

# Improvements to the APBS biomolecular solvation software suite

Elizabeth Jurrus\*, Dave Engel\*, Keith Star\*, Kyle Monson\*, Juan Brandi\*, Dave Engel\*,  
Lisa E. Felberg†, David Brookes†, Leighton Wilson‡, Jiahui Chen§, Karina Liles\*,  
Minju Chun\*, Peter Li\*, Todd Dolinsky||, Robert KonecnyΔ, David Koes▷,  
Jens Erik Nielsen¶, Theresa Head-Gordon†, Weihua Geng§, Robert Krasny‡,  
Marilyn Gunner°, Guo-Wei Wei∇, Michael J. HolstΔ, J. Andrew McCammonΔ, and  
Nathan A. Baker△

\*Pacific Northwest National Laboratory

†University of California Berkeley

‡University of Michigan

§Southern Methodist University

||FoodLogiQ

▷University of Pittsburgh

¶Novozymes

∇Michigan State University

ΔUniversity of California San Diego

°City University of New York

△To whom correspondence should be addressed. Advanced Computing, Mathematics, and Data Division; Pacific Northwest National Laboratory; Richland, WA 99352, USA. Division of Applied Mathematics; Brown University; Providence, RI 02912, USA.

Email: nathan.baker@pnnl.gov

June 24, 2017

## Abstract

The Adaptive Poisson-Boltzmann Solver (APBS) software was developed to solve the equations of continuum electrostatics for large biomolecular assemblages [? ]. The understanding of electrostatic interactions is essential for the study of biomolecular processes and impacts many chemical, biological, and biomedical applications. APBS addresses three key technology challenges for understanding solvation and electrostatics in biomedical applications: accurate and efficient models for biomolecular solvation and electrostatics, robust and scalable software for applying those theories to biomolecular systems, and mechanisms for sharing and analyzing biomolecular electrostatics data in the scientific community. To address new research applications and advancing computational capabilities, we have continually updated APBS and its suite of accompanying software since its release in 2001. In this paper, we discuss the models and capabilities that have recently been implemented within the APBS software package including: a Poisson-Boltzmann analytical and a semi-analytical solver, an optimized boundary element solver, a geometry-based geometric flow solvation model, a graph theory based algorithm for determining  $pK_a$  values, and an improved web-based visualization tool for viewing electrostatics.

# 1 Introduction

This manuscript provides an overview of the new capabilities in the APBS software and its associated tools since their release in 2001 [1? –3]; more detailed documentation for these tools is available on the APBS website [4]. **\*\* Dave: please make sure this is true! \*\*** The manuscript is not intended to be a review of solvation models; readers interested in this area should refer to several previous reviews [5–10]. Robust models of electrostatic interactions are important for understanding intermolecular interactions and the effects of solvation on biomolecular processes. Such models can be loosely grouped into two categories: explicit and implicit. Explicit methods treat the solvent in atomic detail whereas implicit methods generally replace the explicit solvent with a dielectric continuum. Explicit methods tend to offer the highest levels of detail; however, they generally require extensive sampling to converge properties of interest. Implicit methods, on the other hand, trade detail and some accuracy for efficiency by eliminating the sampling and equilibration associated with explicit solvent models. The APBS software implements implicit models of solvation.

A variety of implicit solvent methods have been developed for the description of polar solvation [7], with Poisson-Boltzmann (PB) methods being among the most popular. The PB equation [11–13] provides a global solution for the electrostatic potential ( $\phi$ ) within and around a biomolecule by solving the partial differential equation

$$-\nabla \cdot \epsilon \nabla \phi - \sum_i^M c_i q_i e^{-\beta(q_i \phi + V_i)} = \rho. \quad (1)$$

The solvent is described by the bulk solvent dielectric constant  $\epsilon_s$  as well as the properties of  $i = 1, \dots, M$  mobile ion species, described by their charges  $q_i$ , concentrations  $c_i$ , and steric ion-solute interaction potential  $V_i$ . The biomolecular structure incorporated into the equation through  $V_i$ , a dielectric coefficient function  $\epsilon$ , and a charge distribution function  $\rho$ . The dielectric  $\epsilon$  is often set to a constant value  $\epsilon_m$  in the interior of the molecule and varies sharply across the molecular boundary to the value  $\epsilon_s$  which describes the bulk solvent. The shape of the boundary is determined by the size and location of the solute atoms as well as model-specific parameters such as a characteristic solvent molecule size [? ]. The charge distribution  $\rho$  is usually a sum of atomic charge distributions which are located at atom centers. Finally,  $\beta = (kT)^{-1}$  is the inverse thermal energy where  $k$  is the Boltzmann constant and  $T$  is the temperature.

The potential  $\phi$  can be used in a variety of applications, including visualization, other structural analyses, diffusion simulations, and a number of other calculations that require global electrostatic properties. The PB theory is approximate and, as a result, has several well-known limitations which can affect its accuracy, particularly for strongly charged systems or high salt concentrations [11, 14]. Despite these limitations, PB methods are still very important for biomolecular structural analysis, modeling, and simulation.

Several software packages have been developed to analyze the solvation properties of small and macro-molecules, solving the Poisson-Boltzmann equations to evaluate energies, potentials, and other solvation properties. The most significant (based on user base and citations) of these include CHARMM [15], AMBER [16], DelPhi [17], Jaguar [18], Zap [19], MIBPB [20], and APBS [? ]. This article describes the new functionality added to APBS, and its associated software, since its release in 2001.

Many users interact with APBS and PDB2PQR through web servers linked from the APBS website [4]. These servers support preparation of biomolecular structures (see Section ??) and finite-difference solution of the Poisson-Boltzmann equation (see Section 3.1). However, a broader

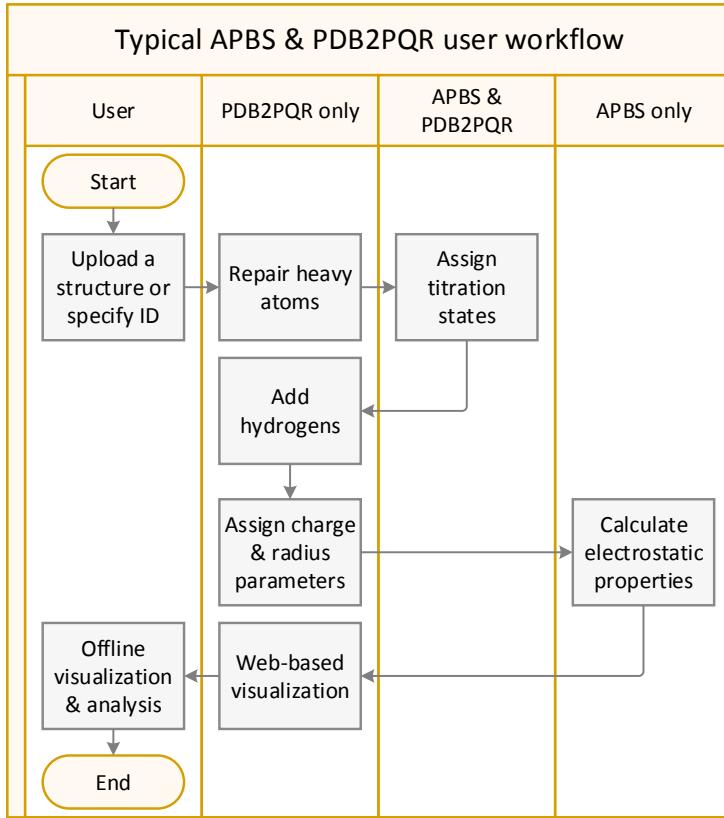


Figure 1: Workflow for biomolecular electrostatics calculations using the APBS-PDB2PQR software suite.

range of features and more flexible configuration is available when APBS and PDB2PQR are run from the command line (on Linux, Mac, and Windows). Command-line versions of APBS can be run locally or through web services provided by the NCBR-developed Opal toolkit [? ]. This toolkit allows for the computing load for processor intensive scientific applications to be shifted to remote computing resources such as those provided by the National Biomedical Computation Resource (NCBR). Finally, APBS can run through other molecular simulation programs such as AMBER [? ], CHARMM [? ], NAMD [? ], and TINKER [? ]. General support for integration of APBS with 3<sup>rd</sup>-party programs is provided by the iAPBS library [21, 22].

Most APBS electrostatics calculations follow the general workflow outlined in Figure 1. This workflow is supported by the APBS tool suite with specific support by PDB2PQR for preparing biomolecular structures (see Section ??) and support by APBS for performing electrostatics calculations (see Section ??).

## 2 Preparing biomolecular structures

Electrostatics calculations begin with specification of the molecule structure and parameters for the charge and size of the constituent atoms. Constituent atoms are generally grouped in types with charge and size values specified by atom type in a variety of force field files developed for implicit solvent calculations [7]. APBS incorporates this information into calculations via the “PQR” format. PQR is a file format of unknown origins used by several software packages such

as MEAD [? ] and AutoDock [? ]. The PQR file simply replaces the temperature and occupancy columns of a PDB flat file [? ] with the per-atom charge ( $Q$ ) and radius ( $R$ ). There are much more elegant ways to implement the PQR functionality through more modern extensible file formats such as mmcCIF [? ] or PDB-XML; however, the simple PDB format is still one of the most widely used formats and, therefore, continued use of the PQR format supports broad compatibility across biomolecular modeling tools and workflows.

