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[◇], 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
[△]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 22, 2017

Abstract

The Adaptive Poisson-Boltzmann Solver (APBS) software was developed to solve the equations of continuum electrostatics for large biomolecular assemblages [1]. The understanding of electrostatic interactions is essential for the study of biomolecular processes and impacts many chemical, biological, and biomedical applications. APBS addresses three key technology challenges for understanding solvation and electrostatics in biomedical applications: accurate and efficient models for biomolecular solvation and electrostatics, robust and scalable software for applying those theories to biomolecular systems, and mechanisms for sharing and analyzing biomolecular electrostatics data in the scientific community. To address new research applications and advancing computational capabilities, we have continually updated APBS and its suite of accompanying software since its release in 2001. In this paper, we discuss the models and capabilities that have recently been implemented within the APBS software package including: a Poisson-Boltzmann analytical and a semi-analytical solver, an optimized boundary element solver, a geometry-based geometric flow solvation model, a graph theory based algorithm for determining pK_a values, and an improved web-based visualization tool for viewing electrostatics.

1 Introduction

This manuscript provides an overview of the new capabilities in the APBS software and its associated tools since their release in 2001. The manuscript is not intended to be a review of solvation models; readers interested in this area should refer to several previous reviews [2–7]. 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 [4], with Poisson-Boltzmann (PB) methods being among the most popular. The PB equation [8–10] 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 [8, 11]. Despite these limitations, PB methods are still very important for biomolecular structural analysis, modeling, and simulation.

2 Workflow support for biomolecular electrostatic calculations

Electrostatics calculations begin with the 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 [4]. 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

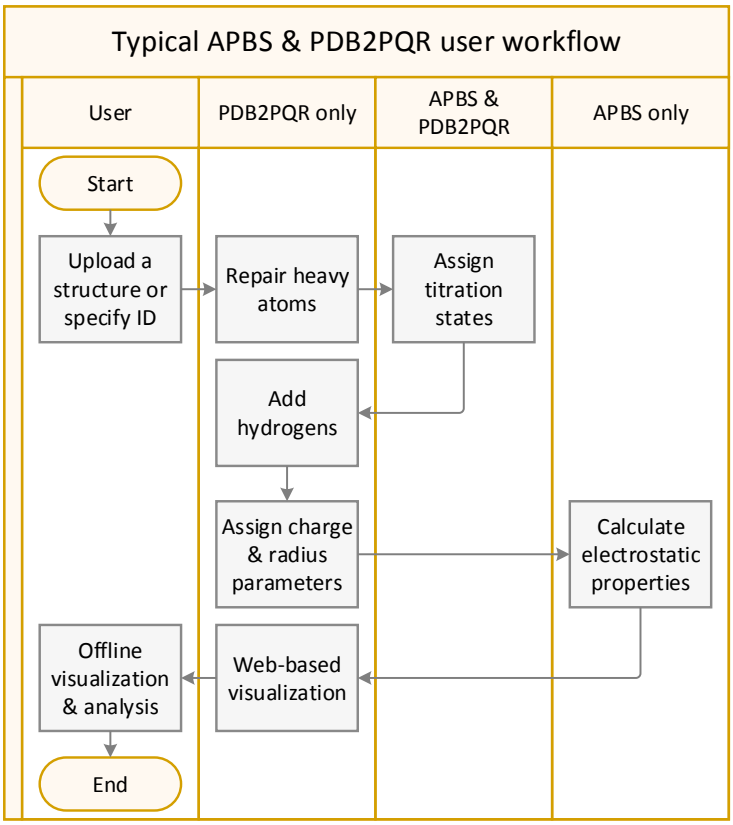


Figure 1: Workflow for biomolecular electrostatics calculations using the APBS software suite.

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 compability 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 [12, 13]. 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 [14]) 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 [15].

- *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 [16]. 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 [17]. 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 equation

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 [18], AMBER [19], DelPhi [20], Jaguar [21], Zap [22], MIBPB [23], and APBS [**?**]. The CHARMM and AMBER packages are multipurpose software designed for a wide range of modeling tasks while DelPhi and MIBPB are the most similar to APBS. Like APBS, DelPhi has a significant and established user base; MIBPB is a newer package. At the time of writing this manuscript, APBS had over 27,000 registered users.

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 equation.

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 (`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.

⋆⋆ **STOPPED HERE** ⋆⋆

3.2 Boundary element methods

In order to address some of the issues with the numerical methods in APBS, a treecode accelerated boundary integral PB solver (TABI-PB) developed by Geng and Krasny [24] has recently been implement. This method provides a boundary element approach to the linearized PB equation. In this method, two coupled integral equations defined on the solute-solvent boundary define a mathematical relationship between the electrostatic surface potential ϕ_1 and its normal derivative with a set of integral kernels consisting of Coulomb and screened Coulomb potentials with their normal derivatives. The method requires a surface triangulation, generated by a program such as MSMS [31] or NanoShaper [32], on which to discretize the integral equations. The resulting linear system is then solved with GMRES iteration [33]. To reduce the computational cost of this dense system from $O(N^2)$ to $O(N \log N)$, a Cartesian particle-cluster treecode is employed to compute matrix-vector products.

