THE PROTEIN SOCIETY

TOOLS FOR PROTEIN SCIENCE

Improvements to the APBS biomolecular solvation software suite

Elizabeth Jurrus,[1] Dave Engel,[1] Keith Star,[1] Kyle Monson,[1] Juan Brandi,[1] Lisa E. Felberg,[2] David H. Brookes,[2] Leighton Wilson,[3] Jiahui Chen,[4] Karina Liles,[1] Minju Chun,[1] Peter Li,[1] David W. Gohara,[5] Todd Dolinsky,[6] Robert Konecny,[7] David R. Koes,[8] Jens Erik Nielsen,[9] Teresa Head-Gordon,[2] Weihua Geng,[4] Robert Krasny,[3] Guo-Wei Wei,[10] Michael J. Holst,[7] J. Andrew McCammon,[7] and Nathan A. Baker[1,11]*

[1]Pacific Northwest National Laboratory, Richland, Washington
[2]University of California, Berkeley, California
[3]University of Michigan, Ann Arbor, Michigan
[4]Southern Methodist University, Dallas, Texas
[5]St. Louis University, St. Louis, Missouri
[6]FoodLogiQ, Durham, North Carolina
[7]University of California San Diego, San Diego, California
[8]University of Pittsburgh, Pittsburgh, Pennsylvania
[9]Protein Engineering, Novozymes A/S, Copenhagen, Denmark
[10]Michigan State University, East Lansing, Michigan
[11]Brown University, Providence, Rhode Island

Received 30 June 2017; Accepted 22 August 2017
DOI: 10.1002/pro.3280
Published online 24 August 2017 proteinscience.org

Abstract: The Adaptive Poisson–Boltzmann Solver (APBS) software was developed to solve the equations of continuum electrostatics for large biomolecular assemblages that have provided impact in the study of a broad range of chemical, biological, and biomedical applications. APBS addresses the 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

Additional Supporting Information may be found in the online version of this article.

Grant sponsor: National Institute of General Medical Sciences, NIH; Grant numbers: GM069702, GM31749, GM103426; Grant sponsor: Office of Science, Office of Basic Energy Sciences, U.S. Department of Energy; Grant numbers: DE-AC02-05CH11231, DE-AC05-76RL01830; Grant sponsor: National Science Foundation Graduate Research Fellowship, Division of Graduate Education; Grant number: 110640; Grant sponsor: Department of Defense, National Defense Science & Engineering Graduate Fellowship (NDSEG); Grant sponsor: National Science Foundation, Division of Mathematical Sciences; Grant number: 1418966.

*Correspondence to: Nathan A. Baker, Advanced Computing, Mathematics, and Data Division; Pacific Northwest National Laboratory; Richland, WA 99352. E-mail: nathan.baker@pnnl.gov

updated APBS and its suite of accompanying software since its release in 2001. In this article, 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.

Keywords: electrostatics; software; solvation; titration; pK_a

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.[1] Continuum 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

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

The solvent is described by the bulk solvent dielectric constant ε_s and 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 and 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 have 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 the section "Preparing Biomolecular Structures") and a fast finite-difference solution of the Poisson–Boltzmann equation (see the section "Finite-difference and finite-element solvers") 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 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 third-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

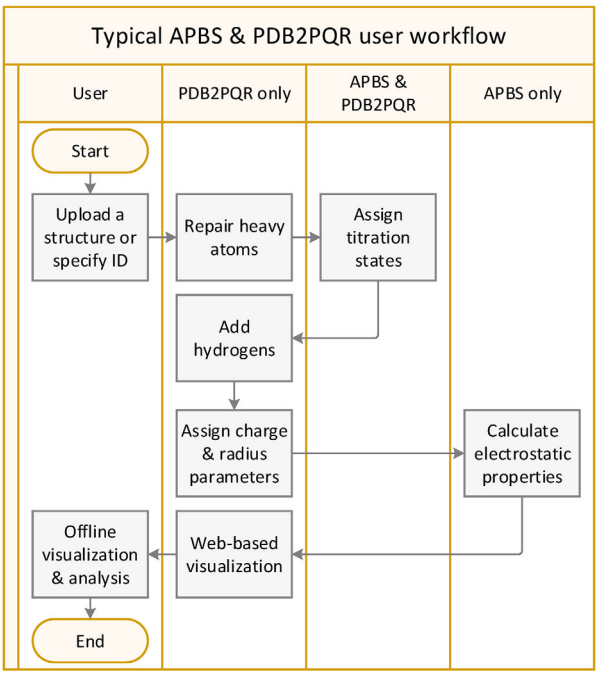

Figure 1. Workflow for biomolecular electrostatics calculations using the APBS-PDB2PQR software suite. This workflow is supported by the APBS tool suite with specific support by PDB2PQR for preparing biomolecular structures (see the section "Preparing Biomolecular Structures") and support by APBS for performing electrostatics calculations (see the section "Solving the Poisson–Boltzmann and Related Solvation Equations").

treats full mutual polarization accurately, known as PBAM (Poisson-Boltzmann Analytical Model)[29,30] and 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] As discussed in the following sections, APBS-PDB2PQR offers a wide range of features for users across all levels of computational biology expertise.

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.[1] 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] 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 (nonhydrogen) 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.[38]

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[40]) 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.[41]
- *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 calculate 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.[42] The paper by Purvine et al.[42] describes the performance of the new graph cut method compared to existing approaches. While the new deterministic method offers advantages for large systems, the traditional stochastic approach is often sufficient for small to moderate systems. 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.[43] 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,[44] CHARMM22,[45] PARSE,[46] PEOE_PB,[47] Swanson et al.,[48] and Tan et al.[49] Many of these force fields only provide parameters for amino acid biomolecules. The PEOE approach[47] provides algorithms to parameterize ligands. However, we welcome contributions for other biomolecular force fields, particularly for lipids and amino acids.

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]

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 multilevel adaptive finite-element library.[50,51] Adaptive finite-element methods can resolve extremely fine features of a complex system (such as 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.[52] The finite-difference PMG solver[51,53] 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.

