Software News and Update
MIBPB: A Software Package for Electrostatic Analysis

DUAN CHEN,[1] ZHAN CHEN,[1] CHANGJUN CHEN,[1] WEIHUA GENG,[1*] GUO-WEI WEI[1,2]
[1]*Department of Mathematics, Michigan State University, East Lansing, Michigan 48824*
[2]*Department of Electrical and Computer Engineering, Michigan State University, East Lansing, Michigan 48824*

Received 28 August 2009; Revised 17 January 2010; Accepted 3 July 2010
DOI 10.1002/jcc.21646
Published online in Wiley Online Library (wileyonlinelibrary.com).

Abstract: The Poisson–Boltzmann equation (PBE) is an established model for the electrostatic analysis of biomolecules. The development of advanced computational techniques for the solution of the PBE has been an important topic in the past two decades. This article presents a matched interface and boundary (MIB)-based PBE software package, the MIBPB solver, for electrostatic analysis. The MIBPB has a unique feature that it is the first interface technique-based PBE solver that rigorously enforces the solution and flux continuity conditions at the dielectric interface between the biomolecule and the solvent. For protein molecular surfaces, which may possess troublesome geometrical singularities, the MIB scheme makes the MIBPB by far the only existing PBE solver that is able to deliver the second-order convergence, that is, the accuracy increases four times when the mesh size is halved. The MIBPB method is also equipped with a Dirichlet-to-Neumann mapping technique that builds a Green's function approach to analytically resolve the singular charge distribution in biomolecules in order to obtain reliable solutions at meshes as coarse as 1 Å — whereas it usually takes other traditional PB solvers 0.25 Å to reach similar level of reliability. This work further accelerates the rate of convergence of linear equation systems resulting from the MIBPB by using the Krylov subspace (KS) techniques. Condition numbers of the MIBPB matrices are significantly reduced by using appropriate KS solver and preconditioner combinations. Both linear and nonlinear PBE solvers in the MIBPB package are tested by protein–solvent solvation energy calculations and analysis of salt effects on protein–protein binding energies, respectively.

 J Comput Chem 00: 000–000, 2010

Key words: MIBPB; Poisson — Bottzmann equation; interface method; electrostatics; proteins

Introduction

Under physiological conditions, almost all important biological processes, for example, signal transduction, DNA specification, transcription, post-transcription modification, translation, protein folding, and protein–ligand binding, occur in water, which comprises 65–90% of cellular mass. An elementary prerequisite for the quantitative description and analysis of the above-mentioned processes is the understanding of solvation, which involves energetics of interactions between solute molecules and solvent molecules or ions in aqueous environment. Solute–solvent interactions are typically classified as the polar-type and the nonpolar-type. Although widely used, this classification is arbitrary and has caveats associated with the nonunique descriptions, as well as the intrinsic coupling between these two types of interactions. The polar-type of solute–solvent interactions is the main interest of this work. It originates from electrostatic effects, which play important roles in biophysics, biochemistry, structural biology, electrochemistry, and electrophoresis. The solvent has a substantial volume and a significant contribution to electrostatics via numerous mobile ions. However, it is the solvated solute molecule that is the focus of the interest in most research. As such, the solute is described in atomic or electronic details, whereas atomic details of the solvent and mobile ions are approximated by a mean-force description and probability distribution, respectively. This multiscale treatment, denoted as implicit solvent method, can greatly reduce the computational cost of the traditional explicit solvent methods, in which a microscopic description of the solvent is retained. Various implicit solvent models are available to describe polar solvation.[1–5] The most widely used methods are currently the generalized Born method,[4,6–9] polarizable continuum,[10–12] and Poisson–Boltzmann

*Present address: Department of Mathematics, University of Michigan, Ann Arbor, MI, USA.

