G_{solv,1}$ and $\Delta G_{solv,2}$ are the solvation energies of the two compounds, and ΔG_{coul} is the difference in Coulomb energy between bound and unbound states. This calculation presents a major challenge, as small errors in solvation energy may have a large impact result in in binding energy, making accuracy crucial.

Following the data set proposed by Fenley and co-workers,[6] available in http://www.sb.fsu.edu/~mfenley/convergence/downloads/convergence_pqr_sets.tar.gz, and later used by Wei et al.[78] to assess the accuracy of different Poisson-Boltzmann solvers, we tested our method

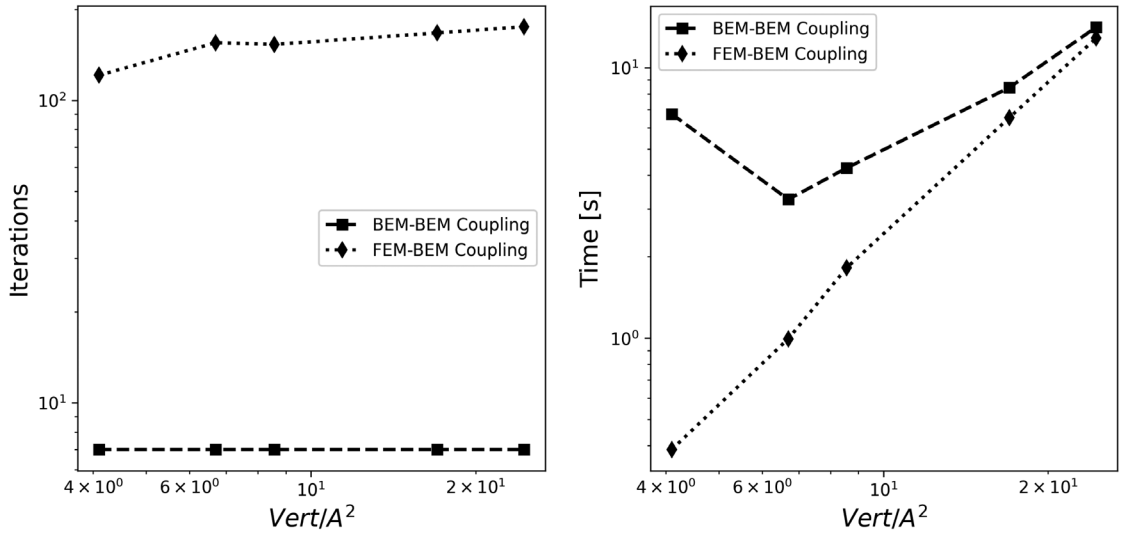

FIGURE 4 Iteration count (left) and time-to-solution (right) for arginine with a constant permittivity.

by computing binding energies. This data set consists of 51 complexes, totalling 153 structures, separated in three families: DNA-drug complexes, different versions of the barnase-barstar complex, and RNA-peptide complexes. Following the problem setting for these data sets, we use $\epsilon_1 = 1$, $\epsilon_2 = 80$ and an inverse of the Debye length of $\kappa = 0.10265\ \text{Å}^{-1}$.

According to the arginine results in Figure 3, a mesh density of 24 vertices per Å^2 gives a solvation energy that is close to the true solution for a realistic geometry. In that case, the difference in solvation energy between calculations using finite element grids generated from a surface mesh of 8.6 and 24.5 vertices per Å^2 (a 3× refinement) is less than 1%. Then, we used surface meshes with 24 vertices per Å^2 and a quality parameter of 1.1 in TetGen to generate the meshes used to compute binding energies in Fenley's database.

Several structures in the database contained voids inside the molecular region where a water molecule fits. In a finite difference description, these voids are naturally considered as pockets of solvent, with the permittivity of water ($\epsilon = \epsilon_2$). Regardless if this is an appropriate physical model, we also placed the permittivity of bulk water inside cavities for comparison purposes. In the FEM formalism, we had to detect the elements contained by cavities and use a piece-wise constant varying description of the permittivity inside the molecular region, which can be done without any modification in the formulation. A BEM-BEM formulation can also account for cavities as regions with water's bulk permittivity,[16,19] but involves certain adjustments in the stiffness matrix.