Geometric flow

Several recent papers have described our work on a geometric flow formulation of Poisson-based implicit solvent models.[54–58] The components of this geometric model are described in previous publications.[56,57] The geometric flow approach couples the polar and nonpolar components of the implicit solvent model with two primary advantages over existing methods. 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.

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)].[59] A discussion of the relative merits between finite difference/element and boundary integral PB methods are provided in Geng et al. and Li et al.[59,60] 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

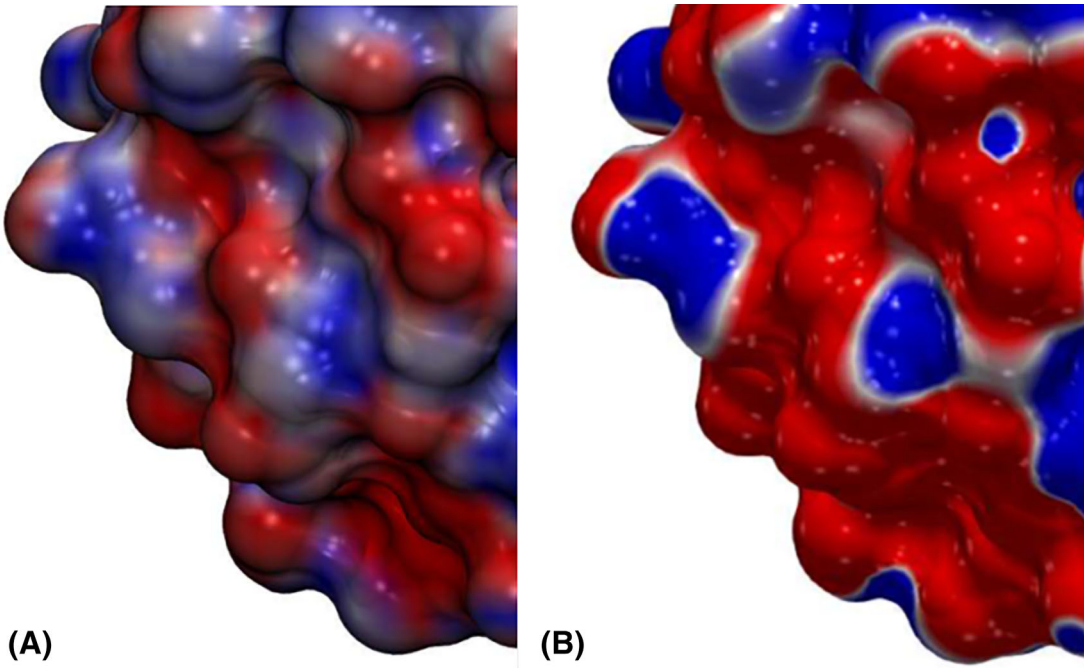

Figure 2. Electrostatic potential visualization of protein with PDB ID 3app for (A) APBS multigrid and (B) TABI-PB. VMD[64] was used to generate the figure in (A), and VTK to generate the figure in (B). The potentials are on a $[-4, 4]$ red–white–blue color map in units of kJ/mol/e. Calculations were performed at 0.15 M ionic strength in monovalent salt, 298.15 K, protein dielectric 2, and solvent dielectric 78.

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.[60,61] A comparison of electrostatic potential with PDB ID 3app from the APBS multigrid solution method and the new TABI-PB solver are shown in Figure 2. Additional information about the boundary element method and its implementation in APBS is provided in Appendix D.

Analytical and semi-analytical methods

Numerical solution methods tend to be computationally intensive, which has led to the adoption of 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 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.

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 3.

Because it is fully analytical, PB-AM can be used for model validation and 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.[66] Additional details about these methods and their use in APBS are presented in Appendix E. In addition to these advantages, the strengths and weaknesses of PB-AM and PB-SAM over existing methods are discussion in Yap et al.[31] and Lotan et al.[29]

Using APBS Results

Visualization

One of the primary uses of the APBS tools is to generate electrostatic potentials for use in biomolecular

