arXiv:submit/1936712 [q-bio.BM] 30 Jun 2017

Improvements to the APBS biomolecular solvation software suite

Elizabeth Jurrus[⋆], Dave Engel[⋆], Keith Star[⋆], Kyle Monson[⋆], Juan Brandi[⋆], Lisa E. Felberg[†], David H. Brookes[†], Leighton Wilson[‡], Jiahui Chen[§], Karina Liles[⋆], Minju Chun[⋆], Peter Li[⋆], Todd Dolinsky[‖], Robert Konecny[Δ], David R. Koes[▷], Jens Erik Nielsen[¶], Teresa Head-Gordon[†], Weihua Geng[§], Robert Krasny[‡], Guo-Wei Wei[▽], Michael J. Holst[Δ], J. Andrew McCammon[Δ], and Nathan A. Baker[◁]

[⋆]Pacific Northwest National Laboratory
[†]University of California, Berkeley
[‡]University of Michigan
[§]Southern Methodist University
[‖]FoodLogiQ
[▷]University of Pittsburgh
[¶]Protein Engineering, Novozymes A/S
[▽]Michigan State University
[Δ]University of California San Diego
[◁]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 30, 2017

Abstract

The Adaptive Poisson-Boltzmann Solver (APBS) software was developed to solve the equations of continuum electrostatics for large biomolecular assemblages that has provided impact in the study of a broad range of 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 manuscript, 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

Robust models of electrostatic interactions are important for understanding early molecular recognition events where long-ranged intermolecular interactions and the effects of solvation on biomolecular processes dominate. While explicit electrostatic models that treat the solute and solvent in atomic detail are common, these approaches generally require extensive equilibration and sampling to converge properties of interest in the statistical ensemble of interest. Coarse-graining approaches that integrate out important, but largely uninteresting degrees of freedom, sacrifice numerical precision in favor of robust but qualitative accuracy and efficiency by eliminating the need for sampling and equilibration associated with explicit solute and solvent models.

While there is a choice among several implicit solvation models [1–6], one of the most popular implicit solvent models for biomolecules is based on the Poisson-Boltzmann (PB) equation [7–9]. The PB equation provides a global solution for the electrostatic potential (ϕ) within and around a biomolecule by solving the partial differential equation

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

The solvent is described by the bulk solvent dielectric constant ε_s as well as the properties of $i = 1, \ldots, M$ mobile ion species, described by their charges q_i, concentrations c_i, and steric ion-solute interaction potential V_i. The biomolecular structure is incorporated into the equation through V_i, a dielectric coefficient function ϵ, and a charge distribution function ρ. The dielectric ϵ is often set to a constant value ε_m in the interior of the molecule and varies sharply across the molecular boundary to the value ε_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 [10]. The charge distribution ρ is usually a sum of Dirac delta 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 ϕ 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 [7, 11]. Despite these limitations, PB methods are still very important and popular for biomolecular structural analysis, modeling, and simulation.

Several software packages have been developed that solve 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 [12], AMBER [13], DelPhi [14], Jaguar [15], Zap [16], MIBPB [17], and APBS [18]. However the APBS and associated software package PDB2PQR has served a large community of ∼27,000 users by creating web servers linked from the APBS website [19] that support preparation of biomolecular structures (see Section 2) and a fast finite-difference solution of the Poisson-Boltzmann equation (see Section 3.1) that are further augmented with a set of analysis tools. Most APBS electrostatics calculations follow the general workflow outlined in Figure 1. An even broader

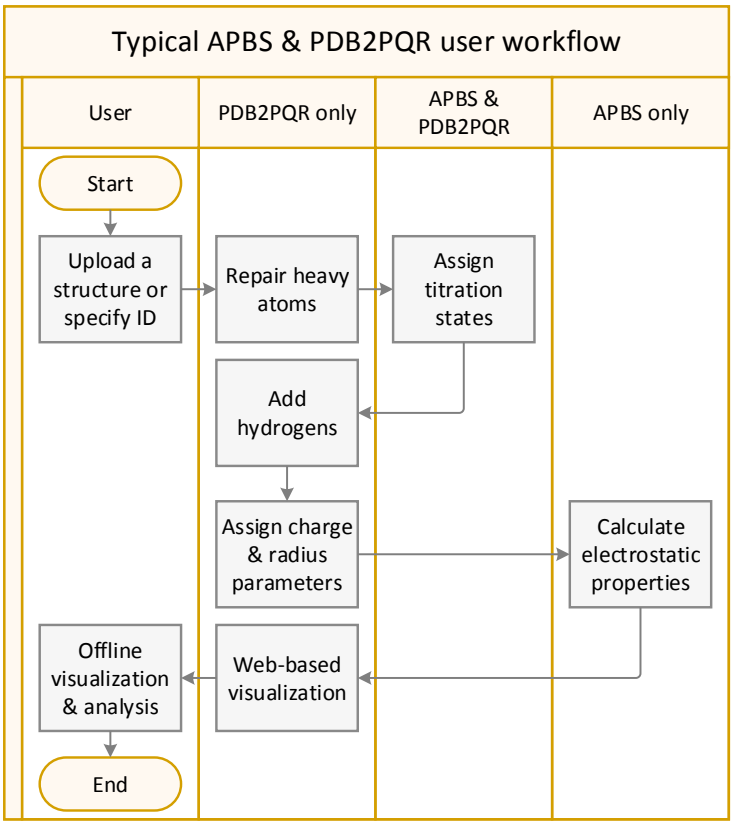

Figure 1: Workflow for biomolecular electrostatics calculations using the APBS-PDB2PQR software suite. This workflow is supported by the APBS tool suite with specific support by PDB2PQR for preparing biomolecular structures (see Section 2) and support by APBS for performing electrostatics calculations (see Section 3).

range of features and more flexible configuration is available when APBS and PDB2PQR are run from the command line on Linux, Mac, and Windows platforms, and which can be run locally or through web services provided by the NBCR-developed Opal toolkit [20]. 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 [13], CHARMM [12], NAMD [21], Rosetta [22] and TINKER [23]. General support for integration of APBS with 3rd-party programs is provided by the iAPBS library [24, 25].

This article provides an overview of the new capabilities in the APBS software and its associated tools since their release in 2001 [18, 26–28]. In particular new solutions to the PB equation have been developed and incorporated into APBS: a fully analytical model based on simple spherical geometries that treats full mutual polarization accurately, known as PBAM (Poisson-Boltzmann Analytical Model) [29, 30] as well as its extension to a semi-analytical PBE solver PB-SAM (Poisson-Boltzmann Semi-Analytical Model) that treats arbitrary shaped dielectric boundaries [31, 32]. APBS-PDB2PQR also now includes

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. We describe these new approaches in the remainder of the paper, and more detailed documentation for these tools is available on the APBS website [19]. ⋆⋆ **Dave: please make sure this is true!** ⋆⋆

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 [3]. 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 [33] and AutoDock [34]. The PQR file simply replaces the temperature and occupancy columns of a PDB flat file [35] 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 mmCIF [36] or PDBML [37]; 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 [38, 39]. PDB2PQR began its existence as a `sed` [40] 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 χ 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 [41]) that uses a heuristic method to compute pK_a perturbations due to desolvation, hydrogen bonding, and charge-charge interactions. PROPKA is included with PDB2PQR. The empirical PROPKA method has surprising accuracy for fast evaluation of protein pK_a values [42].

- *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 $\mathcal{O}\left(2^N\right)$ titration states of N titratable residues. The second method is a new polynomial-time algorithm for the optimization of discrete states in macromolecular systems [43]. This method transforms interaction energies between titratable groups into a graphical flow network. The polynomial-time $\mathcal{O}\left(N^4\right)$ 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 [44]. 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 ρ) and radii (for ϵ and V_i) are assigned based on the chosen force field. PDB2PQR currently supports charge/radii force fields from AMBER99 [45], CHARMM22 [46], PARSE [47], PEOE_PB [48], Swanson et al. [49], and Tan et al. [50].

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 A. The remainder of this section describes the specific solvers available for electrostatic calculations, also described in more detail on the APBS website [19].