The PDB2PQR software is part of the APBS suite that was developed to assist with the conversion of PDB files to PQR format [23, 24]. PDB2PQR began its existence as a `sed` [? ] script and evolved to work with APBS and support the functionality shown in Figure 1. In particular, PDB2PQR automatically sets up, executes, and optimizes the structure for Poisson-Boltzmann electrostatics calculations, outputting a PQR file that can be used with APBS or other modeling software. Some of the key steps in PDB2PQR are described below.

**Repairing missing heavy atoms** Within PDB2PQR, the PDB file is examined to see if there are missing heavy (non-hydrogen) atoms. Missing heavy atoms can be rebuilt using standard amino acid topologies in conjunction with existing atomic coordinates to determine new positions for the missing heavy atoms. A *debump* option performs very limited minimization of sidechain  $\chi$  angles to reduce steric clashes between rebuilt and existing atoms.

**Optimizing titration states** Amino acid titration states are important determinants of biomolecular (particularly enzymatic) function and can be used to assess functional activity and identify active sites. The APBS-PDB2PQR system contains several methods for this analysis.

- *Empirical methods.* PDB2PQR provides an empirical model (PROPKA [25]) which uses a heuristic method to compute  $pK_a$  perturbations due to desolvation, hydrogen bonding, and charge-charge interactions is included in PDB2PQR. The empirical PROPKA method has surprising accuracy for fast evaluation of protein  $pK_a$  values [26].
- *Implicit solvent methods.* PDB2PQR also contains two methods for using implicit solvent (Poisson-Boltzmann) models for predicting residue titration states. The first method uses Metropolis Monte Carlo to calculating titration curves and  $pK_a$  values (PDB2PKA); however, sampling issues can be a major problem with Monte Carlo methods when searching over the  $\mathcal{O}(2^N)$  titration states of  $N$  titratable residues. The second method is a new polynomial-time algorithm for the optimization of discrete states in macromolecular systems [27]. This method transforms interaction energies between titratable groups into a graphical flow network. The polynomial-time  $\mathcal{O}(N^4)$  behavior makes it possible to rigorously evaluate titration states for much larger proteins than Monte Carlo methods.

**Adding missing hydrogens.** The majority of PDB entries do not include hydrogen positions. Given a titration state assignment, PDB2PQR uses Monte Carlo sampling to position hydrogen atoms and optimize the global hydrogen-bonding network in the structure [28]. Newly added hydrogen atoms are checked for steric conflicts and optimized via the debumping procedure discussed above.

**Assigning charge and radius parameters** Given the titration state, atomic charges (for  $\rho$ ) and radii (for  $\epsilon$  and  $V_i$ ) are assigned based on the chosen force field. PDB2PQR currently supports charge/radii force fields from AMBER99 [? ], CHARMM22 (`par_all27_prot_na.prm`) [? ], PARSE [? ], PEOEPB [29], Swanson et al. [30], and Tan et al. [31].

### 3 Solving the Poisson-Boltzmann and related solvation equations

The APBS software was designed from the ground up using modern design principles to ensure its ability to interface with other computational packages and evolve as methods and applications change over time. APBS input files contain several keywords that are generic with respect to the type of calculation being performed; these are described in Appendix ???. The remainder of this section describes the specific solvers available for electrostatic calculations, also described in more detail on the APBS website [4].

#### 3.1 Finite difference and finite element solvers

The original version of APBS was based on two key libraries from the Holst research group. FEtk is a general-purpose multi-level adaptive finite element library [? ? ]. Adaptive finite element methods can resolve extremely fine features of a complex system (like biomolecules) while solving the associated equations over large problem domain. For example, FEtk has been used to solve electrostatic and diffusion equations over six orders of magnitude in length scale [? ]. The finite difference PMG solver [? ? ] trades speed and efficiency for the high-accuracy and high-detail solutions of the finite element FEtk library. However, many APBS users need only a relatively coarse-grained solution of  $\phi$  for their visualization or simulation applications. Therefore, most APBS users employ the Holst group's finite difference grid-based PMG solver for biomolecular electrostatics calculations. Appendix B describes some of the common configuration options for finite difference and finite element calculations in APBS.

#### 3.2 Geometric flow

Several recent papers have described our work on a geometric flow formulation of Poisson-based implicit solvent models [32? ]. The geometric flow approach couples the polar and nonpolar components of the implicit solvent model with two primary benefits. First, this coupling eliminates the need for an *ad hoc* geometric definition for the solute-solvent boundary. In particular, the solute-solvent interface is optimized as part of the geometric flow calculation. Second, the optimization of this boundary ensures self-consistent calculation of polar and nonpolar energetic contributions (using the same surface definitions, etc.), thereby reducing confusion and the likelihood of user error. Additional information about the geometric flow implementation in APBS is provided in Appendix C. This equation is solved in APBS using a finite difference method.

#### 3.3 Boundary element methods

Boundary element methods offer the ability to focus numerical effort on a much smaller region of the problem domain: the interface between the molecule and the solvent. APBS now includes a treecode accelerated boundary integral PB solver (TABI-PB) developed by Geng and Krasny to solve the linearized PB equation [33]. In this method, two coupled integral equations defined on the solute-solvent boundary define a mathematical relationship between the electrostatic surface potential and its normal derivative with a set of integral kernels consisting of Coulomb and screened Coulomb potentials with their normal derivatives. The boundary element method requires a surface triangulation, generated by a program such as MSMS [34] or NanoShaper [35], on which to discretize the integral equations. The resulting linear system is then solved with GMRES iteration [36]. A Cartesian particle-cluster treecode is used to compute matrix-vector products and reduce the computational cost of this dense system from  $\mathcal{O}(N^2)$  to  $\mathcal{O}(N \log N)$  [37, 38]. Typical results from a boundary element calculation are shown in Figure. 2. Additional information about the boundary

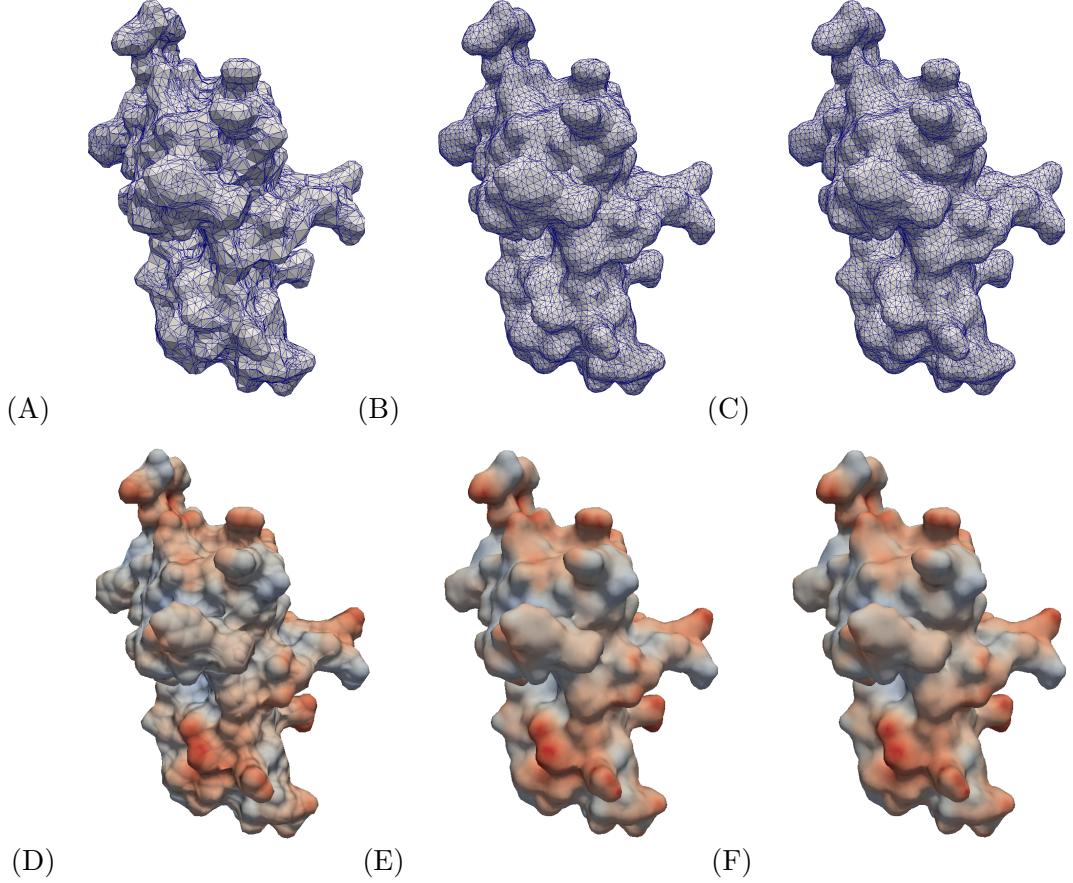


Figure 2: Boundary element calculations for PDB ID 1a63. Surfaces were generated with (A) 20228 triangles via MSMS [34], (B) 20744 triangles via NanoShaper SES, and (C) 21084 triangles via NanoShaper Skin [35]. These discretizations resulted in surface potentials of (D) MSMS in range  $[-8.7, 8.6]$   $kT/e$ , (E) NanoShaper SES in range  $[-13.4, 7.5]$   $kT/e$ , and (F) NanoShaper Skin in range  $[-33.8, 8.0]$   $kT/e$ .

