Improvements to the APBS biomolecular solvation software suite

Elizabeth Jurrus[⋆], Dave Engel[⋆], Keith Star[⋆], Kyle Monson[⋆], Juan Brandi[⋆], Dave Engel[⋆], Lisa E. Felberg[†], David Brookes[†], Leighton Wilson[‡], Jiahua Chen[§], Karina Liles[⋆], Minju Chun[⋆], Peter Li[⋆], Todd Dolinsky[‖], Robert Konecny[△], Jens Erik Nielsen[¶], Theresa Head-Gordon[†], Weihua Geng[§], Robert Krasny[‡], Marilyn Gunner[◇], Guo-Wei Wei[▽], Michael J. Holst[△], J. Andrew McCammon[△], and Nathan A. Baker[○]

[⋆]Pacific Northwest National Laboratory
[†]University of California Berkeley
[‡]University of Michigan
[§]Southern Methodist University
[‖]FoodLogiQ
[¶]Novozymes
[▽]Michigan State University
[△]University of California San Diego
[◇]City University of New York
[○]To whom correspondence should be addressed. Advanced Computing, Mathematics, and Data Division; Pacific Northwest National Laboratory; Richland, WA 99352, USA. Division of Applied Mathematics; Brown University; Providence, RI 02912, USA.
Email: nathan.baker@pnnl.gov

June 23, 2017

Abstract

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

1 Introduction

This manuscript provides an overview of the new capabilities in the APBS software and its associated tools[1] since their release in 2001 [1**?** –3]. The manuscript is not intended to be a review of solvation models; readers interested in this area should refer to several previous reviews [4–9]. Robust models of electrostatic interactions are important for understanding intermolecular interactions and the effects of solvation on biomolecular processes. Such models can be loosely grouped into two categories: explicit and implicit. Explicit methods treat the solvent in atomic detail whereas implicit methods generally replace the explicit solvent with a dielectric continuum. Explicit methods tend to offer the highest levels of detail; however, they generally require extensive sampling to converge properties of interest. Implicit methods, on the other hand, trade detail and some accuracy for efficiently by eliminating the sampling and equilibration associated with explicit solvent models. The APBS software implements implicit models of solvation.

A variety of implicit solvent methods have been developed for the description of polar solvation [6], with Poisson-Boltzmann (PB) methods being among the most popular. The PB equation [10–12] provides a global solution for the electrostatic potential (ϕ) within and around a biomolecule by solving the partial differential equation

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

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

The potential 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 [10, 13]. Despite these limitations, PB methods are still very important for biomolecular structural analysis, modeling, and simulation.

Several software packages have been developed to analyze the solvation properties of small and macro-molecules, solving the Poisson-Boltzmann equations to evaluate energies, potentials, and other solvation properties. The most significant (based on user base and citations) of these include CHARMM [14], AMBER [15], DelPhi [16], Jaguar [17], Zap [18], MIBPB [19], and APBS [**?**]. This article describes the new functionality added to APBS, and its associated software, since its release in 2001. At the time of writing this article, APBS had broad adoption with 27,000 registered users worldwide.

[1] APBS-PDB2PQR website: `http://www.poissonboltzmann.org`.

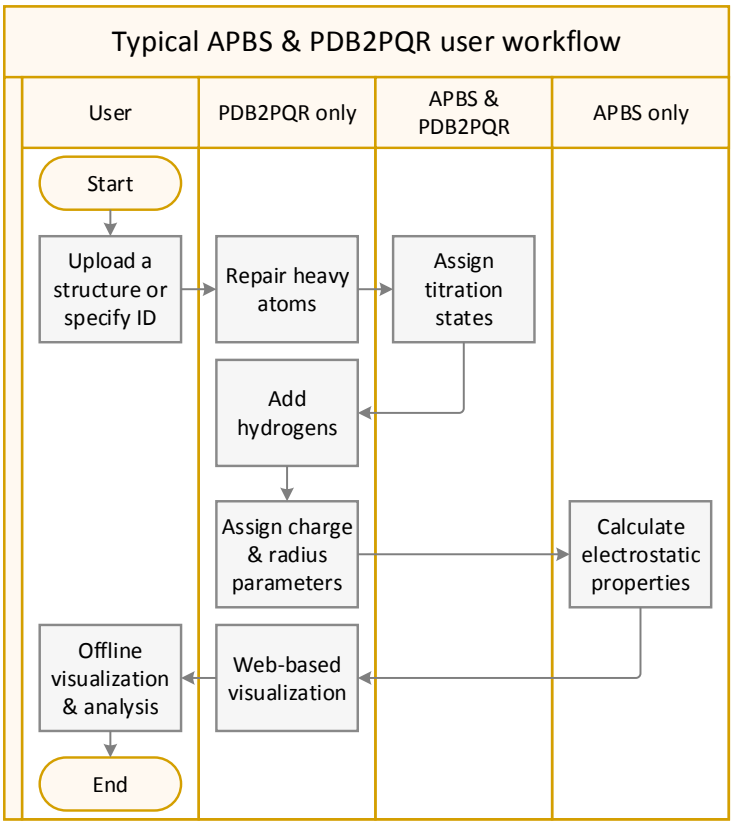


Figure 1: Workflow for biomolecular electrostatics calculations using the APBS-PDB2PQR software suite.

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 [6]. Popular implicit solvent force fields supported by the APBS tool suite include ⋆⋆ **INCLUDE force fields and radii** ⋆⋆. APBS incorporates this information into calculations via the "PQR" format. PQR is a file format of unknown origins used by several software packages such as MEAD [**?**] and AutoDock [**?**]. The PQR file simply replaces the temperature and occupancy columns of a PDB flat file [**?**] with the per-atom charge (Q) and radius (R). There are much more elegant ways to implement the PQR functionality through more modern extensible file formats such as mmCIF [**?**] or PDB-XML; however, the simple PDB format is still one of the most widely used formats and, therefore, continued use of the PQR format supports broad compatibility across biomolecular modeling tools and workflows.