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 [51, 52]. 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 [53]. The finite difference PMG solver [52, 54] 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 ϕ 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 [55–59]. 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 a linearized version of the PB equation (Eq. 1) [60]. 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 [61]. The boundary element method requires a surface triangulation, generated by a program such as MSMS [62] or NanoShaper [63], on which to discretize the integral equations. 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 $\mathcal{O}(N \log N)$ for N points on the discretized molecular surface [61, 64]. Typical results from a boundary element calculation are shown in Figure 2. Additional information about the boundary element method and its implementation 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 [65] and the approaches developed by Head-Gordon and implemented in APBS. In particular, the Poisson-Boltzmann Analytical Method (PB-AM), was developed by Lotan and Head-Gordon in 2006 [29]. 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

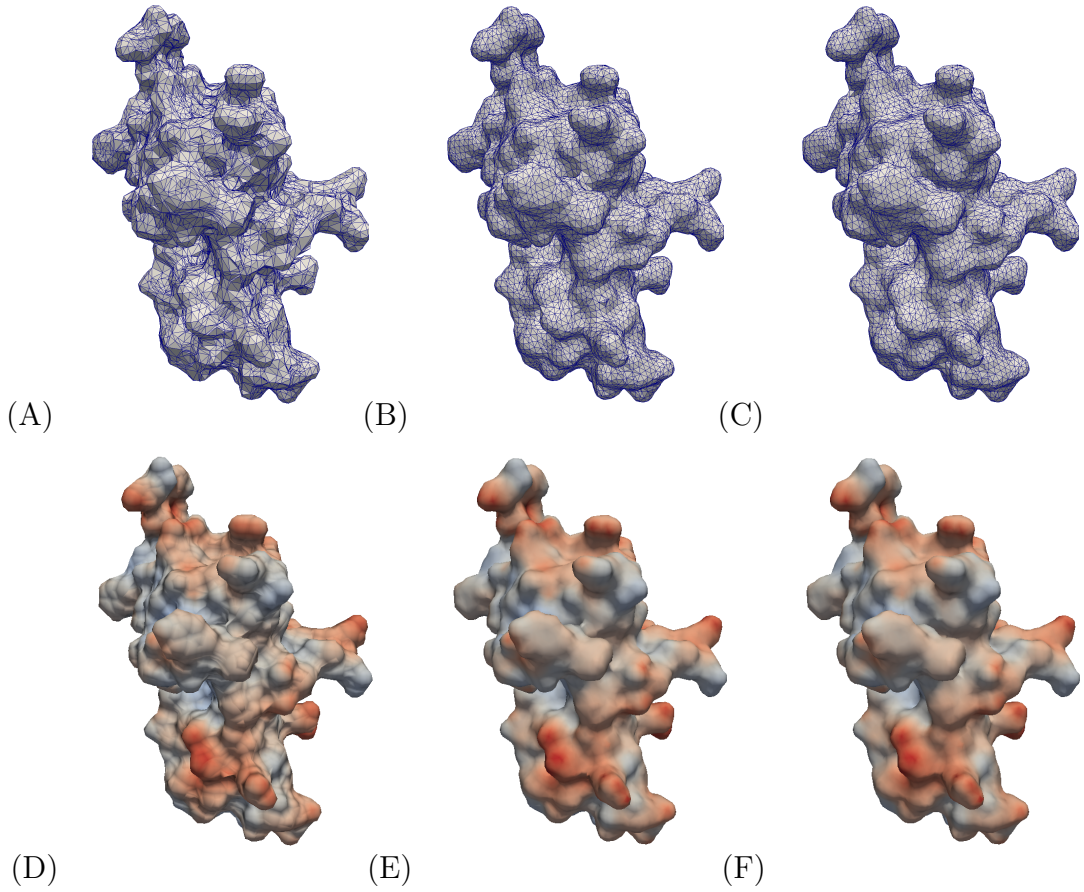

Figure 2: APBS TABI-PB electrostatic potential results for PDB ID 1a63. Surfaces were generated with (A) 20228 triangles via MSMS [62], (B) 20744 triangles via NanoShaper SES, and (C) 21084 triangles via NanoShaper Skin [63]. These discretizations resulted in surface potentials (with units kT/e) of (D) MSMS in range $[-8.7, 8.6]$, (E) NanoShaper SES in range $[-13.4, 7.5]$, and (F) NanoShaper Skin in range $[-33.8, 8.0]$. All calculations were performed at 0.15 M ionic strength in 1:1 salt, with protein dielectric 1, solvent dielectric 80, and temperature 300K.

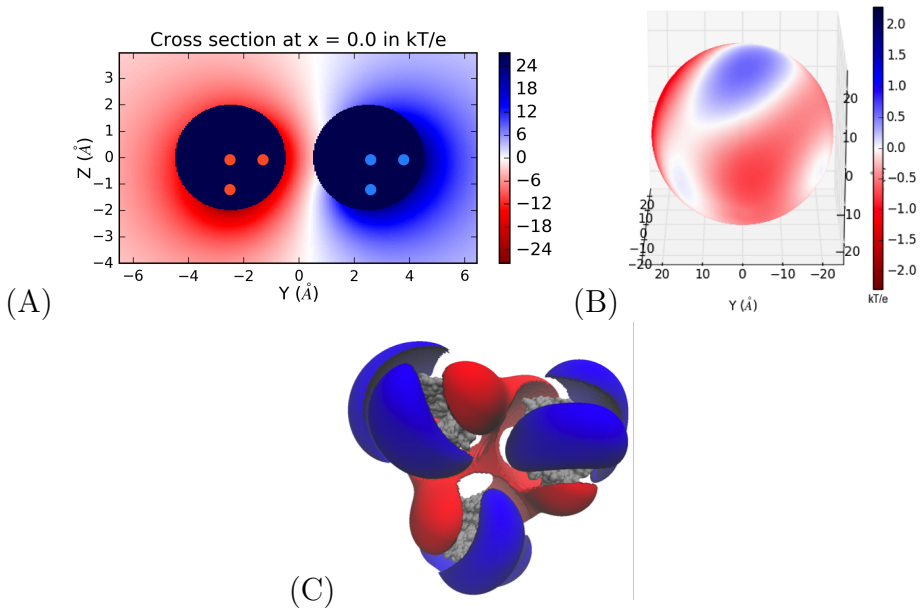

Figure 3: APBS PB-AM electrostatic potential results for (A) 2D cross section for the electrostatic potential outside of two collections of point charges (each with net charge ± 6, charge positions indicated by red and blue circles), (B) potential on the spherical coarse-grain surface of a barstar molecule, (C) VMD [66] of the transmembrane trimer Omp32 porin (molecules are given in gray and isosurfaces are drawn at 1.0 (blue) and -1.0 (red). All calculations were performed at 0.0 M ionic strength, pH 7, protein dielectric 2, and solvent dielectric 78. All electrostatic potentials are given in units of kT/e.

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 [31]. 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 [67]. Additional details about these methods and their use in APBS are presented in Appendix E.

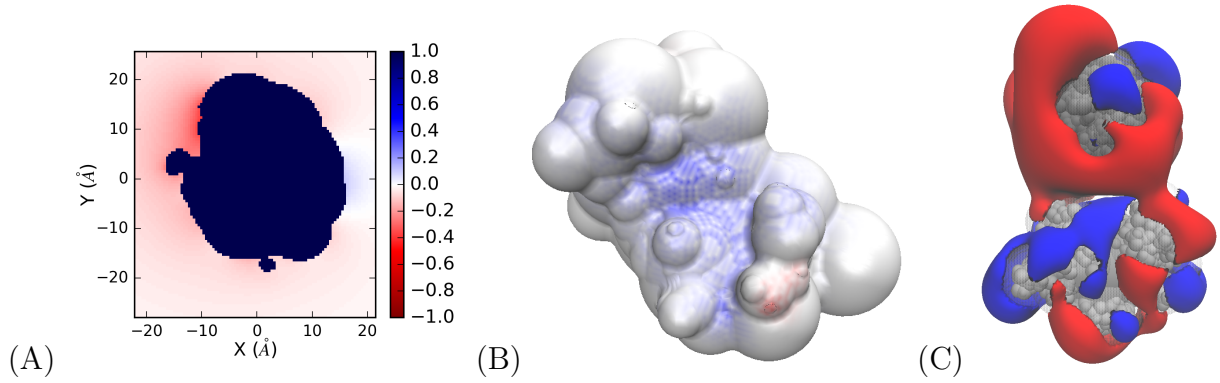

Figure 4: APBS PB-SAM electrostatic potential results for (A) potential in a 2D plane surrounding the barstar molecule, (B) potential over range $[-1, 1]$ on the coarse-grain surface of the barnase molecule (blue region is the location of barstar association, (C) VMD [66] visualization of the ESP around the association of barnase and barstar (molecules are given in gray and isosurfaces are drawn at 1.0 (blue) and -1.0 (red). All calculations were performed at 0.0 M ionic strength, pH 7, protein dielectric 2, and solvent dielectric 78. All electrostatic potentials are given in units of kT/e.

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 [68, 69], VMD [66, 70], PMV [71, 72], and Chimera [73, 74]. We have also worked with the developers of Jmol [75, 76] and 3Dmol.js [77, 78]. to provide web-based setup and visualization of APBS-PDB2PQR calculations and related workflows. APBS integration with Jmol has been described previously [79]. 3Dmol.js 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. As part of the integration with 3Dmol.js, we implemented additional enhancements, 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.js interface is shown in Figure 5. Examples of 3Dmol.js visualization options are shown in Figure 6.