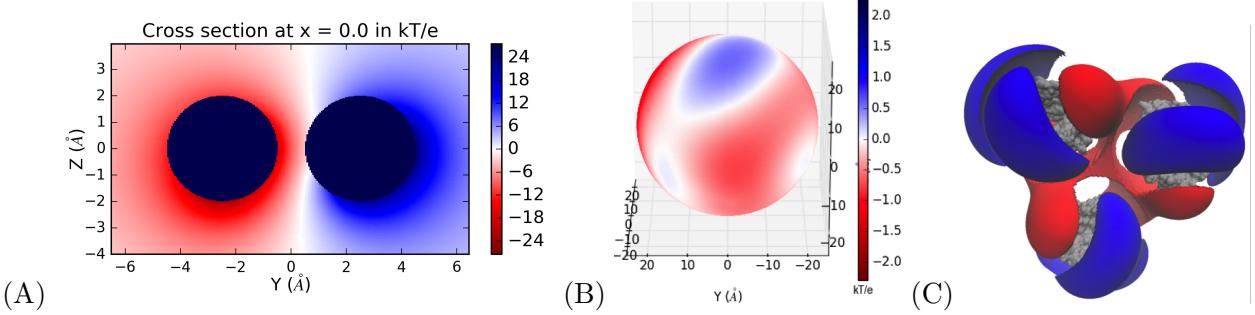


Figure 3: APBS PB-AM electrostatic potential results for (A) 2D cross section of two collections of point charges (each with net charge  $\pm 6$ ) **\*\* Why is the zero-point of the isocontour off center? \*\***, (B) potential on the spherical coarse-grain surface of a barstar molecule ( $k_B T/e$ ), (C) VMD [40] of the transmembrane trimer Omp32 porin (molecules are given in grey and isosurfaces are drawn at 1.0 (blue) and -1.0 ( $k_B T/e$ ). All calculations were performed at 0.0 M ionic strength, pH 7, protein dielectric 2, and solvent dielectric 78.

element method and its in APBS is provided in Appendix D.

### 3.4 Analytical and semi-analytical methods

Numerical solution methods tend to be computationally intensive which has led to the adoption analytical approaches for solvation calculations such as generalized Born [?] and the approaches developed by Head-Gordon implemented in APBS. The Poisson-Boltzmann Analytical Method (PB-AM), was developed by Lotan and Head-Gordon in 2006 [39]. PB-AM produces a fully analytical solution to the linearized PB equation for multiple macromolecules, represented as coarse-grained low-dielectric spheres. This spherical domain enables the use of a multipole expansion to represent charge-charge interactions and higher order cavity polarization effects. The interactions can then be used to compute physical properties such as interaction energies, forces, and torques. An example of this approximation, and the resulting electrostatic potentials from PB-AM, are shown in Figure 3.

The Poisson Boltzmann Semi-Analytical method (PB-SAM) is a modification of PB-AM that incorporates the use of boundary integrals into its formalism to represent a complex molecular domain as a collection of overlapping low dielectric spherical cavities [41]. PB-SAM produces a semi-analytical solution to the linearized PB equation for multiple macromolecules in a screened environment. This semi-analytical method provides a better representation of the molecular boundary when compared to PB-AM, while maintaining computational efficiency. An example of this approximation, and the resulting electrostatic potentials from PB-SAM, are shown in Figure 4.

Because it is fully analytical, PB-AM can be used for model validation as well as for representing systems that are relatively spherical in nature, such as globular proteins and colloids. PB-SAM, on the other hand has a much more detailed representation of the molecular surface and can therefore be used for many systems that other APBS (numerical) methods are currently used for. Through APBS, both programs can be used to compute the electrostatic potential at any point in space, report energies, forces, and torques of a system of macromolecules, and simulate a system using a BD scheme [42]. Additional details about these methods and their use in APBS are presented in Appendix E.

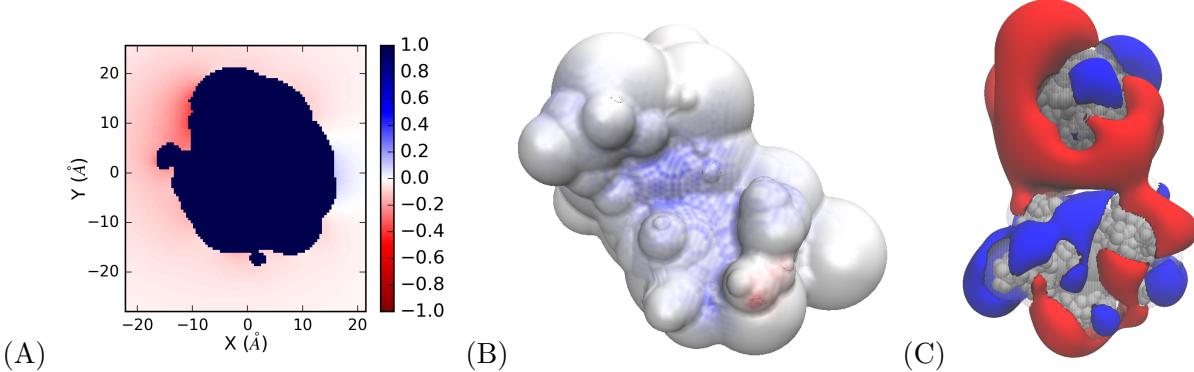


Figure 4: APBS PB-SAM electrostatic potential results for (A) potential in a 2D plane surrounding the barstar molecule ( $k_B T/e$ ), (B) potential on the coarse-grain surface of the barnase molecule (blue region is the location of barstar association, max =  $1.0\ k_B T/e$ , min =  $-1.0\ k_B T/e$ ), (C) VMD [40] visualization of the ESP around the association of barnase and barstar (molecules are given in grey and isosurfaces are drawn at  $1.0$  (blue) and  $-1.0$  (red)  $k_B T/e$ ). All calculations were performed at  $0.0\ \text{M}$  ionic strength, pH 7, protein dielectric 2, and solvent dielectric 78.

## 4 Using APBS results

### 4.1 Visualization

One of the primary uses of the APBS tools is to generate electrostatic potentials for use in biomolecular visualization software. These packages offer both the ability to visualize APBS results as well as a graphical interface for setting up the calculation. Several of these software packages are thick clients that run from users' computers, including PyMOL<sup>1</sup> [? ], VMD<sup>2</sup> [40], PMV<sup>3</sup> [? ], and Chimera<sup>4</sup> [43]. We have also worked with the developers of Jmol<sup>5</sup> [44] and 3Dmol<sup>6</sup> [45] to provide web-based setup and visualization of APBS-PDB2PQR calculations and related workflows. APBS integration with Jmol has been described previously [? ]. 3Dmol is a molecular viewer that offers the performance of a desktop application and convenience of a web-based viewer which broadens accessibility for all users. In order to integrate 3Dmol, enhancements were needed for our tools, including extending our output file formats and creating a customized user interface. Data from the APBS output file is used to generate surfaces, apply color schemes, and display different molecular styles such as cartoons and spheres. An example of the 3Dmol interface is shown in Figure 5. Examples of 3Dmol visualization options are shown in Figure 6.

### 4.2 Other applications

Besides visualization and the processes described in Section ??, there are a number of other applications where APBS can be used. For example, during the past four years, the APBS-PDB2PQR software has been used in the post-simulation energetic analyses of molecular dynamics trajectories [46], understanding protein-nanoparticle interactions [47? ? ], understanding nucleic acid-ion in-

<sup>1</sup>PyMOL website: <http://www.pymol.org>.

<sup>2</sup>VMD website: <http://www.ks.uiuc.edu/Research/vmd/>.

<sup>3</sup>PMV website: <http://mglttools.scripps.edu/>.

<sup>4</sup>Chimera website: <https://www.cgl.ucsf.edu/chimera/>.

<sup>5</sup>Jmol website: <http://jmol.sourceforge.net/>.

<sup>6</sup>3Dmol website: <http://3dmol.csb.pitt.edu/>.

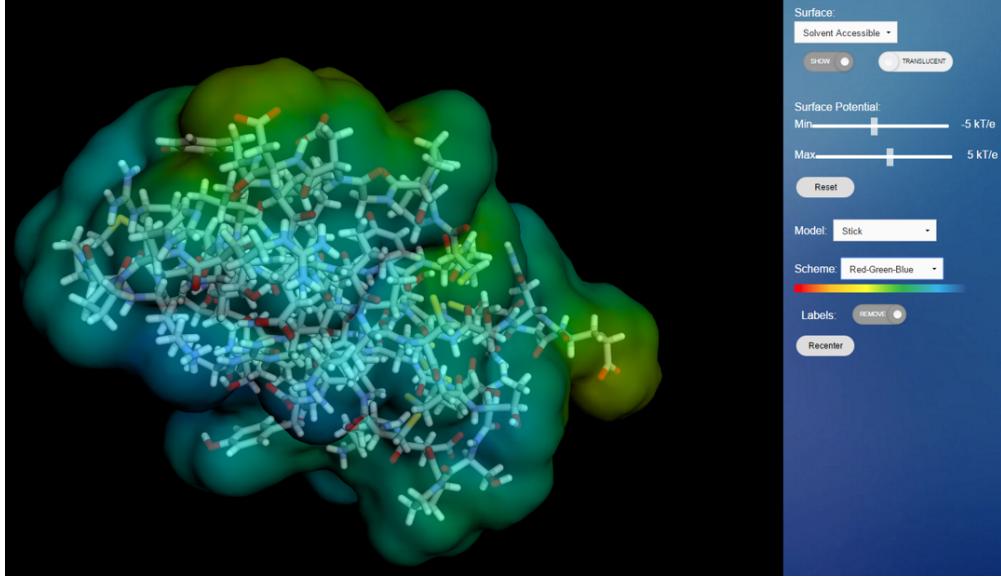


Figure 5: 3Dmol interface displaying a rendering of fasciculin-2 (1FAS) protein with translucent, solvent accessible surface using a stick model and red-green-blue color scheme.

teractions [48? ], biomolecular docking [49] and ligand binding [? ], developing new coarse-grained protein models [50], setting up membrane protein simulations [51], etc. APBS also plays a key role in PIPSA for protein surface electrostatics analysis [52] and SDA for simulation of protein-protein interactions through Brownian dynamics [53]. As discussed above with PB-SAM, another application area for implicit solvent methods is in the evaluation of biomolecular kinetics where implicit solvent models are generally used to provide solvation forces (or energies) for performing (or analyzing) discrete molecular or continuum diffusion simulations with APBS in both of these areas [46, 53? ? –56].