The PDB2PQR software is part of the APBS suite that was developed to assist with the conversion of PDB files to PQR format [20, 21]. PDB2PQR began its existence as a sed [**?**] script and evolved to work with APBS and support the functionality shown in Figure 1. In particular, PDB2PQR automatically sets up, executes, and optimizes the structure for Poisson-Boltzmann electrostatics calculations, outputting a PQR file that can be used with APBS or other modeling software. Some of the key steps in PDB2PQR are described below.

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

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

- *Empirical methods.* PDB2PQR provides an empirical model (PROPKA [22]) which uses a heuristic method to compute pK_a perturbations due to desolvation, hydrogen bonding, and charge-charge interactions is included in PDB2PQR. The empirical PROPKA method has surprising accuracy for fast evaluation of protein pK_a values [23].
- *Implicit solvent methods.* PDB2PQR also contains two methods for using implicit solvent (Poisson-Boltzmann) models for predicting residue titration states. The first method uses Metropolis Monte Carlo to calculating titration curves and pK_a values (PDB2PKA); however, sampling issues can be a major problem with Monte Carlo methods when searching over the $\mathcal{O}\left(2^N\right)$ titration states of N titratable residues. The second method is a new polynomial-time algorithm for the optimization of discrete states in macromolecular systems [24]. This method transforms interaction energies between titratable groups into a graphical flow network. The polynomial-time $\mathcal{O}\left(N^4\right)$ behavior makes it possible to rigorously evaluate titration states for much larger proteins than Monte Carlo methods.

Adding missing hydrogens. The majority of PDB entries do not include hydrogen positions. Given a titration state assignment, PDB2PQR uses Monte Carlo sampling to position hydrogen atoms and optimize the global hydrogen-bonding network in the structure [25]. 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 ⋆⋆ **list force fields** ⋆⋆.

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. As described in the remainder of this section, APBS provides several different methods for solving the Poisson-Boltzmann and related equations.

3.1 General electrostatics calculation usage

⋆⋆ **FINISH** ⋆⋆ ⋆⋆ **STOPPED HERE** ⋆⋆ APBS was designed as a standalone system and also to facilitate use with other programs. This section outlines execution methods and some of the

programs with which APBS is known to work with. All of these methods are described in depth on the APBS-PDB2PQR website.[2]

3.2 Command-line execution

Most users will likely interact with APBS and PDB2PQR through web servers. However, it is also possible to install local versions. These local installations give a command line version of the software that can be customized through a variety of extensions. Compiled from source, a local version can also provide a web server.

3.3 Execution through other programs

APBS can also be executed from other programs. As an example, the following programs have had capabilities to run APBS:

- PDB2PQR
- iAPBS - Robert Konecny (McCammon Group) has developed iAPBS, an interface between APBS and the simulation packages AMBER, CHARMM, and NAMD. The iAPBS package provides a C/C++ and Fortran interface to APBS through a single function call. The interface code can be compiled as a library (libiapbs.a) which can be linked with a Fortran or C/C++ application thus making most of APBS functionality available from within any C/C++/Fortran code. [3]
- TINKER - APBS 1.3 is available with TINKER; TINKER is a molecular modeling software package for molecular mechanics and dynamics, and includes some special features for bioploymers.[4]

3.4 Web-based execution

As discussed earlier, web-based executation of APBS is accomplished using the Opal Toolkit produced by the NBCR. This toolkit allows for the computing load for processor intensive scientific applications to be shifted to a 3rd party and/or generic computing grid. This can be tremendously advantageous in situations where a large amount of computing power is not locally available, but is required, for the task at hand. In particular, many users have discovered that their local computational resources are insufficient for certain types of APBS calculations on large systems or at extremely high accuracy. This client removes this resource limitation by allowing users to run on clusters at NBCR. Recent developmental versions of APBS add optional support for the off-loading of APBS calculations to an Opal service. The APBS Opal support is in the form of a Python script `ApbsClient.py` and is installed by default when following the installation procedure outlined elsewhere. The script has been tested on Python 2.5; newer/older versions of Python may or may be functional. As mentioned above, the basic invocation is the same as the main binary. The only difference is the executable, which is called `ApbsClient.py`, rather than `apbs`. This client should be installed by default when APBS is installed.

[2] www.poissonboltzmann.org
[3] www.mccammon.ucsd.edu/iapbs
[4] www.dasherwustl.edu/tinker

3.5 Finite difference and finite element solvers

The original version of APBS was based on two key libraries from the Holst research group. FEtk (`http://www.fetk.org/`) is a general-purpose multi-level adaptive finite element library [**?**]. Adaptive finite element methods can resolve extremely fine features of a complex system (like biomolecules) while solving the associated equations over large problem domain. For example, FEtk has been used to solve electrostatic and diffusion equations over six orders of magnitude in length scale [**?**]. [**?**] to solve the Poisson-Boltzmann equation. The finite difference PMG solver trades speed and efficiency for the high-accuracy and high-detail solutions of the finite element FEtk library. However, many APBS users need only a relatively coarse-grained solution of 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.

Finite difference use in APBS $\star\star$ **Finish this section. Include keywords and typical setting values.** $\star\star$

Finite elements use in APBS $\star\star$ **Finish this section and/or move all of these sections to an appendix. Include keywords and typical setting values.** $\star\star$

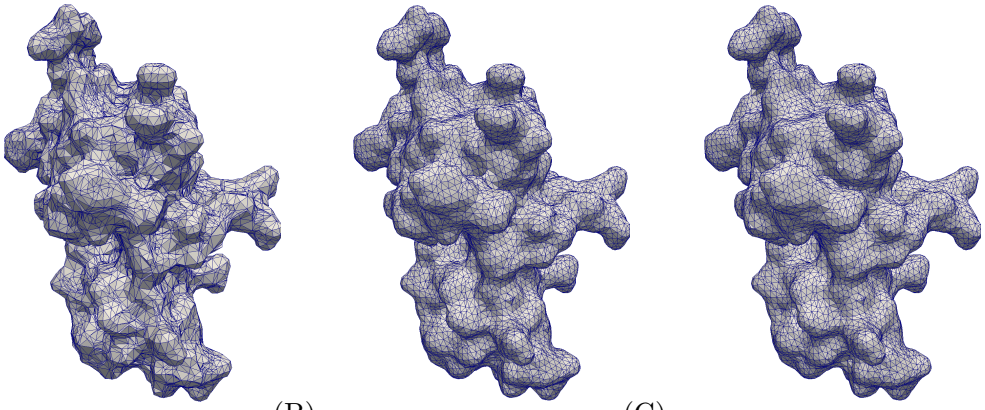
3.6 Geometric flow

Several recent papers have described our work on a geometric flow formulation of Poisson-based implicit solvent models [26**?**]. The geometric flow approach couples the polar and nonpolar components of the implicit solvent model with two primary benefits. First, this coupling eliminates the need for an *ad hoc* geometric definition for the solute-solvent boundary. In particular, the solute-solvent interface is optimized as part of the geometric flow calculation. Second, the optimization of this boundary ensures self-consistent calculation of polar and nonpolar energetic contributions (using the same surface definitions, etc.), thereby reducing confusion and the likelihood of user error. Additional information about the geometric flow implementation in APBS is provided in Appendix A. This equation is solved in APBS using a finite difference method.