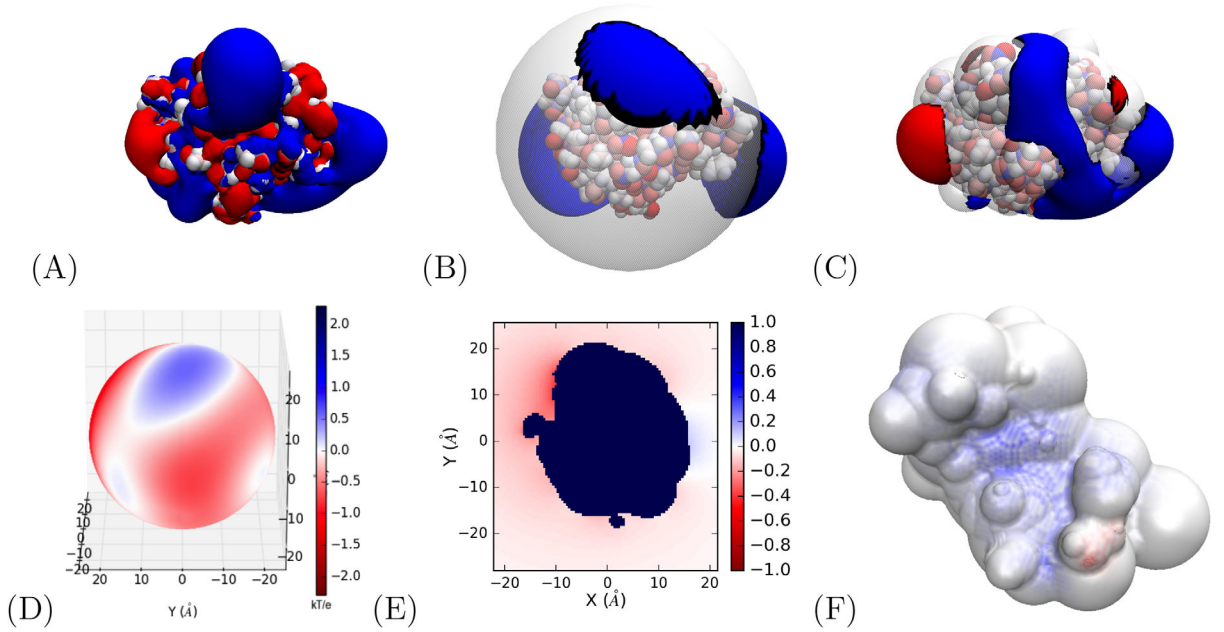

Figure 3. Comparison of APBS, PB-AM, and PB-SAM results (A–C) and electrostatic potential visualization for APBS PB-AM and PB-SAM (D–E). VMD[64] isosurface of barnase molecule generated using (A) APBS boundary element method, (B) APBS PB-AM method, and (C) APBS PB-SAM method. (A–C) atoms colored according to their charge and isosurfaces are drawn at 1.0 (blue) and −1.0 (red) kT/e electrostatic potential. (D) APBS PB-AM potential on the coarse-grain surface of the barnase molecule, (E) APBS PB-SAM potential in a 2D plane surrounding the barstar molecule, and (F) APBS PB-SAM potential over range $[-1, 1]$ on the coarse-grain surface of the barnase molecule (blue region is the location of positive electrostatic potential and barstar association). All calculations were performed at 0.0 M ionic strength, 300 Kelvin, pH 7, protein dielectric 2, and solvent dielectric 78. All electrostatic potentials are given in units of kT/e.

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,[67,68] VMD,[64,69] PMV,[70,71] and Chimera.[72,73] We have also worked with the developers of Jmol[74,75] and 3Dmol.js[76,77] to provide web-based setup and visualization of APBS-PDB2PQR calculations and related workflows. APBS integration with Jmol has been described previously.[78] 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 are 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 4. Examples of 3Dmol.js visualization options are shown in Figure 5.

Other applications

Besides visualization and the processes described in "Preparing Biomolecular Structures," 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,[79] understanding protein–nanoparticle interactions,[80,81] understanding nucleic acid–ion interactions,[82] biomolecular docking[83] and ligand binding,[84] developing new coarse-grained protein models,[85] setting up membrane protein simulations,[86] and so on. APBS also plays a key role in PIPSA for protein surface electrostatics analysis[87] and BrownDye[88] and SDA[89] 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 these areas.[79,89–94]

The Future of APBS

To help with modularity and to facilitate extensibility, APBS was based on an object-oriented programming paradigm with strict ANSI-C compliance. This "Clean OO C"[95] has enabled the long-term sustainability of APBS 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.

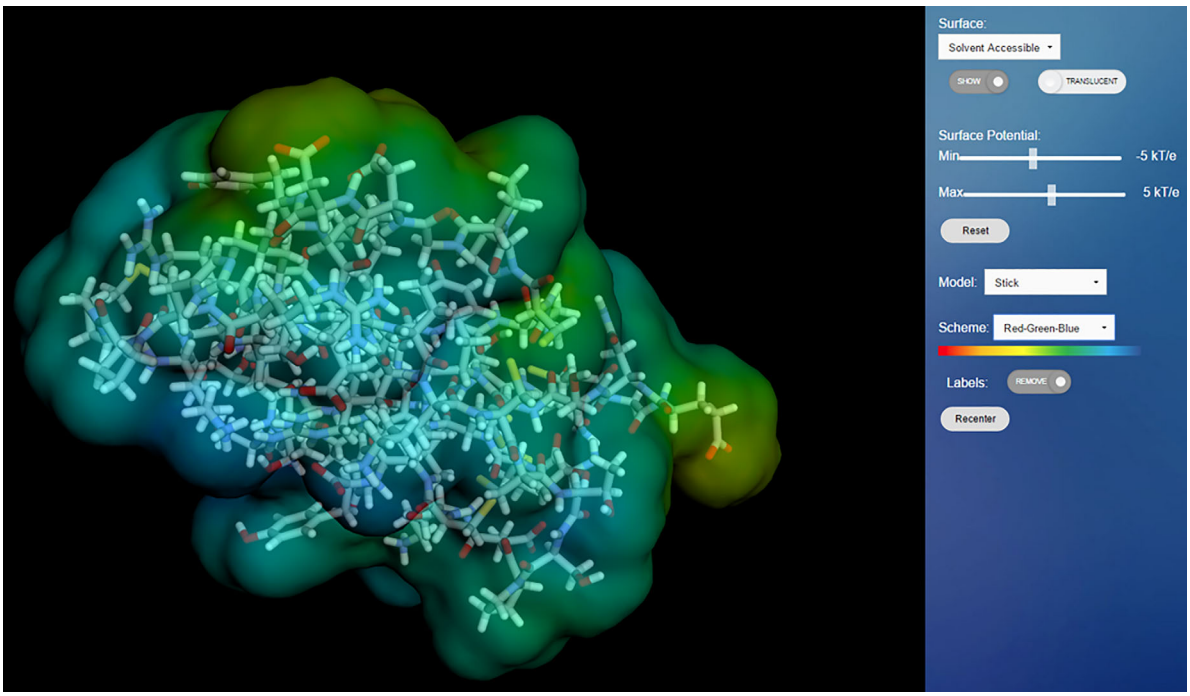

Figure 4. 3Dmol.js interface displaying a rendering of fasciculin-2 (1FAS) protein with translucent, solvent-accessible surface using a stick model and red–green–blue color scheme.