Figures 5,6, and 7 show the binding energies of structures of DNA-drug complexes, barnase-barstar complexes, and RNA-peptide complexes, respectively, presented in Fenley's data set.[6] The data for CPB[6] and MIBPB, Delphi, APBS and PBSA were taken from the literature,[78] for the finest mesh reported. From Figures 5,6, and 7 it is evident that our FEM-BEM approach gives results that are within the range of the difference between other software for all 51 complexes. Given the challenging nature of binding energy calculations, and that our results are consistently comparable to other widely-trusted solvers that use a different numerical scheme for all 51 complexes, we can positively conclude on the correctness of our code.

3.6 | Modeling the solute with a Gaussian-based variable permittivity

In contrast to a purely BEM approach, FEM-BEM coupling gives the flexibility to consider space-varying field parameters. A popular description of the molecule is to consider a permittivity that varies like a Gaussian around each atom,[25] which has shown enhanced accuracy in some applications, like pKa calculations.[26] In this setting, we define a density function ρ depending on the position $\mathbf{r}$ as

$$\rho(r) := \prod_i \left(1 - \exp\left(\frac{\|\mathbf{r} - \mathbf{x}_i\|}{\sigma^2 R_i^2}\right)\right), \tag{13}$$

where the product runs over all the atoms of the solute, R_i is the van der Waals radius of atom i, and we used $\sigma = 1$. Then, we can compute the permittivity as

$$\epsilon := (1-\rho)\epsilon_1 + \rho\epsilon_2 \tag{14}$$

As ϵ is variable, Equation (2) does not have a known analytic solution, and the electrostatic potential in the vacuum state has to be computed numerically. For vacuum calculations, we considered the same distribution of ϵ inside the molecule as in the solvated case, but the solvent permittivity was set to $\epsilon_2 = 2$. Other implementations of Gaussian permittivities also modify the solute permittivity in vacuum calculations, according to a set cutoff.[26] We did not consider a cutoff in our calculations.

3.6.1 | Convergence for arginine with APBS

We used Equation (14) to generate dielectric maps, which we ran on APBS [7] for comparison. We chose APBS because it provides an easy interface to control dielectric maps to ensure their agreement with the maps imposed in our FEM-BEM coupled approach.

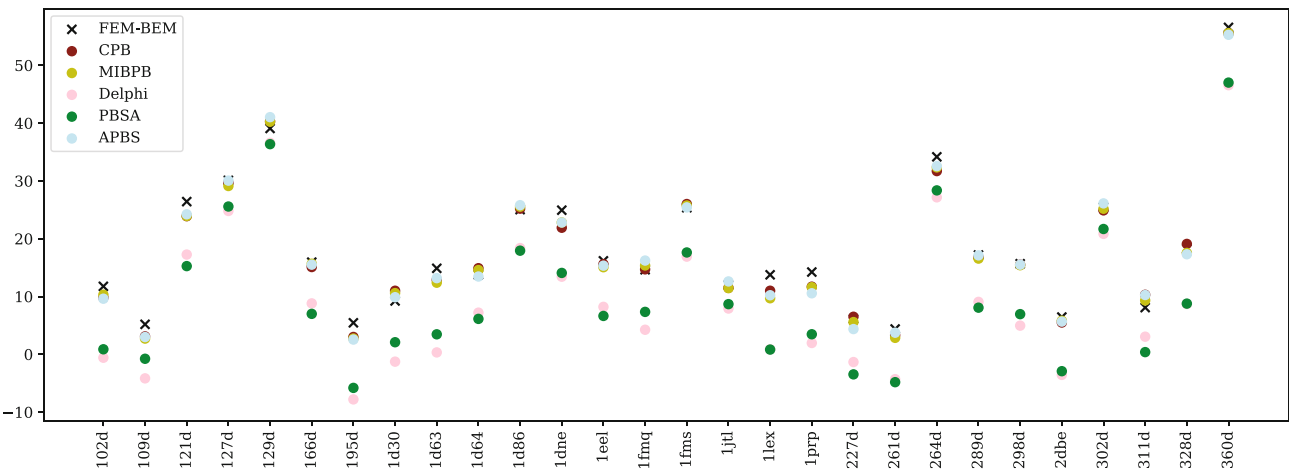

FIGURE 5 Binding energies for structures corresponding to DNA-drug complexes.