## 5 The future of APBS

To help with modularity and to facilitate extensibility, APBS was based on a object-oriented programming paradigm with strict ANSI-C compliance. This “Clean OO C” [57] has enabled the long-term sustainability of APBS and by combining an object-oriented design framework with the portability and speed of C. This object-oriented design framework has made it relatively straightforward to extend APBS functionality and incorporate new features.

The Clean OO C design has served APBS well for the past 17 years. This design still forms the basis for many modules of APBS and PDB2PQR. Additionally, we have begun to explore a framework for integrating components that exploits the common workflows used by APBS-PDB2PQR users and maps naturally to cloud-based resources. Our vision for APBS is to build the infrastructure that can enable our users to implement their own models and methods so that they can run on a common system. Our goal is to have a well-designed, well-tested and well-documented industry-grade framework that will integrate the APBS-PDB2PQR capabilities and make straightforward the incorporation of new features and new workflows.

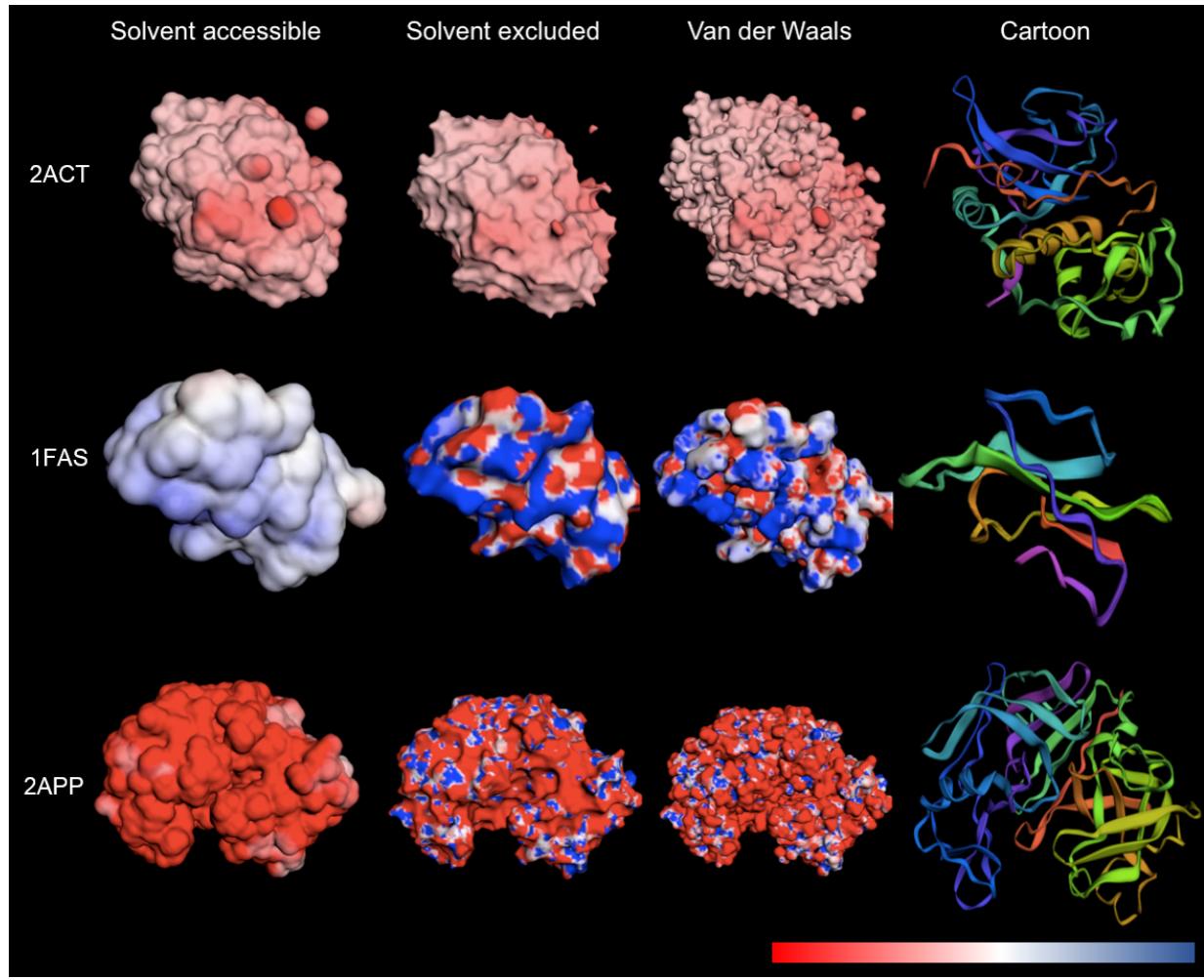


Figure 6: Renderings of three different proteins using renderings of actinidin (2ACT) (top), fasciculin-2 (1FAS) (center), and pepsin-penicillium (2APP) (bottom). To demonstrate the different visualization options. From left to right: solvent-accessible surface, solvent-excluded surface, van der Waals surface, and cartoon models are shown all using red-white-blue color scheme (excluding cartoon model).

## Acknowledgments

The authors gratefully acknowledge NIH grant GM069702 for support of this research. PNNL is operated by Battelle for the U.S. DOE under contract DE-AC05-76RL01830.

## A General keywords for implicit solvent calculations

This section contains information about the general keywords used to configure implicit solvent calculations in the ELEC section of APBS input files. These keywords are used in all of the solver types described in this paper:

- **ion charge**  $\langle \text{charge} \rangle$  **conc**  $\langle \text{conc} \rangle$  **radius**  $\langle \text{radius} \rangle$ : This line can appear multiple times to specify the ionic species in the calculation. This specifies the following terms in Eq. 1:  $\langle \text{charge} \rangle$  gives  $q_i$  in units of electrons,  $\langle \text{conc} \rangle$  gives  $c_i$  in units of M, and  $\langle \text{radius} \rangle$  specifies the ionic radius (in Å) used to calculate  $V_i$ .
- **lpbe|npbe**: This keyword indicates whether to solve the full nonlinear version of Eq. 1 (**npbe**) or a linearized version (**lpbe**).
- **pdie** and **sdie**: These keywords specify the dielectric coefficient values for the biomolecular interior (**pdie**) and bulk solvent (**sdie**). A typical value for **sdie** is 78.54; values for **pdie** are much more variable and often range from 2 to 40.
- **temp**  $\langle \text{temp} \rangle$ : The temperature (in K) for the calculation. A typical value is 298 K.

More detailed information can be found on the APBS website [4].

## B Finite element and finite difference calculations in APBS

### B.1 Finite difference calculation configuration

APBS users have several ways to invoke the finite difference solver capabilities of APBS:

\*\* FIX \*\* APBS users can invoke TABI-PB with the **bem-manual** flag in the ELEC section of the input file. \*\* Finish this section. Include keywords and typical setting values. \*\*

### B.2 Finite elements calculation configuration

\*\* FIX \*\* APBS users can invoke TABI-PB with the **bem-manual** flag in the ELEC section of the input file. \*\* Finish this section and/or move all of these sections to an appendix. Include keywords and typical setting values. \*\*

## C Geometric flow calculations in APBS

This section contains additional information about the geometric flow equation implementation in APBS introduced in Section 3.2.

\*\* STOPPED HERE \*\*

### C.1 Geometric flow equation background

For our modeling, a generalized form of the Poisson equation for computing the electrostatic is used as shown in Eq. 2, where  $\epsilon_m$  and  $\epsilon_s$  are the dielectric coefficients of the solute and solvent, respectively, and  $\rho_m$  is the charge distribution of the fixed solute molecule. For this model, the solutions for the electrostatic potential ( $\phi$ ) and the characteristic function ( $S$ ) are obtained by minimizing the free energy functional. The dielectric function  $\epsilon(S)$  takes on the value  $\epsilon_m$  in the solute region ( $S=1$ ) and the value  $\epsilon_s$  in the solvent region ( $S=0$ ).

$$-\nabla \cdot (\epsilon(S) \nabla \phi) = S \rho_m \quad (2)$$

For this model, we have utilized an Eulerian formulation of the geometric flow problem. This means that  $S$  varies smoothly across the solute-solvent interfaces and leads to a non-linear partial differential equation for the characteristic function,  $S$ , as shown in Eq. 3.

$$-\nabla \cdot \left( \gamma \frac{\nabla S}{\|\nabla S\|} \right) + p - \rho_0 U^{att} + \rho_m \phi - \frac{1}{2} \epsilon_m |\nabla \phi|^2 + \frac{1}{2} \epsilon_s |\nabla \phi|^2 = 0 \quad (3)$$

where  $\gamma$  is the microscopic surface tension,  $p$  is the hydrodynamic pressure, and  $U^{att}$  is the attractive potential of the van der Waals dispersion interaction between the solute and the solvent.

