iAPBS: a programming interface to Adaptive Poisson-Boltzmann Solver (APBS)

Robert Konecny[1,2,3]‡, **Nathan A. Baker**[3,4] **and J. Andrew McCammon**[1,2,3,5]

[1] Department of Chemistry and Biochemistry, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0365
[2] Center for Theoretical Biological Physics, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0374
[3] National Biomedical Computational Resource, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093
[4] Pacific Northwest National Laboratory, P.O. Box 999, MS K7-28, Richland, WA 99352
[5] Howard Hughes Medical Institute and Department of Pharmacology, University of California at San Diego, La Jolla, CA 92093-0365

Abstract. The Adaptive Poisson-Boltzmann Solver (APBS) is a state-of-the-art suite for performing Poisson-Boltzmann electrostatic calculations on biomolecules. The iAPBS package provides a modular programmatic interface to the APBS library of electrostatic calculation routines. The iAPBS interface library can be linked with a FORTRAN or C/C++ program thus making all of the APBS functionality available from within the application. Several application modules for popular molecular dynamics simulation packages – Amber, NAMD and CHARMM are distributed with iAPBS allowing users of these packages to perform implicit solvent electrostatic calculations with APBS.

PACS numbers: 87.15.A-, 87.15.B-, 87.15.H-

AMS classification scheme numbers: 8U20, 92C40

Submitted to: *Computational Science & Discovery, June 2012*

‡ Contact: rok@ucsd.edu

1. Introduction

The important role of solvation in biomolecular systems has led to the development of a variety of computational methods for studying the properties of these interactions [1]. Two of the most popular methods are explicit solvent methods, which treat the solvent in full atomic detail, and implicit solvent methods, which represent the solvent through its average effect on solute. Although explicit solvent methods offer a very detailed description of biomolecular solvation they are computationally demanding due to the large number of degrees of freedom associated with the explicit solvent and ions. Consequently, implicit solvent methods have become popular alternatives to explicit solvent approaches [2–5].

The Adaptive Poisson-Boltzmann Solver (APBS) [6] is a software package for the numerical solution of the Poisson-Boltzmann equation, one of the most popular continuum descriptions of solvation for biomolecular systems. The Poisson-Boltzmann equation (PBE) [7, 8]

$$-\nabla \cdot \epsilon(x)\nabla\phi(x) + \bar{\kappa}^2(x)\sinh\phi(x) = f(x), \tag{1}$$

relates the electrostatic potential (ϕ) to the dielectric properties of the solute and solvent (ϵ), the ionic strength of the solution and the accessibility of ions to the solute interior ($\bar{\kappa}^2$), and the distribution of solute atomic partial charges (f). This nonlinear PBE equation is often simplified to the linearized PBE (LPBE) by assuming $\sinh\phi(x) \approx \phi(x)$.

APBS was designed to efficiently evaluate electrostatic properties for a wide range of length scales and has been used for calculations on systems from small to very large [6, 9, 10]. APBS is robust and offers state-of-the-art computational algorithms for the finite difference (FD) and finite element (FE) discretization schemes of the Poisson-Boltzmann equation. The iAPBS interface library programmatically abstracts all APBS features and makes them available to 3rd-party applications, like molecular dynamics and Monte Carlo simulation packages. APBS integration augments these 3rd-party packages with an advanced facility for evaluating implicit solvent electrostatic energies and forces. It also adds the ability for 3rd-party software to output spatial distributions of the calculated properties (charge, electrostatic potential, energy density, etc.) for visualization and post-processing.

2. Software Design and Methods

2.1. Implementation

The APBS package§ is written in an object-oriented form of ANSI C with some parts of the code in FORTRAN||. The numerical basis of the code is FEtk, a scientific computing toolkit developed by the Holst group [11, 12]. Through FEtk, APBS uses MALOC, a hardware abstraction library for a great portability, PMG (a multigrid solver) [13, 14] and MC (an adaptive finite element solver) [11, 12] for the problem domain discretization routines. The object-oriented nature of the APBS code lends itself to encapsulation and inclusion in

§ APBS is distributed under the BSD/MIT-style open source license.
|| FORTRAN support is currently deprecated in APBS and will be removed from future versions.