4.2 Other applications

Besides visualization and the processes described in Section 2, 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 [80], understanding protein-nanoparticle interactions [81, 82], understanding nucleic acid-ion interactions [83], biomolecular docking [84] and ligand binding [85],

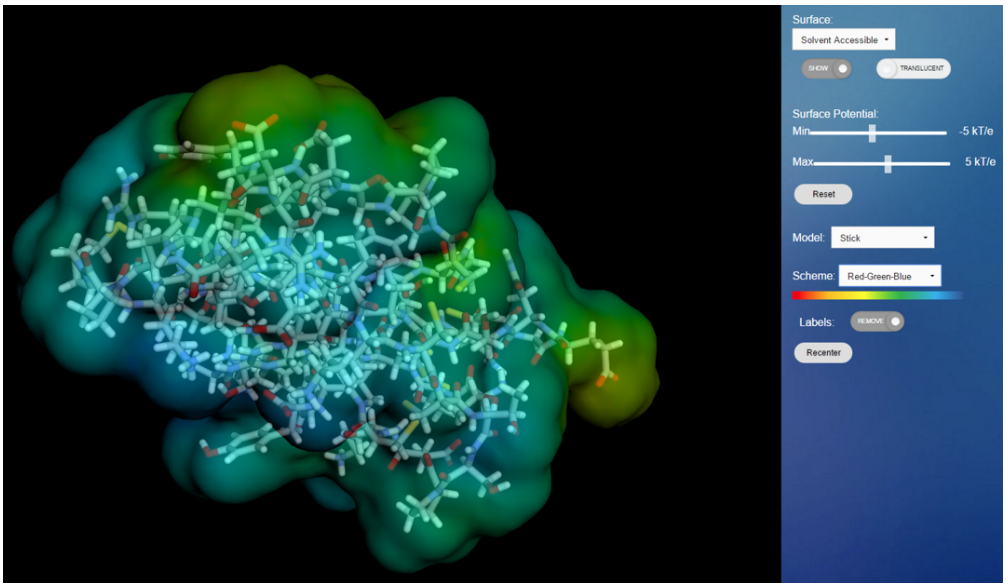

Figure 5: 3Dmol.js interface displaying a rendering of fasciculin-2 (1FAS) protein with translucent, solvent accessible surface using a stick model and red-green-blue color scheme.

developing new coarse-grained protein models [86], setting up membrane protein simulations [87], etc. APBS also plays a key role in PIPSA for protein surface electrostatics analysis [88] as well as BrownDye [89] and SDA [90] for simulation of protein-protein association kinetics through Brownian dynamics. 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 [80, 90–95].

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" [96] 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

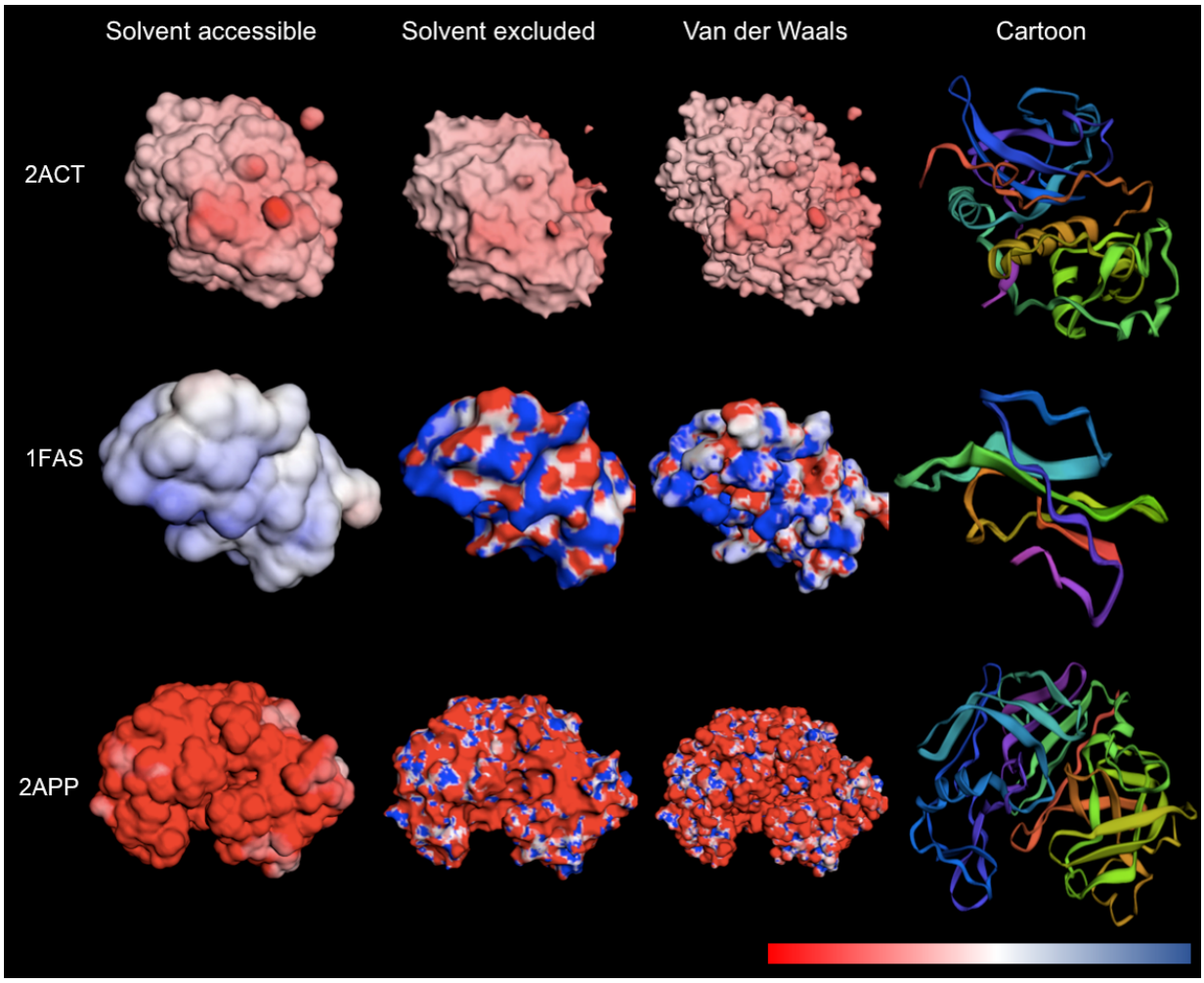

Figure 6: Renderings of three different proteins using renderings of actinidin (2ACT) (top), fasciculin-2 (1FAS) (center), and pepsin-penicillium (2APP) (bottom). To demonstrate the different visualization options. From left to right: solvent-accessible surface, solvent-excluded surface, van der Waals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

workflows.

Acknowledgments

The authors gratefully acknowledge NIH grant GM069702 for support of APBS and PDB2-PQR. PNNL is operated by Battelle for the U.S. DOE under contract DE-AC05-76RL01830. THG and DHB were supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under contract DE-AC02-05CH11231. LEF was supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE 110640. JAM and RK were supported by NIH grants GM31749 and GM103426. LWW was supported by the Department of Defense through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. WG and RK were supported by NSF grant DMS-1418966.

These appendices provide basic information about configuring APBS electrostatics calculations. Rather than duplicating the user manual on the APBS website [19], we have only focused on input file keywords that directly relate to the setup and execution of solvation calculations.

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` ⟨*charge*⟩ `conc` ⟨*conc*⟩ `radius` ⟨*radius*⟩: This line can appear multiple times to specify the ionic species in the calculation. This specifies the following terms in Eq. 1: ⟨*charge*⟩ gives q_i in units of electrons, ⟨*conc*⟩ gives c_i in units of M, and ⟨*radius*⟩ 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`).
- `mol` ⟨*id*⟩: Specify the ID of the molecule on which the calculations are to be performed. This ID is determined by `READ` statements which specify the molecules to import.
- `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` ⟨*temp*⟩: The temperature (in K) for the calculation. A typical value is 298 K.

Additionally, `READ` statements in APBS input files are used to load molecule information, parameter sets, and finite element meshes. More detailed information about these and other commands can be found on the APBS website [19].

B Finite element and finite difference calculations in APBS

The finite difference and finite element methods used by APBS have been described extensively in previous publications [18, 26–28, 54]; this section focuses on the configuration and use of these methods in APBS.

B.1 Finite difference calculation configuration

APBS users have several ways to invoke the PMG finite difference solver [54] capabilities of APBS through keywords in the APBS input file `ELEC` block. The different types of finite difference calculations include:

- `mg-manual`: This specifies a manually configured multigrid finite difference calculation in APBS.
- `mg-auto`: This specifies an automatically configured multigrid finite difference calculation in APBS, using focusing [97] to increase the resolution of the calculation in areas of interest.
- `mg-para`: This specifies a parallel version of the multigrid finite difference calculating, using parallel focusing to increase the resolution of the calculation in areas of interest [18].

These different types of calculations have several keywords described in detail on the APBS website [19]. Some of the most important settings are described below.