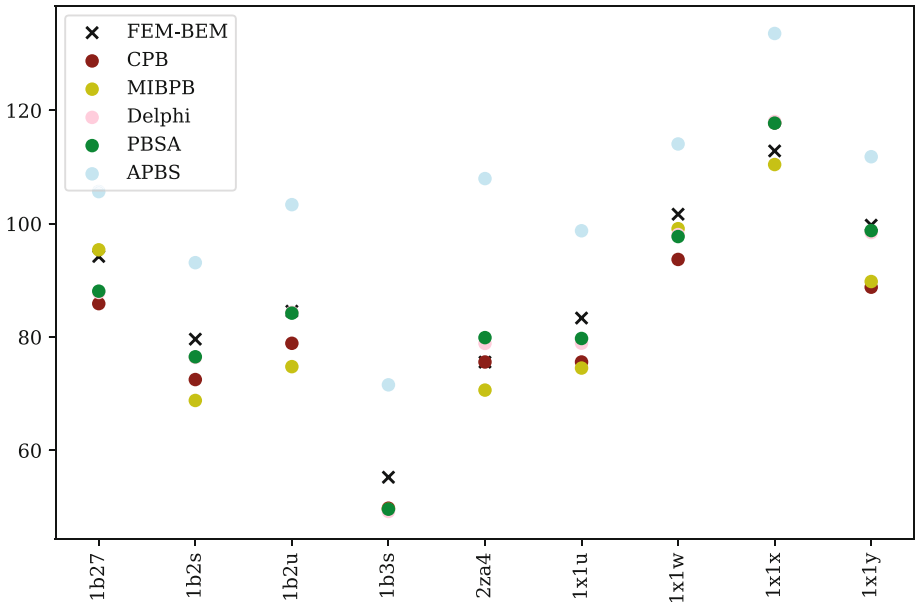

FIGURE 6 Binding energies for structures corresponding to the barnase-barstar complex.

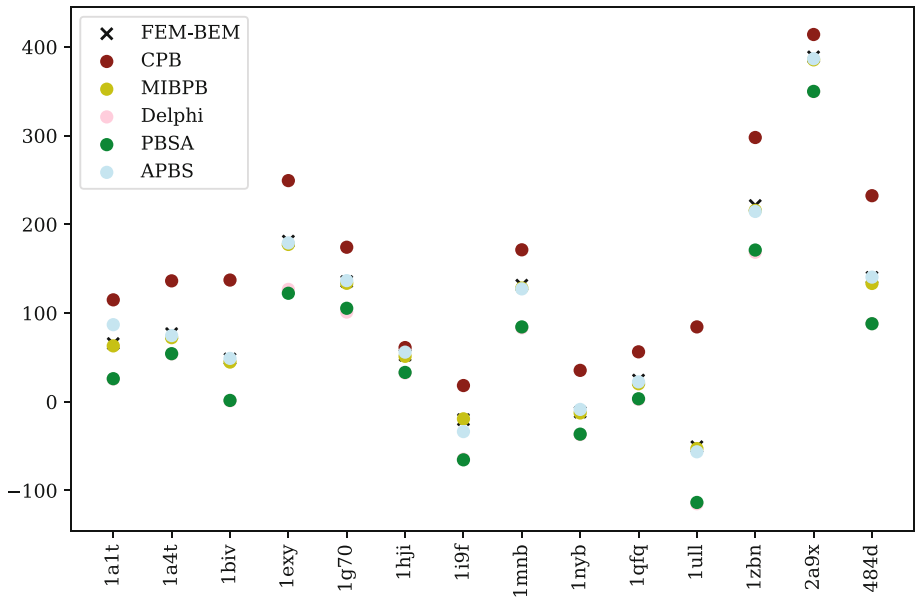

FIGURE 7 Binding energies for structures corresponding to RNA-peptide complexes.