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 [34] is applied to speed up the matrix-vector products in each GMRES iteration [35]. 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 [35], are as follows:

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

where $\varepsilon = \varepsilon_m/\varepsilon_s$, the ratio of the dielectric constant in the solute region and the dielectric constant in the solvent region. The integral kernels K_1, K_2, K_3, K_4 are defined in Eq. 21.

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

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

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

The normal derivatives of the potential kernels G are:

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

for the three spatial components n of the normal direction. Additionally, the source terms S_1 and S_2 in Eq. 20 are:

$$S_1(\mathbf{x}) = \frac{1}{\varepsilon_m}\sum_{k=1}^{N_c} q_k G_0(\mathbf{x},\mathbf{y}_k), \quad S_2(\mathbf{x}) = \frac{1}{\varepsilon_m}\sum_{k=1}^{N_c} q_k \frac{\partial G_0(\mathbf{x},\mathbf{y}_k)}{\partial \nu_{\mathbf{x}}} \tag{6}$$

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

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

$$\begin{aligned}
\frac{1}{2}(1+\varepsilon)\,\phi_1(\mathbf{x}_i) &= \sum_{\substack{j=1\\ j\neq i}}^{N} \left[K_1(\mathbf{x}_i,\mathbf{x}_j)\,\frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_2(\mathbf{x}_i,\mathbf{x}_j)\,\phi_1(\mathbf{x}_j) \right] A_j + S_1(\mathbf{x}_i), \\
\frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial \phi_1(\mathbf{x}_i)}{\partial \nu} &= \sum_{\substack{j=1\\ j\neq i}}^{N} \left[K_3(\mathbf{x}_i,\mathbf{x}_j)\,\frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_4(\mathbf{x}_i,\mathbf{x}_j)\,\phi_1(\mathbf{x}_j) \right] A_J + S_2(\mathbf{x}_I)
\end{aligned} \tag{7}$$

The omission of the $j = i$ term in the summation avoids the singularity of the kernels at that point. Note that the right hand sides of these equations consist of sums of products of kernels and the surface potential or its normal derivative. These are the analogues to the N-body potential in the treecode, with the surface potential or its normal derivatives playing the role of the charges. In the discretized form, the total electrostatic energy of solvation is given by Eq. 26.

$$E_{\text{sol}} = \frac{1}{2}\sum_{k=1}^{N_c} q_k \sum_{j=1}^{N} \left[K_1(\mathbf{x}_k,\mathbf{x}_j)\,\frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu} + K_2(\mathbf{x}_k,\mathbf{x}_j)\,\phi_1(\mathbf{x}_j) \right] A_j \tag{8}$$

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

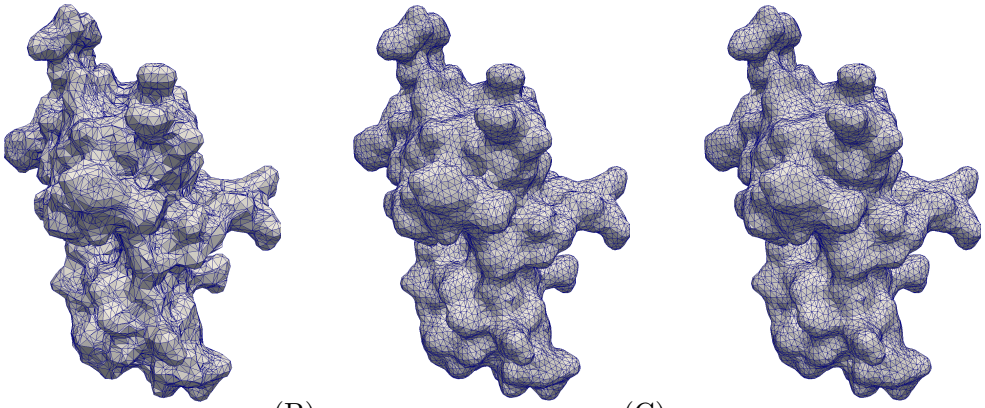
Several unique features of the boundary integral formulation are worth mentioning. Because the integral equations are defined on the molecular boundary, the singular charges are handled analytically and do not introduce the same issues present in grid-based schemes. The integral equations also rigorously 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.

Usage in APBS The APBS user can invoke TABI-PB with the `bem-manual` flag in the ELEC section of the input file. TABI-PB will produce output including the potential and normal derivative of potential for every element and vertex of the triangularization, as well as the electrostatic solvation energy. The user can additionally specify the order of the Taylor expansions used in the far field potential contribution approximations, the maximum number of particles allowable in a leaf (clusters in the last level of the tree), and the multipole acceptance criterion, which determines the distance at which potential contributions from particles in a cluster are evaluated directly or through the Taylor expansion (see Table 2).

In general, a lower multipole acceptance criterion (θ) will result in a more accurate but more expensive computation; a lower θ causes more direct summations and forces the particle-cluster interaction to descend to a finer cluster level. Similarly, a higher Taylor expansion order will also result in a more accurate but more expensive computation. A typical first-pass choice for these parameters would be a multipole acceptance criterion of 0.8, a Taylor expansion order of 3, and a maximum leaf size of 500. More investigation is needed to determine optimal parameter choices.