Correspondence to: G.-W. Wei; e-mail: wei@math.msu.edu

Contract/grant sponsor: NSF; contract/grant numbers: DMS-0616704, CCF-0936830

Contract/grant sponsor: NIH; contract/grant numbers: GM-090208, CA-127189

equation (PBE)[2,3,13,14] models. The use of polarizable continuum models is mostly restricted to small molecular systems. Generalized Born methods are very fast but are only heuristic models for estimating polar solvation energies of biomolecular structures. These methods are often used in high-throughput applications, such as molecular dynamics simulations.[15–17] PBE models can be formally derived from Maxwell's equations[18] and offer a somewhat slower, but more accurate way for evaluating polar solvation properties.[19–21] Additionally, PBE techniques are often used to parameterize and assess the accuracy/performance of generalized Born models.[20,22] Finally, unlike most generalized Born methods, PB models provide a global solution for the electrostatic potential and field within and around a biomolecule, therefore make them uniquely suited to visualization and other analysis[23,24] that require global information about electrostatic properties. One of the primary quantitative applications of implicit solvent models in computational biology and chemistry is the calculation of thermodynamic properties via a pre-equilibration.[1] An example of such pre-equilibration approach is the MM/PBSA model,[25,26] which combines implicit solvent models with molecular mechanical approaches to evaluate binding free energies from an ensemble of biomolecular structures. Other important applications of implicit solvent models include the assignment of protein titration states, the calculation of binding energies, and the estimation of solvation energies.[27,28] One more application area for implicit solvent methods is the evaluation of biomolecular kinetics where implicit solvent models are generally used to provide solvation forces for molecular Langevin dynamics,[29,30] Brownian dynamics,[31–33] or continuum diffusion simulations.[34–36] A major qualitative use of implicit solvent methods in experimental work is the visualization and qualitative analysis of electrostatic potentials on and around biomolecular surfaces,[13,37,38] which is now a standard procedure for the analysis of biomolecular structures.

Although the PBE can be analytically solved for a few simple cases,[39] it relies on numerical approaches to obtain useful solutions for realistic biological systems. A vast variety of computational approaches, such as finite difference methods (FDMs),[30,40–42] finite element methods (FEMs),[43] and boundary integral methods (BIMs),[44,45] have been developed in the past few decades. Each of these methods is subject to certain inherent advantages and limitations due to its associated underlying formulations. In general, FEMs have advantages in applications that require the rapid adaptation of grid points to account for the structural variation of biomolecules. However, the generation of unstructured grids with good quality for complex biomolecular interfaces is very time-consuming, especially for large biomolecules. BIMs have several intrinsic advantages, such as fewer unknowns, exact far-field treatment, and good representation of surface geometry and charge singularity. In light of these advantages, BIMs, especially when accelerated with fast methods, such as treecode and fast multiple methods, can also provide an efficient computational approach. However, BIMs are not very efficient in dealing with the nonlinear term in the PB model. FDMs have been the main workhorse for solving the PBE in computational structural biology for their convenience of using 3D Cartesian coordinates to save the cost on mesh generation and electrostatic potential mapping, as well as its adaptability of existing linear algebraic solvers. FDMs-based PB solvers, particularly in conjunction with advanced linear algebraic solvers, can offer the best combination of accuracy and efficiency, therefore make them the most popular approaches in structural biology.[46]

Many computational technologies for the PBE were incorporated into popular molecular simulation software packages, such as DelPhi,[47] ZAP,[29] UHBD,[31] MEAD,[48] APBS,[49] AMBER,[30,50] and CHARMM.[51,52] These software packages deliver PBE solvers to users who are interested in the study of electrostatics in solution, making the PBE model a widely accepted approach in structural biology. The aim of this article is to introduce an interface technique-based PBE solver, the matched interface, and boundary method based-PB solver (MIBPB), with online software sharing information. Compared with other existing PB solvers, the MIBPB provides rigorous mathematical treatment of interface jump conditions, geometric singularities, and charge singularities as described in the following paragraphs.