Boundary conditions Boundary conditions must be imposed on the exterior of the finite difference calculation domain. In general, biomolecular electrostatics calculations use Dirichlet boundary conditions, setting the value of the potential to an asymptotically correct approximation of the true solution. There are several forms of these boundary conditions available in APBS with approximately equal accuracy given a sufficiently large calculation domain (see below). The only boundary condition which is *not* recommended for typical calculations is the zero-potential Dirichlet condition.

Grid dimensions, center, and spacing Finite difference calculations in APBS are performed in rectangular domains. The key aspects of this domain include its length L_i and number of grid points n_i in each direction i. These parameters are related to the spacing h_i of the finite difference grid by $h_i = L_i/(n_i - 1)$. Grid spacings below 0.5 Å are recommended for most calculations. The number of grid points is specified by the `dime` keyword. For `mg-manual` calculations, the grid spacing can be specified by *either* the `grid` or the `glen` keywords, which specify the grid spacing or length, respectively. For `mg-auto` or `mg-para` calculations, the grid spacing is determined by the `cglen` and `fglen` keywords, which specify the lengths of the coarse and fine grids for the focusing calculations. Grid lengths should extend sufficient distance away from the biomolecule so that the chosen boundary condition is accurate. In general, setting the length of the coarsest grid to approximately 1.5 times the size of the biomolecule gives reasonable results. However, as a best practice, it is important to ensure that the calculated quantities of interest do not change significantly with grid spacing or grid length. The center of the finite difference grid can be specified by the `gcent` command for `mg-manual` calculations or by the `cgcent` and fgcent keywords for the coarse and fine grids (respectively) in `mg-auto` or `mg-para` focusing calculations. These keywords can be used to specify absolute grid centers (in Cartesian coordinates) or relative centers based on molecule location. Because of errors associated with charge discretization, it is generally a good idea to keep the positions of molecules on a grid fixed for all calculations. For example, a binding calculation for rigid molecules should keep all molecules in the same positions on the grid.

Charge discretization As mentioned above, atomic charge distributions are often modeled as a collection of Dirac delta functions or other basis functions with extremely small spatial support. However, finite difference calculations on performed on grids with finite spacing, requiring charges to be mapped across several grid points. This mapping creates significant dependence of the electrostatic potential on the grid spacing which is why it is always important to use the same grid setup for all parts of an electrostatic calculation. The `chgm` keyword controls the interpolation scheme used for charge distributions and includes the following types of discretization schemes:

- `spl0`: Traditional trilinear interpolation (linear splines). The charge is mapped onto the nearest-neighbor grid points. Resulting potentials are very sensitive to grid spacing, length, and position.
- `spl2`: Cubic B-spline discretization as described by Im et al. [98]. The charge is mapped onto the nearest- and next-nearest-neighbor grid points. Resulting potentials are somewhat less sensitive (than `spl0`) to grid spacing, length, and position; however, this discretization can often require smaller grid spacings for accurate representation of charge positions.
- `spl4`: Quintic B-spline discretization as described by Schnieders et al. [99]. Similar to `spl2`, except the charge/multipole is additionally mapped to include next-next-nearest neighbors (125 grid points receive charge density). This discretization results in less sensitivity to grid spacing and position but nearly always requires smaller grid spacings for accurate representation of charge positions.

Surface definition APBS provides several difference surface definitions through the `srfm` keyword:

- `mol`: The dielectric coefficient is defined based on a molecular surface definition. The problem domain is divided into two spaces. The "free volume" space is defined by the union of solvent-sized spheres (size determined by the `srad` keyword) which do not overlap with biomolecular atoms. This free volume is assigned bulk solvent dielectric values. The complement of this space is assigned biomolecular dielectric values. With a non-zero solvent radius (`srad`), this choice of coefficient corresponds to the traditional definition used for PB calculations. When the solvent radius is set to zero, this corresponds to a van der Waals surface definition. The ion-accessibility coefficient is defined by an "inflated" van der Waals model. Specifically, the radius of each biomolecular atom is increased by the radius of the ion species (as specified with the ion keyword). The problem domain is then divided into two spaces. The space inside the union of these inflated atomic spheres is assigned an ion-accessibility value of 0; the complement space is assigned bulk ion accessibility values.
- `smol`: The dielectric and ion-accessibility coefficients are defined as for mol (see above). However, they are then "smoothed" by a 9-point harmonic averaging to somewhat reduce sensitivity to the grid setup [100].

- `spl2`: The dielectric and ion-accessibility coefficients are defined by a cubic-spline surface [98]. The width of the dielectric interface is controlled by the `swin` parameter. These spline-based surface definitions are very stable with respect to grid parameters and therefore ideal for calculating forces. However, they require substantial reparameterization of the force field [101].

- `spl4`: The dielectric and ion-accessibility coefficients are defined by a 7^{th}-order polynomial. This surface definition has characteristics similar to `spl2`, but provides higher order continuity necessary for stable force calculations with atomic multipole force fields (up to quadrupole) [99].

B.2 Finite elements calculation configuration

Users invoke the FEtk finite element solver [51] in APBS by including the `fe-manual` keyword in the `ELEC` section of the input file. Many aspects of the finite element configuration closely follow the finite difference options described above. This section only describes the configuration options which are unique to finite element calculations in APBS.

Finite element calculations begin with an initial mesh. This mesh can be imported from an external file via the `usemesh` keyword or generated by APBS. APBS generates the initial mesh from a very coarse 8-tetrahedron mesh of size `domainLength` which is then refined uniformly or selectively at the molecular surface and charge locations, based on the value of the `akeyPRE` keyword. This initial refinement occurs until the mesh has `targetNum` vertices or has reached `targetRes` edge length (in Å).

As described previously [26, 27], APBS uses FEtk in a solve-estimate-refine iteration which involves the following steps:

1. Solve the problem with the current finite element mesh.

2. Estimate the error in the solution as a function of position on the mesh. The method of error estimation is determined by the `ekey` keyword which can have the values:

   - `simp`: Per-simplex error threshold; simplices with error above this limit are flagged for refinement.
   - `global`: Global (whole domain) error limit; flag enough simplices for refinement to reduce the global error below this limit.
   - `frac`: The specified fraction of the simplices with the highest amount of error are flagged for refinement.

3. Adaptively refine the mesh to reduce the error using the error metric described by `ekey`.

This iteration is repeated until a target error level `etol` is reached or a maximum number of solve-estimate refine iterations (`maxsolve`) or vertices (`maxvert`) is reached.

C Geometric flow calculations in APBS

This section contains additional information about the geometric flow equation implementation in APBS introduced in Section 3.2. The geometric flow methods used by APBS have been described extensively in previous publications [55–58, 102]; this section focuses on the configuration and use of these methods in APBS.

C.1 Geometric flow calculation configuration

Users invoke the geometric flow solver in APBS by including the `geoflow-auto` keyword in the `ELEC` section of the input file. Because the geometric flow solver is based on finite difference solvers, many of the keywords for this section are similar to those described in Section B.1. Three additional parameters are needed for geometric flow calculations to specify how nonpolar solvation is linked to the polar implicit solvent models:

- `gamma` ⟨*tension*⟩: Specify the surface tension of the solvent in units of kJ mol^{-1} Å^{-2}. Based on Daily et al. [58], a recommended value for small molecules is 0.431 kJ mol^{-1} Å^{-2}.
- `press` ⟨*pressure*⟩: Specify the internal pressure of the solvent in units of kJ mol^{-1} Å^{-3}. Based on Daily et al. [58], a recommended value for small molecules is 0.104 kJ mol^{-1} Å^{-3}.
- `bconc` ⟨*concentration*⟩: Specify the bulk concentration of solvent in Å^{-3}. The bulk density of water, 0.0334 Å^{-3}, is recommended.
- `vdwdisp` ⟨*bool*⟩: Indicate whether van der Waals interactions should be included in the geometric flow calculation through ⟨*bool*⟩ (1 = include, 0 = exclude). If these interactions are included, then a force field with van der Waals terms must be included through a `READ` statement in the APBS input file.

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

This section provides additional background on the TABI-PB boundary element solver [60], introduced in Section 3.3. As described earlier, this method involves solving 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 [62] or NanoShaper [63], on which to discretize the integral equations.