Table 1 shows a comparison of the solvation energy computed with APBS and our FEM-BEM coupling approach. We can see that they are both converging to equivalent values, where the solutions on the finest meshes agree to within 0.5% (0.5 kcal/mol). The coarsest meshes used in both cases are within the recommended densities for accurate solvation energy simulations with constant permittivity: a finite difference meshes with $h = 0.5$ Å or less is recommended for binding energy calculations[79] (the result of subtracting two solvation energies), and a similar study with BEM[19] showed that a mesh with 2 vertices per Å^2 gives acceptable results when computing binding and solvation energies. We see a jump in the solvation energy for the FEM-BEM results going from 4.1 to 6.7 vertices per Å^2, but later the energy monotonically decreases, converging to a solution. This is an indication that mesh requirements for variable permittivities may be tighter than with a constant permittivity, even though the jump in dielectric constant across the molecular surface is usually smaller.

3.7 | Performance analysis for larger structures

In the experiments presented so far, we have only tested the FEM-BEM coupled approach with small structures. In this section, we study its behaviour with larger structures to evaluate its applicability in more realistic settings, and in test cases where BEM-BEM coupling is not an alternative. Using the same Gaussian permittivity field inside the molecule, we computed the solvation free energy of protein G B1 (955 atoms, PDB code 1pgb), lysozyme (1960 atoms, PDB code 1lyz), the barnase-barstar complex (9464 atoms, PDB code 1x1u), and

TABLE 1 Solvation energy of arginine with a Gaussian-like permittivity, computed using APBS (left) and the FEM-BEM approach (right).

Mesh size (Å)	Grid points	ΔG_{solv} (kcal/mol)	Mesh density (vert/Å^2)	DOFs	ΔG_{solv} (kcal/mol)
$0.52 \times 0.52 \times 0.52$	$97 \times 97 \times 97$	−107.6186	4.1	3 491	−109.931
$0.39 \times 0.39 \times 0.39$	$129 \times 129 \times 129$	−107.8752	6.7	5 787	−110.237
$0.26 \times 0.26 \times 0.26$	$193 \times 193 \times 193$	−108.3378	8.6	8 844	−109.661
$0.195 \times 0.195 \times 0.195$	$257 \times 257 \times 257$	−108.5837	17.0	19 911	−109.369
$0.098 \times 0.098 \times 0.098$	$513 \times 513 \times 513$	−108.8844	24.5	32 302	−109.315

Note: The mesh density for FEM-BEM corresponds to the vertex density of the surface mesh used to generate the volumetric mesh.

TABLE 2 Results for larger structures with a variable permittivity.

Molecule	FEM DOFs	BEM DOFs	ΔG_{solv} (kcal/mol)	Iterations	Setup time (s)	Solving time (s)	Time per iter. (s)	Total time (s)
1pgb	29,434	10,058	−300.888	703	86	472	0.67	789
1lyz	56,114	18,606	−599.310	1073	336	1350	1.26	1686
1x1u	263,181	81,258	−1982.085	2791	8340	11000	3.94	19,340
1igt	597,575	187,712	−3294.0157	5366	41,998	40,100	7.47	82,098

immunoglobulin G (20148 atoms, PDB code 1igt). All structures were parameterized with the Amber[80] force field using pdb2pqr.[81] The surfaces were meshed with Nanoshaper[69] using a grid scale of 1.5, and then volume meshes were generated using pyGAMer[70] and TetGen[71] with a radius-edge ratio of 1.0. The solvation free energy and timings for these runs are presented in Table 2. These results demonstrate that our FEM-BEM coupling implementation can reach medium-to-large-sized proteins on a simple workstation. For these larger structures, we enabled the FMM capabilities of Bempp-cl.