The solution to this non-linear PDE can be solved by utilizing the parabolic PDE shown in Eq. 4. This equation is known as the generalized geometric flow equation and it is coupled with the Poisson equation through the characteristic function  $S$  [58].

$$\frac{\partial S}{\partial t} = \gamma \|\nabla S\| \left[ \nabla \cdot \left( \frac{\nabla S}{\|\nabla S\|} \right) + \frac{V}{\gamma} \right], \quad (4)$$

where  $V$  is known as the generalized flow potential as described in Eq. 5.

$$V = -p + \rho_0 U^{att} - \rho_m \phi + \frac{\epsilon_m}{2} |\nabla \phi|^2 - \frac{\epsilon_s}{2} |\nabla \phi|^2. \quad (5)$$

These equations were solved using a second-order central finite difference scheme, which utilizes a bi-conjugate gradient stabilized solver.

The sensitivity and accuracy of an implicit solvent method typically depends on the values selected for the force-field parameters. However, the geometric flow method is only dependent on the parameters of the implicit solvent model (surface tension, dielectric coefficient, solvent pressure, etc.).

In a sensitivity analysis performed by Thomas et. al. [32], changing the distance between the axis boundary and the surface of each molecule and the grid spacing for the solver showed little effect on the results (3-5%). However, changes in the internal dielectric ( $\epsilon_m$ ) resulted in much larger sensitivity (i.e., changes in the results).

### C.2 Geometric flow calculation configuration

**\*\* FINISH \*\*\* STOPPED HERE \*\***

## D Boundary element method implementation

This appendix provides additional information about the boundary element method introduced in Section 3.3.

## D.1 Boundary element method background

\*\* STOPPED HERE \*\* The matrix-vector products involve evaluation of the integral kernels over the surface elements. These evaluations effectively take the form of an  $N$ -body potential: a sum over a set of  $N$  positions of products between a kernel and a "charge" at each position. In this case, the locations of the  $N$  particles are the centroids of the surface triangularization elements. To rapidly evaluate the  $N$ -body potential at the  $N$  particle locations, the treecode subdivides the particles into a tree-like hierarchical structure of clusters. At each location, the potential contribution from nearby particles is computed by direct sum, while for far particle clusters, a Taylor approximation about the center of the cluster is used to evaluate the contribution. The Taylor coefficients are calculated through recurrence relations.

The TABI-PB method utilizes a well-posed boundary integral PB formulation to ensure rapid convergence. In addition, a fast treecode algorithm for the screened Coulomb potential is applied to speed up the matrix-vector products in each GMRES iteration [38]. The TABI-PB solver serves as an effective alternative to the multi-grid finite difference solver when the surface potential accuracy or system memory capacity becomes a concern.

The coupled second kind integral equations employed by TABI-PB for calculating the surface potential  $\phi_1$  and its normal derivative [38], are as follows:

$$\begin{aligned} \frac{1}{2}(1+\varepsilon)\phi_1(\mathbf{x}) &= \int_{\Gamma} \left[ K_1(\mathbf{x}, \mathbf{y}) \frac{\partial\phi_1(\mathbf{y})}{\partial\nu} + K_2(\mathbf{x}, \mathbf{y})\phi_1(\mathbf{y}) \right] dS_{\mathbf{y}} + S_1(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \\ \frac{1}{2}\left(1+\frac{1}{\varepsilon}\right) \frac{\partial\phi_1(\mathbf{x})}{\partial\nu} &= \int_{\Gamma} \left[ K_3(\mathbf{x}, \mathbf{y}) \frac{\partial\phi_1(\mathbf{y})}{\partial\nu} + K_4(\mathbf{x}, \mathbf{y})\phi_1(\mathbf{y}) \right] dS_{\mathbf{y}} + S_2(\mathbf{x}), \quad \mathbf{x} \in \Gamma \end{aligned} \quad (6)$$

where  $\varepsilon = \varepsilon_m/\varepsilon_s$ , the ratio of the dielectric constant in the solute region and the dielectric constant in the solvent region. The integral kernels  $K_1, K_2, K_3, K_4$  are defined in Eq. 7.

$$\begin{aligned} K_1(\mathbf{x}, \mathbf{y}) &= G_0(\mathbf{x}, \mathbf{y}) - G_{\kappa}(\mathbf{x}, \mathbf{y}), \quad K_2(\mathbf{x}, \mathbf{y}) = \varepsilon \frac{\partial G_{\kappa}(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{y}}} - \frac{\partial G_0(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{y}}} \\ K_3(\mathbf{x}, \mathbf{y}) &= \varepsilon \frac{\partial G_0(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{x}}} - \frac{1}{\varepsilon} \frac{\partial G_{\kappa}(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{x}}}, \quad K_4(\mathbf{x}, \mathbf{y}) = \varepsilon \frac{\partial^2 G_{\kappa}(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{x}}\partial\nu_{\mathbf{y}}} - \frac{1}{\varepsilon} \frac{\partial^2 G_0(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{x}}\partial\nu_{\mathbf{y}}} \end{aligned} \quad (7)$$

where  $G_0$  and  $G_{\kappa}$  are the Coulomb and screened Coulomb potentials defined as:

$$G_0(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi|\mathbf{x}-\mathbf{y}|}, \quad G_{\kappa}(\mathbf{x}, \mathbf{y}) = \frac{e^{-\kappa|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|} \quad (8)$$

The normal derivatives of the potential kernels  $G$  are:

$$\begin{aligned} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{y}}} &= \sum_{n=1}^3 \nu_n(\mathbf{y}) \partial_{y_n} G(\mathbf{x}, \mathbf{y}), \\ \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{x}}} &= - \sum_{n=1}^3 \nu_n(\mathbf{y}) \partial_{x_n} G(\mathbf{x}, \mathbf{y}), \\ \frac{\partial^2 G(\mathbf{x}, \mathbf{y})}{\partial\nu_{\mathbf{x}}\partial\nu_{\mathbf{y}}} &= - \sum_{m=1}^3 \sum_{n=1}^3 \nu_m(\mathbf{x}) \nu_n(\mathbf{y}) \partial_{x_n} \partial_{y_m} G(\mathbf{x}, \mathbf{y}) \end{aligned} \quad (9)$$

for the three spatial components  $n$  of the normal direction. Additionally, the source terms  $S_1$  and  $S_2$  in Eq. 6 are:

$$S_1(\mathbf{x}) = \frac{1}{\varepsilon_m} \sum_{k=1}^{N_c} q_k G_0(\mathbf{x}, \mathbf{y}_k), \quad S_2(\mathbf{x}) = \frac{1}{\varepsilon_m} \sum_{k=1}^{N_c} q_k \frac{\partial G_0(\mathbf{x}, \mathbf{y}_k)}{\partial\nu_{\mathbf{x}}} \quad (10)$$

where  $N_c$  is the number of atoms in the solute molecule, and  $q_k$  is the charge of the  $k$ th atom. Note that  $S_1$  is a linear superposition of the point charge electrostatic potentials, and  $S_2$  is a linear superposition of the normal derivatives of the potentials.

Given a surface triangularization with  $N$  elements, where  $\mathbf{x}_i$  and  $A_i$  are the centroid and area, respectively, of the  $i$ th triangle, the integral equations are discretized as:

$$\begin{aligned} \frac{1}{2}(1+\varepsilon)\phi_1(\mathbf{x}_i) &= \sum_{\substack{j=1 \\ j \neq i}}^N \left[ K_1(\mathbf{x}_i, \mathbf{x}_j) \frac{\partial\phi_1(\mathbf{x}_j)}{\partial\nu} + K_2(\mathbf{x}_i, \mathbf{x}_j) \phi_1(\mathbf{x}_j) \right] A_j + S_1(\mathbf{x}_i), \\ \frac{1}{2}\left(1+\frac{1}{\varepsilon}\right) \frac{\partial\phi_1(\mathbf{x}_i)}{\partial\nu} &= \sum_{\substack{j=1 \\ j \neq i}}^N \left[ K_3(\mathbf{x}_i, \mathbf{x}_j) \frac{\partial\phi_1(\mathbf{x}_j)}{\partial\nu} + K_4(\mathbf{x}_i, \mathbf{x}_j) \phi_1(\mathbf{x}_j) \right] A_j + S_2(\mathbf{x}_i) \end{aligned} \quad (11)$$

The omission of the  $j = i$  term in the summation avoids the singularity of the kernels at that point. Note that the right hand sides of these equations consist of sums of products of kernels and the surface potential or its normal derivative. These are the analogues to the  $N$ -body potential in the treecode, with the surface potential or its normal derivatives playing the role of the charges. In the discretized form, the total electrostatic energy of solvation is given by Eq. 12.

$$E_{\text{sol}} = \frac{1}{2} \sum_{k=1}^{N_c} q_k \sum_{j=1}^N \left[ K_1(\mathbf{x}_k, \mathbf{x}_j) \frac{\partial\phi_1(\mathbf{x}_j)}{\partial\nu} + K_2(\mathbf{x}_k, \mathbf{x}_j) \phi_1(\mathbf{x}_j) \right] A_j \quad (12)$$

where  $q_k$  is the charge on the  $k$ th atom of the solute molecule, and  $\mathbf{x}_k$  is its position.