The coupled second kind integral equations employed by TABI-PB for calculating the surface potential ϕ and its normal derivative [61] are:

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

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. 3.

$$\begin{aligned}K_1\left(\mathbf{x},\mathbf{y}\right) &= G_0\left(\mathbf{x},\mathbf{y}\right)-G_\kappa\left(\mathbf{x},\mathbf{y}\right),\\ K_2\left(\mathbf{x},\mathbf{y}\right) &= \varepsilon\frac{\partial G_\kappa\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{y}}}-\frac{\partial G_0\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{y}}},\\ K_3\left(\mathbf{x},\mathbf{y}\right) &= \varepsilon\frac{\partial G_0\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{x}}}-\frac{1}{\varepsilon}\frac{\partial G_\kappa\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{x}}},\\ K_4\left(\mathbf{x},\mathbf{y}\right) &= \varepsilon\frac{\partial^2 G_\kappa\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{x}}\partial\nu_{\mathbf{y}}}-\frac{1}{\varepsilon}\frac{\partial^2 G_0\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{x}}\partial\nu_{\mathbf{x}}},\end{aligned} \tag{3}$$

where G_0 and G_κ are the Coulomb and screened Coulomb potentials defined as:

$$\begin{aligned}G_0\left(\mathbf{x},\mathbf{y}\right) &= \frac{1}{4\pi\left|\mathbf{x}-\mathbf{y}\right|},\\ G_\kappa\left(\mathbf{x},\mathbf{y}\right) &= \frac{e^{-\kappa|\mathbf{x}-\mathbf{y}|}}{4\pi\left|\mathbf{x}-\mathbf{y}\right|}.\end{aligned} \tag{4}$$

The normal derivatives of the potential kernels G are:

$$\begin{aligned}\frac{\partial G\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{y}}} &= \sum_{n=1}^{3}\nu_n\left(\mathbf{y}\right)\partial_{y_n}G\left(\mathbf{x},\mathbf{y}\right),\\ \frac{\partial G\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{x}}} &= -\sum_{n=1}^{3}\nu_n\left(\mathbf{y}\right)\partial_{x_n}G\left(\mathbf{x},\mathbf{y}\right),\\ \frac{\partial^2 G\left(\mathbf{x},\mathbf{y}\right)}{\partial\nu_{\mathbf{x}}\partial\nu_{\mathbf{y}}} &= -\sum_{m=1}^{3}\sum_{n=1}^{3}\nu_m\left(\mathbf{x}\right)\nu_n\left(\mathbf{y}\right)\partial_{x_n}\partial_{y_n}G\left(\mathbf{x},\mathbf{y}\right)\end{aligned} \tag{5}$$

for the three spatial components n of the normal direction. Additionally, the source terms S_1 and S_2 in Eq. 2 are:

$$\begin{aligned}S_1(\mathbf{x}) &= \frac{1}{\varepsilon_m}\sum_{k=1}^{N_c}q_k G_0\left(\mathbf{x},\mathbf{y}_k\right),\\ S_2(\mathbf{x}) &= \frac{1}{\varepsilon_m}\sum_{k=1}^{N_c}q_k\frac{\partial G_0\left(\mathbf{x},\mathbf{y}_k\right)}{\partial\nu_{\mathbf{x}}},\end{aligned} \tag{6}$$

where N_c is the number of atoms in the solute molecule, and q_k is the charge of the kth 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 ith triangle – the integral equations are discretized as:

$$\begin{aligned}
\frac{1}{2}\left(1+\varepsilon\right)\phi\left(\mathbf{x}_i\right) &= \sum_{\substack{j=1\\ j\neq i}}^{N}\left[K_1\left(\mathbf{x}_i,\mathbf{x}_j\right)\frac{\partial\phi\left(\mathbf{x}_j\right)}{\partial\nu}+K_2\left(\mathbf{x}_i,\mathbf{x}_j\right)\phi\left(\mathbf{x}_j\right)\right]A_j+S_1(\mathbf{x}_i),\\
\frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial\phi\left(\mathbf{x}_i\right)}{\partial\nu} &= \sum_{\substack{j=1\\ j\neq i}}^{N}\left[K_3\left(\mathbf{x}_i,\mathbf{x}_j\right)\frac{\partial\phi\left(\mathbf{x}_j\right)}{\partial\nu}+K_4\left(\mathbf{x}_i,\mathbf{x}_j\right)\phi\left(\mathbf{x}_j\right)\right]A_J+S_2(\mathbf{x}_I).
\end{aligned} \tag{7}$$

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. 8:

$$E_{\text{sol}}=\frac{1}{2}\sum_{k=1}^{N_c}q_k\sum_{j=1}^{N}\left[K_1\left(\mathbf{x}_k,\mathbf{x}_j\right)\frac{\partial\phi\left(\mathbf{x}_j\right)}{\partial\nu}+K_2\left(\mathbf{x}_k,\mathbf{x}_j\right)\phi\left(\mathbf{x}_j\right)\right]A_j \tag{8}$$

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

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. 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 $\mathcal{O}(N\log N)$ for N points on the discretized molecular surface [64]. In particular, 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 well-separated particle cluster interactions, a Taylor approximation about the center of the cluster is used to evaluate the contribution. The Taylor coefficients are calculated through recurrence relations. The resulting linear system is then solved with GMRES iteration [103].

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.

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` ⟨*flag*⟩: 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` ⟨*flag*⟩: Type of output data file generated; 0 = APBS OpenDX format [104] and 1 = ParaView format [105].

Additional information about parameter settings is provided via the APBS website [19]. 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 [29, 30] introduced in Section 3.4.

E.1 Analytical method (PB-AM) background

The solution to the PB-AM model is represented as a system of linear equations:

$$A = \Gamma \cdot (\Delta \cdot T \cdot A + E), \tag{9}$$

where A represents a vector of the effective multipole expansion of the charge distributions of each molecule, E is a vector of the fixed charge distribution of all molecules, Γ is a dielectric boundary-crossing operator, Δ 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 A determined using the Gauss-Seidel iterative method and can then be used to compute physical properties such as interaction energies, forces, and torques. The interaction energy for molecule i, $(\Omega^{(i)})$ is given in Eq. 10.

$$\Omega^{(i)} = \frac{1}{\epsilon_s} \left\langle \sum_{j \neq i}^{N} T \cdot A^{(j)}, A^{(i)} \right\rangle \tag{10}$$

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

$$\mathbf{F}^{(i)} = \nabla_i \Omega^{(i)} = \frac{1}{\epsilon_s} [\langle \nabla_i \, T \cdot A^{(i)}, A^{(i)} \rangle + \langle T \cdot A^{(i)}, \nabla_i \, A^{(i)} \rangle] \tag{11}$$

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. 12.

$$\tau^{(i)} = \frac{1}{\epsilon_s} \left[{}^{x}H^{(i)}, {}^{y}H^{(i)}, {}^{z}H^{(i)} \right] \times \left[\nabla_i L^{(i)} \right] \tag{12}$$

where ${}^{\alpha}H^{(i)}_{n,m} = \sum_{j=1}^{M_i} \alpha_j^{(i)} \gamma_n^{(i)} q_j^{(i)} (\rho_j^{(i)})^n Y_{n,m}(\vartheta_j^{(i)}, \varphi_j^{(i)})$, $\alpha = x, y, z$, 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 $p_j^{(i)} = \left[\rho_j^{(i)}, \vartheta_j^{(i)}, \varphi_j^{(i)} \right]$ is its position in spherical coordinates. For more details on the PB-AM derivation, see Lotan and Head-Gordon [29].

E.2 Semi-analytical method (PM-SAM) background

The derivation details of PB-SAM have been reported previously [31, 32], with the main points being summarized in this section. The electrostatic potential (ϕ_r) of the system at any point r is governed by the linearized form of the PB equation:

$$-\nabla \cdot \epsilon \nabla \phi + \kappa^2 \phi = \rho, \tag{13}$$

where κ is the inverse Debye length. Eq. 13 is a linearization of Eq. 1 for $\beta q_i \phi \ll 1$. For the case of spherical cavities, we can solve Eq. 13 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:

$$\phi_{out}^{(i)}(r) = \sum_{I=1}^{N_{mol}} \left(4\pi \int_{d\Omega^{(I)}} \frac{e^{-\kappa |r - r'|}}{|r - r'|} h^{(I)}(r') dr' \right) \tag{14}$$

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

$$\phi_{in}^{(i)}(r) = \sum_{\alpha=1}^{N_C^{(I)}} \frac{1}{|r - r_\alpha^{(I)}|} \cdot \frac{q_\alpha^{(I)}}{\epsilon_{in}} + \frac{1}{4\pi} \int_{d\Omega^{(I)}} \frac{1}{|r - r'|} f^{(I)}(r') dr' \tag{15}$$

where $N_C^{(I)}$ is the number of charges in molecule I, q_α is the magnitude of the α-th charge, $r_\alpha^{(I)} = \left[\rho_\alpha^{(I)}, \theta_\alpha^{(I)}, \phi_\alpha^{(I)}\right]$ 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:

$$F_{n,m}^{(I,k)} \equiv \frac{1}{4\pi} \int_{d\Omega^{(I,k)}} f^{(I,k)}(r') \left(\frac{a^{(I,k)}}{r'}\right)^{n+1} \overline{Y_{n,m}^{(I,k)}}(\theta', \phi') dr' \tag{16}$$

$$H_{n,m}^{(I,k)} \equiv \frac{1}{4\pi} \int_{d\Omega^{(I,k)}} h^{(I,k)}(r') \left(\frac{r'}{a^{(I,k)}}\right)^{n} \hat{i}_n(\kappa r') \overline{Y_{n,m}^{(I,k)}}(\theta', \phi') dr' \tag{17}$$