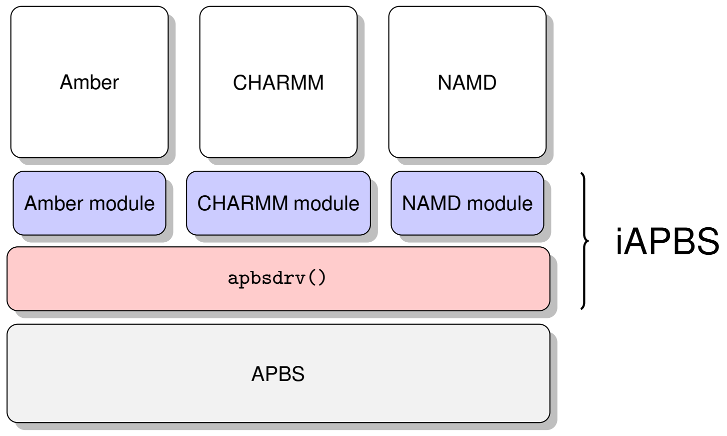

Figure 1. Schematic design of the iAPBS architecture. The low level interface (`apbsdrv()`, light red) abstracts all the APBS functionality and presents it through an API to higher level, application specific modules (blue).

other applications. The iAPBS software was written with this goal in mind: encapsulating the APBS functionality for easy integration with biochemical simulation codes which use molecular dynamics, Monte Carlo and other methods where implicit solvent contributions are evaluated.

A schematic representation of the iAPBS architecture design is shown in Figure 1. iAPBS consists of two software layers: a low-level APBS library interface function (`apbsdrv()`) and high level, application-specific modules. The `apbsdrv()` interface function abstracts all APBS functionality and exposes it through an application programming interface (API) to the higher level application module. The function accepts several input parameters – coordinates of atoms, their charges and radii and APBS calculation parameters. All these input parameters are passed to the APBS routines prior to the electrostatic calculation. The function returns calculated polar and non-polar solvation energies and forces. Various calculated properties like electrostatic potential, solvent accessible surface definition, charge distribution, can also be written to files and subsequently post-processed or visualized by external applications. The interface is written in an object-oriented form of the C programming language and can be transparently called from any C, C++ or FORTRAN code.

The calculated electrostatic and non-polar energies and forces are then incorporated into a biochemical simulation program by the application-specific module. This module should also parse the application input files and perform error checking and output printing. Currently, application modules for Amber [15], CHARMM [16] and NAMD [17] molecular dynamics packages are distributed with iAPBS.

The iAPBS distribution also includes, in addition to the extensive testing facilities a reference module implementation in FORTRAN (`src/wrapper.f`) which reads input data from an external file, calls the `apbsdrv()` interface function and prints out calculated electrostatic energy. This reference code can be used as a template for writing additional application modules. The iAPBS documentation includes an extensive user guide with practical examples and also a programmer's guide describing the iAPBS API which should aid in application module development.

2.2. Application Modules

The role of the application modules is to integrate APBS calculated electrostatic properties into the information flow in a molecular dynamics simulation program. When carrying out implicit solvent molecular dynamics simulations, the calculated solvation energies and forces are typically added to the total system free energy (G_{tot}) and forces (F_{tot}) in each MD step. A flowchart presented in Figure 2 shows how the iAPBS modules can be incorporated into a molecular dynamics modeling program.

First, the application parses the general MD calculation parameters, reads in the molecule coordinates, bonding information and the necessary force field parameters. Next, the iAPBS module parses input files for the APBS-related calculation parameters; for example, if solution of the nonlinear or linearized Poisson-Boltzmann equation was selected, discretization scheme, type of boundary conditions, size of the grid and its resolution, dielectric constants for the protein and solvent, ionic strength, etc. In addition to the molecular Cartesian coordinates, per-atom charges and radii are also read in. Since calculated continuum solvation properties using a discontinuous dielectric function are very sensitive to the surface definitions, the selection of radius parameters is an important step in any PB calculation [18–20]. iAPBS modules offer several flexible options how the charge and radius set is constructed. These values can be either automatically extracted from the appropriate molecular dynamics force field or they can be read in from an external file. All available modules support reading the charge and radius information from PQR files. The format of PQR files is a modified PDB format with added charge and radius parameters. A PQR file can be generated from a PDB file using the PDB2PQR utility [21] or the PDB2PQR web service [22]. The charge and radius information can also be read from a simply formatted keyword/value list.

The iAPBS interface function then performs the actual electrostatic calculation utilizing the necessary APBS library routines¶. Upon exit the function returns the calculated electrostatic and non-polar energies and forces which are then added to the total system free energy and forces. This process is described in detail in the following two sections.