The user can also choose to specify whether TABI-PB uses MSMS, the SES implementation in NanoShaper, or the Skin surface implementation of NanoShaper to generate a surface triangularization, as shown in Figure 4. NanoShaper, an open-source program for surface triangularization, was recently added to TABI-PB, and preliminary results suggest superior performance to MSMS. Examples of surface potential on 1a63 are given in Figure 5 by using MSMS and NanoShaper.



(A) (B) (C)

Figure 2: PDB ID 1a63 surfaces generated with (A) MSMS, number of triangles=20228; (B) NanoShaper SES, number of triangles=20744; (C) NanoShaper Skin, number of triangles=21084.

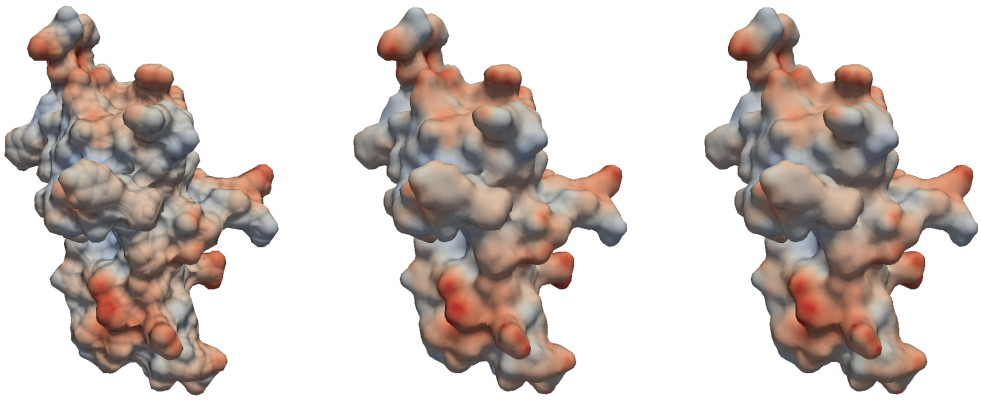
3.3 Analytical and semi-analytical methods

As mentioned above, numerical solution methods tend to be computationally intensive. Therefore, analytical methods have been included in the APBS system, which is beneficial for a rapid and accurate solution of the PB equation.

Poisson-Boltzmann Analytical Method The Poisson-Boltzmann Analytical Method (PB-AM), was developed by Lotan and Head-Gordon in 2006 [25]. It presents a fully analytical solu-

Table 1: Keywords used in TABI-PB. For more details and examples of implementation, please see the APBS website.

Keyword	Parameters	Description
tree_order	<order>	Integer indicating the order of the Taylor expansion for determining treecode coefficients.
tree_n0	<max_number>	Maximum number of particles allowable in a treecode leaf.
mac	<theta>	Multipole acceptance criterion, specifies the distance ratio at which the



(A) (B) (C)

Figure 3: 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.

tion to the linearized PB equation for multiple macromolecules in a screened environment. Each molecule in the system is coarse-grained and represented as a single low dielectric spherical cavity. This domain enables the use of the multipole expansion to represent charge-charge interactions and higher order cavity polarization effects. The solution to this model is represented as a system of linear equations as follows:

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

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. Once the unknown $\mathbf{A}$'s are solved using the Gauss-Sieidel iterative method, many physical properties, such as interaction energies, forces and torques can be computed. The interaction energy for molecule i, $(\Omega^{(i)})$ is given in Eq. 3.

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

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

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

By definition, the torque on a charge in the molecule is the cross product of its position relative to the center of mass of the molecule with the force it experiences. The total torque on the molecule is a linear combination of the torque on all charges of the molecule, as illustrated in Eq. 5.

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

where ${}^{\alpha}H^{(i)}_{n,m} = \sum_{j=1}^{M_i} \alpha_j^{(i)} \gamma_n^{(i)} q_j^{(i)} (\rho_j^{(i)})^n Y_{n,m}(\vartheta_j^{(i)}, \varphi_j^{(i)})$, $\alpha = x, y, z$, is a coefficient vector for each of the charges in the molecule, M_i is the number of charges in molecule i, $q_j^{(i)}$ is the magnitude of the j^{th} charge, and $p_j^{(i)} = \left[\rho_j^{(i)}, \vartheta_j^{(i)}, \varphi_j^{(i)} \right]$ is its position in spherical coordinates. For more details on the PB-AM derivation, see Lotan, Head-Gordon [25].

Poisson Boltzmann Semi-Analytical Method An extension of the fully analytical solution, the Poisson Boltzmann Semi-Analytical method (PB-SAM) incorporates the use of boundary integrals into its formalism to represent a complex molecular domain as a collection of overlapping low dielectric spherical cavities [26]. It presents a semi-analytical solution to the linearized PB equation for multiple macromolecules in a screened environment. The 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. Additionally, both methods have been implemented in a Brownian dynamics (BD) scheme [27] and can perform dynamics with full mutual polarization, a novel feature for APBS.