Geometric flow use in APBS $\star\star$ **FINISH** $\star\star$ $\star\star$ **STOPPED HERE** $\star\star$

3.7 Boundary element methods

Boundary element methods offer the ability to focus numerical effort on a much smaller region of the problem domain: the interface between the molecule and the solvent. APBS now includes a treecode accelerated boundary integral PB solver (TABI-PB) developed by Geng and Krasny to solve the linearized PB equation [27]. In this method, two coupled integral equations defined on the solute-solvent boundary define a mathematical relationship between the electrostatic surface potential and its normal derivative with a set of integral kernels consisting of Coulomb and screened Coulomb potentials with their normal derivatives. The boundary element method requires a surface triangulation, generated by a program such as MSMS [28] or NanoShaper [29], on which to discretize the integral equations. The resulting linear system is then solved with GMRES iteration [30]. 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 $\mathcal{O}(N \log N)$ [31, 32]. Additional information about the boundary element method implementation in APBS is provided in Appendix B.



(A) (B) (C)

Figure 2: PDB ID 1a63 surfaces generated with (A) 20228 triangles via MSMS [28], (B) 20744 triangles via NanoShaper SES, and (C) 21084 triangles via NanoShaper Skin [29].

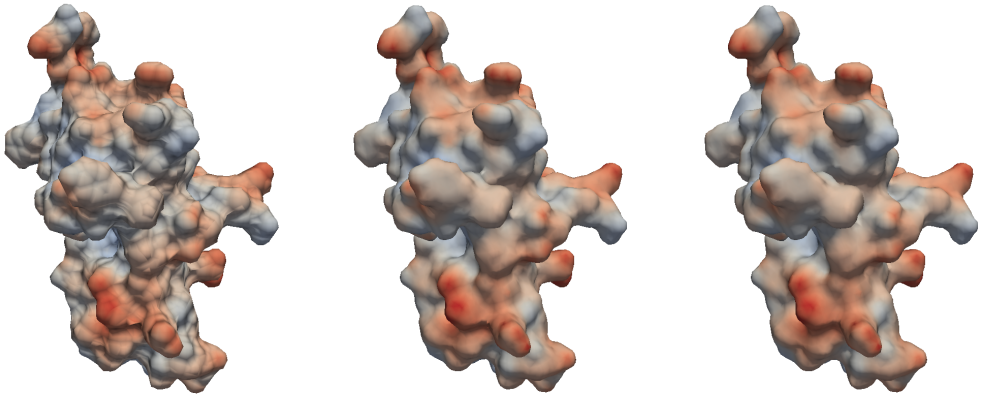
Boundary element use in APBS APBS users can invoke TABI-PB with the `bem-manual` flag in the ELEC section of the input file. Major options include:

- `tree_order` ⟨ *order* ⟩: An integer indicating the order of the Taylor expansion for determining treecode coefficients. Higher values of ⟨ *order* ⟩ will result in a more accurate – but more expensive – calculations. A typical choice for this parameter is 3.
- `tree_n0` ⟨ *number* ⟩: The maximum number of particles allowable in a leaf of the treecode (clusters in the last level of the tree). A typical choice for this parameter is 500.
- `mac` ⟨ *criterion* ⟩: Multipole acceptance criterion specifies the distance ratio at which the Taylor expansion is used. In general, a higher value of ⟨ *criterion* ⟩ will result in a more accurate but more expensive computation; while a lower value causes more direct summations and forces the particle-cluster interaction to descend to a finer cluster level. A typical choice for this parameter is 0.8.
- `mesh` ⟨ *flag* ⟩: The software used to mesh the molecular surface; 0 = MSMS, 1 = NanoShaper's SES implementation, and 2 = NanoShaper's Skin implementation. See Figure 2 for an example of surface meshes.
- `outdata` ⟨ *flag* ⟩: Type of output data file generated; 0 = APBS OpenDX format [**?**] and 1 = ParaView format [**?**].

Additional information about parameter settings is provided via the APBS website (`http://www.poissonboltzmann.org`). 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 3 by using MSMS and NanoShaper.

3.8 Analytical and semi-analytical methods

Numerical solution methods tend to be computationally intensive which has led to the adoption analytical approaches for solvation calculations such as generalized Born [**?**] and the approaches developed by Head-Gordon implemented in APBS. The Poisson-Boltzmann Analytical Method



(A) (B) (C)

Figure 3: PDB ID 1a63 visualization of surface potential for (A) MSMS, max = 5.06 kcal/mol, min = −5.15 kcal/mol; (B) NanoShaper SES, max = 4.44 kcal/mol, min = −7.95 kcal/mol; (C) NanoShaper Skin, max = 4.25 kcal/mol, min = −20.03 kcal/mol. ⋆⋆ **Check units on these potential values.** ⋆⋆

(PB-AM), was developed by Lotan and Head-Gordon in 2006 [33]. 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.

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 [34]. 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.