2.2.1. Calculating Solvation Energy

The total free energy of the system (G_{tot}) in implicit solvent stochastic molecular dynamics simulations can be written as a sum of the internal energy of the molecule (bonded and non-bonded) and free energy of solvation

$$G_{tot} = E_{bonded} + E_{non\text{-}bonded} + \Delta G_{solv}. \tag{2}$$

The bonded internal energy E_{bonded} includes contributions from covalent interactions and the non-bonded internal energy $E_{non\text{-}bonded}$ usually contains the van der Waals and Coulombic electrostatic terms. Both, the bonded and non-bonded energy terms are

¶ Periodic boundary conditions simulations are currently not supported via APBS.

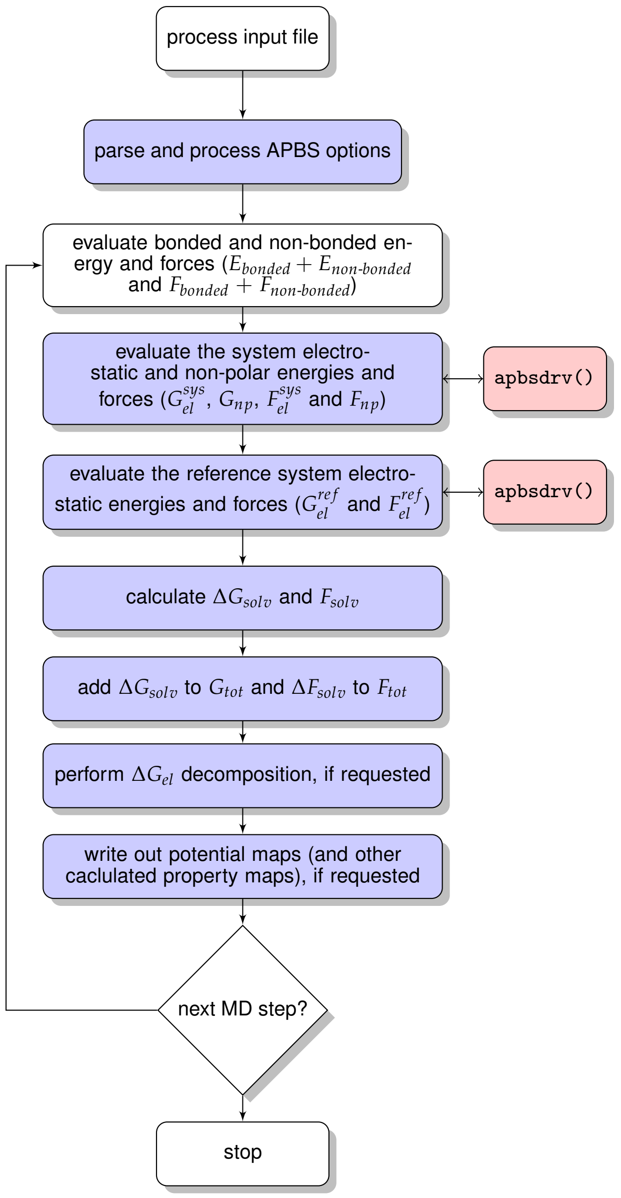
process input file
parse and process APBS options
evaluate bonded and non-bonded energy and forces ($E_{bonded} + E_{non\text{-}bonded}$ and $F_{bonded} + F_{non\text{-}bonded}$)
evaluate the system electrostatic and non-polar energies and forces (G_{el}^{sys}, G_{np}, F_{el}^{sys} and F_{np})
apbsdrv()
evaluate the reference system electrostatic energies and forces (G_{el}^{ref} and F_{el}^{ref})
apbsdrv()
calculate ΔG_{solv} and F_{solv}
add ΔG_{solv} to G_{tot} and ΔF_{solv} to F_{tot}
perform ΔG_{el} decomposition, if requested
write out potential maps (and other caclulated property maps), if requested
next MD step?
stop

Figure 2. A symbolic flowchart of the iAPBS module integration to a molecular dynamics simulation program. The white boxes represent data processing and calculation by the MD application. The iAPBS module code (blue blocks) parses and pre-processes APBS calculation parameters, calls the low level `apbsdrv()` interface (light red), updates the total system forces and energy with the calculated solvation terms ($\Delta F_{el} + F_{np}$ and $\Delta G_{el} + G_{np}$, respectively) and, if requested carries out post-processing of the calculated quantities.