The implicit solvent models require an interface definition to indicate the separation of solute atoms from the surrounding solvent. All of the physical properties of interests, such as electrostatic free energies, biomolecular surface areas, molecular cavitation volumes, solvation free energies, and pK_a values are very sensitive to the interface definition.[53–55] The van der Waals surface, the solvent accessible surface,[56] and the molecular surface (MS)[57,58] are often used for this purpose. Different dielectric constants of the solvent and molecular domain lead to discontinuous coefficients in the PBE, resulting in nonsmoothness of the solution. Additionally, the MS admits geometric singularities, such as cusps and self-intersecting surfaces.[58,59] Explicit interface treatment of geometric singularities has not been considered in the popular PB solvers. It is desirable to have an interface-based solver that is able to deliver highly accurate solutions to the PBE in the presence of such solution and geometric singularities. The third type of singularity comes from the singular charge terms, that is, the delta functions at the right hand side of the PBE, often reduce the accuracy of the numerical solution. The treatment of charge singularities is also an important issue in solving the PBE. Finally, because of the complex matrix structure of the linear systems resulting from PBE solvers, the choice of appropriate linear system solvers and the selection of matrix acceleration algorithms are very important issues as well.

The development of the MIBPB solver focuses on resolving the above-mentioned difficulties or singularities. The adoption of the molecular surface, the treatment of discontinuity of coefficients, and flux jumps require the application of interface methods.[60–65] Commonly seen interface techniques were not used in the biomolecular context because of the complexity of the biomolecular boundaries. The MIB method[66–70] has been developed for solving elliptic equations with discontinuous interfaces. It is of arbitrarily high-order accuracy in principle and up to sixth-order accurate MIB schemes have been constructed.[67,70] The MIB has been successfully applied to the analysis of mechanical structures,[71,72] waveguides,[72] biomedical imaging,[73] and electromagnetic waves.[74] Three generations of MIB-based PB solvers, the MIBPB-I,[75] the MIBPB-II,[76] and the MIBPB-III[77] have been developed (http://www.math.msu.edu/~wei/MIBPB/). The MIBPB-I is the first PB solver that directly enforces the flux continuity conditions at the dielectric interface in the biomolecular context. However, it cannot maintain its designed order of accuracy in the presence of MS singularities, such as cusps and self-intersecting surfaces. This problem was addressed in the MIBPB-II by using an advanced MIB technique developed by Yu et al.,[70] who offered special treatments for geometric singularities.

However, the MIBPB-II loses its accuracy when the mesh size is as large as half of the smallest van der Waals radius, because of the interference of the interface and singular charges. To split the singular charge part of the solution, a Dirichlet to Neumann mapping approach[78] was designed in the MIBPB-III, which is by far the most accurate and reliable PB solver. This new solver remains accurate at the smallest van der Waals radius, that is, about 1.0 Å grid resolution for proteins. Comparing with traditional PB solvers, the MIPPB-III is a few orders of magnitude more accurate at a given mesh size and about three times faster at a given accuracy.[76,77] The MIBPB is the first and still the only known second-order convergent PB solver for the singular molecular surfaces of biomolecules, where the second-order convergence means that the accuracy of the solution improves four times when the mesh size is halved.

Apart from the accuracy, the efficiency of linear system solvers is another important issue crucial to many applications. Previous MIBPB solvers are typically slow in comparing with other FDMs that do not invoke an interface treatment. In this article, we have paid special efforts on the strategies for the selection of most suitable linear system solvers for the resulting MIBPB matrices. Two linear solver libraries, the SLATEC (http://people.sc.fsu.edu/~burkardt/f_src/slatec/slatec.html) and the PETSc (http://www.mcs.anl.gov/petsc/petsc-as/) are considered in the exploration of linear solvers. Another remaining issue in previous MIBPB solvers is the treatment of the nonlinearity in the PBE. Although this issue was tackled in a dissertation,[79] no reliable MIBPB nonlinear solver was produced. This work develops a reliable nonlinear solver for the PBE with a salt solvent.

The rest of this article is organized as follows. "Theory and Algorithm" section is devoted to the theoretical formulation and computational algorithm of the MIBPB solver. Krylov subspace (KS) technique accelerated MIBPB solvers are constructed in "Preconditioner Accelerated MIBPB Solvers" Section. Extensive experimental validation is given to the different combinations of solvers and preconditioners. "Usage Illustration and Application" section illustrates the usage of the MIBPB software package with some example applications, such as calculating the free energy of solvation of biomolecular systems and salt effect on protein–protein binding. Concluding remarks are provided in "Conclusion" section. Finally, a brief introduction to the linear algebraic systems generated from the MIBPB discretization matrices and theoretical underpinnings of the KS methods are presented in Appendices.

