iAPBS: a programming interface to the adaptive Poisson–Boltzmann solver

2012 Comput. Sci. Disc. 5 015005

(http://iopscience.iop.org/1749-4699/5/1/015005)

COMPUTATIONAL SCIENCE&DISCOVERY

iAPBS: a programming interface to the adaptive Poisson–Boltzmann solver

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, USA
[2] Center for Theoretical Biological Physics, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0374, USA
[3] National Biomedical Computational Resource, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA
[4] Pacific Northwest National Laboratory, PO Box 999, MS K7-28, Richland, WA 99352, USA
[5] Howard Hughes Medical Institute and Department of Pharmacology, University of California at San Diego, La Jolla, CA 92093-0365, USA
E-mail: rok@ucsd.edu

Received 22 May 2012, in final form 27 June 2012
Published 26 July 2012
Computational Science & Discovery **5** (2012) 015005 (8pp)
doi:10.1088/1749-4699/5/1/015005

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.

Contents

www.iop.org/journals/csd
1749-4699/12/015005+08$33.00

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 the 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 (PBE), one of the most popular continuum descriptions of solvation for biomolecular systems. The 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 is often simplified to the linearized PBE by assuming that $\sinh\phi(x) \approx \phi(x)$.

APBS was designed to efficiently evaluate electrostatic properties for a wide range of length scales and has been used for calculations on systems ranging 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 PBE. The iAPBS interface library programmatically abstracts all APBS features and makes these available to third-party applications, such as molecular dynamics (MD) and Monte Carlo simulation packages. APBS integration augments these third-party packages with an advanced facility for evaluating implicit solvent electrostatic energies and forces. It also adds the ability for third-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[6] is written in an object-oriented form of ANSI C with some parts of the code in FORTRAN[7]. 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 greater portability, PMG (a multigrid solver) [13, 14] and MC (an adaptive FE solver) [11, 12] for the problem domain discretization routines. The object-oriented nature of the APBS code lends itself to encapsulation and inclusion in other applications. The iAPBS software was written with the following goal in mind: encapsulating the APBS functionality for easy integration with biochemical simulation codes that use MD, 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 such as electrostatic potential, solvent accessible surface definition and charge distribution can also be written to files

[6] APBS is distributed under the BSD/MIT-style open source license.

[7] FORTRAN support is currently deprecated in APBS and will be removed from future versions.

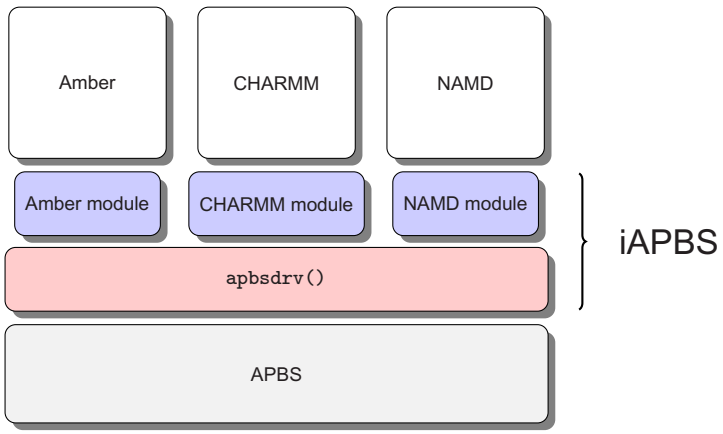

Figure 1. Schematic design of the iAPBS architecture. The low-level interface (`apbsdrv()`, light red) abstracts all the APBS functionality and presents it through an API to higher-level, application-specific modules (blue).

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 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 an MD simulation program. When carrying out implicit solvent MD 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 an MD 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 PBE was selected, discretization scheme, the type of boundary conditions, the 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 the calculated continuum solvation properties using a discontinuous dielectric function are very sensitive to the surface definitions, selection of the radius parameters is an important step in any Poisson–Boltzmann calculation [18–20]. iAPBS modules offer several flexible options for how the charge and radius set is constructed. These values can be either automatically extracted from the appropriate MD force field or 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.