Because it is fully analytical, PB-AM can be used for model validation as well as for representing systems that are relatively spherical in nature, such as globular proteins and colloids. PB-SAM, on the other hand has a much more detailed representation of the molecular surface and can therefore be used for many systems that other APBS (numerical) methods are currently used for. Through APBS, both programs can be used to compute the electrostatic potential at any point in space, report energies, forces, and torques of a system of macromolecules, and simulate a system using a BD scheme [35]. Additional details about these methods are presented in Appendix C

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

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

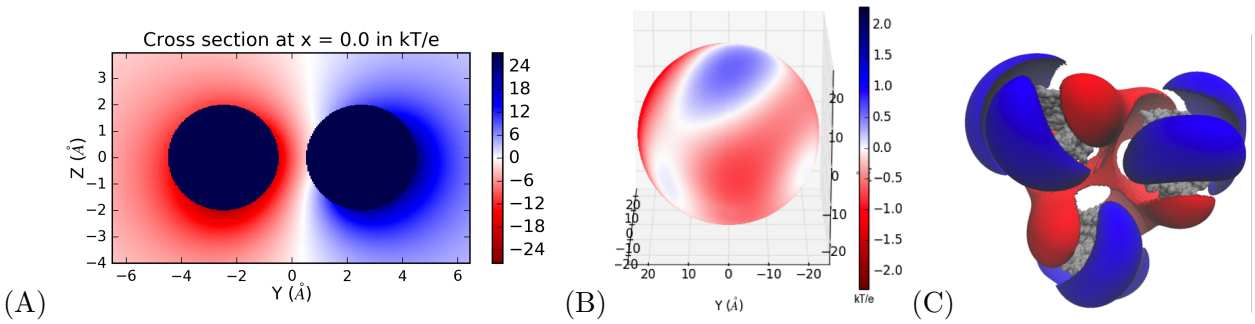


Figure 4: APBS PB-AM electrostatic potential results for (A) 2D cross section of two collections of point charges (each with net charge ±6) ⋆⋆ **Why is the zero-point of the isocontour off center?** ⋆⋆, (B) potential on the spherical coarse-grain surface of a barstar molecule (k_BT/e), (C) VMD [36] of the transmembrane trimer Omp32 porin (molecules are given in grey and isosurfaces are drawn at 1.0 (blue) and -1.0 (red) k_BT/e). All calculations were performed at 0.0 M ionic strength, pH 7, protein dielectric 2, and solvent dielectric 78.

- `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* ⟩: ⋆⋆ **FINISH** ⋆⋆
  - `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 1 for everything else. ⋆⋆ **I don't understand this. Why have this parameter at all? I would have expected the parameter to be the molecule index.** ⋆⋆
- `xyz` ⟨ *idx* ⟩ ⟨ *fpath* ⟩: Molecule index ⟨ *idx* ⟩ and file path ⟨ *fpath* ⟩ for the molecule starting configurations. ⋆⋆ **Why not just use the existing APBS `READ` statements?** ⋆⋆ A starting configuration is needed for each molecule and each trajectory. Therefore, 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. ⋆⋆ **Where are those keywords described?** ⋆⋆ Additional information about parameter settings is provided via the APBS website (`http://www.poissonboltzmann.org`). Examples of the electrostatic potentials produced from PB-AM and PB-SAM are shown in Figures 4 and 5, respectively.

4 Visualizing APBS results

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

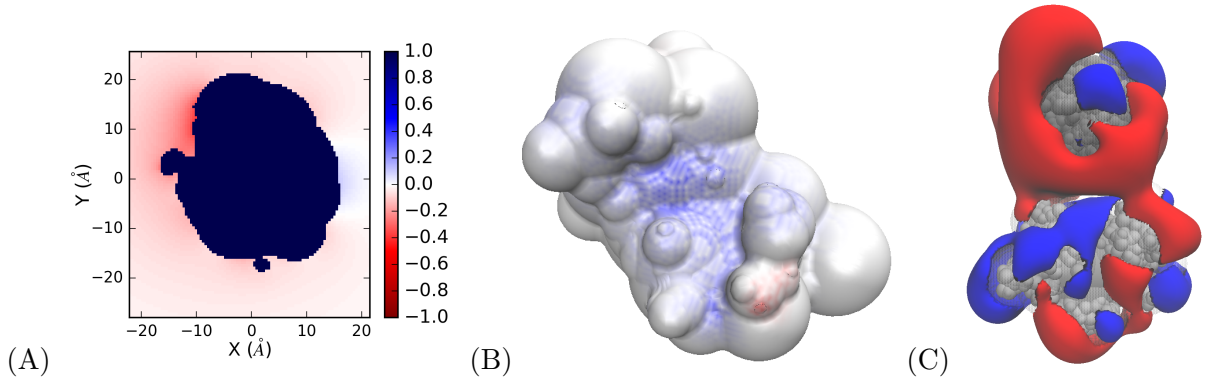


Figure 5: APBS PB-SAM electrostatic potential results for (A) potential in a 2D plane surrounding the barstar molecule (k_BT/e), (B) potential on the coarse-grain surface of the barnase molecule (blue region is the location of barstar association, max = 1.0 k_BT/e, min = -1.0 k_BT/e), (C) VMD [36] visualization of the ESP around the association of barnase and barstar (molecules are given in grey and isosurfaces are drawn at 1.0 (blue) and -1.0 (red) k_BT/e). All calculations were performed at 0.0 M ionic strength, pH 7, protein dielectric 2, and solvent dielectric 78.

as a graphical interface for setting up the calculation. Several of these software packages are thick clients that run from users' computers, including PyMOL[5] [**?**], VMD[6] [36], PMV[7] [**?**], and Chimera[8] [37]. We have also worked with the developers of Jmol[9] [38] and 3Dmol[10] [39]. to provide web-based setup and visualization of APBS-PDBD2PQR calculations and related workflows. APBS integration with Jmol has been described previously [**?**]. 3Dmol is a molecular viewer that offers the performance of a desktop application and convenience of a web-based viewer which broadens accessibility for all users. In order to integrate 3Dmol, enhancements were need for our tools, including extending our output file formats and creating a customized user interface. Data from the APBS output file is used to generate surfaces, apply color schemes, and display different molecular styles such as cartoons and spheres. An example of the 3Dmol interface is shown in Figure 6. Examples of 3Dmol visualization options are shown in Figure 7.