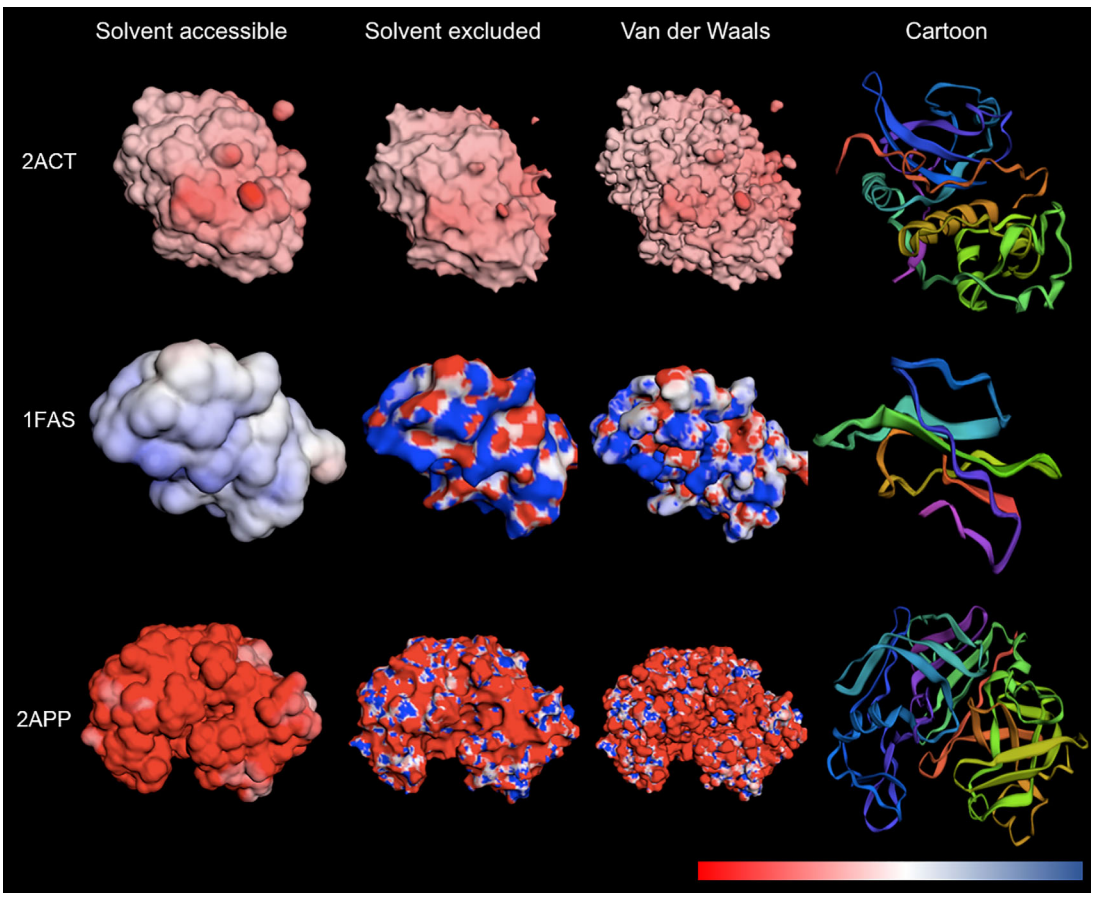

Figure 5. Renderings of three different proteins: actinidin (2ACT) (top), fasciculin-2 (1FAS) (center), and pepsin-penicillium (2APP) (bottom). To demonstrate the different visualization options. From left to right: solvent-accessible surface, solvent-excluded surface, van der Waals surface, and cartoon models are shown all using red–white–blue color scheme (excluding cartoon model), where red and blue correspond to negative and positive electrostatic potentials, respectively.

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.

APPENDICES

The 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.

APPENDIX 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 article:

- 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

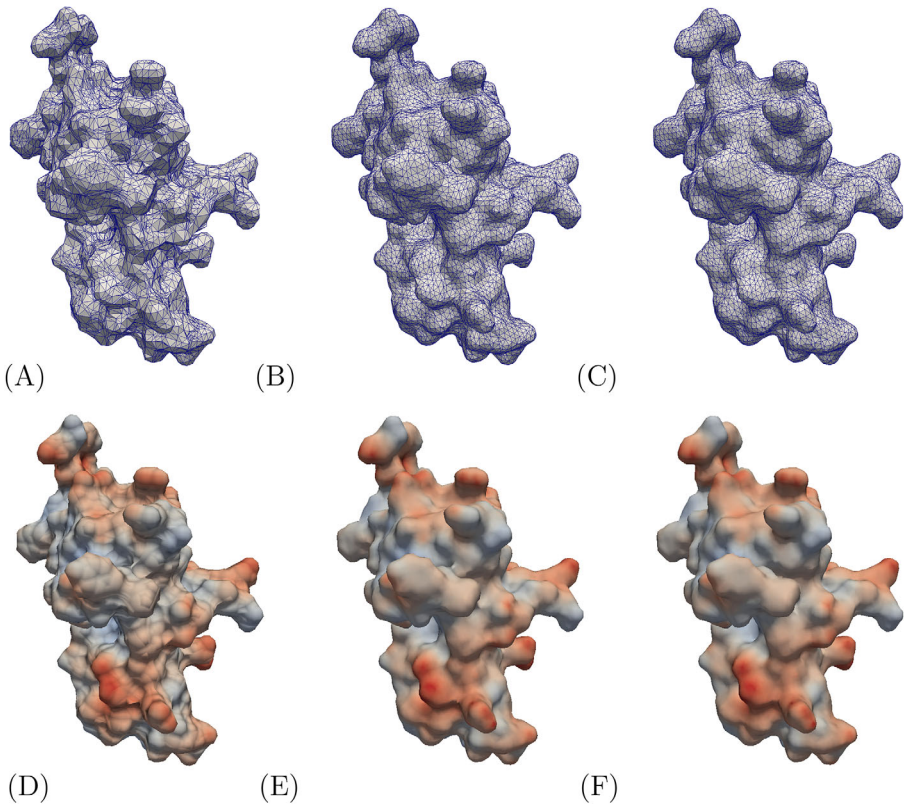

Figure A1. APBS TABI-PB electrostatic potential results for PDB ID 1a63. Surfaces were generated with (A) 20228 triangles via MSMS,[62] (B) 20744 triangles via NanoShaper SES, and (C) 21084 triangles via NanoShaper Skin.[63] These discretizations resulted in surface potentials (with units kT/e) of (D) 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 300 K. Red and blue surface colors correspond to negative and positive electrostatic surface potentials, respectively.

(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]

APPENDIX 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[18,26–28,53]; this section focuses on the configuration and use of these methods in APBS.