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:

$$F_{n,m}^{(I,k)} = \langle I_{E,n,m}^{(I,k)}, WF^{(I,k)} \rangle \tag{18}$$

$$H_{n,m}^{(I,k)} = \langle I_{E,n,m}^{(I,k)}, WH^{(I,k)} \rangle \tag{19}$$

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:

$$I_{E,n,m}^{(I,k)} \equiv \frac{1}{4\pi} \int_{\phi_E} \int_{\theta_E} Y_{l,s}^{(I,k)}(\theta', \phi') \overline{Y_{n,m}^{(I,k)}}(\theta', \phi') \sin\theta' d\theta' d\phi' \tag{20}$$

Using the above formalism, physical properties of the system, such as interaction energy, forces and torques can also 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:

$$\Omega^{(i)} = \frac{1}{\epsilon_s} \sum_{k}^{N_k^{(I)}} \langle LHN^{(I,k)}, H^{(I,k)} \rangle \tag{21}$$

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:

$$F^{(I)} = -\nabla\Omega^{(I)} = -\frac{1}{\epsilon_s} \sum_{k}^{N_k^{(I)}} f_{I,k} = -\frac{1}{\epsilon_s} \sum_{k}^{N_k^{(I)}} (\langle \nabla LHN^{(I,k)}, H^{(I,k)} \rangle + \langle LHN^{(I,k)}, \nabla H^{(I,k)} \rangle) \tag{22}$$

As in the analytical PB-AM 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. 23.

$$\tau^{(I)} = \sum_k^{N_k^{(I)}} c^{(I,k)} \times f_{I,k} + \sum_k^{N_k^{(I)}} \sum_{P \in k} r_P^{(I,k)} \times f_P \tag{23}$$

where $f_{I,k}$ is given in Eq. 22 and

$$f_P = -\frac{1}{\epsilon_s} \sum_k^{N_k^{(I)}} (\langle \nabla_I LHN^{(I,k)}, H_P^{(I,k)} \rangle + \langle LHN^{(I,k)}, \nabla_I H_P^{(I,k)} \rangle) \tag{24}$$

where

$$H_{P,n,m}^{(I,k)} = h(\theta_p, \phi_p) Y_{n,m}^{(I,k)}(\theta_p, \phi_p) \tag{25}$$

$$\nabla_j H_{P,\alpha,n,m}^{(I,k)} = \left[\nabla_j \, h(\theta_p, \phi_p)\right]_\alpha Y_{n,m}^{(I,k)}(\theta_p, \phi_p) \tag{26}$$

where $\alpha = x, y, z$. For the derivation of the PB-SAM solver please see the previous publications [31, 32] .

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.
- `runtype dynamics`: Perform a Brownian Dynamics simulation.
- `ntraj` ⟨*number*⟩: Number of Brownian Dynamics simulations to run.
- `term` ⟨*type*⟩ ⟨*value*⟩ ⟨*mol*⟩: Allows the user to indicate conditions for the termination of each BD trajectory. The following values of ⟨*type*⟩ are allowed:
  - `time` ⟨*time*⟩: A limit on the total simulation time.
  - `x` or `y` or `y` or `z` or `r` and ⟨>=⟩ or ⟨<=⟩: Represents the approach of two molecules to a certain distance `r` or certain region of space given by `x` or `y` or `y`. The operators >= and <= represent the corresponding inequalities.

  The parameter ⟨*mol*⟩ is the molecular index that this condition applies. ⟨*mol*⟩ should be 0 for `time` and for a termination condition of `x`, `y` or `z`, the molecule index that this termination condition applies to.

- `xyz` ⟨*idx*⟩ ⟨*fpath*⟩: Molecule index ⟨*idx*⟩ and file path ⟨*fpath*⟩ for the molecule starting configurations. A starting configuration is needed for each molecule and each trajectory. Therefore, jf there are m molecules and `ntraj` ⟨n⟩ trajectories, then the input file must contain $m \times n$ `xyz` entries.

- `tolsp` ⟨*val*⟩: Modify the coarseness of the molecular description. ⟨*val*⟩ is the distance (in Å) beyond the solvent-excluded surface that the coarse-grained representation extends. Increasing values of ⟨*val*⟩ leads to fewer coarse-grained spheres, faster calculation times, but less accurate solutions. Typical values for ⟨*val*⟩ 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. These parameters are similar to those found in the `ELEC` section of a usual APBS run and are documented on the Contributions portion of the APBS website [19]. Additional information about parameter settings is provided via the APBS website [19]. Examples of the electrostatic potentials produced from PB-AM and PB-SAM are shown in Figures 3 and 4, respectively.

References

[1] M. E. Davis and J A McCammon. "Electrostatics in biomolecular structure and dynamics". In: *Chemical Reviews* 90.3 (1990), pp. 509–521.

[2] M F Perutz. "Electrostatic effects in proteins". In: *Science* 201.4362 (1978), pp. 1187–91.

[3] P Ren et al. "Biomolecular electrostatics and solvation: a computational perspective". In: *Quarterly Reviews of Biophysics* 45.4 (2012), pp. 427–91. DOI: 10.1017/S003358351200011X.

[4] K A Sharp and B Honig. "Electrostatic interactions in macromolecules: theory and applications". In: *Annual Review of Biophysics and Biophysical Chemistry* 19 (1990), pp. 301–32. DOI: 10.1146/annurev.bb.19.060190.001505.

[5] B Roux and T Simonson. "Implicit solvent models". In: *Biophysical Chemistry* 78.1-2 (1999), pp. 1–20.

[6] A Warshel et al. "Modeling electrostatic effects in proteins". In: *Biochimica et Biophysica Acta* 1764.11 (2006), pp. 1647–76. DOI: 10.1016/j.bbapap.2006.08.007.

[7] M Fixman. "The Poisson-Boltzmann equation and its application to polyelectrolytes". In: *Journal of Chemical Physics* 70 (1979), pp. 4995–5005.

[8] P Grochowski and J Trylska. "Continuum molecular electrostatics, salt effects, and counterion binding–a review of the Poisson-Boltzmann theory and its modifications". In: *Biopolymers* 89.2 (2008), pp. 93–113. DOI: 10.1002/bip.20877.

[9] G Lamm et al. "The Poisson-Boltzmann Equation". In: *Reviews in Computational Chemistry* 19 (2003), pp. 147–365.

[10] B. Lee and F. M Richards. "The interpretation of protein structures: Estimation of static accessibility". In: *Journal of Molecular Biology* 55.3 (1971), pp. 379–400. DOI: 10.1016/0022-2836(71)90324-X.

[11] R. R. Netz and H. Orland. "Beyond Poisson-Boltzmann: Fluctuation effects and correlation functions". In: *European Physical Journal E: Soft Matter* 1.2-3 (2000), pp. 203–214.

[12] B R Brooks et al. "CHARMM: the biomolecular simulation program". In: *Journal of Computational Chemistry* 30.10 (2009), pp. 1545–614. DOI: 10.1002/jcc.21287.

[13] D A Case et al. "The AMBER biomolecular simulation programs". In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1668–88. DOI: 10.1002/jcc.20290.

[14] S Sarkar et al. "DelPhi web server: A comprehensive online suite for electrostatic calculations of biological macromolecules and their complexes". In: *Communications in Computational Physics* 13.1 (2013), pp. 269–284.

[15] AD Bochevarov et al. "Jaguar: A high-performance quantum chemistry software program with strengths in life and materials sciences". In: *International Journal of Quantum Chemistry* 113.18 (2013), pp. 2110–2142.

[16] JA Grant, BT Pickup, and A Nicholls. "A smooth permittivity function for Poisson-Boltzmann solvation methods". In: *Journal of Computational Chemistry* 22 (2001), pp. 608–640.

[17] Y. C. Zhou, M. Feig, and G. W. Wei. "Highly accurate biomolecular electrostatics in continuum dielectric environments". In: *Journal of Computational Chemistry* 29.1 (2008), pp. 87–87. DOI: 10.1002/jcc.20769.

[18] N A Baker et al. "Electrostatics of nanosystems: application to microtubules and the ribosome". In: *Proceedings of the National academy of Sciences of the United States of America* 98.18 (2001), pp. 10037–41. DOI: 10.1073/pnas.181342398.

[19] "APBS website". http://www.poissonboltzmann.org/. URL: http://www.poissonboltzmann.org/.

[20] Sriram Krishnan et al. "Design and evaluation of Opal2: A toolkit for scientific software as a service". In: *Services-I, 2009 World Conference on*. IEEE. 2009, pp. 709–716. URL: http://nbcr-222.ucsd.edu/opal2/dashboard.

[21] J C Phillips et al. "Scalable molecular dynamics with NAMD". In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1781–802. DOI: 10.1002/jcc.20289.

[22] David Baker. "Rosetta website". https://www.rosettacommons.org/. URL: https://www.rosettacommons.org/.

[23] Jay Ponder. "TINKER website". https://dasher.wustl.edu/tinker/. URL: https://dasher.wustl.edu/tinker/.