Several unique features of the boundary integral formulation are worth mentioning. Because the integral equations are defined on the molecular boundary, the singular charges are handled analytically and do not introduce the same issues present in grid-based schemes. The integral equations also rigorously enforce the interface conditions on the surface, and the boundary condition at infinity is exactly satisfied. Thus, the boundary integral formulation can potentially be superior to other methods for investigating electrostatic potential on the boundary. However, because the method is only well defined for the linearized PB equation, it may perform poorly for high salt concentration in the solvent.

## D.2 Boundary element calculation configuration

APBS users can invoke TABI-PB with the `bem-manual` flag in the ELEC section of the input file. Major options include:

- `tree_order <order>`: An integer indicating the order of the Taylor expansion for determining treecode coefficients. Higher values of `<order>` will result in a more accurate – but more expensive – calculations. A typical choice for this parameter is 3.
- `tree_n0 <number>`: The maximum number of particles allowable in a leaf of the treecode (clusters in the last level of the tree). A typical choice for this parameter is 500.
- `mac <criterion>`: Multipole acceptance criterion specifies the distance ratio at which the Taylor expansion is used. In general, a higher value of `<criterion>` will result in a more accurate but more expensive computation; while a lower value causes more direct summations and forces the particle-cluster interaction to descend to a finer cluster level. A typical choice for this parameter is 0.8.

- **mesh**  $\langle flag \rangle$ : The software used to mesh the molecular surface; 0 = MSMS, 1 = NanoShaper’s SES implementation, and 2 = NanoShaper’s Skin implementation. See Figure 2 for an example of surface meshes.
- **outdata**  $\langle flag \rangle$ : Type of output data file generated; 0 = APBS OpenDX format [? ] and 1 = ParaView format [? ].

Additional information about parameter settings is provided via the APBS website [4]. TABI-PB produces output including the potential and normal derivative of potential for every element and vertex of the triangularization, as well as the electrostatic solvation energy. Examples of electrostatic surface potential on the protein 1a63 are shown in Figure 2 by using MSMS and NanoShaper.

## E Analytical and semi-analytical method implementations

This appendix provides additional information about the analytical and semi-analytic methods introduced in Section 3.4.

### E.1 Analytical method background

\*\* STOPPEDHERE \*\* The solution to this model is represented as a system of linear equations:

$$\mathbf{A} = \Gamma \cdot (\Delta \cdot T \cdot \mathbf{A} + \mathbf{E}), \quad (13)$$

where  $\mathbf{A}$  represents a vector of the effective multipole expansion of the charge distributions of each molecule,  $\mathbf{E}$  is a vector of the fixed charge distribution of all molecules,  $\Gamma$  is a dielectric boundary-crossing operator,  $\Delta$  is a cavity polarization operator, and  $T$  is an operator that transforms the multipole expansion from the global (lab) coordinates to a local coordinate frame. The unknown  $\mathbf{A}$  determined using the Gauss-Seidel iterative method and can then be used to compute physical properties such as interaction energies, forces, and torques. \*\* STOPPED-HERE \*\* The interaction energy for molecule  $i$ ,  $(\Omega^{(i)})$  is given in Eq. 14.

$$\Omega^{(i)} = \frac{1}{\epsilon_s} \left\langle \sum_{j \neq i}^N T \cdot A^{(j)}, A^{(i)} \right\rangle \quad (14)$$

where  $\epsilon_s$  is the dielectric constant of the solvent and  $\langle M, N \rangle$  denotes the inner product. When energy is computed, forces follow as:

$$\mathbf{F}^{(i)} = \nabla_i \Omega^{(i)} = \frac{1}{\epsilon_s} [\langle \nabla_i T \cdot A^{(i)}, A^{(i)} \rangle + \langle T \cdot A^{(i)}, \nabla_i A^{(i)} \rangle] \quad (15)$$

By definition, the torque on a charge in the molecule is the cross product of its position relative to the center of mass of the molecule with the force it experiences. The total torque on the molecule is a linear combination of the torque on all charges of the molecule, as illustrated in Eq. 16.

$$\tau^{(i)} = \frac{1}{\epsilon_s} \left[ {}^x H^{(i)}, {}^y H^{(i)}, {}^z H^{(i)} \right] \times \left[ \nabla_i L^{(i)} \right] \quad (16)$$

where  ${}^\alpha H_{n,m}^{(i)} = \sum_{j=1}^{M_i} \alpha_j^{(i)} \gamma_n^{(i)} q_j^{(i)} (\rho_j^{(i)})^n Y_{n,m}(\vartheta_j^{(i)}, \varphi_j^{(i)})$ ,  $\alpha = x, y, z$ , is a coefficient vector for each of the charges in the molecule,  $M_i$  is the number of charges in molecule  $i$ ,  $q_j^{(i)}$  is the magnitude of the  $j^{th}$  charge, and  $p_j^{(i)} = [\rho_j^{(i)}, \vartheta_j^{(i)}, \varphi_j^{(i)}]$  is its position in spherical coordinates. For more details on the PB-AM derivation, see Lotan, Head-Gordon [39].

## E.2 Semi-Analytical method background

The derivation details of PB-SAM have been reported previously [? ? ], with the main points being summarized in this section. The electrostatic potential ( $\phi_r$ ) of the system at any point  $r$  is governed by the linearized form of the PB equation:

$$\nabla \cdot [\epsilon(\mathbf{r}) \nabla \phi(\mathbf{r})] - \epsilon(\mathbf{r}) \kappa^2 \phi(\mathbf{r}) = 4\pi \rho(\mathbf{r}) \quad (17)$$

where  $\kappa$  is the inverse Debye length. For the case of spherical cavities, we can solve Eq. 17 by dividing the system into inner sphere and outer sphere regions, and enforcing a set of boundary conditions that stipulate the continuity of the electrostatic potential and the electrostatic field at the surface of each sphere. The electrostatic potential outside molecule ( $I$ ) is described by:

$$\phi_{out}^{(i)}(\mathbf{r}) = \sum_{I=1}^{N_{mol}} \left( 4\pi \int_{d\Omega^{(I)}} \frac{e^{-\kappa|r-r'|}}{|r-r'|} h^{(I)}(r') dr' \right) \quad (18)$$

where  $h(r)$  is an effective surface charge that can be transformed into the unknown multipole expansion  $H^{(I,k)}$  with inside molecule  $I$  and sphere  $k$ . In a similar manner, the interior potential is given as

$$\phi_{in}^{(i)}(\mathbf{r}) = \sum_{\alpha=1}^{N_C^{(I)}} \frac{1}{|r-r_{\alpha}^{(I)}|} \cdot \frac{q_{\alpha}^{(I)}}{\epsilon_{in}} + \frac{1}{4\pi} \int_{d\Omega^{(I)}} \frac{1}{|r-r'|} f^{(I)}(r') dr' \quad (19)$$

where  $N_C^{(I)}$  is the number of charges in molecule  $I$ ,  $q_{\alpha}$  is the magnitude of the  $\alpha$ -th charge,  $r_{\alpha}^{(I)} = [\rho_{\alpha}^{(I)}, \theta_{\alpha}^{(I)}, \phi_{\alpha}^{(I)}]$  is its position in spherical coordinates, and  $f(r)$  is a reactive surface charge that can be transformed into the unknown multipole expansion  $F^{(I,k)}$ . The reactive multipole and the effective multipole,  $H^{(I,k)}$ , are given as:

$$F_{n,m}^{(I,k)} \equiv \frac{1}{4\pi} \int_{d\Omega^{(I,k)}} f^{(I,k)}(r') \left( \frac{a^{(I,k)}}{r'} \right)^{n+1} \overline{Y_{n,m}^{(I,k)}}(\theta', \phi') dr' \quad (20)$$

$$H_{n,m}^{(I,k)} \equiv \frac{1}{4\pi} \int_{d\Omega^{(I,k)}} h^{(I,k)}(r') \left( \frac{r'}{a^{(I,k)}} \right)^n \hat{i}_n(\kappa r') \overline{Y_{n,m}^{(I,k)}}(\theta', \phi') dr' \quad (21)$$

where  $Y_{n,m}$  is the spherical harmonics,  $\overline{Y_{n,m}}$  is the complete conjugate, and  $a^{(I,k)}$  is the radius of sphere  $k$  of molecule  $I$ . These multipole expansions can be iteratively solved using:

$$F_{n,m}^{(I,k)} = \langle I_{E,n,m}^{(I,k)}, WF^{(I,k)} \rangle \quad (22)$$

$$H_{n,m}^{(I,k)} = \langle I_{E,n,m}^{(I,k)}, WH^{(I,k)} \rangle \quad (23)$$

where  $WF^{(I,k)}$  and  $WH^{(I,k)}$  are scaled multipoles computed from fixed charges and polarization charges from other spheres.  $I_{E,n,m}^{(I,k)}$  is a matrix of the surface integrals over the exposed surface:

$$I_{E,n,m}^{(I,k)} \equiv \frac{1}{4\pi} \int_{\phi_E} \int_{\theta_E} Y_{l,s}^{(I,k)}(\theta', \phi') \overline{Y_{n,m}^{(I,k)}}(\theta', \phi') \sin\theta' d\theta' d\phi' \quad (24)$$