The derivation details of PB-SAM have been reported previously [26, 28], 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:

$$\nabla \cdot [\epsilon(\mathbf{r})\nabla\phi(\mathbf{r})] - \epsilon(\mathbf{r})\kappa^2\phi(\mathbf{r}) = 4\pi\rho(\mathbf{r}) \tag{13}$$

where κ is the inverse Debye length. For the case of spherical cavities, we can solve Eq. 6 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:

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

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

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

where $N_C^{(I)}$ is the number of charges in molecule I, q_α is the magnitude of the α-th charge, $r^{(I)}_\alpha = \left[\rho^{(I)}_\alpha, \theta^{(I)}_\alpha, \phi^{(I)}_\alpha\right]$ 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:

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

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

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

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

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

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

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

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

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

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

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

As in the analytical 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. 16.

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

where $f_{I,k}$ is given in Eq. 15 and

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

where

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

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

where $\alpha = x, y, z$. For the derivation of the PB-SAM solver please see the previous publications [26, 28] .

Usage 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.[1] From 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 [27]. For an example of the key simulation commands used in a run of PB-AM or PB-SAM, see Table 1. The `term` keyword, included in Table 1, allows the user to indicate conditions for the termination of each BD trajectory. This may include the contact of two molecules or the diffusion of a specified molecule beyond a specified point in space, like the conditions defined by Northrup, Allison and McCammon for the NAM simulation protocol [29]. In addition to parameters specific to each type of run (BD, electrostatic potential printing and physical calculations), the user can modify the coarseness of the molecular description by modifying the coarse-graining parameter `tolsp`, which indicates how far beyond the solvent-excluded surface the coarse-grain molecular boundary is allowed to extend. The larger the value of `tolsp`, the coarser the description of the molecule, e.g. less coarse grain spheres used to represent it. This will make computational evaluation of the system more rapid, but less accurate. Conversely, the smaller the `tolsp` value chosen, the more coarse grain spheres needed to represent the system, which means a more accurate but also more costly representation of the system. The `tolsp` parameter is generally in the range of 1 to 5 Å. The commands (keywords) not included in Table 1 are used to specify system conditions, such as temperature and salt concentration.

Table 2: Major keywords used in PB-AM and PB-SAM. For more keywords, details and examples of implementation, please see the APBS website.

Keyword	Parameters	Description
runname	<name>	Desired name to be used for outputs of each run.
pbc	<boxlength>	Size of periodic box.
runtype dynamics		Perform a Brownian Dynamics run.
ntraj	<trajct>	Number of trajectories to run.
term	<type> <val> <mols>	Attributes of a termination condition: <type> can be time, x<=, y<=, z<=, r<=, (or the >= equivalents), <val> is the value of the condition, <mols> is the molecular index that this condition applies to (time requires 0, all else, 1).
xyz	<idx> <fpath>	Molecule index and file name (path) for the xyz file. A starting configuration is needed for each trajectory for all the molecules, so there should be <trajct> xyz lines for each molecule.
PB-SAM Keywords		
msms		Will run the MSMS program for the coarse-graining process.
tolsp	<tol>	For the coarse-graining process, a tolerance in (Å) to indicate how far from the molecule surface the coarse grain surface is allowed to extend.

Examples of electrostatic potential (ESP) visualization results from PB-AM are given in Figure 2. In Figure 2 (A), 2-D cross sections of electrostatic potential surrounding two spheres of opposite charge are shown. The 3-D surface potential visualization given in Figure 2 (B) describes the potential on the surface of a coarse-grain barstar molecule (a ribonuclease inhibitor). Figure 2 (C) is an example of three-dimensional isosurface visualization in VMD [30]. The molecule system

[1] www.poissonboltzmann.org/docs/apbs-overview

which was analyzed was the periplasmic face of the Omp32 Porin trimer, a transmembrane protein. The periplasmic face (channel exit) has a positive surface (blue) at the channel exit which may further enhance anion transport through the channel.

ESP visualization results are given in Figure 3 for the PB-SAM model. They include the visualizations of barnase and barstar molecules, a ribonuclease and its inhibitor, whose association is driven by electrostatic interactions. Figure 3 (A) is a two-dimensional cross section visualization of a barnase molecule. The molecule interior is given in dark blue and the color bar indicates the spatial variation of the potential with the molecule centered at the origin. Figure 3 (B) is a visualization of the potential of a barstar molecule at its association interface. The positive potential distribution at the surface interacts favorably with the negative potential distribution on the barnase surface. Figure 3 (C) visualizes the electrostatic potential surrounding the barnase-barstar interaction complex.