calculated by the MD application. The total solvation free energy of a molecule ΔG_{solv} is calculated by the iAPBS module using the APBS electrostatic routines. This term is computed [23] as a sum of the electrostatic energy of the system (ΔG_{el}) and the non-polar contribution to biomolecular energetics (G_{np})

$$\Delta G_{solv} = \Delta G_{el} + G_{np}. \tag{3}$$

The total electrostatic energy G_{el} can be obtained by solving the Poisson-Boltzmann equation. However, the calculated electrostatic energy (and force) contains large "self-energy" terms associated with the interaction of a particular charge with itself these terms are highly dependent on the discretization of the problem [24]. Therefore, the self-energy terms are removed by a reference calculation using the same numerical discretization

$$\Delta G_{el} = G_{el}^{sys} - G_{el}^{ref}, \tag{4}$$

where G_{el}^{sys} is the electrostatic free energy of the system with different dielectric constant inside (in the protein, ϵ_p) and outside (in the solvent, ϵ_s) of the biomolecule, a fixed charge distribution corresponding to the atom locations and charges, and a varying ion accessibility coefficient ($\bar{\kappa}^2(x)$) that is zero inside the biomolecule and equal to the bulk ionic strength outside. The reference free energy G_{el}^{ref} uses the same fixed charge distribution but has a constant dielectric coefficient equal to the value in biomolecular interior ($\epsilon_s = \epsilon_p$) and a constant zero ion accessibility coefficient ($\bar{\kappa}^2(x) = 0$).

The ΔG_{el} evaluation is implemented in iAPBS as a two step calculation (see Figure 2). First, the electrostatic energy of the system (G_{el}^{sys}) is computed by calling `apbsdrv()` with the requested calculation parameters. Then the same calculation is repeated in the reference environment – in homogeneous dielectric $\epsilon_s = \epsilon_p$ and zero ionic strength ($\bar{\kappa}^2(x) = 0$). Both calculations use the same discretization (i.e., same FD grid spacing or FE refinement) to ensure cancellation of self-energies in computing ΔG_{el}.

In iAPBS, the non-polar solvation term G_{np} in Equation 3 is approximated by a linear function of the solvent-accessible surface area:

$$G_{np} = \gamma A \tag{5}$$

where A is the solvent-accessible surface area and γ is the energetic coefficient or surface tension. APBS also offers more accurate approximate methods [25] for evaluating G_{np}; these will be implemented in the future releases of iAPBS.

2.2.2. Calculating Solvation Forces Similar to the total free energy calculation, the total force on the system in implicit solvent molecular dynamics simulations is evaluated as a sum of bonded forces (F_{bonded}), non-bonded forces ($F_{non\text{-}bonded}$) and forces due to the solvent environment (ΔF_{solv}):

$$F_{tot} = F_{bonded} + F_{non\text{-}bonded} + \Delta F_{solv}. \tag{6}$$

Both F_{bonded} and $F_{non\text{-}bonded}$ forces are computed by the MD application and the solvation forces ΔF_{solv} are calculated by the iAPBS interface function. Like their energetic counterparts, solvation force evaluations must also be performed using reference calculation due to the presence of self-interactions in the charge distribution. The total solvation force on the system is calculated as

$$\Delta F_{solv} = F_{el}^{sys} - F_{el}^{ref} + F_{np}, \tag{7}$$

where F_{el}^{sys} is the total electrostatic force on atoms of the system due to all atoms. This is calculated from the numerical solution of the PB equation and the dielectric and ion accessibility coefficients are inhomogeneous. F_{el}^{ref} is the total electrostatic force on atoms in the reference system due to all atoms. This is again calculated from the numerical solution of the PB equation, however the dielectric and ion accessibility coefficients are homogeneous ($\epsilon_s = \epsilon_p$) and $\bar{\kappa}^2(x) = 0$. As with the computation of free energies, this calculation is performed using the same discretization as the calculation of F_{el}^{sys}. The F_{np} term represents the non-polar forces.

