Improvements to the APBS biomolecular solvation software suite


*Pacific Northwest National Laboratory
†University of California, Berkeley
‡University of Michigan
§Southern Methodist University
○St. Louis University
‖FoodLogiQ
¶University of Pittsburgh
◦Protein Engineering, Novozymes A/S
∇Michigan State University
∆University of California San Diego
◁To whom correspondence should be addressed.

August 21, 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 [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

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

(1)

The solvent is described by the bulk solvent dielectric constant \(\epsilon_s\) 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 \(\epsilon\), and a charge distribution function \(\rho\).

The dielectric \(\epsilon\) is often set to a constant value \(\epsilon_m\) in the interior of the molecule and varies sharply across the molecular boundary to the value \(\epsilon_s\) which describes the bulk solvent. The shape of the boundary is determined by the size and location of the solute atoms as well as model-specific parameters such as a characteristic solvent molecule size [10]. The charge distribution \(\rho\) 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 \(\phi\) can be used in a variety of applications, including visualization, other structural analyses, diffusion simulations, and a number of other calculations that require global electrostatic properties. The PB theory is approximate and, as a result, has several well-known limitations which can affect its accuracy, particularly for strongly charged systems or high salt concentrations [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 \(\sim 27,000\) 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 visual-
ization 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.

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 [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 (non-hydrogen) atoms. Missing heavy atoms can be rebuilt using
standard amino acid topologies in conjunction with existing atomic coordinates to deter-
mine new positions for the missing heavy atoms. A debump option performs very limited
minimization of sidechain \( \chi \) angles to reduce steric clashes between rebuilt and existing
atoms [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 $O(2^N)$ 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 $O(N^4)$ 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 $\rho$) and radii (for $\epsilon$ and $V_i$) are assigned based on the chosen force field. PDB2PQR currently supports charge/radii force fields from AMBER99 [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.

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.1. 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 [50, 51]. 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 [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 $\phi$ for their visualization or simulation applications. Therefore, most APBS users employ the Holst group’s finite difference grid-based PMG solver for biomolecular electrostatics calculations. Appendix A.2 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 [54–58]. The components of this geometric model are described in previous publications [56, 57]. The geometric flow approach couples the polar and non-polar 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 A.3. 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) [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 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 multi-grid solution method and the new TABI-PB solver are shown in Figure A1. Additional information about the boundary element method and its implementation in APBS is provided in Appendix A.4.
3.4 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.

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 3.

Because it is fully analytical, PB-AM can be used for model validation as well as for repre-
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, (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, (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 $kT/e$. 
senting 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 A.5. In addition to these advantages, the strengths and weaknesses of PB-AM and PB-SAM over existing methods are discussed in Yap et al. [31] and Lotan et al. [29].

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 [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 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 4. Examples of 3Dmol.js visualization options are shown in Figure 5.

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 [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], etc. APBS also plays a key role in PIPSA for protein surface electrostatics analysis [87] as well as 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 of 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.

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.

Acknowledgments

The authors gratefully acknowledge NIH grant GM069702 for support of APBS and PDB2PQR. 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
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.
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.
A Appendix

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.1 General keywords for implicit solvent calculations

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

- **ion charge** \(\langle charge \rangle\) **conc** \(\langle conc \rangle\) **radius** \(\langle radius \rangle\): This line can appear multiple times to specify the ionic species in the calculation. This specifies the following terms in Eq. 1: \(\langle charge \rangle\) gives \(q_i\) in units of electrons, \(\langle conc \rangle\) gives \(c_i\) in units of M, and \(\langle radius \rangle\) specifies the ionic radius (in Å) used to calculate \(V_i\).

- **lpbe|npbe**: This keyword indicates whether to solve the full nonlinear version of Eq. 1 (npbe) or a linearized version (lpbe).

- **mol** \(\langle id \rangle\): 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** \(\langle temp \rangle\): 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].

A.2 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.

A.2.1 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:

- **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 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 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 im-
portant to use the same grid setup for all parts of an electrostatic calculation. The \texttt{chgm}
keyword controls the interpolation scheme used for charge distributions and includes the
following types of discretization schemes:

- \texttt{sp10}: 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.