Using the above formalism, physical properties of the system, such as interaction energy, forces and torques may be computed. The interaction energy of each molecule, ( $\Omega^{(i)}$ ), is the product of the molecule's total charge distribution (from fixed and polarization charges) with the potential

due to external sources. This is computed as the inner product between the molecule's multipole expansion, ( $H^{(I,k)}$ ), and the multipole expansions of the other molecules in the system, ( $LHN^{(I,k)}$ ) as follows:

$$\Omega^{(i)} = \frac{1}{\epsilon_s} \sum_k^{N_k^{(I)}} \langle LHN^{(I,k)}, H^{(I,k)} \rangle \quad (25)$$

which allows us to define the force which is computed as the gradient of the interaction energy with respect to the position of the center of molecule  $I$ :

$$F^{(I)} = -\nabla \Omega^{(I)} = -\frac{1}{\epsilon_s} \sum_k^{N_k^{(I)}} f_{I,k} = -\frac{1}{\epsilon_s} \sum_k^{N_k^{(I)}} (\langle \nabla LHN^{(I,k)}, H^{(I,k)} \rangle + \langle LHN^{(I,k)}, \nabla H^{(I,k)} \rangle) \quad (26)$$

As in the analytical method, the torque on a charge in the molecule is the cross product of its position relative to the center of mass of the molecule with the force it experiences. For a charge at position,  $P$ , about the center of mass  $c^{(I)}$  for molecule  $I$ , the torque is given by the cross product of its position,  $r_P^{(I,k)}$ , with respect to the center of mass and the force on that charge,  $f_P$ . We can re-express  $r_P^{(I,k)}$  as the sum of vectors from the center of molecule  $I$  to the center of sphere  $k$  ( $c^{(I,k)}$ ) and from the center of sphere  $k$  to point  $P$  ( $r_P^{(I,k)}$ ). The total torque on molecule  $I$  is then given by Eq. 27.

$$\tau^{(I)} = \sum_k^{N_k^{(I)}} c^{(I,k)} \times f_{I,k} + \sum_k^{N_k^{(I)}} \sum_{P \in k} r_P^{(I,k)} \times f_P \quad (27)$$

where  $f_{I,k}$  is given in Eq. 26 and

$$f_P = -\frac{1}{\epsilon_s} \sum_k^{N_k^{(I)}} (\langle \nabla_I LHN^{(I,k)}, H_P^{(I,k)} \rangle + \langle LHN^{(I,k)}, \nabla_I H_P^{(I,k)} \rangle) \quad (28)$$

where

$$H_{P,n,m}^{(I,k)} = h(\theta_p, \phi_p) Y_{n,m}^{(I,k)}(\theta_p, \phi_p) \quad (29)$$

$$\nabla_j H_{P,\alpha,n,m}^{(I,k)} = [\nabla_j h(\theta_p, \phi_p)]_\alpha Y_{n,m}^{(I,k)}(\theta_p, \phi_p) \quad (30)$$

where  $\alpha = x, y, z$ . For the derivation of the PB-SAM solver please see the previous publications [? ?].

### E.3 PB-AM and PB-SAM configuration in APBS

PB-AM and PB-SAM have been fully integrated into APBS, and is invoked using the keyword `pbam-auto` or `pbsam-auto` in the ELEC section of an APBS input file. Major options include:

- `runname <name>`: Desired name to be used for outputs of each run.
- `pbc <length>`: Size of the periodic simulation/calculation domain. Typical values for this option are **\*\* FINISH \*\***.
- `runtype dynamics`: Perform a Brownian Dynamics simulation.
- `ntraj <number>`: Number of Brownian Dynamics simulations to run.

- **term**  $\langle type \rangle \langle value \rangle \langle mol \rangle$ : Allows the user to indicate conditions for the termination of each BD trajectory. The following values of  $\langle type \rangle$  are allowed:
  - **time**  $\langle time \rangle$ : **\*\* FINISH \*\***
  - **x** or **y** or **z** or **r** and  $\langle \geq \rangle$  or  $\langle \leq \rangle$ : Represents the approach of two molecules to a certain distance **r** or certain region of space given by **x** or **y** or **z**. The operators  $\geq$  and  $\leq$  represent the corresponding inequalities.

The parameter  $\langle mol \rangle$  is the molecular index that this condition applies.  $\langle mol \rangle$  should be 0 for **time** and 1 for everything else. **\*\* I don't understand this. Why have this parameter at all? I would have expected the parameter to be the molecule index. \*\***

- **xyz**  $\langle idx \rangle \langle fpath \rangle$ : Molecule index  $\langle idx \rangle$  and file path  $\langle fpath \rangle$  for the molecule starting configurations. **\*\* Why not just use the existing APBS READ statements? \*\*** A starting configuration is needed for each molecule and each trajectory. Therefore, if there are  $m$  molecules and **ntraj**  $\langle n \rangle$  trajectories, then the input file must contain  $m \times n$  **xyz** entries.
- **tolsp**  $\langle val \rangle$ : Modify the coarseness of the molecular description.  $\langle val \rangle$  is the distance (in Å) beyond the solvent-excluded surface that the coarse-grained representation extends. Increasing values of  $\langle val \rangle$  leads to fewer coarse-grained spheres, faster calculation times, but less accurate solutions. Typical values for  $\langle val \rangle$  are between 1 and 5 Å.

The commands (keywords) not included in this list are used to specify system conditions, such as temperature and salt concentration. **\*\* Where are those keywords described? \*\*** Additional information about parameter settings is provided via the APBS website [4]. Examples of the electrostatic potentials produced from PB-AM and PB-SAM are shown in Figures 3 and 4, respectively.

## References

- [1] M Holst, N Baker, and F Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. algorithms and examples. *J. Comput. Chem.*, 21(15):1319–1342, November 2000.
- [2] N Baker, M Holst, and F Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. refinement at solvent-accessible surfaces in biomolecular systems. *J. Comput. Chem.*, 21(15):1343–1352, November 2000.
- [3] N A Baker, D. Sept, M J Holst, and J A McCammon. The adaptive multilevel finite element solution of the Poisson-Boltzmann equation on massively parallel computers. *IBM J. Res. Dev.*, 45(3-4):427–438, May-Jul 2001.
- [4] APBS website.
- [5] M. E. Davis and J A McCammon. Electrostatics in biomolecular structure and dynamics. *Chemical Reviews*, 90(3):509–521, 1990.
- [6] M F Perutz. Electrostatic effects in proteins. *Science*, 201(4362):1187–91, Sep 1978.
- [7] P Ren, J Chun, D G Thomas, M J Schnieders, M Marucho, J Zhang, and N A Baker. Biomolecular electrostatics and solvation: a computational perspective. *Q Rev Biophys*, 45(4):427–91, Nov 2012.

- [8] K A Sharp and B Honig. Electrostatic interactions in macromolecules: theory and applications. *Annu Rev Biophys Biophys Chem*, 19:301–32, 1990.
- [9] B Roux and T Simonson. Implicit solvent models. *Biophys Chem*, 78(1-2):1–20, Apr 1999.
- [10] A Warshel, P K Sharma, M Kato, and W W Parson. Modeling electrostatic effects in proteins. *Biochim Biophys Acta*, 1764(11):1647–76, Nov 2006.
- [11] M Fixman. The Poisson-Boltzmann equation and its application to polyelectrolytes. *J. Chem. Phys.*, 70:4995–5005, 1979.
- [12] P Grochowski and J Trylska. Continuum molecular electrostatics, salt effects, and counterion binding—a review of the Poisson-Boltzmann theory and its modifications. *Biopolymers*, 89(2):93–113, 2008.
- [13] G Lamm, K B Lipkowitz, R Larter, T R Cundari, and D B Boyd. The Poisson-Boltzmann equation. *Rev. Comput. Chem.*, 19:147–365, 2003.
- [14] R. R. Netz and H. Orland. Beyond Poisson-Boltzmann: Fluctuation effects and correlation functions. *Eur. Phys. J. E*, 1(2-3):203–214, Feb-Mar 2000.
- [15] B R Brooks, C L Brooks, 3rd, A D Mackerell, Jr, L Nilsson, R J Petrella, B Roux, Y Won, G Archontis, C Bartels, S Boresch, A Caflisch, L Caves, Q Cui, A R Dinner, M Feig, S Fischer, J Gao, M Hodoscek, W Im, K Kuczera, T Lazaridis, J Ma, V Ovchinnikov, E Paci, R W Pastor, C B Post, J Z Pu, M Schaefer, B Tidor, R M Venable, H L Woodcock, X Wu, W Yang, D M York, and M Karplus. Charmm: the biomolecular simulation program. *J Comput Chem*, 30(10):1545–614, Jul 2009.
- [16] D A Case, T E Cheatham, 3rd, T Darden, H Gohlke, R Luo, K M Merz, Jr, A Onufriev, C Simmerling, B Wang, and R J Woods. The AMBER biomolecular simulation programs. *J Comput Chem*, 26(16):1668–88, Dec 2005.
- [17] S Sarkar, S Witham, J Zhang, M Zhenirovskyy, W Rocchia, and E Alexov. Delphi web server: A comprehensive online suite for electrostatic calculations of biological macromolecules and their complexes. *Comm. Comp. Phys.*, 13(1):269–284, 2013.
- [18] AD Bochevarov, E Harder, TF Hughes, JR Greenwood, DA Braden, DM Philipp, D Rinaldo, MD Halls, J Zhang, and RA Friesner. Jaguar: A high-performance quantum chemistry software program with strengths in life and materials sciences. *International J. of Quantum Chem.*, 113(18):2110–2142, 2013.
- [19] JA Grant, BT Pickup, and A Nicholls. A smooth permittivity function for poisson-boltzmann solvation methods. *J. Comp. Chem.*, 22:608–640, 2001.
- [20] Z Chen, N A Baker, and G W Wei. Differential geometry based solvation model II: Lagrangian formulation. *J Math Biol*, 63(6):1139–200, Dec 2011.
- [21] R Konecny, N A Baker, and J A McCammon. iAPBS: a programming interface to adaptive Poisson-Boltzmann solver (APBS). *Comput Sci Discov*, 5(1), Jul 2012.
- [22] iAPBS website.