As shown in Figure 2, the total solvation energy and force (ΔG_{solv} and ΔF_{solv}) calculation requires two calls to the low level `apbsdrv()` function per MD step. During the first call, the system electrostatic energy and forces (G_{el}^{sys} and F_{el}^{sys}) and the non-polar energy and forces (G_{np} and F_{np}, respectively) are calculated. During the second call the reference electrostatic energy and force (G_{el}^{ref} and F_{el}^{ref}) are computed. These two calculations must be repeated for each MD step (unless an alternative solvation force update scheme is used, see below). Since the `apbsdrv()` evaluation is the slowest step in the workflow there are some performance issues to be considered. Although it is possible to implement, with certain assumptions, a solvation energy and force calculation protocol using only one `apbsdrv()` call per MD step (i.e., skipping the reference system calculation) this comes with a significant accuracy penalty. The current versions of Amber, CHARMM and NAMD modules use the reference system calculation every MD step to ensures cancellation of self energies and, therefore, accurate solvation energies and forces. These modules also reconstruct the numerical grid every MD step instead of re-using it. This improves the PB calculation stability and accuracy due to maintaining the rotational invariability of the PB solution with respect to the grid (at adequate grid spacing) with only a negligible impact on the performance.

The forces and energies due to Coulomb's law (pairwise Coulombic interactions between all atoms in the molecule) are not calculated by iAPBS and must be supplied by the molecular dynamics application. This is usually not an issue since the calculation of Coulombic energies and forces is implemented as a part of the non-bonded energy and force evaluation in these programs.

2.2.3. Other Features The currently available iAPBS application modules also contain several features extending the capabilities of these applications:

Adaptive grid size The Amber, NAMD and CHARMM modules implement an adaptive grid size algorithm for minimization and molecular dynamics calculations. This feature allows the user to specify a target grid resolution (for example 0.5 Å); the grid span and size are then recalculated on the fly during each minimization or MD step, ensuring that the molecule is completely enveloped by the grid of the requested resolution. This adaptive grid resizing increases accuracy of the calculation of both electrostatic energy and electrostatic forces due to the consistent grid discretization parameters between the MD steps. It also prevents the molecule from "falling off" the grid when the volume of the molecule is changed from one MD step to another during the simulation.

On-the-fly electrostatic potential map generation The Amber module can also write out electrostatic potential maps after each individual step in the molecular dynamics simulation. The maps (one per each MD step or optionally per *n-th* MD step) can be post-processed to create a dynamic picture of the electrostatic potential during the molecular dynamics (for example as a movie).

Electrostatic energy decomposition Another feature implemented in the Amber module is the availability of electrostatic energy decomposition – either at per-atom or per-residue levels. This information can be used, for example, to study the changes in the electrostatic potential at active site residues during a MD simulation.

Alternative solvation force update scheme The Poisson-Boltzmann-based solvation force evaluation is one of the slowest steps in molecular dynamics simulations and there has been a considerable effort to address this performance issue [26]. Since the electrostatic potential distribution varies only slowly with small conformation changes during dynamic simulations, it is reasonable to assume that updating this potential only every *n-th* step should not compromise the integrity of the calculated thermodynamic and kinetic data. The iAPBS Amber module implements an alternative solvation force update scheme based on the mollified impulse method [27]. Additionally, the solvation update force code is written in a modular way so new update schemes can be easily implemented.

MM-PBSA based post-processing The availability of very accurate solvation energies via the iAPBS module can also be exploited in MM-PBSA [28] type of calculations. The mmpbsa.py script [29] which is distributed with Amber allows use of the iAPBS-calculated solvation energies (both electrostatic and non-polar parts) in the MM-PBSA protocol.

Visualization and post-processing of volumetric data Visualization is a very important tool when analyzing biomolecular electrostatics. Some of the most common ways to study the three dimensional distribution of electrostatic potential around the molecule include projection of the calculated potential on the solvent accessible surface area (SASA) of the molecule or construction of electrostatic potential isocontours around the molecule.

APBS features an advanced facility for writing out calculated three-dimensional volumetric data, from electrostatic maps to charge distributions. The iAPBS interface allows for this facility to be used in the molecular dynamics simulation packages thus providing a way to generate "dynamic" electrostatic data. The ability to investigate how the electrostatic spatial data change during the molecular dynamics simulation can offer additional insights to the structure and function of biomolecules. There are many possible applications of this approach for example observing dynamical changes in three-dimensional properties of the electrostatic potential around an active site in a biomolecule, visualizing changes in per-residue (or per-atom) electrostatic properties, monitoring electrostatic interactions between selected residues during the molecular dynamics simulation, etc. The iAPBS distribution includes several PyMOL [30] script templates for generating movies of electrostatic properties from the individual MD simulation frames.