- \texttt{sp12}: Cubic B-spline discretization as described by Im et al. \cite{97}. The charge is
mapped onto the nearest- and next-nearest-neighbor grid points. Resulting potentials
are somewhat less sensitive (than \texttt{sp10}) to grid spacing, length, and position; however,
this discretization can often require smaller grid spacings for accurate representation
of charge positions.

- \texttt{sp14}: Quintic B-spline discretization as described by Schnieders et al. \cite{98}. Similar to
\texttt{sp12}, 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 \texttt{srfm}
keyword:

- \texttt{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 \texttt{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 (\texttt{srad}), this choice of coefficient corresponds to the traditional
definition used for PB calculations. When the solvent radius is set to zero, this corre-
sponds 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 \texttt{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.

- \texttt{smol}: The dielectric and ion-accessibility coefficients are defined as for \texttt{mol} (see above).
However, they are then “smoothed” by a 9-point harmonic averaging to somewhat
reduce sensitivity to the grid setup \cite{99}.

- \texttt{sp12}: The dielectric and ion-accessibility coefficients are defined by a cubic-spline sur-
face \cite{97}. The width of the dielectric interface is controlled by the \texttt{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 reparam-
eterization of the force field \cite{100}.
• sp14: The dielectric and ion-accessibility coefficients are defined by a 7th-order polynomial. This surface definition has characteristics similar to sp12, but provides higher order continuity necessary for stable force calculations with atomic multipole force fields (up to quadrupole) [98].

A.2.2 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.

A.3 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 [54–57, 101]; this section focuses on the configuration and use of these methods in APBS.
A.3.1 Geometric flow calculation configuration

Users invoke the geometric flow solver in APBS by including the \texttt{geoflow-auto} keyword in the \texttt{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 A.2.1. Three additional parameters are needed for geometric flow calculations to specify how nonpolar solvation is linked to the polar implicit solvent models:

- \texttt{gamma} \langle \text{tension} \rangle: 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$^{-2}$.

- \texttt{press} \langle \text{pressure} \rangle: 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$^{-3}$.

- \texttt{bconc} \langle \text{concentration} \rangle: Specify the bulk concentration of solvent in Å$^{-3}$. The bulk density of water, 0.0334 Å$^{-3}$, is recommended.

- \texttt{vdwdisp} \langle \text{bool} \rangle: Indicate whether van der Waals interactions should be included in the geometric flow calculation through \langle \text{bool} \rangle (1 = include, 0 = exclude). If these interactions are included, then a force field with van der Waals terms must be included through a \texttt{READ} statement in the APBS input file.

A.4 Boundary element method implementation

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

A.4.1 Boundary element method background

This section provides additional background on the TABI-PB boundary element solver [59], introduced in Section 3.3. 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 as well as example electrostatic potential output.

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

\begin{equation}
\frac{1}{2} (1 + \varepsilon) \phi(x) = \int_\Gamma \left[ K_1(x,y) \frac{\partial \phi(y)}{\partial \nu} + K_2(x,y) \phi(y) \right] dS_y + S_1(x), \ x \in \Gamma,
\end{equation}

\begin{equation}
\frac{1}{2} \left( 1 + \frac{1}{\varepsilon} \right) \frac{\partial \phi(x)}{\partial \nu} = \int_\Gamma \left[ K_3(x,y) \frac{\partial \phi(y)}{\partial \nu} + K_4(x,y) \phi(y) \right] dS_y + S_2(x), \ x \in \Gamma
\end{equation}
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 300K. Red and blue surface colors correspond to negative and positive electrostatic surface potentials, respectively.
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.

$$
K_1(x, y) = G_0(x, y) - G_\kappa(x, y),
$$

$$
K_2(x, y) = \varepsilon \frac{\partial G_\kappa(x, y)}{\partial \nu_y} - \frac{\partial G_0(x, y)}{\partial \nu_y},
$$

$$
K_3(x, y) = \varepsilon \frac{\partial G_0(x, y)}{\partial \nu_x} - \frac{1}{\varepsilon} \frac{\partial G_\kappa(x, y)}{\partial \nu_x},
$$

$$
K_4(x, y) = \varepsilon \frac{\partial^2 G_\kappa(x, y)}{\partial \nu_x \partial \nu_y} - \frac{1}{\varepsilon} \frac{\partial^2 G_0(x, y)}{\partial \nu_x \partial \nu_y},
$$