Theory and Algorithm

Implicit Solvent Model: The PBE

In the implicit solvent model, the solvent is treated as a continuous medium while the description for solute is kept at the atomic level. The electrostatic potential ϕ of a solvent–solute system can be determined by the PBE in a regular domain Ω whose dimension usually has the order from 10 Å^3 to 500 Å^3 for biomolecular applications. Figure 1 gives the sketch of the protein–solute system and the computational domain.

The protein region and the solvent region are denoted as Ω_1 and Ω_2, respectively. Naturally the whole computational domain is $\Omega = \Omega_1 \bigcup \Omega_2$, and the molecular surface is labeled as Γ. For simplicity, the ion-exclusive layer is ignored in the present model.

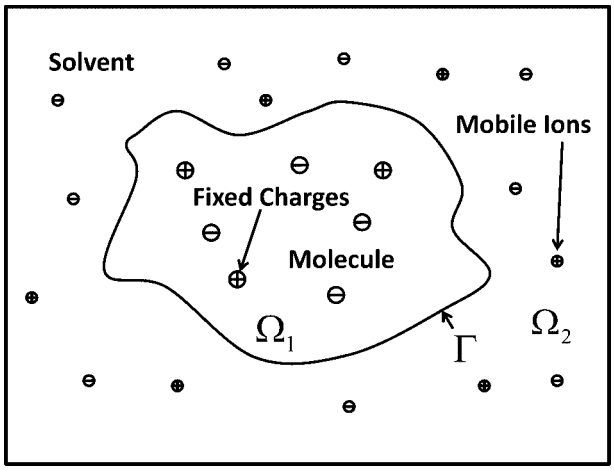

Figure 1. The implicit protein–solvent system.

Although mobile ions in the solvent are explicitly indicated in the figure, the whole solvent region is actually modeled by an implicit continuum. Under these assumptions, the PBE reads

$$-\nabla\cdot(\epsilon(\mathbf{x})\nabla u(\mathbf{x})) + \overline{\kappa}^2(\mathbf{x})\sinh(u(\mathbf{x})) = C\sum_{i=1}^{N_m} q_i\delta(\mathbf{x}-\mathbf{x}_i), \quad \mathbf{x}\in\Omega\subset\mathbb{R}^3 \tag{1}$$

$$u(\mathbf{x}) = \frac{C}{4\pi\epsilon_s}\sum_{i=1}^{N_m}\left[q_i e^{-\overline{\kappa}^2(\mathbf{x})|\mathbf{x}-\mathbf{x}_i|/\sqrt{\epsilon_s}}\right]/|\mathbf{x}-\mathbf{x}_i|, \quad \mathbf{x}\in\partial\Omega, \tag{2}$$

where $u = e_c\phi/k_BT$ is the dimensionless electrostatic potential for computational simplicity, e_c the electron charge, k_B is the Boltzmann constant and T is the temperature, ϵ and $\overline{\kappa}$ are dielectric constant, and modified Debye-Hückel screening function describing the ion strength, respectively. Here q_i is the charge fraction of the fixed charge in the protein and $\mathbf{x}_i$ denotes the position of the fixed charge, and N_m is the total number of fractional charges. The constant $C = 4\pi e_c^2/k_BT$ is resulting from dimensionless procedure and ϵ_s is for the dielectric constant of the solvent.

In principle, the electrostatics $u(\mathbf{x})$ satisfies the boundary condition at infinity, that is:

$$u(\infty) = 0. \tag{3}$$

However, the practical computation has to be restricted onto a bounded domain Ω. Usually, it is taken as a cuboid that contains the target protein and $\partial\Omega$ represents the boundary. In this approximation, proper boundary conditions need to be imposed and various treatments are used upon different numerical schemes. Equation (2) describes the Dirichlet boundary condition, which is widely used in finite difference method. The biological meaning of the rationale is the (screened) Coulomb potential originating from all the fixed charges and mapping on the walls of the cuboid Ω.