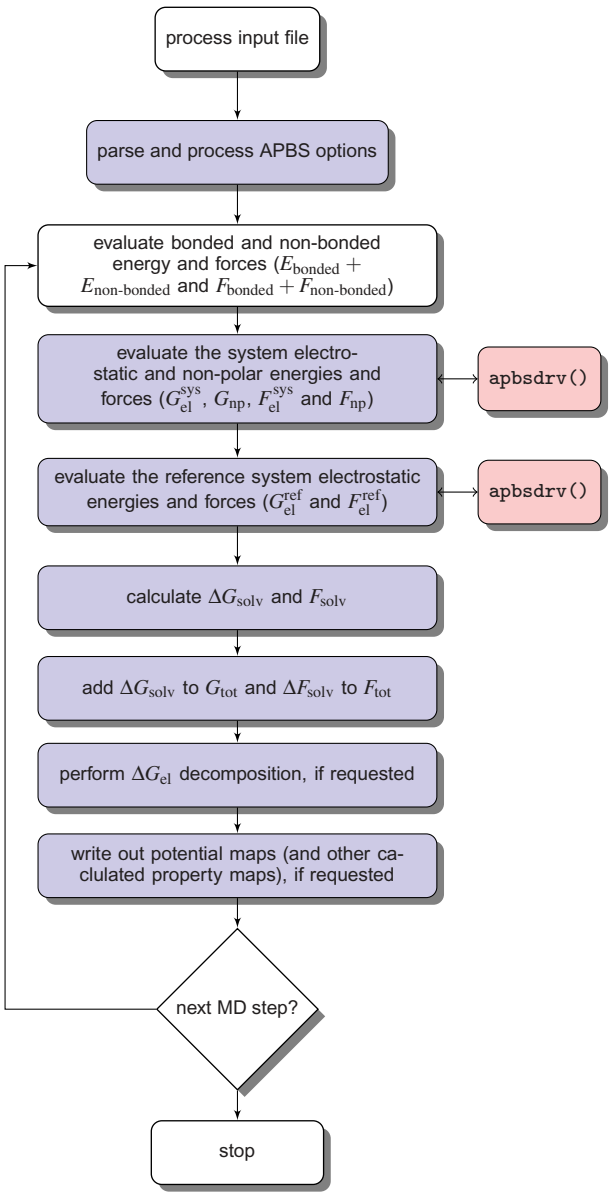

Figure 2. A symbolic flowchart of the iAPBS module's integration into a molecular dynamics simulation program. The white boxes represent data processing and calculation by the MD application. The iAPBS module code (blue blocks) parses and pre-processes APBS calculation parameters, calls the low-level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

The iAPBS interface function then performs the actual electrostatic calculation utilizing the necessary APBS library routines[8]. 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 MD simulations can be written as a sum of the internal energy of the molecule (bonded and non-bonded) and

[8] Periodic boundary condition simulations are currently not supported via APBS.

free energy of solvation

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

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

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

The total electrostatic energy G_{el} can be obtained by solving the PBE. 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

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

where $G_{\text{el}}^{\text{sys}}$ is the electrostatic free energy of the system with different dielectric constants 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 $G_{\text{el}}^{\text{ref}}$ uses the same fixed charge distribution but has a constant dielectric coefficient equal to the value in biomolecular interior ($\epsilon_{\text{s}} = \epsilon_{\text{p}}$) 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 ($G_{\text{el}}^{\text{sys}}$) is computed by calling `apbsdrv()` with the requested calculation parameters. Then the same calculation is repeated in the reference environment—in homogeneous dielectric $\epsilon_{\text{s}} = \epsilon_{\text{p}}$ and zero ionic strength ($\bar{\kappa}^2(x) = 0$). Both calculations use the same discretization (i.e. the same FD grid spacing or FE refinement) to ensure cancelation 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:

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

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 MD simulations is evaluated as a sum of bonded forces (F_{bonded}), non-bonded forces ($F_{\text{non-bonded}}$) and forces due to the solvent environment (ΔF_{solv}):

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

Both F_{bonded} and $F_{\text{non-bonded}}$ 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

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

where $F_{\text{el}}^{\text{sys}}$ is the total electrostatic force on atoms of the system due to all atoms. This is calculated from the numerical solution of the PBE and the dielectric and ion accessibility coefficients are inhomogeneous. $F_{\text{el}}^{\text{ref}}$ 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 PBE; 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 ensure cancelation 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 Poisson–Boltzmann calculation stability and accuracy due to maintaining the rotational invariability of the Poisson–Boltzmann 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 MD 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 contain also 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 MD 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 the 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 MD 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 MD (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 at per-residue levels. This information can be used, for example, to study the changes in the electrostatic potential at active site residues during an MD simulation.

Alternative solvation force update scheme. The Poisson–Boltzmann-based solvation force evaluation is one of the slowest steps in MD simulations and considerable effort has been put 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.

Molecular mechanics Poisson–Boltzmann surface area (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 calculations. The `mmpbsa.py` script [29] that is distributed with Amber allows the 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 (3D) distribution of the electrostatic potential around the molecule include projection of the calculated potential on the solvent accessible surface area of the molecule or construction of electrostatic potential isocontours around the molecule.

APBS features an advanced facility for writing out calculated 3D volumetric data, from electrostatic maps to charge distributions. The iAPBS interface allows for this facility to be used in the MD simulation packages, thus providing a way to generate 'dynamic' electrostatic data. The ability to investigate how the electrostatic spatial data change during the MD simulation can offer additional insights into the structure and function of biomolecules. There are many possible applications of this approach, for example, observing dynamical changes in 3D 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 MD simulation and so on. The iAPBS distribution includes several PyMOL [30] script templates for generating movies of electrostatic properties from the individual MD simulation frames.

3. Conclusions

The APBS software package is a widely used tool for evaluating electrostatic interactions in biomolecular systems in the implicit solvent. The iAPBS interface library offers a straightforward integration of the APBS capabilities into biomolecular simulation programs. Several application modules for popular MD 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 MD 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 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 of the JAM group is supported by NIH, NSF and HHMI. Additional funding for this work was provided by NIH grant no. R01 GM069702 to NAB.

References

[1] Ren P, Chun J, Thomas D G, Schnieders M J, Marucho M, Zhang J and Baker N A Biomolecular electrostatics and solvation: a computational perspective *Q. Rev. Biophys.* at press

[2] Baker N A 2005 Improving implicit solvent simulations: a Poisson-centric view *Curr. Opin. Struct. Biol.* **15** 137–43