4.1 Other applications

Besides visualization and the processes described in Section **??**, there are a number of other applications where APBS can be used. For example, during the past four years, the APBS-PDB2PQR software has been used in the post-simulation energetic analyses of molecular dynamics trajectories [40], understanding protein-nanoparticle interactions [41**? ?**], understanding nucleic acid-ion interactions [42**?**], biomolecular docking [43] and ligand binding [**?**], developing new coarse-grained protein models [44], setting up membrane protein simulations [45], etc. APBS also plays a key role in PIPSA for protein surface electrostatics analysis [46] and SDA for simulation of protein-protein interactions through Brownian dynamics [47]. 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

[5]PyMOL website: `http://www.pymol.org`.
[6]VMD website: `http://www.ks.uiuc.edu/Research/vmd/`.
[7]PMV website: `http://mgltools.scripps.edu/`.
[8]Chimera website: `https://www.cgl.ucsf.edu/chimera/`.
[9]Jmol website: `http://jmol.sourceforge.net/`.
[10]3Dmol website: `http://3dmol.csb.pitt.edu/`.

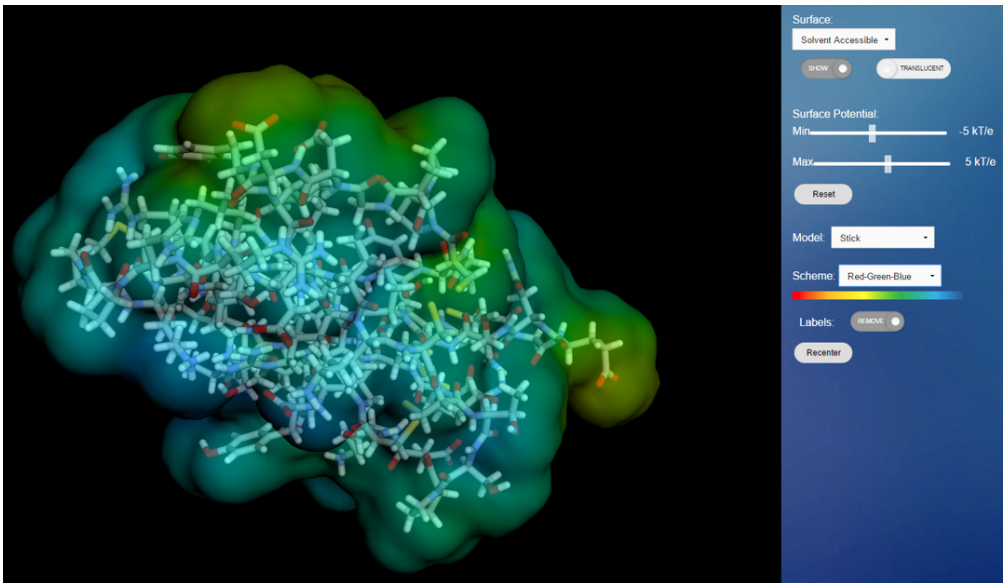


Figure 6: 3Dmol interface displaying a rendering of fasciculin-2 (1FAS) protein with translucent, solvent accessible surface using a stick model and red-green-blue color scheme.

(or analyzing) discrete molecular or continuum diffusion simulations with APBS in both of these areas [40**? ? ? ? ? ? ?**].

⋆⋆ **STOPPED HERE** ⋆⋆

To allow users web-based access to PDB2PQR and APBS, we utilize a web server interface hosted at the National Biomedical Computation Resource (NBCR)[11] Center which uses the Opal2 toolkit to interface with the NBCR cluster. Opal2 handles the scheduling and execution of jobs and stores the results for later retrieval [48]. The NBCR server is designed to facilitate the setup and execution of continuum electrostatics calculation from PDB data, particularly by non-experts. The web service is driven by a modular Python-based collection of routines, which provides considerable flexibility to the software and permits non-interactive, high-throughput usage.

5 Using the software

5.1 Extending APBS capabilities

There are several challenges when working with large code bases. Careful planning must go into the design, especially if the program is to be extended. Ideally, one would like the ability to link new code, which extends functionality, without having to change the existing base code. However, this is rarely the case. In the best scenario, a few line changes may be required. On the other hand, if proper care has not been taken, large modifications may be unavoidable.

To help with modularity, several programming paradigms have been introduced over the years. Object Oriented Programming (OOP), the paradigm used in the design of APBS, focuses on principal objects. Classes are used to define objects with given characteristics and functionality. This gives rise to a natural hierarchical structure where subclasses share any number of characteristics with the main class. APBS works with atoms, that form residues, that form molecules, etc, which