Table 2 shows that the iteration count increases with the problem size. This is expected for Johnson–Nédélec coupling, which is the simplest coupling strategy. To isolate the effect of the increased iterations in the analysis, we separate timings into the setup and solving time, where the setup time is independent of the number of iterations, and the solving time corresponds to the time spent in the GMRES solver. The time-per-iteration is computed by dividing the solving time by the number of iterations. The time per iteration scales closer to $\mathcal{O}(N)$, which is expected as we are using FMM for the BEM portion of the matrix. This is an indication that the high solving time is mainly due to the increase in iteration count, and having better-conditioned coupling methods, such as the so-called hybrid approach,[82] or more effective preconditioners for the blocked system would have a large impact on the time to solution.

Even though this scheme is capable of calculating the electrostatic potential in medium-to-large proteins, the largest test case in Table 2 (1ight) used up 30% of the available RAM. If we were aiming at larger structures, such as full viruses,[43,44] we would require the use of optimised fast algorithms[83,84] and parallelization of the storage and solver, alongside the necessary improvements in the conditioning of the system.

4 | CONCLUSIONS

This paper presents the first implementation of a FEM-BEM coupling approach to solve the Poisson–Boltzmann equation for molecular electrostatics. This brings the best of both worlds: the accuracy and efficiency of BEM to exactly enforce the boundary conditions at infinity, and the flexibility of FEM to account for space variations of the material properties and nonlinearities. After presenting verification results of solvation energy for a sphere and arginine with a constant permittivity inside the solute, and binding energy for an extensive data set, we showcased our implementation with an advanced modelling technique that considers Gaussian-varying permittivities in a confined region, with the results validated against the widely-used APBS software. The final scaling results for larger molecules start from protein G B1 (955 atoms) and go up to immunoglobulin G (20 148 atoms), proving the applicability of this approach for realistic problems. Even though our implementation was able to reach medium-to-large systems, we recognise the need for further research towards better preconditioning of the linear system, optimising the coupling technique, acceleration algorithms, and parallel execution, especially as we look towards much larger solutes, like viruses. We hope this proof-of-concept work will serve as motivation for future model development that considers space-varying permittivities and Debye lengths, and mixed linear-nonlinear techniques, especially for highly-charged systems, like nucleic acids.

ACKNOWLEDGMENTS

Michał Bosy acknowledges the support from Kingston University through the First Kingston University grant. Matthew W. Scroggs acknowledges support from EPSRC grant EP/W007460/1. Timo

Betcke acknowledges support from EPSRC grants EP/W007460/1 and EP/W026260/1. Erik Burman acknowledges support from EPSRC grants EP/V050400/1 and EP/W007460/1. Christopher D. Cooper acknowledges the support from CCTVal through ANID PIA/APOYO AFB220004 and Universidad Técnica Federico Santa María through Proyectos Internos USM 2023 PI-LIR-23-03. We would like to thank Jørgen Dokken for his help with various implementation details related to FEniCSx.

DATA AVAILABILITY STATEMENT

All codes to reproduce the results of this manuscript can be found in the GitHub repository https://github.com/MichalBosy/FEM_BEM_coupling/, alongside links to Docker images that include the codes alongside installations of appropriate versions of Bempp-cl and FEniCSx. The numerical results of the binding energy calculations, alongside all the required input files to reproduce them, are available in the Zenodo repository https://doi.org/10.5281/zenodo.8393170.