[24] R Konecny, N A Baker, and J A McCammon. "iAPBS: a programming interface to Adaptive Poisson-Boltzmann Solver (APBS)". In: *Computational Science and Discovery* 5.1 (2012). DOI: 10.1088/1749-4699/5/1/015005.

[25] "iAPBS website". https://mccammon.ucsd.edu/iapbs/. URL: https://mccammon.ucsd.edu/iapbs/.

[26] M Holst, N Baker, and F Wang. "Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples". In: *Journal of Computational Chemistry* 21.15 (2000), pp. 1319–1342.

[27] 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". In: *Journal of Computational Chemistry* 21.15 (2000), pp. 1343–1352.

[28] N A Baker et al. "The adaptive multilevel finite element solution of the Poisson-Boltzmann equation on massively parallel computers". In: *IBM Journal of Research and Development* 45.3-4 (2001), pp. 427–438.

[29] Itay Lotan and Teresa Head-Gordon. "An Analytical Electrostatic Model for Salt Screened Interactions between Multiple Proteins". In: *Journal of Chemical Theory and Computation* 2.3 (2006). PMID: 26626662, pp. 541–555. DOI: 10.1021/ct050263p. eprint: http://dx.doi.org/10.1021/ct050263p.

[30] Lisa E. Felberg et al. "PB-AM: An open-source, fully analytical linear Poisson-Boltzmann solver". In: *Journal of Computational Chemistry* 38.15 (2017), pp. 1275–1282. ISSN: 1096-987X. DOI: 10.1002/jcc.24528.

[31] E-H Yap and T Head-Gordon. "New and Efficient Poisson-Boltzmann Solver for Interaction of Multiple Proteins". In: *Journal of Chemical Theory and Computation* 6.7 (2010), pp. 2214–2224. DOI: 10.1021/ct100145f.

[32] E. H. Yap and T. Head-Gordon. "Calculating the Bimolecular Rate of Protein-Protein Association with Interacting Crowders". In: *Journal of Chemical Theory and Computation* 9.5 (2013), pp. 2481–9. ISSN: 1549-9618 (Print) 1549-9618 (Linking). DOI: 10.1021/ct400048q. URL: http://www.ncbi.nlm.nih.gov/pubmed/26583736.

[33] Donald Bashford. "An Object-Oriented Programming Suite for Electrostatic Effects in Biological Molecules". In: *Scientific Computing in Object-Oriented Parallel Environments*. Ed. by Yutaka Ishikawa et al. Vol. 1343. Lecture Notes in Computer Science. Berlin: Springer, 1997, pp. 233–240.

[34] Garrett M. Morris et al. "AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility". In: *Journal of Computational Chemistry* 30 (2009), pp. 2785–2791. URL: http://www3.interscience.wiley.com/cgi-bin/abstract/122365050/ABSTRACT.

[35] "Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description. Version 3.3". http://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html. 2012. URL: http://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html.

[36] "PDBx/mmCIF Dictionary Resources". http://mmcif.wwpdb.org/. URL: http://mmcif.wwpdb.org/.

[37] J. Westbrook et al. "PDBML: the representation of archival macromolecular structure data in XML". In: *Bioinformatics* 21.7 (2005), pp. 988–992. DOI: 10.1093/bioinformatics/bti082.

[38] Todd J Dolinsky et al. "PDB2PQR: an automated pipeline for the setup of Poisson–Boltzmann electrostatics calculations". In: *Nucleic Acids Research* 32.suppl 2 (2004), W665–W667.

[39] T J Dolinsky et al. "PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations". In: *Nucleic Acids Research* 35 (2007), W522–5. DOI: 10.1093/nar/gkm276.

[40] "sed - a stream editor". https://www.gnu.org/software/sed/manual/sed.html. URL: https://www.gnu.org/software/sed/manual/sed.html.

[41] Chresten R. Sondergaard et al. "Improved Treatment of Ligands and Coupling Effects in Empirical Calculation and Rationalization of pK_a Values". In: *Journal of Chemical Theory and Computation* 7.7 (2011), pp. 2284–2295.

[42] H. Li, A. D. Robertson, and J. H. Jensen. "Very fast empirical prediction and rationalization of protein pK_a values". In: *Proteins* 61 (2005), p. 704721.

[43] Emilie Purvine et al. "Energy Minimization of Discrete Protein Titration State Models Using Graph Theory". In: *Journal of Physical Chemistry* 120 (2016), pp. 8354–8360. DOI: 10.1021/acs.jpcb.6b02059.

[44] J E Nielsen and G Vriend. "Optimizing the hydrogen-bond network in Poisson-Boltzmann equation-based pK_a calculations". In: *Proteins* 43.4 (2001), pp. 403–12.

[45] Junmei Wang, Piotr Cieplak, and Peter A. Kollman. "How well does a restrained electrostatic potential (RESP) model perform in calculating conformational energies of organic and biological molecules?" In: *Journal of Computational Chemistry* 21.12 (2000), pp. 1049–1074. DOI: 10.1002/1096-987X(200009)21:12<1049::AID-JCC3>3.0.CO;2-F.

[46] A.D. MacKerell Jr., M. Feig, and C.L. Brooks III. "Extending the treatment of backbone energetics in protein force fields: limitations of gas-phase quantum mechanics in reproducing protein conformational distributions in molecular dynamics simulations". In: *Journal of Computational Chemistry* 25.11 (2004), pp. 1400–1415. DOI: 10.1002/jcc.20065.

[47] Doree Sitkoff, Kim A. Sharp, and Barry Hong. "Accurate calculation of hydration free energies using macroscopic solvation models". In: *Journal of Physical Chemistry* 98.7 (1994), pp. 1978–1988.

[48] Paul Czodrowski et al. "Development, validation, and application of adapted PEOE charges to estimate pK_a values of functional groups in protein-ligand complexes". In: *Proteins* 65.2 (2006), pp. 424–437. ISSN: 1097-0134. DOI: 10.1002/prot.21110.

[49] J M J Swanson, S A Adcock, and J A McCammon. "Optimized Radii for Poisson-Boltzmann Calculations with the AMBER Force Field". In: *Journal of Chemical Theory and Computation* 1.3 (2005), pp. 484–93. DOI: 10.1021/ct049834o.

[50] Chunhu Tan, Lijiang Yang, and Ray Luo. "How Well Does Poisson-Boltzmann Implicit Solvent Agree with Explicit Solvent? A Quantitative Analysis". In: *Journal of Physical Chemistry B* 110.37 (2006), pp. 18680–18687. ISSN: 1520-6106. DOI: 10.1021/jp063479b.

[51] Michael Holst. "Adaptive numerical treatment of elliptic systems on manifolds". In: *Journal of Computational Mathematics* 15 (2001), pp. 139–191.

[52] "FEtk website". http://www.fetk.org. URL: http://www.fetk.org.

[53] Kaihsu Tai et al. "Finite element simulations of acetylcholine diffusion in neuromuscular junctions". In: *Biophysical Journal* 84.4 (2003), pp. 2234–2241. DOI: 10.1016/S0006-3495(03)75029-2.

[54] Michael Holst and Faisal Saied. "Multigrid solution of the Poisson—Boltzmann equation". In: *Journal of Computational Chemistry* 14.1 (1993), pp. 105–113. DOI: 10.1002/jcc.540140114.

[55] Z Chen, N A Baker, and G W Wei. "Differential geometry based solvation model I: Eulerian formulation". In: *Journal of Computational Physics* 229.22 (2010), pp. 8231–8258. DOI: 10.1016/j.jcp.2010.06.036.

[56] Z Chen, N A Baker, and G W Wei. "Differential geometry based solvation model II: Lagrangian formulation". In: *Journal of Mathematical Biology* 63.6 (2011), pp. 1139–200. DOI: 10.1007/s00285-011-0402-z.

[57] Z Chen et al. "Variational approach for nonpolar solvation analysis". In: *Journal of Chemical Physics* 137.8 (2012), p. 084101. DOI: 10.1063/1.4745084.

[58] M D Daily et al. "Origin of parameter degeneracy and molecular shape relationships in geometric-flow calculations of solvation free energies". In: *Journal of Chemical Physics* 139.20 (2013), p. 204108. DOI: 10.1063/1.4832900.

[59] D G Thomas et al. "ISA-TAB-Nano: a specification for sharing nanomaterial research data in spreadsheet-based format". In: *BMC Biotechnology* 13 (2013), p. 2. DOI: 10.1186/1472-6750-13-2.

[60] Weihua Geng and Robert Krasny. "A treecode-accelerated boundary integral Poisson-Boltzmann solver for electrostatics of solvated biomolecules". In: *Journal of Computational Physics* 247 (2013), pp. 62–78.

[61] André Juffer et al. "The electric potential of a macromolecule in a solvent: A fundamental approach". In: *Journal of Computational Physics* 97.1 (1991), pp. 144–171.