[11] www.nbrc.ucsd.edu

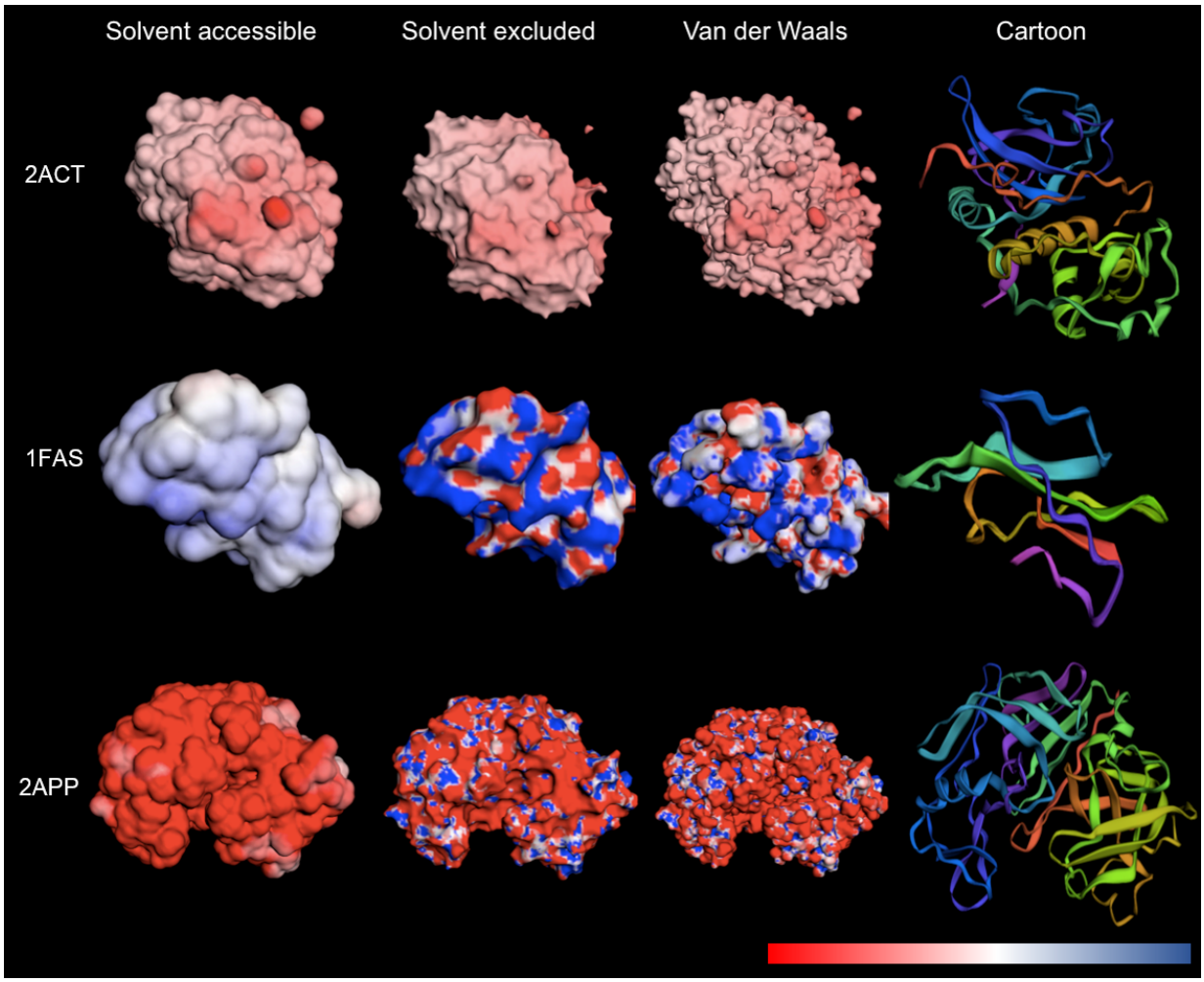


Figure 7: Renderings of three different proteins using renderings of actinidin (2ACT) (top), fasciculin-2 (1FAS) (center), and pepsin-penicillium (2APP) (bottom). To demonstrate the different visualization options. From left to right: solvent-accessible surface, solvent-excluded surface, van der Waals surface, and cartoon models are shown all using red-white-blue color scheme (excluding cartoon model).

naturally lends itself to the OOP paradigm. The C programming language is known for fast performance but does not natively supports OOP. To overcome this drawback, APBS was written using a subset of ANSI C called Clean OO C style by Mike Holst [49].

The main APBS file takes inputs from the command line and initializes some of the variables necessary for I/O and passing data to the Routines manager. The Routines manager, in turn, creates atom objects and invokes whichever solver is to be used. The advantages of using Clean OO C becomes clear when adding new solver methods to APBS. Because of APBS' OOP style, all one must do is make Routines aware of the new method. As long as the new methods adhere to Clean OO C calling conventions, the passing and return of data is seamless and most other methods (e.g. reading and writing to DX files) can be used. New methods which are external or do not adhere to Clean OO C standards can be managed by writing auxiliary methods for translating data from one to the other. These auxiliary methods are given in a header and source file. The external methods are linked at compilation time.

6 Future directions

The focus of APBS and its accompanying software is the modeling of solvation and electrostatic properties of macromolecules in biomedical research. As such, continued research and development of APBS is focused on modeling different possible application scenarios that may arise in the modeling of the solvation and electrostatic properties.

APBS contains many different models and methods, which gives it a unique capability as a software package to be utilized in many different research applications. Methods that have been examined for future implementation into APBS and PDB2PQR include: advanced methods (gPC) for quantifying the influence of uncertainty on simple solvation-related properties (such as solvent-accessible surface area), ensemble averaging methods for titration state predictions, and virtual reality visualization of electrostatic potentials and related properties.

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. Imagine all the duplication of effort in building UIs, file loaders, parsers, output writers, etc. All that could be eliminated.

Our goal is to have a well designed, well tested and well documented industry grade framework that will make it possible for all of these various bimolecular software packages to work together. To leverage each other's capabilities, to exceed the sum of the individual software package capabilities.

7 Acknowledgments

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

References

[1] M Holst, N Baker, and F Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. algorithms and examples. *J. Comput. Chem.*, 21(15):1319–1342, November 2000.

[2] N Baker, M Holst, and F Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. refinement at solvent-accessible surfaces in biomolecular systems. *J. Comput. Chem.*, 21(15):1343–1352, November 2000.

[3] N A Baker, D. Sept, M J Holst, and J A McCammon. The adaptive multilevel finite element solution of the Poisson-Boltzmann equation on massively parallel computers. *IBM J. Res. Dev.*, 45(3-4):427–438, May-Jul 2001.

[4] M. E. Davis and J A McCammon. Electrostatics in biomolecular structure and dynamics. *Chemical Reviews*, 90(3):509–521, 1990.

[5] M F Perutz. Electrostatic effects in proteins. *Science*, 201(4362):1187–91, Sep 1978.

[6] P Ren, J Chun, D G Thomas, M J Schnieders, M Marucho, J Zhang, and N A Baker. Biomolecular electrostatics and solvation: a computational perspective. *Q Rev Biophys*, 45(4):427–91, Nov 2012.

[7] K A Sharp and B Honig. Electrostatic interactions in macromolecules: theory and applications. *Annu Rev Biophys Biophys Chem*, 19:301–32, 1990.

[8] B Roux and T Simonson. Implicit solvent models. *Biophys Chem*, 78(1-2):1–20, Apr 1999.

[9] A Warshel, P K Sharma, M Kato, and W W Parson. Modeling electrostatic effects in proteins. *Biochim Biophys Acta*, 1764(11):1647–76, Nov 2006.

[10] M Fixman. The Poisson-Boltzmann equation and its application to polyelectrolytes. *J. Chem. Phys.*, 70:4995–5005, 1979.

[11] P Grochowski and J Trylska. Continuum molecular electrostatics, salt effects, and counterion binding–a review of the Poisson-Boltzmann theory and its modifications. *Biopolymers*, 89(2):93–113, 2008.