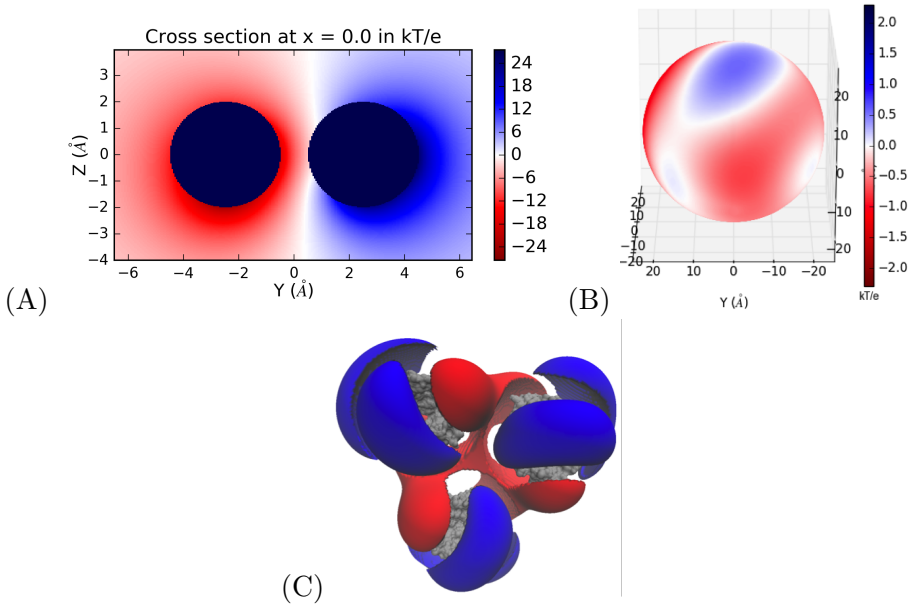
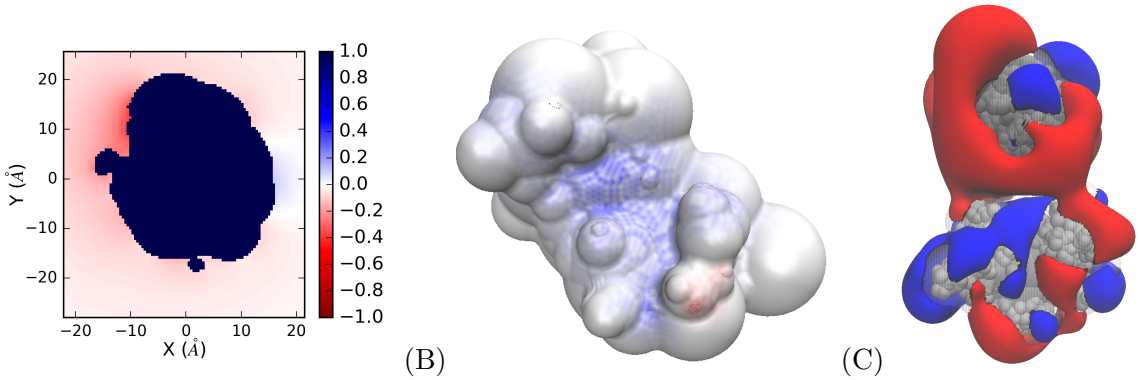


Figure 4: ESP results derived from APBS PB-AM (0.0M salt, 7 pH, dielectric 2 (protein), and 78 (solution)): (A) 2D cross section of two collections of point charges (each with net charge ±6), (B) potential on the spherical coarse-grain surface of a barstar molecule (k_BT/e), (C) VMD visualization of the transmembrane trimer Porin (molecules are given in grey and isosurfaces are drawn at 1.0 (blue) and -1.0 (red) k_BT/e).

foo These methods use fast iterative techniques that solve a sparse linear system. However, issues have been identified which effect the results and performance of the models. The following issues were identified by Geng and Krasny [24] when solving the PB equation utilizing numerical methods:

- the memory requirements for a three-dimensional grid can be prohibitively large,
- the geometric details of the molecular surface may be obscured on a regular grid,
- the singular atomic charges are smoothed by interpolation onto the grid,
- the interface conditions may not be rigorously enforced on the molecular surface., and
- the far-field boundary condition is often satisfied only approximately on a truncated domain.

To address these issues, new methods have been added to APBS to improve the flexibility for more diverse applications, address issues identified with grid-based methods, increase performance, and to expand the analysis capabilities of APBS. These methods are described in the following sections. For each method, we briefly discuss the algorithms that were implemented, how a user may use them, and show example results for several of the methods.



(A) (B) (C)

Figure 5: ESP results derived from APBS PB-SAM (0.0M salt, 7 pH, dielectric 2 (protein), and 78 (solution)): (A) potential in a 2D plane surrounding a 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 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).

3.3.1 Geometric Flow

To increase the accuracy of our implicit solvent modeling, we have recently implemented a differential geometry based geometric flow solvation model. In this model, polar and nonpolar free energies are coupled through a characteristic function. This function describes a smooth dielectric interface profile across the solvent-solute boundary [36].

For our modeling, a generalized form of the Poisson equation for computing the electrostatic is used as shown in Eq. 27, 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).

$$-\nabla\cdot(\epsilon(S)\nabla\phi) = S\rho_m \tag{27}$$

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. 28.

$$-\nabla\cdot\left(\gamma\frac{\nabla S}{\parallel\nabla S\parallel}\right)+p-\rho_0U^{att}+\rho_m\phi-\frac{1}{2}\epsilon_m\mid\nabla\phi\mid^2+\frac{1}{2}\epsilon_s\mid\nabla\phi\mid^2=0 \tag{28}$$

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. 29. This equation is known as the generalized geometric flow equation and it is coupled with the Poisson equation through the characteristic function S [37].

$$\frac{\partial S}{\partial t}=\gamma\parallel\nabla S\parallel\left[\nabla\cdot\left(\frac{\nabla S}{\parallel\nabla S\parallel}\right)+\frac{V}{\gamma}\right], \tag{29}$$

where V is known as the generalized flow potential as described in Eq. 30.