- [23] Todd J Dolinsky, Jens E Nielsen, J Andrew McCammon, and Nathan A Baker. Pdb2pqr: an automated pipeline for the setup of poisson–boltzmann electrostatics calculations. *Nucleic Acids Res.*, 32(suppl 2):W665–W667, 2004.
- [24] T J Dolinsky, P Czodrowski, Hui Li, J E Nielsen, Jan H Jen, G Klebe, and N A Baker. PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucleic Acids Res.*, 35:W522–5, Jul 2007.
- [25] Chresten R. Sondergaard, Mats HM Olsson, Michal Rostkowski, and Jan H. Jensen. Improved treatment of ligands and coupling effects in empirical calculation and rationalization of pka values. *J. Chem. Theory Comp.*, 7(7):2284–2295, 2011.
- [26] H. Li, A. D. Robertson, and J. H. Jensen. Very fast empirical prediction and rationalization of protein pka values. *Proteins*, 61:704721, 2005.
- [27] Emilie Purvine, Kyle Monson, Elizabeth Jurrus, Keith Starr, and Nathan A. Baker. Energy minimization of discrete protein titration state models using graph theory. *J. Phys. Chem.*, 120:8354–8360, 2016.
- [28] J E Nielsen and G Vriend. Optimizing the hydrogen-bond network in Poisson-Boltzmann equation-based  $pK_a$  calculations. *Proteins*, 43(4):403–12, Jun 2001.
- [29] Paul Czodrowski, Ingo Dramburg, Christoph A. Sotriffer, and Gerhard Klebe. Development, validation, and application of adapted PEOE charges to estimate  $pK_a$  values of functional groups in protein-ligand complexes. *Proteins*, 65(2):424–437, November 2006.
- [30] J M J Swanson, S A Adcock, and J A McCammon. Optimized radii for Poisson-Boltzmann calculations with the AMBER force field. *J Chem Theory Comput*, 1(3):484–93, May 2005.
- [31] Chunhu Tan, Lijiang Yang, and Ray Luo. How well does poissonboltzmann implicit solvent agree with explicit solvent? a quantitative analysis. *J. Phys. Chem. B*, 110(37):18680–18687, September 2006.
- [32] D G Thomas, S Gaheen, S L Harper, M Fritts, F Klaessig, E Hahn-Dantona, D Paik, S Pan, G A Stafford, E T Freund, J D Klemm, and N A Baker. ISA-TAB-Nano: a specification for sharing nanomaterial research data in spreadsheet-based format. *BMC Biotechnol*, 13:2, 2013.
- [33] Weihua Geng and Robert Krasny. A treecode-accelerated boundary integral poisson-boltzmann solver for electrostatics of solvated biomolecules. *J. Comp. Phys.*, 247:62–78, 2013.
- [34] Michel Sanner, Arthur Olson, and Jean Claude Spehner. Fast and robust computation of molecular surfaces. In *Proc 11th ACM Symp Comp Geom*, pages C6–C7. ACM, 1995.
- [35] Sergio Decherchi and Walter Rocchia. A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale. *PLoS ONE*, 8(4):e59744, 2013.
- [36] Y. Saad and M. Schultz. Gmres: A generalized minimal residual algorithm for solving non-symmetric linear systems. *J. Sci. Statist. Comput.*, 7:856–869, 1986.
- [37] Peijun Li, Hans Johnston, and Robert Krasney. A cartesian treecode for screened coulomb interactions. *J. Comp. Phys.*, 228:3858–3868, 2009.

- [38] André Juffer, Eugen Botta, Bert van Keulen, Auke van der Ploeg, and Herman Berendsen. The electric potential of a macromolecule in a solvent: A fundamental approach. *J. Comp. Phys.*, 97(1):144–171, 1991.
- [39] Itay Lotan and Teresa Head-Gordon. An analytical electrostatic model for salt screened interactions between multiple proteins. *J. Chem. Theory and Comp.*, 2(3):541–555, 2006. PMID: 26626662.
- [40] W Humphrey, A Dalke, and K Schulten. VMD: visual molecular dynamics. *J Mol Graph*, 14(1):33–8, 27–8, Feb 1996.
- [41] E-H Yap and T Head-Gordon. New and efficient Poisson-Boltzmann solver for interaction of multiple proteins. *J. Chem. Theory Comput.*, 6(7):2214–2224, July 2010.
- [42] Donald Ermak and J. A. McCammon. Brownian dynamics with hydrodynamic interactions. *J. Chem. Phys.*, 69(4):1352–1360, 1978.
- [43] E F Pettersen, T D Goddard, C C Huang, G S Couch, D M Greenblatt, E C Meng, and T E Ferrin. Ucsf chimera—a visualization system for exploratory research and analysis. *J Comput Chem*, 25(13):1605–12, Oct 2004.
- [44] Angel Herraez. Biomolecules in the computer: Jmol to the rescue. *Biochemistry and Molecular Biology Education*, 34(4):255–261, 2006.
- [45] Nicholas Rego and David Koes. 3dmol. js: molecular visualization with webgl. *Bioinformatics*, 31(8):1322–1324, 2015.
- [46] RO Dror, HF Green, C Valant, DW Borhani, JR Valcourt, AC Pan, DH Arlow, M Canals, JR Lane, R Rahmani, JB Baell, PM Sexton, A Christopoulos, and DE Shaw. Structural basis for modulation of a g-protein-coupled receptor by allosteric drugs. *Nature*, 503(7475):295–299, 2013.
- [47] L Treuel, S Brandholt, P Maffre, S Wiegele, L Shang, and GU Nienhaus. Impact of protein modification on the protein corona on nanoparticles and nanoparticlecell interactions. *ACS Nano*, 8(1):503–513, 2013.
- [48] J Lipfert, Sebastian Doniach, R Das, and D Herschlag. Understanding nucleic acid-ion interactions. *Annu Rev Biochem*, 83:813–41, 2014.
- [49] V A Roberts, E E Thompson, M E Pique, M S Perez, and L F Ten Eyck. DOT2: Macromolecular docking with improved biophysical models. *J Comput Chem*, 34(20):1743–58, Jul 2013.
- [50] E Spiga, D Alemani, M T. Degiacomi, M Casella, and M Dal Peraro. Electrostatic-consistent coarse-grained potentials for molecular simulations of proteins. *J. Chem. Theory Comput.*, 9(8):3515–3526, August 2013.
- [51] Phillip J Stansfeld, Joseph E Goose, M Caffrey, Elisabeth P Carpenter, Joanne L Parker, S Newstead, and Mark SP Sansom. Memprotmd: Automated insertion of membrane protein structures into explicit lipid membranes. *Structure*, 23(7):1350–1361, 2015.
- [52] S Richter, A Wenzel, M Stein, R R Gabdoulline, and R C Wade. webPIPSA: a web server for the comparison of protein interaction properties. *Nucleic Acids Res*, 36(Web Server issue):W276–80, Jul 2008.

- [53] Michael Martinez, Neil J Bruce, Julia Romanowska, Daria B Kokh, Musa Ozboyaci, Xiaofeng Yu, Mehmet Ali Öztürk, Stefan Richter, and Rebecca C Wade. Sda 7: A modular and parallel implementation of the simulation of diffusional association software. *J Comput Chem*, 36(21):1631–45, Aug 2015.
- [54] Y Song, Y Zhang, CL Bajaj, and NA Baker. Continuum diffusion reaction rate calculations of wild-type and mutant mouse acetylcholinesterase: adaptive finite element analysis. *Biophysical Journal*, 87(3):1558–1566, 2004.
- [55] A H Elcock. Molecular simulations of diffusion and association in multimacromolecular systems. *Methods Enzymol*, 383:166–98, 2004.
- [56] P Mereghetti and RC Wade. Atomic detail brownian dynamics simulations of concentrated protein solutions with a mean field treatment of hydrodynamic interactions. *J. Phy. Chem.*, 116(29):8523–8533, 2012.
- [57] Michael Holst. Adaptive numerical treatment of elliptic systems on manifolds. *J. Comp. Math.*, 15:139–191, 2001.
- [58] Z Chen, N A Baker, and G W Wei. Differential geometry based solvation model I: Eulerian formulation. *J Comput Phys*, 229(22):8231–8258, Nov 2010.