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

$$
G_0(x, y) = \frac{1}{4\pi |x - y|},
$$

$$
G_\kappa(x, y) = \frac{e^{-\kappa|x - y|}}{4\pi |x - y|}.
$$

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

$$
\frac{\partial G(x, y)}{\partial \nu_y} = \sum_{n=1}^{3} \nu_n(y) \partial_{y_n} G(x, y),
$$

$$
\frac{\partial G(x, y)}{\partial \nu_x} = -\sum_{n=1}^{3} \nu_n(y) \partial_{x_n} G(x, y),
$$

$$
\frac{\partial^2 G(x, y)}{\partial \nu_x \partial \nu_y} = -\sum_{m=1}^{3} \sum_{n=1}^{3} \nu_m(x) \nu_n(y) \partial_{x_m} \partial_{y_n} G(x, y)
$$

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

$$
S_1(x) = \frac{1}{\varepsilon_m} \sum_{k=1}^{N_c} q_k G_0(x, y_k),
$$

$$
S_2(x) = \frac{1}{\varepsilon_m} \sum_{k=1}^{N_c} q_k \frac{\partial G_0(x, y_k)}{\partial \nu_x},
$$

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

Given a surface triangularization with $N$ elements – where $x_i$ and $A_i$ are the centroid
and area, respectively, of the $i$th triangle – the integral equations are discretized as:

$$\frac{1}{2} \left(1 + \varepsilon\right) \phi(x_i) = \sum_{j=1 \atop j \neq i}^{N} \left[ K_1(x_i, x_j) \frac{\partial \phi(x_j)}{\partial \nu} + K_2(x_i, x_j) \phi(x_j) \right] A_j + S_1(x_i),$$

$$\frac{1}{2} \left(1 + \frac{1}{\varepsilon}\right) \frac{\partial \phi(x_i)}{\partial \nu} = \sum_{j=1 \atop j \neq i}^{N} \left[ K_3(x_i, x_j) \frac{\partial \phi(x_j)}{\partial \nu} + K_4(x_i, x_j) \phi(x_j) \right] A_j + S_2(x_i). \quad (7)$$

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:

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

where $q_k$ is the charge on the $k$th atom of the solute molecule, and $x_k$ 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 $O(N^2)$ to $O(N \log N)$ 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.

**A.4.2 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** \( ⟨\text{order}⟩ \): An integer indicating the order of the Taylor expansion for determining treecode coefficients. Higher values of \( ⟨\text{order}⟩ \) will result in a more accurate but more expensive calculations. A typical choice for this parameter is 3.

- **tree_n0** \( ⟨\text{number}⟩ \): 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** \( ⟨\text{criterion}⟩ \): Multipole acceptance criterion specifies the distance ratio at which the Taylor expansion is used. In general, a higher value of \( ⟨\text{criterion}⟩ \) 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** \( ⟨\text{flag}⟩ \): 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** \( ⟨\text{flag}⟩ \): 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, 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.

### A.5 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.

#### A.5.1 Analytical method (PB-AM) background

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

\[
A = \Gamma \cdot (\Delta \cdot T \cdot A + E),
\]

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, \( \Gamma \) is a dielectric boundary-crossing operator, \( \Delta \) is a cavity polarization operator, and \( T \) is an operator that transforms the multipole expansion from the global (lab) coordinates to a local coordinate frame. The unknown \( A \) 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.

\[
\Omega^{(i)} = \frac{1}{\epsilon_s} \left\langle \sum_{j \neq i}^N T \cdot A^{(j)}, A^{(i)} \right\rangle
\]
where $\epsilon_s$ is the dielectric constant of the solvent and $(M, N)$ denotes the inner product. When energy is computed, forces follow as:

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

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.