Finite-Difference Calculation Configuration

APBS users have several ways to invoke the PMG finite-difference solver[53] capabilities of APBS through keywords in the APBS input file ELEC block. The different types of finite-difference calculations include the following:

- 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[96] 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 ~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 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.[97] 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.[98] 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 nonzero 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.[99]
- spl2: The dielectric and ion-accessibility coefficients are defined by a cubic-spline surface.[97] 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.[100]
- spl4: The dielectric and ion-accessibility coefficients are defined by a seventh-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).[98]

Finite-Elements Calculation Configuration

Users invoke the FEtk finite-element solver[50] 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.

APPENDIX C: GEOMETRIC FLOW CALCULATIONS IN APBS

This section contains additional information about the geometric flow equation implementation in APBS introduced in the section "Geometric flow." The geometric flow methods used by APBS have been described extensively in previous publications[54–57,101]; this section focuses on the configuration and use of these methods in APBS.

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 the section "Finite-Difference Calculation Configuration." 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.,[57] 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.,[57] 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 an READ statement in the APBS input file.

APPENDIX D: BOUNDARY ELEMENT METHOD IMPLEMENTATION

This appendix provides additional information about the boundary element method introduced in the section "Boundary element methods".

Boundary Element Method Background

This section provides additional background on the TABI-PB boundary element solver,[59] introduced in the section "Boundary element methods." As described earlier, this method involves solving two coupled integral equations defined on the solute–solvent boundary which 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. Figure A1 shows different types of surface discretizations and example electrostatic potential output.

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

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

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).

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

where G_0 and G_κ are the Coulomb and screened Coulomb potentials, respectively, defined as

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

The normal derivatives of the potential kernels G are

$$\begin{aligned}
\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial v_\mathbf{y}} &= \sum_{n=1}^{3} v_n(\mathbf{y}) \partial_{y_n} G(\mathbf{x},\mathbf{y}), \\
\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial v_\mathbf{x}} &= -\sum_{n=1}^{3} v_n(\mathbf{x}) \partial_{x_n} G(\mathbf{x},\mathbf{y}), \\
\frac{\partial^2 G(\mathbf{x},\mathbf{y})}{\partial v_\mathbf{x} \partial v_\mathbf{y}} &= -\sum_{m=1}^{3}\sum_{n=1}^{3} v_m(\mathbf{x}) v_n(\mathbf{y}) \partial_{x_m} \partial_{y_n} G(\mathbf{x},\mathbf{y})
\end{aligned} \tag{5}$$

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

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

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

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

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 analogs 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):

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

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.[60] 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.[102]

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.

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 the following:

- 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 ⟨*crit erion*⟩ 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 A1 for an example of surface meshes.
- outdata ⟨*flag*⟩: Type of output data file generated; 0 = APBS OpenDX format[103] and 1 = ParaView format.[104]

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, and the electrostatic solvation energy. Examples of electrostatic surface potential on the protein 1a63 are shown in Figure A1 using MSMS and NanoShaper.

APPENDIX E: ANALYTICAL AND SEMI-ANALYTICAL METHOD IMPLEMENTATIONS

This appendix provides additional information about the analytical and semi-analytic methods[29,30]

introduced in the section "Analytical and semi-analytical methods."

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 the vector of the effective multipole expansion of the charge distributions of each molecule, E is the vector of the fixed charge distribution of all molecules, Γ is the dielectric boundary-crossing operator, Δ is the 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).

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

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 jth 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 and Head-Gordon.[29]

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. Equation (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

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

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

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

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

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

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

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 the scaled multipoles computed from fixed charges and polarization

charges from other spheres. $I^{(I,k)}_{E,n,m}$ is the matrix of the surface integrals over the exposed surface:

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

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:

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

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

$$H^{(I,k)}_{P,n,m} = h(\theta_p,\phi_p) Y^{(I,k)}_{n,m}(\theta_p,\phi_p) \tag{25}$$

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

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

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 the following:

- 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 Figure 3.