$$V=-p+\rho_0U^{att}-\rho_m\phi+\frac{\epsilon_m}{2}\mid\nabla\phi\mid^2-\frac{\epsilon_s}{2}\mid\nabla\phi\mid^2. \tag{30}$$

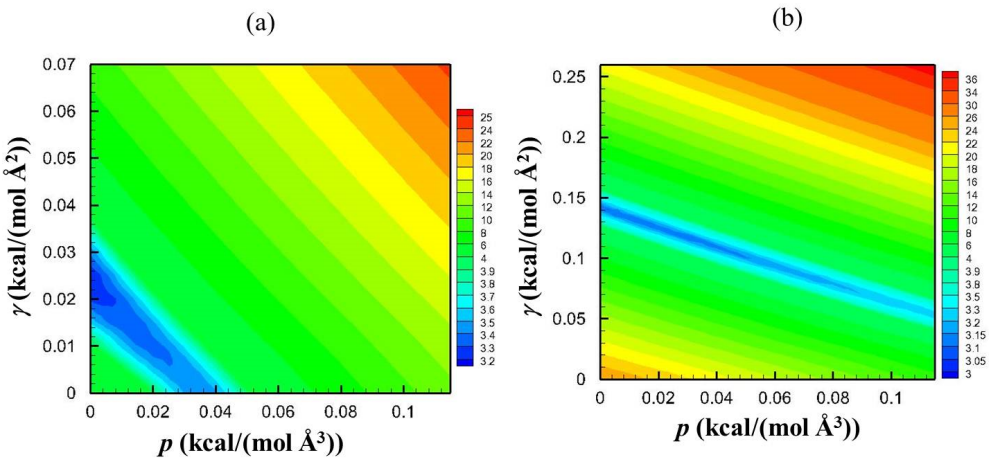


Figure 6: Contour plot of the error between experimental and calculated solvation free energies based on OPLS-AA calculations: (a) without vdW; (b) with vdW.

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. [36], 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).

Usage in APBS The differential geometry-based geometric flow solvation model is used to describe a smooth dielectric interface profile across the solvent-solute boundary in a thermodynamically self-consistent fashion. The main parameters of the model are the solute/solvent dielectric coefficients, solvent pressure on the solute, microscopic surface tension, solvent density, and molecular force-field parameters.

This model is accessed under the ELEC section of the input file. Results using the OPLS-AA force-field calculations, with and without van der Waals dispersion (vdW), are shown in Figure 6 and were discussed by Thomas and friends [36].

3.4 Visualizing and using the results

APBS was designed to facilitate use with other programs. This section outlines some of the programs with which APBS is known to work with.

3.4.1 Thick-client visualization

Graphical User Interfaces

PyMOL - molecular visualization and animation package which provides an interface to APBS.

The APBS plugin to PyMOL permits isocontour and surface map visualization of APBS results.[2]

VMD - molecular visualization and animation package which provides an interface to APBS. It permits visualization of APBS results as isocontours, electric field lines, or on biomolecular surfaces. VMD also has graphical plugin to setup APBS calculations and execute them either locally or remotely via BioCoRE [30].

PMV - Python-based molecular visualization package which provides an interface to APBS. It not only permits visualization of APBS results but it also integrates setup and executation of APBS calculations.[3]

Chimera - molecular visualization package which provides an interface to APBS.The APBS plugin integrated in Chimera allows the user to run a protein through both PDB2PQR and APBS to optimize the protein and view the electrostatic potential [38].

Visualization Software

Electrostatic potentials are commonly visualized in the context of biomolecular structure to better understand functional aspects of biological systems. Note that the graphical user interfaces discussed above can also be used to visualize APBS output. The following molecular graphics software can display potentials and other data output from APBS:

Off-Target Pipeline - implements APBS to estimate ligand binding energies and compare electrostatic potential distributions within binding cavities [**?**].

Dino3D - molecular graphics program which can read UHBD-formatted electrostatic data. APBS can write multigrid results in UHBD format and therefore can be used with Dino3D.[4]

OpenDX - general data visualization package which can read APBS output using the scripts provided in tools/visualization/opendx. However, as there is no straightforward way to visualize the potential in the context of the atomic structure, OpenDX should not be a first choice for APBS visualization.[5]

3.4.2 Web-based visualization

Computational chemists and biologists rely on molecular visualization tools to view model proteins and predict unobserved protein simulations [39]. These tools allow scientists to efficiently visualize and explore proteins. Thus allowing for a better understanding of relationships and functions of proteins.

Visualization tools for evaluating simulation results are critical for the discovery of new insights. Molecular viewers are available as stand-alone desktop programs or web-based applications. Desktop users are often burdened with obtaining software licenses or downloads when using desktop viewers such as VMD and web-based users are troubled with java security settings, permissions, or browser plug-ins when using web-based viewers such as Jmol [40]. While most viewers offer the same features such as rendering styles and coloring methods, lack of accessibility and poor performance can become a hindrance for scientists as they conduct research.

To help with these issues, we have integrated 3Dmol [41] into the PDB2PQR software and APBS pipeline. 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. In the end, we have enabled a new light weight capability for visualization of complex electrostatic properties.