[3] Baker N A, Bashford D and Case D A 2006 Implicit solvent electrostatics in biomolecular simulation *New Algorithms for Macromolecular Simulation (Lecture Notes in Computational Science and Engineering* vol 49*)* ed B Leimkuhler, C Chipot, R Elber, A Laaksonen, A Mark, T Schlick, C Schütte and R Skeel (Berlin: Springer) pp 263–95
[4] Roux B and Simonson T 1999 Implicit solvent models *Biophys. Chem.* **78** 1–20
[5] Roux B 2001 Implicit solvent models *Computational Biochemistry and Biophysics* ed O M Becker, A D Mackerell Jr, B Roux and M Watanabe (New York: Dekker) pp 133–52
[6] Baker N A, Sept D, Joseph S, Holst M J and McCammon J A 2001 Electrostatics of nanosystems: application to microtubules and the ribosome *Proc. Natl Acad. Sci. USA* **98** 10037–41
[7] Davis M E and McCammon J A 1990 Electrostatics in biomolecular structure and dynamics *Chem. Rev.* **90** 509–21
[8] Honig B and Nicholls A 1995 Classical electrostatics in biology and chemistry *Science* **268** 1144–9
[9] Trylska J, Konecny R, Tama F, Brooks C L and McCammon A J 2004 Ribosome motions modulate electrostatic properties *Biopolymers* **74** 423–31
[10] Konecny R, Trylska J, Tama F, Zhang D, Baker N A, Brooks C L and McCammon J A 2006 Electrostatic properties of cowpea chlorotic mottle virus and cucumber mosaic virus capsids *Biopolymers* **82** 106–20
[11] Holst M 2001 Adaptive numerical treatment of elliptic systems on manifolds *Adv. Comput. Math.* **15** 139–91
[12] Bank R E and Holst M 2003 A new paradigm for parallel adaptive meshing algorithms *SIAM Rev.* **45** 291
[13] Holst M and Saied F 1993 Multigrid solution of the Poisson–Boltzmann equation *J. Comput. Chem.* **14** 105–13
[14] Holst M J and Saied F 1995 Numerical solution of the nonlinear Poisson–Boltzmann equation: developing more robust and efficient methods *J. Comput. Chem.* **16** 337–64
[15] Case D A *et al* 2012 *AMBER 12* (San Francisco, CA: University of California)
[16] Brooks B R *et al* 2009 CHARMM: the biomolecular simulation program *J. Comput. Chem.* **30** 1545–615
[17] Phillips J C, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel R D, Kale L and Schulten K 2005 Scalable molecular dynamics with NAMD *J. Comput. Chem.* **26** 1781–02
[18] Rick S W and Berne B J 1994 The aqueous solvation of water: a comparison of continuum methods with molecular dynamics *J. Am. Chem. Soc.* **116** 3949–54
[19] Nina M, Beglov D and Roux B 1997 Atomic radii for continuum electrostatics calculations based on molecular dynamics free energy simulations *J. Phys. Chem.* B **101** 5239–48
[20] Swanson J M J, Wagoner J A, Baker N A and McCammon J A 2007 Optimizing the Poisson dielectric boundary with explicit solvent forces and energies: lessons learned with atom-centered dielectric functions *J. Chem. Theory Comput.* **3** 170–83
[21] Dolinsky T J, Nielsen J E, McCammon J A and Baker N A 2004 PDB2PQR: an automated pipeline for the setup of Poisson–Boltzmann electrostatics calculations *Nucleic Acids Res.* **32(Web Server)** W665–7
[22] Unni S, Huang Y, Hanson R M, Tobias M, Krishnan S, Li W W, Nielsen J E and Baker N A 2011 Web servers and services for electrostatics calculations with APBS and PDB2PQR *J. Comput. Chem.* **32** 1488–91
[23] Baker N A 2004 Poisson–Boltzmann methods for biomolecular electrostatics *Methods in Enzymology* vol 383 ed L Brand and M L Johnson (New York: Academic) pp 94–118
[24] Dong F, Olsen B and Baker N A 2008 Computational Methods for Biomolecular Electrostatics *Methods in Cell Biology* vol 84 ed J J Correia and H W Detrich III (New York: Academic) pp 843–70
[25] Wagoner J A and Baker N A 2006 Assessing implicit models for nonpolar mean solvation forces: the importance of dispersion and volume terms *Proc. Natl Acad. Sci. USA* **103** 8331–6
[26] Luo R, David L and Gilson M K 2002 Accelerated Poisson–Boltzmann calculations for static and dynamic systems *J. Comput. Chem.* **23** 1244–53
[27] Izaguirre J A, Reich S and Skeel R D 1999 Longer time steps for molecular dynamics *J. Chem. Phys.* **110** 9853–64
[28] Kollman P A *et al* 2000 Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models *Acc. Chem. Res.* **33** 889–97
[29] McGee D, Miller B III and Swails J 2012 MMPBSA.py: a script for performing molecular mechanics Poisson Boltzmann surface area calculation to find free energies of binding
[30] Schrödinger L L C 2012 The PyMOL molecular graphics system, version 1.4r1