References

1. Ren P, Chun J, Thomas DG, Schnieders MJ, Marucho M, Zhang J, Baker NA (2012) Biomolecular electrostatics and solvation: a computational perspective. Quarter Rev Biophys 45:427–491. doi: 10.1017/S003358351200011X.
2. Davis ME, McCammon JA (1990) Electrostatics in biomolecular structure and dynamics. Chem Rev 90:509–521.
3. Perutz MF (1978) Electrostatic effects in proteins. Science 201:1187–1191.
4. Sharp KA, Honig B (1990) Electrostatic interactions in macromolecules: theory and applications. Annu Rev Biophys Biophys Chem 19:301–332. doi: 10.1146/annurev.bb.19.060190.001505.
5. Roux B, Simonson T (1999) Implicit solvent models. Biophys Chem 78:1–20.
6. Warshel A, Sharma PK, Kato M, Parson WW (2006) Modeling electrostatic effects in proteins. Biochim Biophys Acta 1764:1647–1676. doi: 10.1016/j.bbapap.2006.08.007.
7. Fixman M (1979) The Poisson-Boltzmann equation and its application to polyelectrolytes. J Chem Phys 70: 4995–5005.
8. Grochowski P, Trylska J (2008) Continuum molecular electrostatics, salt effects, and counterion binding–a review of the Poisson-Boltzmann theory and its modifications. Biopolymers 89:93–113. doi: 10.1002/bip.20877.
9. Lamm G, Lipkowitz KB, Larter R, Cundari TR, Boyd DB (2003) The Poisson-Boltzmann equation. Rev Comput Chem 19:147–365.
10. Lee B, Richards FM (1971) The interpretation of protein structures: estimation of static accessibility. J Mol Biol 55:379–400. doi: 10.1016/0022–2836(71)90324-X.
11. Netz RR, Orland H (2000) Beyond Poisson-Boltzmann: fluctuation effects and correlation functions. Eur Phys J E Soft Matter 1:203–214.
12. Brooks B, Brooks A, Mackerell CL, Nilsson L, Petrella R, Roux B, Won Y, Archontis G, Bartels C, Boresch S, Caflisch A, Caves L, Cui Q, Dinner AR, Feig M, Fischer S, Gao J, Hodoscek M, Im W, Kuczera K, Lazaridis T, Ma J, Ovchinnikov V, Paci E, Pastor RW, Post CB, Pu JZ, Schaefer M, Tidor B, Venable RM, Woodcock HL, Wu X, Yang W, York DM, Karplus M (2009) CHARMM: the biomolecular simulation program. J Comput Chem 30:1545–1614. doi: 10.1002/jcc.21287.
13. Case D, Cheatham T, Darden T, Gohlke H, Luo R, Merz K, Onufriev A, Simmerling C, Wang B, Woods R (2005) The AMBER biomolecular simulation programs. J Comput Chem 26:1668–1688. doi: 10.1002/jcc.20290.
14. Li L, Li C, Sarkar S, Zhang J, Witham S, Zhang Z, Wang L, Smith N, Petukh M, and Alexov E (2012) DelPhi: a comprehensive suite for DelPhi software and associated resources. BMC Biophys 5:9. doi: 10.1186/2046-1682-5-9.
15. Bochevarov A, Harder E, Hughes T, Greenwood J, Braden D, Philipp D, Rinaldo D, Halls M, Zhang J, Friesner R (2013) Jaguar: a high-performance quantum chemistry software program with strengths in life and materials sciences. Int J Quantum Chem 113: 2110–2142.
16. Grant JA, Pickup BT, Nicholls A (2001) A smooth permittivity function for Poisson-Boltzmann solvation methods. J Comput Chem 22:608–640.
17. Zhou YC, Feig M, Wei GW (2008) Highly accurate biomolecular electrostatics in continuum dielectric environments. J Comput Chem 29:87–87. doi: 10.1002/jcc.20769.
18. Baker NA, Sept D, Joseph S, Holst MJ, McCammon JA (2001) Electrostatics of nanosystems: application to microtubules and the ribosome. Proc Natl Acad Sci USA 98:10037–10041. doi: 10.1073/pnas.181342398.
19. "APBS website". URL: http://www.poissonboltzmann.org/.
20. Krishnan S, Clementi L, Ren J, Papadopoulos P, Li W "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. Phillips JC, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel RD, Kalé L, Schulten K (2005) Scalable molecular dynamics with NAMD. J Comput Chem 26:1781–1802. p doi: 10.1002/jcc.20289.
22. Baker D. Rosetta website. URL: https://www.rosettacommons.org/.
23. Ponder J. TINKER website. URL: https://dasher.wustl.edu/tinker/.
24. Konecny R, Baker NA, McCammon JA (2012) iAPBS: a programming interface to Adaptive Poisson-Boltzmann Solver (APBS). Comput Sci Disc 5:015005. doi: 10.1088/1749–4699/5/1/.
25. iAPBS website. URL: https://mccammon.ucsd.edu/iapbs/.
26. Holst M, Baker N, Wang F (2000) Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples. J Comput Chem 21: 1319–1342.
27. Baker N, Holst M, Wang F (2000) Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems. J Comput Chem 21:1343–1352.
28. Baker NA, Sept D, Holst MJ, McCammon JA (2001) The adaptive multilevel finite element solution of the Poisson-Boltzmann equation on massively parallel computers. IBM J Res Dev 45:427–438.
29. Lotan I, Head-Gordon T (2006) An analytical electrostatic model for salt screened interactions between multiple proteins. J Chem Theory Comput 2:541–555. PMID: 26626662, doi: 10.1021/ct050263p.
30. Feldberg LE, Brookes DH, Yap EH, Jurrus E, Baker NA, Head-Gordon T (2017) PB-AM: An open-source, fully analytical linear Poisson-Boltzmann solver. J Comput Chem 38:1275–1282. doi: 10.1002/jcc.24528.
31. Yap E-H, Head-Gordon T (2010) New and efficient Poisson-Boltzmann solver for interaction of multiple proteins. J Chem Theory Comput 6:2214–2224. doi: 10.1021/ct100145f.
32. Yap EH, Head-Gordon T (2013) Calculating the bimolecular rate of protein-protein association with interacting crowders. J Chem Theory Comput 9:2481–2489. issn: 1549–9618 (Print) 1549–9618 (Linking). doi: 10.1021/ct400048q.
33. Bashford D An object-oriented programming suite for electrostatic effects in biological molecules. In: Ishikawa Y, et al., Ed. (1997) Scientific computing in object-oriented parallel environments. Lecture notes in computer science, Vol. 1343. Berlin: Springer, pp 233–240.
34. Morris GM, Huey R, Lindstrom W, Sanner MF, Belew RK, Goodsell DS, Olson AJ (2009) AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility. J Comput Chem 30:2785–2791.

35. Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description. Version 3.3. URL: http://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html. 2012.
36. PDBx/mmCIF Dictionary Resources. URL: http://mmcif.wwpdb.org/.
37. Westbrook J, Ito N, Nakamura H, Henrick K, Berman HM (2005) PDBML: the representation of archival macromolecular structure data in XML. Bioinformatics 21:988–992. doi: 10.1093/bioinformatics/bti082.
38. Dolinsky TJ, Nielsen JE, McCammon JA, Baker NA (2004) PDB2PQR: an automated pipeline for the setup of Poisson–Boltzmann electrostatics calculations. Nucleic Acids Res 32:W665–W667.
39. Dolinsky TJ, Czodrowski P, Hui L, Nielsen JE, Jensen JH, Klebe G, Baker NA (2007) PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. Nucleic Acids Res 35:W522–W525. doi: 10.1093/nar/gkm276.
40. Søndergaard CR, Olsson MHM, Rostkowski M, Jensen JH (2011) Improved treatment of ligands and coupling effects in empirical calculation and rationalization of pK_a values. J Chem Theory Comput 7:2284–2295.
41. Li H, Robertson AD, Jensen JH (2005) Very fast empirical prediction and rationalization of protein pK*a* values. Proteins 61:704–721.
42. Purvine E, Monson K, Jurrus E, Starr K, Baker NA (2016) Energy minimization of discrete protein titration state models using graph theory. J Phys Chem 120:8354–8360. doi: 10.1021/acs.jpcb.6b02059.
43. Nielsen JE, Vriend G (2001) Optimizing the hydrogen-bond network in Poisson-Boltzmann equation-based pK*a* calculations. Proteins 43:403–412.
44. Wang J, Cieplak P, Kollman PA (2000) How well does a restrained electrostatic potential (RESP) model perform in calculating conformational energies of organic and biological molecules? J Comput Chem 21:1049–1074. doi: 10. 1002/1096-987X(200009)21:12 < 1049::AID-JCC3 > 3.0.CO;2-F.
45. MacKerell AD, Jr.,Feig M, Brooks CL III (2004) 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. J Comput Chem 25:1400–1415. doi: 10.1002/jcc.20065.
46. Sitkoff D, Sharp KA, Hong B (1994) Accurate calculation of hydration free energies using macroscopic solvation models. J Phys Chem 98:1978–1988.
47. Czodrowski P, Dramburg I, Sotriffer CA, Klebe G (2006) Development, validation, and application of adapted PEOE charges to estimate pK*a* values of functional groups in protein-ligand complexes. Proteins 65:424–437. ISSN: 1097-0134. doi: 10.1002/prot.21110.
48. Swanson JMJ, Adcock SA, McCammon JA (2005) Optimized radii for Poisson-Boltzmann calculations with the AMBER force field. J Chem Theory Comput 1:484–493. doi: 10.1021/ct049834o.
49. Chunhu T, Lijiang Y, Ray L (2006) How well does Poisson-Boltzmann implicit solvent agree with explicit solvent? A quantitative analysis. J Phys Chem B 110:18680–18687. ISSN: 1520–6106. doi: 10. 1021/jp063479b.
50. Holst M (2001) Adaptive numerical treatment of elliptic systems on manifolds. J Comput Math 15:139–191.
51. FEtk website. URL: http://www.fetk.org.
52. Tai K, Bond SD, MacMillan HR, Baker NA, Holst MJ, McCammon JA (2003) Finite element simulations of acetylcholine diffusion in neuromuscular junctions. Biophys J 84:2234–2241. doi: 10.1016/S0006–3495(03)75029-2.
53. Holst M, Saied F (1993) Multigrid solution of the Poisson—Boltzmann equation. J Comput Chem 14:105–113. doi: 10.1002/jcc.540140114.
54. Chen Z, Baker NA, Wei GW (2010) Differential geometry based solvation model I: Eulerian formulation. J Comput Phys 229:8231–8258. doi: 10.1016/j.jcp.2010.06.036.
55. Chen Z, Baker NA, Wei GW (2011) Differential geometry based solvation model II: Lagrangian formulation. J Math Biol 63:1139–1200. doi: 10.1007/s00285-011-0402-z.
56. Chen Z, Zhao S, Chun J, Thomas DG, Baker NA, Bates PW, Wei GW (2012) Variational approach for nonpolar solvation analysis. J Chem Phys 137:084101. doi: 10.1063/1.4745084.
57. Daily MD, Chun J, Heredia-Langner A, Wei G, Baker NA. (2013) Origin of parameter degeneracy and molecular shape relationships in geometric-flow calculations of solvation free energies. J Chem Phys 139:204108. doi: 10.1063/1.4832900.
58. Thomas DG, Gaheen S, Harper SL, Fritts M, Klaessig F, Hahn-Dantona E, Paik D, Pan S, Stafford GA, Freund ET, Klemm JD, Baker NA (2013) ISA-TAB-Nano: a specification for sharing nanomaterial research data in spreadsheet-based format. BMC Biotechnol 13:2. doi: 10.1186/1472–6750-13-2.
59. Geng W, Krasny R (2013) A treecode-accelerated boundary integral Poisson-Boltzmann solver for electrostatics of solvated biomolecules. J Comput Phys 247:62–78.
60. Li P, Johnston H, Krasny R (2009) A Cartesian treecode for screened Coulomb interactions. J Comput Phys 228:3858–3868.
61. Juffer AH, Botta EFF, van Keulen BAM, van der Ploeg A, Berendsen HJC (1991) The electric potential of a macromolecule in a solvent: a fundamental approach. J Comput Phys 97:144–171.
62. Sanner M, Olson A, Spehner JC. Fast and robust computation of molecular surfaces. In: *Proc 11th ACM Symp Comp Geom*. ACM. 1995, pp. C6–C7.
63. Decherchi S, Rocchia W (2013) A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale. PLoS ONE 8:e59744.
64. Humphrey W, Dalke A, Schulten K (1996) VMD: visual molecular dynamics. J Mol Graphics 14:33–38, 27–28.
65. Bashford D, Case DA (2000) Generalized Born models of macromolecular salvation effects. Annu Rev Phys Chem 51:129–152. doi: 10.1146/annurev.physchem. 51.1.129.
66. Ermak D, McCammon JA (1978) Brownian dynamics with hydrodynamic interactions. J Chem Phys 69:1352–1360.
67. Schrödinger LLC (2015) The PyMOL Molecular Graphics System, Version 1.8. PyMOL.
68. "PyMOL website". URL: http://www.pymol.org.
69. "VMD website". URL: http://www.ks.uiuc.edu/Research/vmd/.
70. Johnson GT, Autin L, Goodsell DS, Sanner MF, Olson AJ (2011) ePMV embeds molecular modeling into professional animation software environments. Structure 19:293–303.
71. MGLTools (AutoDock, PMV, Vision) website. URL: http://mgltools.scripps.edu/.
72. Chimera website. URL: https://www.cgl.ucsf.edu/chimera/.
73. Pettersen E, Goddard T, Huang C, Couch G, Greenblatt D, Meng E, Ferrin T (2004) UCSF Chimera–a visualization system for exploratory research and analysis. J Comput Chem 25:1605–1612. doi: 10.1002/jcc.20084.