The hyperbolic term $\sinh(u(\mathbf{x}))$ takes into account the salt effect with the Boltzmann distribution theory at the equilibrium state. Therefore, eq. (1) is a nonlinear partial differential equation (PDE) of

elliptic-type. Such a nonlinear term can be linearized under the weak potential approximation, that is, when $u(\mathbf{x}) \ll 1, \sinh(u(\mathbf{x})) \sim u(\mathbf{x})$. Thus, the linear approximation of eq. (1) is

$$-\nabla\cdot(\epsilon(\mathbf{x})\nabla u(\mathbf{x}))+\overline{\kappa}^2(\mathbf{x})u(\mathbf{x}) = C\sum_{i=1}^{N_m} q_i\delta(\mathbf{x}-\mathbf{x}_i),\quad \mathbf{x}\in\Omega\subset\mathbb{R}^3. \tag{4}$$

Typically, for biomolecular systems of given ranges of temperature and ionic strength, the PBE is solved with the following coefficient bounds[80]

$$\begin{aligned} 1 &\le \epsilon(\mathbf{x}) \le 80\\ 0 &\le \overline{\kappa}^2 \le 127\\ 5249 &\le C \le 10500\\ -1 &\le q_i \le 1. \end{aligned}$$

The molecular surface of the solute (protein) is considered as an interface. We partition the whole domain into the solute region and the solvent region, on which the $\epsilon(\mathbf{x})$ takes different values. Similar situation happens to the parameter $\overline{\kappa}(\mathbf{x})$. As it represents the ionic strength of the solvent, $\overline{\kappa}(\mathbf{x})$ is nonzero in the solvent region but zero in the solute region. In other words, the spatial-dependent coefficients $\epsilon(x)$ and $\overline{\kappa}(\mathbf{x})$ are discontinuous across the molecular surface. It is a challenge to solve such an elliptic equation with high accuracy because the regularity of its solution is reduced due to the interface and geometric singularity. For this class of problems, numerical accuracy and convergence rate are typically low without special interface treatments. Another challenge is the singular source term which contains many Delta functions, which are infinity at their spatial locations. Accurate approximation to the point-supported singular functions is an important topic in computational mathematics. The above two difficulties hinder the accurate numerical solution to the PB equation. To maintain a given accuracy, the grid spacing of the discretization has to be sufficiently small because of the low regularity of the solution. On the other hand, a small grid spacing implies millions of variables even for a middle-size protein. For example, the cube containing a 2800-atom protein may have a dimension of $50\times 50\times 50$ (Å^3), which leads to 1×10^6 variables if the resolution is 0.5 Å. This gives rise to a major obstacle for PB applications, especially for the calculation of thermodynamic properties via either the molecular dynamics or pre-equilibrium approaches. In the following sections, a robust and efficient mathematical algorithm, the MIB method, is introduced and applied to the solution of the PBE.

MIB Method for the PBE

Mathematically, the PBE admits at least three types of singularities that hinder one's attempt to obtain highly accurate numerical solutions. The first one is the nonsmooth solution singularities. Based on the statement in the previous section, the potential $u(x)$ is naturally restricted to the two subregions as u^1 and u^2, respectively. The solution of the PBE is subject to the interface continuity condition and flux continuity condition along the interface Γ

$$[u]|_\Gamma = u^1|_\Gamma - u^2|_\Gamma = 0 \tag{5}$$

$$[\epsilon u_{\mathbf{n}}]|_\Gamma = \epsilon_m u^1_{\mathbf{n}}\big|_\Gamma - \epsilon_s u^2_{\mathbf{n}}\big|_\Gamma = 0, \tag{6}$$