3. Conclusions

The Adaptive Poisson-Boltzmann Solver (APBS) software package is a widely used tool for evaluating electrostatic interactions in biomolecular systems in implicit solvent. The iAPBS interface library offers a straightforward integration of the APBS capabilities into biomolecular simulation programs. Several application modules for popular molecular dynamics applications – Amber, CHARMM and NAMD are distributed with iAPBS. These modules allow incorporation of APBS-calculated electrostatic and solvation energies and forces into simulations. They also add the ability to easily generate time series of spatial distributions for electrostatic data during the molecular dynamics simulation using the advanced APBS volumetric data output facility. The modular design of the iAPBS interface library allows easy extendability and addition of new application modules.
Availability: iAPBS is distributed as part of the APBS source code, in the `contrib/iapbs` directory. APBS with iAPBS can be downloaded from `http://www.poissonboltzmann.org/apbs`. Instructions on how to build and install iAPBS can be found at `http://mccammon.ucsd.edu/iapbs`. The software is being released under the GNU public license.

Acknowledgments

We are grateful to Todd Dolinsky and David Gohara for their invaluable technical assistance and Jason Swails for very useful and stimulating discussions. We also wish to thank Justin Gullingsrud who wrote the original iAPBS/NAMD module. This work was supported in part by the W. M. Keck Foundation, the National Biomedical Computational Resource (NIH P41 RR0860516 and P41 RR08605) and the Center for Theoretical Biological Physics (NSF PHY-0216576 and 0225630). Work in the JAM group is supported by NIH, NSF, and HHMI. Additional funding for this work was provided by NIH grant R01 GM069702 to NAB.

References

[1] Pengyu Ren, Jaehun Chun, Dennis G. Thomas, Michael J. Schnieders, Marcelo Marucho, Jiajing Zhang, and Nathan A. Baker. Biomolecular electrostatics and solvation: a computational perspective. *Quart. Revs. Biophysics*, in press.

[2] Nathan A. Baker. Improving implicit solvent simulations: a Poisson-centric view. *Curr. Opin. Struct. Biol.*, 15(2):137–143, 2005.

[3] Nathan A. Baker, Donald Bashford, and David A. Case. Implicit solvent electrostatics in biomolecular simulation. In Benedict Leimkuhler, Christophe Chipot, Ron Elber, Aatto Laaksonen, Alan Mark, Tamar Schlick, Christoph Schütte, and Robert Skeel, editors, *New Algorithms for Macromolecular Simulation*, volume 49 of *Lecture Notes in Computational Science and Engineering*, pages 263–295. Springer Berlin Heidelberg, 2006.

[4] Benoît Roux and Thomas Simonson. Implicit solvent models. *Biophys. Chem.*, 78(1-2):1–20, 1999.

[5] B. Roux. Implicit solvent models. In O. M. Becker, A. D. Mackerell Jr., B. Roux, and M. Watanabe, editors, *Computational Biochemistry and Biophysics*, pages 133–152. Marcel Dekker, New York, 2001.

[6] N. A. Baker, D. Sept, S. Joseph, M. J. Holst, and J. A. McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proc. Natl. Acad. Sci. USA*, 98:10037–10041, 2001.

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

[8] Barry Honig and Anthony Nicholls. Classical electrostatics in biology and chemistry. *Science*, 268(5214):1144–1149, 1995.

[9] Joanna Trylska, Robert Konecny, Florence Tama, Charles L Brooks, and J. Andrew McCammon. Ribosome motions modulate electrostatic properties. *Biopolymers*, 74(6):423–431, 2004.

[10] Robert Konecny, Joanna Trylska, Florence Tama, Deqiang Zhang, Nathan A. Baker, Charles L Brooks, and J. Andrew McCammon. Electrostatic properties of cowpea chlorotic mottle virus and cucumber mosaic virus capsids. *Biopolymers*, 82(2):106–120, 2006.

[11] M. Holst. Adaptive numerical treatment of elliptic systems on manifolds. *Adv. Comput. Math.*, 15(1):139–191, 2001.

[12] Randolph E. Bank and Michael Holst. A new paradigm for parallel adaptive meshing algorithms. *SIAM Rev.*, 45(2):291, 2003.

[13] Michael Holst and Faisal Saied. Multigrid solution of the Poisson-Boltzmann equation. *J. Comp. Chem.*, 14(1):105–113, 1993.