[12] G Lamm, K B Lipkowitz, R Larter, T R Cundari, and D B Boyd. The Poisson-Boltzmann equation. *Rev. Comput. Chem.*, 19:147–365, 2003.

[13] R. R. Netz and H. Orland. Beyond Poisson-Boltzmann: Fluctuation effects and correlation functions. *Eur. Phys. J. E*, 1(2-3):203–214, Feb-Mar 2000.

[14] B R Brooks, C L Brooks, 3rd, A D Mackerell, Jr, L Nilsson, R J Petrella, B Roux, Y Won, G Archontis, C Bartels, S Boresch, A Caflisch, L Caves, Q Cui, A R Dinner, M Feig, S Fischer, J Gao, M Hodoscek, W Im, K Kuczera, T Lazaridis, J Ma, V Ovchinnikov, E Paci, R W Pastor, C B Post, J Z Pu, M Schaefer, B Tidor, R M Venable, H L Woodcock, X Wu, W Yang, D M York, and M Karplus. Charmm: the biomolecular simulation program. *J Comput Chem*, 30(10):1545–614, Jul 2009.

[15] D A Case, T E Cheatham, 3rd, T Darden, H Gohlke, R Luo, K M Merz, Jr, A Onufriev, C Simmerling, B Wang, and R J Woods. The AMBER biomolecular simulation programs. *J Comput Chem*, 26(16):1668–88, Dec 2005.

[16] S Sarkar, S Witham, J Zhang, M Zhenirovskyy, W Rocchia, and E Alexov. Delphi web server: A comprehensive online suite for electrostatic calculations of biological macromolecules and their complexes. *Comm. Comp. Phys.*, 13(1):269–284, 2013.

[17] AD Bochevarov, E Harder, TF Hughes, JR Greenwood, DA Braden, DM Philipp, D Rinaldo, MD Halls, J Zhang, and RA Friesner. Jaguar: A high-performance quantum chemistry software program with strengths in life and materials sciences. *International J. of Quantum Chem.*, 113(18):2110–2142, 2013.

[18] JA Grant, BT Pickup, and A Nicholls. A smooth permittivity function for poisson-boltzmann solvation methods. *J. Comp. Chem.*, 22:608–640, 2001.

[19] Z Chen, N A Baker, and G W Wei. Differential geometry based solvation model II: Lagrangian formulation. *J Math Biol*, 63(6):1139–200, Dec 2011.

[20] Todd J Dolinsky, Jens E Nielsen, J Andrew McCammon, and Nathan A Baker. Pdb2pqr: an automated pipeline for the setup of poisson–boltzmann electrostatics calculations. *Nucleic Acids Res.*, 32(suppl 2):W665–W667, 2004.

[21] T J Dolinsky, P Czodrowski, Hui Li, J E Nielsen, Jan H Jen, G Klebe, and N A Baker. PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucleic Acids Res*, 35:W522–5, Jul 2007.

[22] Chresten R. Sondergaard, Mats HM Olsson, Michal Rostkowski, and Jan H. Jensen. Improved treatment of ligands and coupling effects in empirical calculation and rationalization of pka values. *J. Chem. Theory Comp.*, 7(7):2284–2295, 2011.

[23] H. Li, A. D. Robertson, and J. H. Jensen. Very fast empirical prediction and rationalization of protein pka values. *Proteins*, 61:704721, 2005.

[24] Emilie Purvine, Kyle Monson, Elizabeth Jurrus, Keith Starr, and Nathan A. Baker. Energy minimization of discrete protein titration state models using graph theory. *J. Phys. Chem.*, 120:8354–8360, 2016.

[25] J E Nielsen and G Vriend. Optimizing the hydrogen-bond network in Poisson-Boltzmann equation-based pK_a calculations. *Proteins*, 43(4):403–12, Jun 2001.

[26] D G Thomas, S Gaheen, S L Harper, M Fritts, F Klaessig, E Hahn-Dantona, D Paik, S Pan, G A Stafford, E T Freund, J D Klemm, and N A Baker. ISA-TAB-Nano: a specification for sharing nanomaterial research data in spreadsheet-based format. *BMC Biotechnol*, 13:2, 2013.

[27] Weihua Geng and Robert Krasny. A treecode-accelerated boundary integral poisson-boltzmann solver for electrostatics of solvated biomolecules. *J. Comp. Phys.*, 247:62–78, 2013.

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

[29] Sergio Decherchi and Walter Rocchia. A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale. *PLoS ONE*, 8(4):e59744, 2013.

[30] Y. Saad and M. Schultz. Gmres: A generalized minimal residual algorithm for solving non-symmetric linear systems. *J. Sci. Statist. Comput.*, 7:856–869, 1986.

[31] Peijun Li, Hans Johnston, and Robert Krasney. A cartesian treecode for screened coulomb interactions. *J. Comp. Phys.*, 228:3858–3868, 2009.

[32] André Juffer, Eugen Botta, Bert van Keulen, Auke van der Ploeg, and Herman Berendsen. The electric potential of a macromolecule in a solvent: A fundamental approach. *J. Comp. Phys.*, 97(1):144–171, 1991.

[33] Itay Lotan and Teresa Head-Gordon. An analytical electrostatic model for salt screened interactions between multiple proteins. *J. Chem. Theory and Comp.*, 2(3):541–555, 2006. PMID: 26626662.

[34] E-H Yap and T Head-Gordon. New and efficient Poisson-Boltzmann solver for interaction of multiple proteins. *J. Chem. Theory Comput.*, 6(7):2214–2224, July 2010.

[35] Donald Ermak and J. A. McCammon. Brownian dynamics with hydrodynamic interactions. *J. Chem. Phys.*, 69(4):1352–1360, 1978.

[36] W Humphrey, A Dalke, and K Schulten. VMD: visual molecular dynamics. *J Mol Graph*, 14(1):33–8, 27–8, Feb 1996.

[37] E F Pettersen, T D Goddard, C C Huang, G S Couch, D M Greenblatt, E C Meng, and T E Ferrin. Ucsf chimera–a visualization system for exploratory research and analysis. *J Comput Chem*, 25(13):1605–12, Oct 2004.