74. Jmol website. URL: http://jmol.sourceforge.net/.
75. Herraez A (2006) Biomolecules in the computer: Jmol to the rescue. Biochem Mol Biol Educ 34:255–261.
76. Koes D. 3Dmol.js. URL: http://3dmol.csb.pitt.edu/ (visited on 2016).
77. Rego N, Koes D (2015) 3Dmol.js: molecular visualization with WebGL. Bioinformatics 31:1322–1324.
78. Unni S, Huang Y, Hanson RM, Tobias M, Krishnan S, Li WW, Nielsen JE, Baker NA (2011) Web servers and services for electrostatics calculations with APBS and PDB2PQR. J Comput Chem 32:1488–1491. doi: 10.1002/jcc.21720.
79. Dror RO, Green HF, Valant C, Borhani DW, Valcourt JR, Pan AC, Arlow DH, Canals M, Lane JR, Rahmani R, Baell JB, Sexton PM, Christopoulos A, Shaw DE (2013) Structural basis for modulation of a G-protein-coupled receptor by allosteric drugs. Nature 503:295–299.
80. Treuel L, Brandholt S, Maffre P, Wiegele S, Shang L, Nienhaus G (2013) Impact of protein modification on the protein corona on nanoparticles and nanoparticle-cell interactions. ACS Nano 8:503–513.
81. De Paoli SH, Diduch L, Tegegn T, Orecna M, Strader M, Karnaukhova E, Bonevich J, Holada K, Simak J (2014) The effect of protein corona composition on the interaction of carbon nanotubes with human blood platelets. Biomaterials 35:6182–6194. ISSN: 0142–9612.
82. Lipfert J, Doniach S, Das R, Herschlag D (2014) Understanding nucleic acid-ion interactions. Annu Rev Biochem 83:813–841. doi: 10.1146/annurev-biochem-060409–092720.
83. Roberts V, Thompson E, Pique M, Perez M, Ten Eyck L (2013) DOT2: macromolecular docking with improved biophysical models. J Comput Chem 34:1743–1758. doi: 10.1002/jcc.23304.
84. Evangelidis T, Bourne P, Xie L, Xie L (2009) An integrated workflow for proteome-wide off-target identification and polypharmacology drug design. In: *International Conference on Bioinformatics and Biomedicine Workshops*. IEEE. pp. 32–39. doi: 2010.1109/BIBMW.2012.6470348.
85. Spiga E, Alemani D, Degiacomi M, Cascella M, Peraro M (2013) Electrostatic-consistent coarse-grained potentials for molecular simulations of proteins. J Chem Theory Comput 9:3515–3526. doi: 10.1021/ct400137q.
86. Stansfeld PJ, Goose JE, Caffrey M, Carpenter EP, Parker JL (2015) MemProtMD: automated insertion of membrane protein structures into explicit lipid membranes. Structure 23:1350–1361.
87. Richter S, Wenzel A, Stein M, Gabdoulline R, Wade R (2008) webPIPSA: a web server for the comparison of protein interaction properties. Nucleic Acids Res 36: W276–W280. doi: 10.1093/nar/gkn181.
88. Huber GA, McCammon JA (2010) Browndye: a software package for Brownian dynamics. Comput Phys Commun 181:1896–1905. doi: 10.1016/j.cpc.2010.07.022.
89. Martinez M, Bruce N, Romanowska J, Kokh D, Ozboyaci M, Yu X, Öztürk M, Richter S, Wade R (2015) SDA 7: a modular and parallel implementation of the simulation of diffusional association software. J Comput Chem 36:1631–1645. doi: 10.1002/jcc.23971.
90. Cheng Y, Suen JK, Zhang D, Bond SD, Zhang Y, Song Y, Baker NA, Bajaj C, Holst MJ, McCammon JA (2007) Finite element analysis of the time-dependent Smoluchowski equation for acetylcholinesterase reaction rate calculations. Biophys J 92:3397–3406. doi: 10.1529/biophysj.106.102533.
91. Cheng Y, Suen JK, Radic Z, Bond SD, Holst MJ, McCammon JA (2007) Continuum simulations of acetylcholine diffusion with reaction-determined boundaries in neuromuscular junction models. Biophys Chem 127:129–139. doi: 10.1016/j.bpc.2007.01.003.
92. Song Y, Zhang Y, Bajaj C, Baker NA (2004) Continuum diffusion reaction rate calculations of wild-type and mutant mouse acetylcholinesterase: adaptive finite element analysis. Biophys J 87:1558–1566.
93. Elcock AH (2004) Molecular simulations of diffusion and association in multimacromolecular systems. Methods Enzymol 383:166–198. doi: 10.1016/S0076–6879(04)83008-8.
94. Mereghetti P, Wade RC (2012) Atomic detail brownian dynamics simulations of concentrated protein solutions with a mean field treatment of hydrodynamic interactions. J Phys Chem 116:8523–8533. p
95. Holst MJ Clean Object-Oriented C. URL: http://fetk.org/codes/maloc/api/html/index.html.
96. Gilson MK, Sharp KA, Honig BH (1988) Calculating the electrostatic potential of molecules in solution: method and error assessment. J Comput Chem 9:327–335. doi: 10.1002/jcc.540090407.
97. Im W, Beglov D, Roux B (1998) Continuum solvation model: computation of electrostatic forces from numerical solutions to the Poisson-Boltzmann equation. Comput Phys Commun 111:59–75.
98. Schnieders MJ, Baker NA, Ren P, Ponder JW (2007) Polarizable atomic multipole solutes in a Poisson-Boltzmann continuum. J Chem Phys 126:124114.
99. Bruccoleri RE, Novotny J, Davis ME, Sharp KA (1997) Finite difference Poisson-Boltzmann electrostatic calculations: increased accuracy achieved by harmonic dielectric smoothing and charge antialiasing. J Comput Chem 18:268–276.
100. Nina M, Im W, Roux B (1999) Optimized atomic radii for protein continuum electrostatics solvation forces. Biophys Chem 78:89–96. doi: 10.1016/S0301–4622(98)00236-1.
101. Thomas DG, Chun J, Chen Z, Wei G, Baker NA (2013) Parameterization of a geometric flow implicit solvation model. J Comput Chem 34:687–695. doi: 10.1002/jcc.23181.
102. Saad Y, Schultz M (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. J Sci Stat Comput 7:856–869.
103. OpenDX. http://fetk.org/codes/maloc/api/html/index.html. URL: http://www.opendx.org/.
104. Ahrens J, Geveci B, Law C (2005) ParaView: an end-user tool for large data visualization. Elsevier.