ORCID

Michał Bosy https://orcid.org/0000-0003-2723-6913
Matthew W. Scroggs https://orcid.org/0000-0002-4658-2443
Timo Betcke https://orcid.org/0000-0002-3323-2110
Erik Burman https://orcid.org/0000-0003-4287-7241
Christopher D. Cooper https://orcid.org/0000-0003-0282-8998

REFERENCES

[1] A. V. Onufriev, S. Izadi, *Wiley Interdiscip. Rev: Comput. Mol. Sci.* **2018**, *8*, e1347.
[2] B. Roux, T. Simonson, *Biophys. Chem.* **1999**, *78*, 1.
[3] S. Decherchi, M. Masetti, I. Vyalov, W. Rocchia, *Eur. J. Med. Chem.* **2015**, *91*, 27.
[4] N. A. Baker, *Meth. Enzymol.* **2004**, *383*, 94.
[5] J. P. Bardhan, *Comput. Sci. Disc.* **2012**, *5*, 013001.
[6] R. C. Harris, A. H. Boschitsch, M. O. Fenley, *J. Chem. Theory Comput.* **2013**, *9*, 3677.
[7] N. A. Baker, D. Sept, M. J. Holst, J. A. McCammon, P. Natl, *Acad. Sci. USA* **2001**, *98*, 10037.
[8] M. K. Gilson, A. Rashin, R. Fine, B. Honig, *J. Mol. Biol.* **1985**, *184*, 503.
[9] E. Jurrus, D. Engel, K. Star, K. Monson, J. Brandi, L. E. Felberg, D. H. Brookes, L. Wilson, J. Chen, K. Liles, et al., *Protein Sci.* **2018**, *27*, 112.
[10] C. Li, Z. Jia, A. Chakravorty, S. Pahari, Y. Peng, S. Basu, M. Koirala, S. K. Panday, M. Petukh, L. Li, et al., *J. Comput. Chem.* **2019**, *40*, 2502.
[11] M. Holst, J. A. Mccammon, Z. Yu, Y. Zhou, Y. Zhu, *Commun Comput Phys* **2012**, *11*, 179.
[12] S. D. Bond, J. H. Chaudhry, E. C. Cyr, L. N. Olson, *J. Comput. Chem.* **2010**, *31*, 1625.
[13] S. Nakov, E. Sobakinskaya, T. Renger, J. Kraus, *J. Comput. Chem.* **2021**, *42*, 1832.
[14] A. H. Boschitsch, M. O. Fenley, H.-X. Zhou, *J Phys Chem B* **2002**, *106*, 2741.
[15] B. Lu, X. Cheng, J. Huang, J. A. McCammon, P. Natl, *Acad. Sci. USA* **2006**, *103*, 19314.
[16] M. D. Altman, J. P. Bardhan, J. K. White, B. Tidor, *J. Comput. Chem.* **2009**, *30*, 132.
[17] C. Bajaj, S.-C. Chen, A. Rand, *SIAM J Sci Comput* **2011**, *33*, 826.
[18] W. H. Geng, R. Krasny, *J. Comput. Phys.* **2013**, *247*, 62.
[19] C. D. Cooper, J. P. Bardhan, L. A. Barba, *Comput. Phys. Commun.* **2014**, *185*, 720.
[20] E.-H. Yap, T. Head-Gordon, *J. Chem. Theory Comput.* **2010**, *6*, 2214.
[21] L. E. Felberg, D. H. Brookes, E.-H. Yap, E. Jurrus, N. A. Baker, T. Head-Gordon, *J. Comput. Chem.* **2017**, *38*, 1275.
[22] C. Quan, B. Stamm, Y. Maday, *SIAM J Sci Comput* **2019**, *41*, B320.
[23] S. V. Siryk, W. Rocchia, *J Phys Chem B* **2022**, *126*, 10400.
[24] A. Jha, B. Stamm, arXiv preprint arXiv:2309.06862. **2023**.
[25] J. A. Grant, B. T. Pickup, A. Nicholls, *J. Comput. Chem.* **2001**, *22*, 608.
[26] L. Li, C. Li, Z. Zhang, E. Alexov, *J. Chem. Theory Comput.* **2013**, *9*, 2126.
[27] F. Fogolari, P. Zuccato, G. Esposito, P. Viglino, *Biophys. J.* **1999**, *76*, 1.
[28] C. Carstensen, E. Stephan, *Numer Methods Partial Differ Equ* **1995**, *11*, 539.
[29] M. Aurada, M. Feischl, T. Führer, M. Karkulik, J. M. Melenk, D. Praetorius, *Comput Mech* **2013**, *51*, 399.
[30] B. Aour, O. Rahmani, M. Nait-Abdelaziz, *Int. J. Solids Struct.* **2007**, *44*, 2523.