where $u_{\mathbf{n}}$ denotes the normal derivative of the function $u(\mathbf{x})$ and ϵ_m is the dielectric constant of the solute molecule. Interface conditions (5) and (6) indicate that the continuity of the potential function and the potential flux across the molecular surface. However, different dielectric values of ϵ_s and ϵ_m in (6) imply that the normal derivatives of the potential function differ from each other across the surface — in other words, the solution is nonsmooth. The second singularity comes from the solvent–solute interface Γ. The interface generated based on the current molecular surface model inevitably introduces geometric singularities, such as cusps and self-intersecting surfaces, especially for large proteins. This singularity has been a challenging issue to many traditional interface techniques designed for solving elliptic interface problems. The third singularity is the source term of the PBE, which includes the summation of Delta functions. Second-order numerical implementation of Dirac delta functions on the Cartesian grid points is feasible with appropriate approximation. However, the overlap of grid points carrying redistributed charges and those involved in the treatment of geometric interface singularities leads to the accuracy reduction, especially for a coarse mesh. These singularities pose challenges in the numerical implementation of the PBE and make it difficult to balance numerical accuracy and efficiency.

The MIB scheme and the Dirichlet-to-Neumann mapping (DNM) method are used to deal with the above-mentioned three singularities and to achieve high accuracy and efficiency. In the MIB method, the molecular surface is considered as the interface. Interface jump conditions (5) and (6) are enforced at each intersecting point of the interface and mesh lines. As such, the ingredients of the MIB scheme also include local coordinates of the interface to overcome geometric singularity constraints. The necessary information of each local coordinate contains the location (x_0, y_0, z_0), and the normal direction $\mathbf{n}$ of the intersecting point where the interface meets a mesh line of the grid. Here, $\mathbf{n}$ is parameterized as

$$\mathbf{n} = (\sin\psi\cos\theta, \sin\psi\sin\theta, \cos\psi)^T \tag{7}$$

where θ and ψ are the azimuth and zenith angles with respect to the normal vector, notation T represents the transpose for a vector. This information allows us to set up a local coordinate system at every intersecting point and to define its relation to the global Cartesian grid.

The basic idea of the MIB scheme is to define sets of regular and irregular grid points near the interface, according to the desired convergence order. At each regular point, the standard central difference scheme of a given order is applied, whereas at irregular points, special treatments are to be taken. First, the restricted potential function u^1 (and u^2) is smoothly extended from one subregion to another on all of the irregular points. Then artificially introduced function values, named as fictitious values, are defined on the irregular points. As smooth extensions of the function, fictitious values are used in FD schemes at irregular points to guarantee necessary smoothness conditions.

Interface condition (5) is further decomposed into two parts to give flexibility to the calculation of fictitious values

$$[u_{\tau_1}] = u^1_{\tau_1}\big|_\Gamma - u^2_{\tau_1}\big|_\Gamma = 0$$
$$[u_{\tau_2}] = u^1_{\tau_2}|_\Gamma - u^2_{\tau_2}|_\Gamma = 0, \quad (8)$$

where τ_1 and τ_2 are the two tangential vectors derived from (7)

$$\tau_1 = (-\sin\theta, \cos\theta, 0)^T$$
$$\tau_2 = (-\cos\psi\cos\theta, -\cos\psi\sin\theta, \sin\psi)^T. \quad (9)$$

Through jump conditions (5)–(8), fictitious values are determined as the combinations of the to-be-determined numerical solutions on grid points. Although a pair of fictitious values are determined along a mesh at a time, an iterative procedure can be used to determine all the required fictitious values for higher order schemes by repeatedly using the lowest order jump conditions. The essential strategy of the MIB method is to locally reduce a 2D or a 3D interface problem into 1D-like ones.[66–68,75] The essence of the MIB scheme is the use of fictitious values, which entirely assemble the interface conditions, the local geometry of the interface and all the necessary interface information. Therefore, the MIB scheme is very robust for different protein surfaces and successfully overcomes the first two types of singularities.[69,70,76] Rigorous second-order MIB schemes have been developed to solve the PBE with geometric singularities of molecular surfaces.