[62] Michel Sanner, Arthur Olson, and Jean Claude Spehner. "Fast and robust computation of molecular surfaces". In: *Proc 11th ACM Symp Comp Geom.* ACM. 1995, pp. C6–C7.

[63] Sergio Decherchi and Walter Rocchia. "A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale". In: *PLoS ONE* 8.4 (2013), e59744.

[64] Peijun Li, Hans Johnston, and Robert Krasny. "A Cartesian treecode for screened Coulomb interactions". In: *Journal of Computational Physics* 228 (2009), pp. 3858–3868.

[65] D. Bashford and D. A. Case. "Generalized Born models of macromolecular solvation effects". In: *Annual Review in Physical Chemistry* 51 (2000), pp. 129–152. DOI: 10.1146/annurev.physchem.51.1.129.

[66] W Humphrey, A Dalke, and K Schulten. "VMD: visual molecular dynamics". In: *Journal of Molecular Graphics* 14.1 (1996), pp. 33–8, 27–8.

[67] Donald Ermak and J. A. McCammon. "Brownian dynamics with hydrodynamic interactions". In: *Journal of Chemical Physics* 69.4 (1978), pp. 1352–1360.

[68] Schrödinger, LLC. "The PyMOL Molecular Graphics System, Version 1.8". PyMOL. 2015.

[69] "PyMOL website". http://www.pymol.org. URL: http://www.pymol.org.

[70] "VMD website". http://www.ks.uiuc.edu/Research/vmd/.

[71] Graham T. Johnson et al. "ePMV Embeds Molecular Modeling into Professional Animation Software Environments". In: *Structure* 19.3 (2011), pp. 293–303. DOI: 10.1016/j.str.2010.12.023.

[72] "MGLTools (AutoDock, PMV, Vision) website". http://mgltools.scripps.edu/. URL: http://mgltools.scripps.edu/.

[73] "Chimera website". https://www.cgl.ucsf.edu/chimera/. URL: https://www.cgl.ucsf.edu/chimera/.

[74] E F Pettersen et al. "UCSF Chimera–a visualization system for exploratory research and analysis". In: *Journal of Computational Chemistry* 25.13 (2004), pp. 1605–12. DOI: 10.1002/jcc.20084.

[75] "Jmol website". http://jmol.sourceforge.net/. URL: http://jmol.sourceforge.net/.

[76] Angel Herraez. "Biomolecules in the computer: Jmol to the rescue". In: *Biochemistry and Molecular Biology Education* 34.4 (2006), pp. 255–261.

[77] D Koes. "3Dmol.js". http://3dmol.csb.pitt.edu/. URL: http://3dmol.csb.pitt.edu/ (visited on 2016).

[78] Nicholas Rego and David Koes. "3Dmol.js: molecular visualization with WebGL". In: *Bioinformatics* 31.8 (2015), pp. 1322–1324. URL: http://3dmol.csb.pitt.edu/.

[79] S Unni et al. "Web servers and services for electrostatics calculations with APBS and PDB2PQR". In: *Journal of Computational Chemistry* 32.7 (2011), pp. 1488–91. DOI: 10.1002/jcc.21720.

[80] RO Dror et al. "Structural basis for modulation of a G-protein-coupled receptor by allosteric drugs". In: *Nature* 503.7475 (2013), pp. 295–299.

[81] L Treuel et al. "Impact of Protein Modification on the Protein Corona on Nanoparticles and Nanoparticle-Cell Interactions". In: *ACS Nano* 8.1 (2013), pp. 503–513.

[82] Silvia H. De Paoli et al. "The effect of protein corona composition on the interaction of carbon nanotubes with human blood platelets". In: *Biomaterials* 35.24 (2014), pp. 6182–6194. ISSN: 0142-9612. URL: http://www.sciencedirect.com/science/article/pii/S0142961214004657.

[83] J Lipfert et al. "Understanding nucleic acid-ion interactions". In: *Annual Review of Biochemistry* 83 (2014), pp. 813–41. DOI: 10.1146/annurev-biochem-060409-092720.

[84] V A Roberts et al. "DOT2: Macromolecular docking with improved biophysical models". In: *Journal of Computational Chemistry* 34.20 (2013), pp. 1743–58. DOI: 10.1002/jcc.23304.

[85] T. Evangelidis et al. "An integrated workflow for proteome-wide off-target identification and polypharmacology drug design". In: *International Conference on Bioinformatics and Biomedicine Workshops*. IEEE. 2009, pp. 32–39. URL: doi:%2010.1109/BIBMW.2012.6470348.

[86] E Spiga et al. "Electrostatic-Consistent Coarse-Grained Potentials for Molecular Simulations of Proteins". In: *Journal of Chemical Theory and Computation* 9.8 (2013), pp. 3515–3526. DOI: 10.1021/ct400137q.

[87] Phillip J Stansfeld et al. "MemProtMD: Automated Insertion of Membrane Protein Structures into Explicit Lipid Membranes". In: *Structure* 23.7 (2015), pp. 1350–1361.

[88] S Richter et al. "webPIPSA: a web server for the comparison of protein interaction properties". In: *Nucleic Acids Res* 36.Web Server issue (2008), W276–80. DOI: 10.1093/nar/gkn181.

[89] Gary A. Huber and J. Andrew McCammon. "Browndye: A software package for Brownian dynamics". In: *Computer Physics Communications* 181.11 (2010), pp. 1896–1905. DOI: 10.1016/j.cpc.2010.07.022.

[90] Michael Martinez et al. "SDA 7: A modular and parallel implementation of the simulation of diffusional association software". In: *Journal of Computational Chemistry* 36.21 (2015), pp. 1631–45. DOI: 10.1002/jcc.23971.

[91] Y Cheng et al. "Finite element analysis of the time-dependent Smoluchowski equation for acetylcholinesterase reaction rate calculations". In: *Biophysical Journal* 92.10 (2007), pp. 3397–406. DOI: 10.1529/biophysj.106.102533.

[92] Y Cheng et al. "Continuum simulations of acetylcholine diffusion with reaction-determined boundaries in neuromuscular junction models". In: *Biophysical Chemistry* 127.3 (2007), pp. 129–39. DOI: 10.1016/j.bpc.2007.01.003.

[93] Y Song et al. "Continuum diffusion reaction rate calculations of wild-type and mutant mouse acetylcholinesterase: adaptive finite element analysis". In: *Biophysical Journal* 87.3 (2004), pp. 1558–1566.

[94] A H Elcock. "Molecular simulations of diffusion and association in multimacromolecular systems". In: *Methods in Enzymology* 383 (2004), pp. 166–98. DOI: 10.1016/S0076-6879(04)83008-8.

[95] P Mereghetti and RC Wade. "Atomic Detail Brownian Dynamics Simulations of Concentrated Protein Solutions with a Mean Field Treatment of Hydrodynamic Interactions". In: *Journal of Physical Chemistry* 116.29 (2012), pp. 8523–8533.

[96] Michael J Holst. "Clean Object-Oriented C". http://fetk.org/codes/maloc/api/html/index.html.

[97] Michael K. Gilson, Kim A. Sharp, and Barry H. Honig. "Calculating the electrostatic potential of molecules in solution: Method and error assessment". In: *Journal of Computational Chemistry* 9.4 (1988), pp. 327–335. DOI: 10.1002/jcc.540090407.

[98] W. Im, D. Beglov, and B. Roux. "Continuum Solvation Model: Computation of electrostatic forces from numerical solutions to the Poisson-Boltzmann equation". In: *Computer Physics Communications* 111 (1998), pp. 59–75.

[99] M. J. Schnieders et al. "Polarizable atomic multipole solutes in a Poisson-Boltzmann continuum". In: *Journal of Chemical Physics* 126 (2007), p. 124114. URL: https://www.ncbi.nlm.nih.gov/pubmed/17411115.

[100] Robert E. Bruccoleri et al. "Finite difference Poisson-Boltzmann electrostatic calculations: Increased accuracy achieved by harmonic dielectric smoothing and charge antialiasing". In: *Journal of Computational Chemistry* 18.2 (1997), pp. 268–276.

[101] Mafalda Nina, Wonpil Im, and Benoit Roux. "Optimized atomic radii for protein continuum electrostatics solvation forces". In: *Biophysical Chemistry* 78.1-2 (1999), pp. 89–96. DOI: 10.1016/S0301-4622(98)00236-1.

[102] D G Thomas et al. "Parameterization of a geometric flow implicit solvation model". In: *Journal of Computational Chemistry* 34.8 (2013), pp. 687–95. DOI: 10.1002/jcc.23181.

[103] Y. Saad and M. Schultz. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *Journal of Scientific and Statistical Computing* 7 (1986), pp. 856–869.

[104] "OpenDX". http://fetk.org/codes/maloc/api/html/index.html. URL: http://www.opendx.org/.

[105] James Ahrens, Berk Geveci, and Charles Law. *ParaView: An End-User Tool for Large Data Visualization*. Elsevier, 2005.