[31] O. Von Estorff, H. Antes, *Int J Numer Methods Eng* **1991**, *31*, 1151.
[32] R. Hiptmair, P. Meury, *SIAM J Numer Anal* **2006**, *44*, 2107.
[33] F. Matsuoka, A. Kameari, *IEEE Trans. Magn.* **1988**, *24*, 182.
[34] R. Hiptmair, P. Meury, *Modeling and computations in electromagnetics*, Springer, Berlin Heidelberg **2008**, p. 1.
[35] F. Bruckner, C. Vogler, M. Feischl, D. Praetorius, B. Bergmair, T. Huber, M. Fuger, D. Suess, *J. Magn. Magn. Mater.* **2012**, *324*, 1862.
[36] A. H. Boschitsch, M. O. Fenley, *J. Comput. Chem.* **2004**, *25*, 935.
[37] D. Xie, J. Ying, *J Comput Appl Math* **2016**, *307*, 319.
[38] J. Ying, D. Xie, *Appl Math Model* **2018**, *58*, 166.
[39] C. Johnson, J. Nédélec, *Math Comput* **1980**, *35*, 1063.
[40] T. Betcke, M. W. Scroggs, *J Open Source Softw* **2021**, *6*, 2879.
[41] M. W. Scroggs, I. A. Baratta, C. N. Richardson, G. N. Wells, *J Open Source Softw* **2022a**, *7*, 3982.
[42] M. W. Scroggs, J. S. Dokken, C. N. Richardson, G. N. Wells, *ACM Trans Math Softw* **2022b**, *48*, 18:1.
[43] M. Martínez, C. D. Cooper, A. B. Poma, H. V. Guzman, *J. Chem. Inf. Model.* **2019**, *60*, 974.
[44] T. Wang, C. D. Cooper, T. Betcke, L. A. Barba, arXiv preprint arXiv: 2103.01048. **2021a**.
[45] B. Z. Lu, Y. C. Zhou, M. J. Holst, J. A. McCammon, *Commun Comput Phys* **2008**, *3*, 973.
[46] A. Lee, W. Geng, S. Zhao, *J. Comput. Phys.* **2021**, *426*, 109958.
[47] S. Wang, Y. Shao, E. Alexov, S. Zhao, *J. Comput. Phys.* **2022**, *464*, 111340.
[48] S. D. Search, C. D. Cooper, E. van't Wout, *J. Comput. Chem.* **2022**, *43*, 674.
[49] B. J. Yoon, A. M. Lenhoff, *J. Comput. Chem.* **1990**, *11*, 1080.
[50] B. Lu, X. Cheng, J. Huang, J. A. McCammon, *J. Chem. Theory Comput.* **2009**, *5*, 1692.
[51] J. DeBuhr, B. Zhang, A. Tsueda, V. Tilstra-Smith, T. Sterling, *Commun Comput Phys* **2016**, *20*, 1106.
[52] R. Zauhar, R. Morgan, *J. Mol. Biol.* **1985**, *186*, 815.
[53] P. B. Shaw, *Phys. Rev. A* **1985**, *32*, 2476.
[54] C. D. Cooper, N. C. Clementi, G. Forsyth, L. A. Barba, *J. Open Source Softw.* **2016**, *1*, 43.
[55] R. Hiptmair, *SIAM J Numer Anal* **2002**, *40*, 41.
[56] L. Banjai, C. Lubich, F.-J. Sayas, *Numer Math* **2015**, *129*, 611.
[57] O. Steinbach, *Numerical approximation methods for elliptic boundary value problems*, Springer, New York **2008**. https://doi.org/10.1007/978-0-387-68805-3
[58] J. G. Kirkwood, *J. Chem. Phys.* **1934**, *2*, 351.
[59] T. Wang, R. Yokota, L. A. Barba, *J Open Source Softw* **2021b**, *6*, 3145.
[60] M. S. Alnaes, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, G. N. Wells, *Arch Numer Softw* **2015**, *3*, 9. https://doi.org/10.11588/ans.2015.100.20553
[61] A. Logg, K. Mardal, G. N. Wells, *Automated solution of differential equations by the finite element method*, Springer, Berlin Heidelberg **2012a**.
[62] M. S. Alnaes, A. Logg, K. B. Ølgaard, M. E. Rognes, G. N. Wells, *ACM Trans Math Softw* **2014**, *40*, 1.