[38] Angel Herraez. Biomolecules in the computer: Jmol to the rescue. *Biochemistry and Molecular Biology Education*, 34(4):255–261, 2006.

[39] Nicholas Rego and David Koes. 3dmol. js: molecular visualization with webgl. *Bioinformatics*, 31(8):1322–1324, 2015.

[40] RO Dror, HF Green, C Valant, DW Borhani, JR Valcourt, AC Pan, DH Arlow, M Canals, JR Lane, R Rahmani, JB Baell, PM Sexton, A Christopoulos, and DE Shaw. Structural basis for modulation of a g-protein-coupled receptor by allosteric drugs. *Nature*, 503(7475):295–299, 2013.

[41] L Treuel, S Brandholt, P Maffre, S Wiegele, L Shang, and GU Nienhaus. Impact of protein modification on the protein corona on nanoparticles and nanoparticlecell interactions. *ACS Nano*, 8(1):503–513, 2013.

[42] J Lipfert, Sebastian Doniach, R Das, and D Herschlag. Understanding nucleic acid-ion interactions. *Annu Rev Biochem*, 83:813–41, 2014.

[43] V A Roberts, E E Thompson, M E Pique, M S Perez, and L F Ten Eyck. DOT2: Macromolecular docking with improved biophysical models. *J Comput Chem*, 34(20):1743–58, Jul 2013.

[44] E Spiga, D Alemani, M T. Degiacomi, M Cascella, and M Dal Peraro. Electrostatic-consistent coarse-grained potentials for molecular simulations of proteins. *J. Chem. Theory Comput.*, 9(8):3515–3526, August 2013.

[45] Phillip J Stansfeld, Joseph E Goose, M Caffrey, Elisabeth P Carpenter, Joanne L Parker, S Newstead, and Mark SP Sansom. Memprotmd: Automated insertion of membrane protein structures into explicit lipid membranes. *Structure*, 23(7):1350–1361, 2015.

[46] S Richter, A Wenzel, M Stein, R R Gabdoulline, and R C Wade. webPIPSA: a web server for the comparison of protein interaction properties. *Nucleic Acids Res*, 36(Web Server issue):W276–80, Jul 2008.

[47] Michael Martinez, Neil J Bruce, Julia Romanowska, Daria B Kokh, Musa Ozboyaci, Xiaofeng Yu, Mehmet Ali Öztürk, Stefan Richter, and Rebecca C Wade. Sda 7: A modular and parallel implementation of the simulation of diffusional association software. *J Comput Chem*, 36(21):1631–45, Aug 2015.

[48] Sriram Krishnan, Luca Clementi, Jingyuan Ren, Philip Papadopoulos, and Wilfred Li. Design and evaluation of opal2: A toolkit for scientific software as a service. In *Services-I, 2009 World Conference on*, pages 709–716. IEEE, 2009.

[49] Michael Holst. Adaptive numerical treatment of elliptic systems on manifolds. *J. Comp. Math.*, 15:139–191, 2001.

[50] Z Chen, N A Baker, and G W Wei. Differential geometry based solvation model I: Eulerian formulation. *J Comput Phys*, 229(22):8231–8258, Nov 2010.

A Geometric flow implementation

This section contains additional information about the geometric flow equation implementation in APBS introduced in Section 3.6. ⋆⋆ **STOPPED HERE** ⋆⋆ For our modeling, a generalized form of the Poisson equation for computing the electrostatic is used as shown in Eq. 2, where ϵ_m and ϵ_s are the dielectric coefficients of the solute and solvent, respectively, and ρ_m is the charge distribution of the fixed solute molecule. For this model, the solutions for the electrostatic potential (ϕ) and the characteristic function (S) are obtained by minimizing the free energy functional. The dielectric function $\epsilon(S)$ takes on the value ϵ_m in the solute region (S=1) and the value ϵ_s in the solvent region (S=0).

$$-\nabla \cdot (\epsilon(S)\nabla\phi) = S\rho_m \tag{2}$$

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

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

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

The solution to this non-linear PDE can be solved by utilizing the parabolic PDE shown in Eq. 4. This equation is known as the generalized geometric flow equation and it is coupled with the Poisson equation through the characteristic function S [50].

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

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

$$V = -p + \rho_0 U^{att} - \rho_m \phi + \frac{\epsilon_m}{2} \mid \nabla\phi \mid^2 - \frac{\epsilon_s}{2} \mid \nabla\phi \mid^2 . \tag{5}$$

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

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

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

⋆⋆ **STOPPED HERE** ⋆⋆

B Boundary element method implementation

This appendix provides additional information about the boundary element method introduced in Section 3.7.

⋆⋆ **STOPPEDHERE** ⋆⋆

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

The TABI-PB method utilizes a well-posed boundary integral PB formulation to ensure rapid convergence. In addition, a fast treecode algorithm for the screened Coulomb potential is applied to speed up the matrix-vector products in each GMRES iteration [32]. The TABI-PB solver serves as an effective alternative to the multi-grid finite difference solver when the surface potential accuracy or system memory capacity becomes a concern.

The coupled second kind integral equations employed by TABI-PB for calculating the surface potential ϕ_1 and its normal derivative [32], are as follows:

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

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

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

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

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

The normal derivatives of the potential kernels G are:

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

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

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

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

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

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

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

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

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

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

C Analytical and semi-analytical method implementations

This appendix provides additional information about the analytical and semi-analytic methods introduced in Section 3.8. ⋆⋆ **STOPPEDHERE** ⋆⋆ The solution to this model is represented as a system of linear equations:

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

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

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

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:

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

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

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

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

Poisson Boltzmann Semi-Analytical Method The derivation details of PB-SAM have been reported previously [**? ?**], 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:

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

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

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

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

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

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

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

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

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

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

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

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

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

Using the above formalism, physical properties of the system, such as interaction energy, forces and torques may be computed. The interaction energy of each molecule, $(\Omega^{(i)})$, is the product of the molecule's total charge distribution (from fixed and polarization charges) with the potential due to external sources. This is computed as the inner product between the molecule's multipole expansion, $(H^{(I,k)})$, and the multipole expansions of the other molecules in the system, $(LHN^{(I,k)})$ as follows:

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

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

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

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

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

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

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

where

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

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

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