[2] www.pymol.org

[3] www.mgltools.scripps.edu/packages/pmv

[4] www.dino3d.org

[5] www.opendx.org

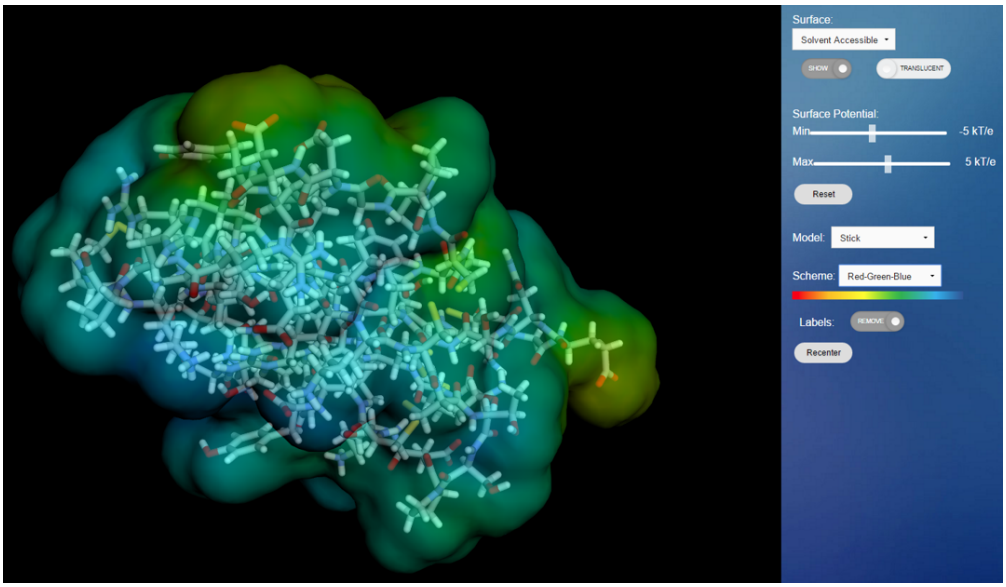


Figure 7: 3Dmol interface displaying a rendering of FASCICULIN (1FAS) protein with translucent, solvent accessible surface using a stick model and red-green-blue color scheme.

To allow users web-based access to PDB2PQR and APBS, we utilize a web server interface hosted at the National Biomedical Computation Resource (NBCR)[6] 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 [42]. 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.

We use the new capabilities in 3Dmol to render common molecular data representations for visualization and analysis. 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 7. The graphical user interface (GUI) has three surface options; solvent accessible, solvent excluded, and van der Waals.

Using the GUI, the user can show or hide the surface or display it as translucent or opaque. The user can apply a minimum and maximum surface potential value. The default minimum and maximum isovalues are -5 kT/e and 5 kT/e, respectfully. The user can select a line, stick, cross, sphere, or cartoon model and apply a red-white-blue or red-green-blue color scheme to the surface. Additionally, 3Dmol renders proteins using different styles and color schemes, detailed in Table 3. Figure 8 shows the visual representation of some selected features.

3Dmol is an effective, interactive molecular viewer that is accessible across web browsers and operating systems. With 3Dmol, users can make selections using a simple GUI such as select different surfaces, apply various models, and choose from different color schemes. The features available in 3Dmol are comparable to Jmol[7], a molecular visualization tool written in Java; however, there are also differences between the two tools. While Jmol offers trace, backbone, cartoon, and ball and stick models, 3Dmol offers cartoon, cross, line, sphere, and stick models. Unlike Jmol,

[6] www.nbrc.ucsd.edu
[7] www.jmol.sourceforge.net

Table 3: Selected 3Dmol features.

Feature	Description
Solvent accessible surface	describes the surface area of a biomolecule that is accessible to a solvent
Red-White-Blue	describes the color scheme, red to white to blue, for charges
Stick model	draws bonds as capped cylinders
Cross model	draws atoms as crossed lines (i.e., stars)
Sphere model	draws atoms as spheres