[63] R. C. Kirby, A. Logg, *ACM Trans Math Softw* **2006**, *32*, 417.
[64] A. Logg, K. B. Ølgaard, M. E. Rognes, G. N. Wells, in *Automated Solution of Differential Equations by the Finite Element Method*. Lecture Notes in Computational Science and Engineering, Vol. 84 (Eds: A. Logg, K. Mardal, G. N. Wells), Springer, Berlin Heidelberg **2012b**.
[65] T. Betcke, M. W. Scroggs, *IEEE Comput Sci Eng* **2021**, *23*, 18.
[66] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., *Nat. Methods* **2020**, *17*, 261.
[67] C. Geuzaine, J.-F. Remacle, *Int J Numer Methods Eng* **2009**, *79*, 1309.
[68] J. W. Ponder, D. A. Case, *Adv. Protein Chem.* **2003**, *66*, 27.
[69] S. Decherchi, W. Rocchia, *PLoS One* **2013**, *8*, e59744.
[70] C. T. Lee, J. G. Laughlin, J. B. Moody, R. E. Amaro, J. A. McCammon, M. Holst, P. Rangamani, *Biophys. J.* **2020**, *118*, 1003.
[71] S. Hang, A. C. M. Trans, *Math. Softw* **2015**, *41*, 11.
[72] W. Geng, S. Yu, G. Wei, *J Chem Phys* **2007**, *127*, 244108.
[73] D. Chen, Z. Chen, C. Chen, W. Geng, G.-W. Wei, *J. Comput. Chem.* **2011**, *32*, 756.
[74] A. H. Boschitsch, M. O. Fenley, *J. Chem. Theory Comput.* **2011**, *7*, 1524.
[75] J. Wang, C. Tan, Y.-H. Tan, Q. Lu, R. Luo, *Commun Comput Phys* **2008**, *3*, 1010.
[76] Q. Cai, M.-J. Hsieh, J. Wang, R. Luo, *J. Chem. Theory Comput.* **2010**, *6*, 203.
[77] W. Rocchia, E. Alexov, B. Honig, *J Phys Chem B* **2001**, *105*, 6507.
[78] D. D. Nguyen, B. Wang, G.-W. Wei, *J. Comput. Chem.* **2017**, *38*, 941.
[79] J. Sørensen, M. O. Fenley, R. E. Amaro, *Computational Electrostatics for Biological Applications*, Springer, Berlin Heidelberg **2015**, p. 39.
[80] J. M. J. Swanson, S. A. Adcock, J. A. McCammon, *J. Chem. Theory Comput.* **2005**, *1*, 484.
[81] T. J. Dolinsky, J. E. Nielsen, J. A. McCammon, N. A. Baker, *Nucleic Acids Res.* **2004**, *32*, W665.
[82] T. Betcke, M. Bosy, E. Burman, *Numer Algorithms* **2022**, *91*, 997.
[83] T. Wang, R. Yokota, L. A. Barba, *J Open Source Softw* **2021c**, *6*, 3145.
[84] S. Kailasa, T. Wang, L. A. Barba, T. Betcke, arXiv preprint arXiv: 2303.08394. **2023**.

How to cite this article: M. Bosy, M. W. Scroggs, T. Betcke, E. Burman, C. D. Cooper, *J. Comput. Chem.* **2024**, *45*(11), 787. https://doi.org/10.1002/jcc.27262