[14] Michael J Holst and Faisal Saied. Numerical solution of the nonlinear Poisson-Boltzmann equation: Developing more robust and efficient methods. *J. Comp. Chem.*, 16(3):337–364, 1995.

[15] D. A. Case, T. A. Darden, T. E. Cheatham, III, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, R. C. Walker, W. Zhang, K. M. Merz, B. Roberts, S. Hayik, A. Roitberg, G. Seabra, J. Swails, A. W. Goetz, I. Kolossvai, K. F. Wong, F. Paesani, J. Vanicek, R. M. Wolf, J. Liu, X. Wu, S. R. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, G. Cui, D. R. Roe, M. G. Seetin D. H. Mathews, R. Salomon-Ferrer, C. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, and P. A. Kollman. *AMBER 12*. University of California, San Francisco, 2012.

[16] B. R. Brooks, C. L. Brooks, III, A. D. Mackerell, 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. Comp. Chem.*, 30:1545–1615, 2009.

[17] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kale, and Klaus Schulten. Scalable molecular dynamics with NAMD. *J. Comp. Chem.*, 26:1781–1802, 2005.

[18] S. W. Rick and B. J. Berne. The aqueous solvation of water: A comparison of continuum methods with molecular dynamics. *J. Am. Chem. Soc.*, 116(9):3949–3954, 1994.

[19] Mafalda Nina, Dmitri Beglov, and Benoît Roux. Atomic radii for continuum electrostatics calculations

based on molecular dynamics free energy simulations. *J. Phys. Chem. B*, 101(26):5239–5248, 1997.

[20] Jessica M. J. Swanson, Jason A. Wagoner, Nathan A. Baker, and J. A. McCammon. Optimizing the Poisson dielectric boundary with explicit solvent forces and energies: Lessons learned with atom-centered dielectric functions. *J. Chem. Theory Comput.*, 3(1):170–183, 2007.

[21] T. J. Dolinsky, J. E. Nielsen, J. A. McCammon, and N. A. Baker. PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. *Nucleic Acids Res.*, 32(Web Server):W665–W667, 2004.

[22] Samir Unni, Yong Huang, Robert M. Hanson, Malcolm Tobias, Sriram Krishnan, Wilfred W. Li, Jens E. Nielsen, and Nathan A. Baker. Web servers and services for electrostatics calculations with APBS and PDB2PQR. *J. Comp. Chem.*, 32(7):1488–1491, 2011.

[23] Nathan A. Baker. Poisson-boltzmann methods for biomolecular electrostatics. In Ludwig Brand and Michael L. Johnson, editors, *Numerical Computer Methods, Part D*, volume 383 of *Methods in Enzymology*, pages 94–118. Academic Press, 2004.

[24] Feng Dong, Brett Olsen, and Nathan A. Baker. Computational methods for biomolecular electrostatics. In Dr. John J. Correia and III Dr. H. William Detrich, editors, *Biophysical Tools for Biologists, Volume One: In Vitro Techniques*, volume 84 of *Methods in Cell Biology*, pages 843–870. Academic Press, 2008.

[25] J. A. Wagoner and N. A. Baker. Assessing implicit models for nonpolar mean solvation forces: the importance of dispersion and volume terms. *Proc. Natl. Acad. Sci. USA*, 103:8331–8336, 2006.

[26] Ray Luo, Laurent David, and Michael K. Gilson. Accelerated Poisson-Boltzmann calculations for static and dynamic systems. *J. Comp. Chem.*, 23(13):1244–1253, 2002.

[27] Jesús A. Izaguirre, Sebastian Reich, and Robert D. Skeel. Longer time steps for molecular dynamics. *J. Chem. Phys.*, 110(20):9853–9864, 1999.

[28] Peter A. Kollman, Irina Massova, Carolina Reyes, Bernd Kuhn, Shuanghong Huo, Lillian Chong, Matthew Lee, Taisung Lee, Yong Duan, Wei Wang, Oreola Donini, Piotr Cieplak, Jaysharee Srinivasan, David A. Case, and Thomas E. Cheatham. Calculating structures and free energies of complex molecules: Combining molecular mechanics and continuum models. *Acc. Chem. Res.*, 33(12):889–897, 2000.

[29] Dwight McGee, Billy Miller, III, and Jason Swails. MMPBSA.py: A script for performing Molecular Mechanics Poisson Boltzmann Surface Area calculation to find free energies of binding. 2012.

[30] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.4r1. 2012.