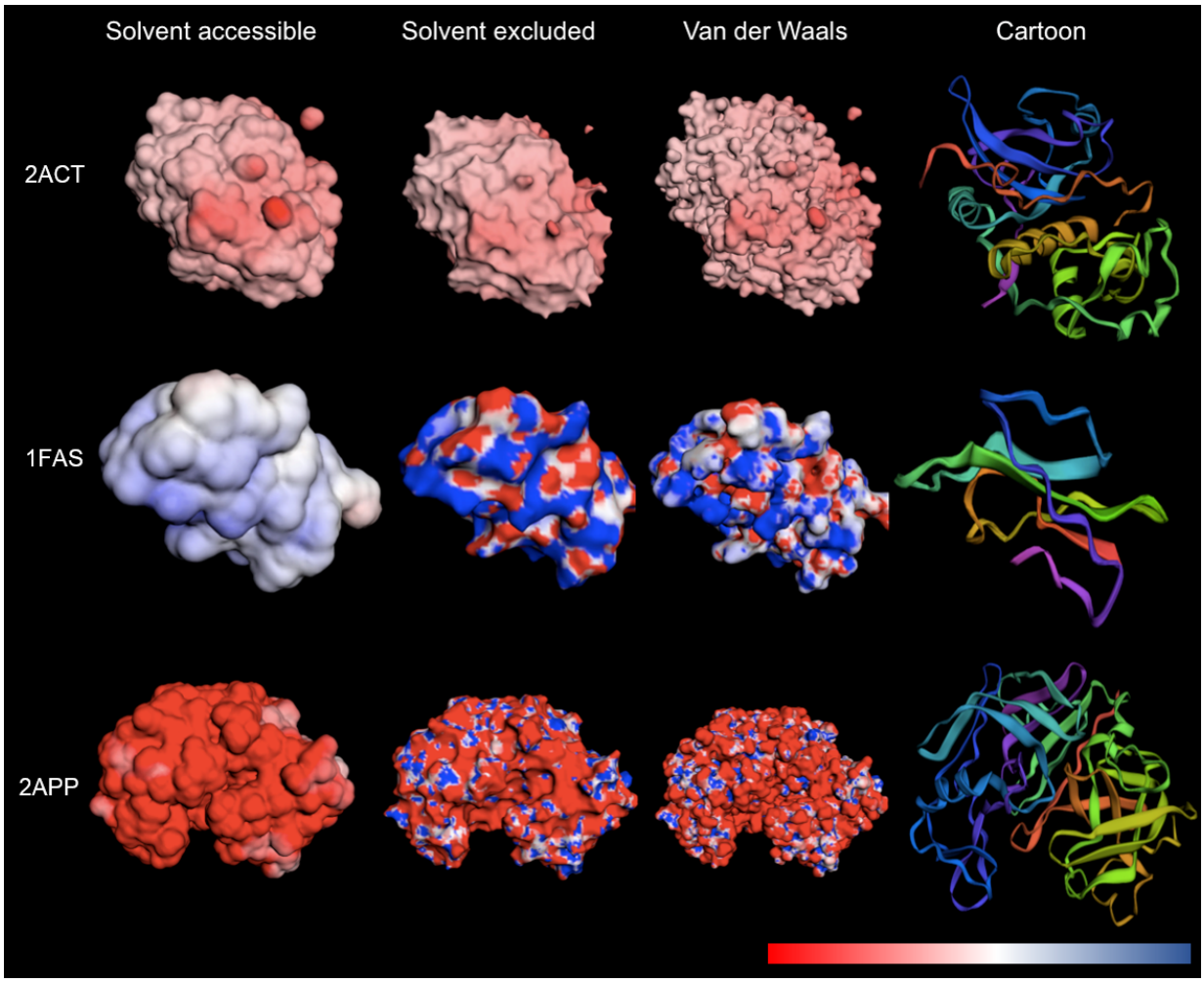


Figure 8: Renderings of three different proteins using renderings of Actinidin (2ACT) (top), Fasciculin (1FAS) (center), and Pepsin, Penicillium (2APP) (bottom). To demonstrate the different visualization options. From left to right, solvent accessible surface, solvent excluded surface, van der Waals surface, and cartoon model are shown all using red-white-blue color scheme (excluding cartoon model).

3Dmol loads quickly and does not require any downloads or plug-ins for use.

3.4.3 Other data uses

There are a number of other applications of implicit solvent models to biomedical problems. For example, during the past four years, our APBS software has been used in the post-simulation energetic analyses of molecular dynamics trajectories [43], understanding protein-nanoparticle interactions [44**? ?**], understanding nucleic acid-ion interactions [45**?**], biomolecular docking [46], developing new coarse-grained protein models [47], setting up membrane protein simulations [48], etc. APBS also plays a key role in software packages from the Wade group, including PIPSA for protein surface electrostatics analysis [49] and SDA for simulation of protein-protein interactions through Brownian dynamics [50]. 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 [43, 50**? ?** –53].

4 Using the software

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.[8]

4.1 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.

4.2 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. [9]
- 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.[10]

[8] www.poissonboltzmann.org
[9] www.mccammon.ucsd.edu/iapbs
[10] www.dasherwustl.edu/tinker

4.3 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.

4.4 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 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 [54].

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.

5 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.

6 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] 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.

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

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

[4] 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.

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

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

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] E. H. Yap and T. Head-Gordon. Calculating the bimolecular rate of protein-protein association with interacting crowders. *J. Chem. Theory Comp.*, 9(5):2481–9, 2013.

[29] S.H. Northrup, S.A. Allison, and J.A. McCammon. Brownian dynamics simulation of diffusion influenced bimolecular reactions. *J. Chem. Phys.*, 80:1517–1524, 1984.

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

[31] 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.

[32] 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.

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] Loretta L Jones, Kenneth D Jordan, and Neil A Stillings. Molecular visualization in chemistry education: the role of multidisciplinary collaboration. *Chemistry Education Research and Practice*, 6(3):136–149, 2005.

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

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

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] Y Song, Y Zhang, CL Bajaj, and NA Baker. Continuum diffusion reaction rate calculations of wild-type and mutant mouse acetylcholinesterase: adaptive finite element analysis. *Biophysical Journal*, 87(3):1558–1566, 2004.

[52] A H Elcock. Molecular simulations of diffusion and association in multimacromolecular systems. *Methods Enzymol*, 383:166–98, 2004.

[53] P Mereghetti and RC Wade. Atomic detail brownian dynamics simulations of concentrated protein solutions with a mean field treatment of hydrodynamic interactions. *J. Phy. Chem.*, 116(29):8523–8533, 2012.

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