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

where $a H_{n,m}^{(i)} = \sum_{j=1}^{M_i} \alpha_j^{(i)} \gamma_n^{(i)} q_j^{(i)} (p_j^{(i)})^n Y_{n,m}(\varphi_j^{(i)}, \varphi_j^{(i)})$, $\alpha = x, y, z$, is a coefficient vector for each of the charges in the molecule, $M_i$ is the number of charges in molecule $i$, $q_j^{(i)}$ is the magnitude of the $j$th charge, and $p_j^{(i)} = [\rho_j^{(i)}, \rho_j^{(i)}, \varphi_j^{(i)}]$ is its position in spherical coordinates. For more details on the PB-AM derivation, see Lotan and Head-Gordon [29].

### A.5.2 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 ($\phi_r$) of the system at any point $r$ is governed by the linearized form of the PB equation:

$$-\nabla \cdot \epsilon \nabla \phi + \kappa^2 \phi = \rho, \quad (13)$$

where $\kappa$ is the inverse Debye length. Eq. 13 is a linearization of Eq. 1 for $\beta q_1 \phi \ll 1$. 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:

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

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

$$\phi_{\text{in}}^{(i)}(r) = \sum_{\alpha=1}^{N_{\text{in}}^{(I)}} \frac{1}{|r - r_{\alpha}^{(I)}|} \frac{1}{4\pi} \int_{d^{(I)}} f^{(I)}(r') dr'$$

where $N_{\text{in}}^{(I)}$ is the number of charges in molecule $I$, $q_\alpha$ is the magnitude of the $\alpha$th charge, $r_{\alpha}^{(I)} = [\rho_{\alpha}^{(I)}, \theta_{\alpha}^{(I)}, \phi_{\alpha}^{(I)}]$ is its position in spherical coordinates, and $f(r)$ is a reactive surface

21
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:

\[
F^{(I,k)}_{n,m} = \frac{1}{4\pi} \int_{d\Omega^{(I,k)}} f^{(I,k)}(r') \left( \frac{a^{(I,k)}}{r'} \right)^{n+1} Y_{n,m}^{(I,k)}(\theta', \phi') dr'
\]
\[
H^{(I,k)}_{n,m} = \frac{1}{4\pi} \int_{d\Omega^{(I,k)}} h^{(I,k)}(r') \left( \frac{r'}{a^{(I,k)}} \right)^{n} i_n(Kr') Y_{n,m}^{(I,k)}(\theta', \phi') dr'
\]

where \( Y_{n,m} \) is the spherical harmonics, \( Y_{n,m}^{\ast} \) 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:

\[
F^{(I,k)} = \langle I_{E,n,m}, WF^{(I,k)} \rangle \\
H^{(I,k)} = \langle I_{E,n,m}, WH^{(I,k)} \rangle
\]

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

\[
I_{E,n,m}^{(I,k)} = \frac{1}{4\pi} \int_{\phi_E} \int_{\theta_E} Y_{l,s}^{(I,k)}(\theta', \phi') Y_{n,m}^{(I,k)}(\theta', \phi') \sin \theta' d\theta' d\phi'
\]

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, \( \langle \Omega^{(i)} \rangle \), 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, \( \langle H^{(I,k)} \rangle \), and the multipole expansions of the other molecules in the system, \( \langle LHN^{(I,k)} \rangle \) as follows:

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

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 \( I \):

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

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 \( r^{(I,k)}_P \) with respect to the center of mass and the force on that charge \( f_P \). We can re-express \( r^{(I,k)}_P \) 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 \( P \) \( (r^{(I,k)}_P) \). The total torque on molecule \( I \) is then given by Eq. 23.

\[
\tau^{(I)} = \sum_k c^{(I,k)} \times f_{I,k} + \sum_k \sum_{P \in k} r^{(I,k)}_P \times f_P
\]
where $f_{I,k}$ is given in Eq. 22 and

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

(24)

where

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

(25)

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

(26)

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

A.5.3 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 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, if 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


“Chimera website”. URL: https://www.cgl.ucsf.edu/chimera/.