The source term singularity is removed by the DNM. In ref. 77, the solution $u(\mathbf{x})$ of the PBE is decomposed into a singular part and a regular part. The singular part of the solution comes from singular delta functions, and is obtained analytically as the Green's function. As a consequence, this separation generates an extra Neumann jump condition at the interface for the regular part. Therefore, after the separation, one only needs to solve the remaining homogeneous PBE subject to corresponding Neumann jump conditions at the interface. This procedure is called Dirichlet to Neumann mapping. Consequently, truly second-order accurate solution to the PBE with molecular surfaces and singular charges can be obtained with a relatively large grid spacing.[77]

Preconditioner Accelerated MIBPB Solvers

In practice, when one pursues the numerical solution of the PBE, the discretization of the PBE results in a linear equation system

$$L_h U_h = f_h \quad (10)$$

where h is the discretization resolution, L_h and f_h represent the matrix and right hand side generated via the MIB and DNM schemes, U_h is the solution vector. It is worth pointing out that in standard FDMs, the matrix L_h only depends on the grid resolution and the dielectric constants. However, in the MIBPB scheme, the structure of L_h also depends on the molecular surface of a specific protein. Because of this reason, we also call L_h the matrix of a protein for simplicity. The MIB and DNM successfully overcome the equation singularities and promise a high-accuracy convergence order by taking into account all the local interface information. However, as a trade-off, the structure of matrix L_h is much more complicated than that from standard FDMs. Specifically, the matrix loses symmetry and may not be positive-definite any more. The lose of these properties will lead to more computational time and memory. Therefore, the selection of appropriate linear solvers becomes subtle when computational efficiency is sought as well.

The review of several basic linear solvers are summarized in Appendix A. However, the matrices from the MIB and the DNM scheme can barely take any advantage from the described methods due to their notoriously complicated structures. Therefore, we put our emphasis on choosing other methods and accelerating techniques. In Appendix B, we include a brief description of KS techniques. Based on the KS theory, proper linear solvers and acceleration techniques (preconditioners) are chosen and compared in this section for the numerical efficiency of MIBPB linear systems. Two KS solvers, the stabilized biconjugate gradient method (BiCG) and the generalized minimal residual method (GMRES), are potentially effective iterative solvers for matrices with general structures. Several preconditioning strategies, the Jacobi preconditioner (JAC), the blocked Jacobi preconditioner (BJAC), and the incomplete LU factorization preconditioner (ILU) are available to incorporate with the two solvers to accelerate the solution of the linear system.

Matrices generated from a set of proteins are used to test the performance of various KS solver-preconditioner (PC) combinations. For each matrix, the condition number, linear system iteration number, and iteration time are used to characterize numerical efficiency. All these measurements of matrices are analyzed numerically by the PETSc (http://www.mcs.anl.gov/petsc/petsc-as/). The grid resolution is taken as 1.0 Å in the following tests unless otherwise specified. The stopping criterion of all KS solvers are taken as 1×10^{-6} to get more accurate solutions, whereas in practical biological applications the criterion can be relaxed to 1×10^{-3} to save CPU time but satisfactory results are also achieved.

First of all, the matrix condition numbers are examined. The condition number can predict the level of difficulty in solving the linear system before it is really solved. The magnitude of a matrix condition number depends on the size and structure of a biomolecule. More specifically, under the same grid resolution, a molecule, which has a larger number of atoms, needs a larger computational domain and a larger matrix size. Meanwhile, a molecule, which has a more complex surface geometry, leads to more involvement of interface conditions and consequentially less sparse matrix. Both cases contribute to higher condition numbers. Therefore, the size and complexity of a biomolecule usually affect the numerical efficiency of the MIBPB solver.

Figure 2(a) presents condition numbers of matrices corresponding to 15 protein structures and indicates the numerical difficulties of solution without proper acceleration techniques. The horizontal axis lists proteins. Protein data bank (PDB) identification numbers (IDs) are listed in the figure. The numbers of atoms of these proteins range from 500 to 2000. Discretizing the PBE with the MIB scheme, without any PC applied, the condition numbers are usually in the order of 10^4, about one order larger than those of the matrices generated from the standard FD discretization, that is, without the interface treatment. This is expected because the use of the molecular surface as the interface and all included local information around the interface, the MIBPB matrices do not maintain the symmetry and are not

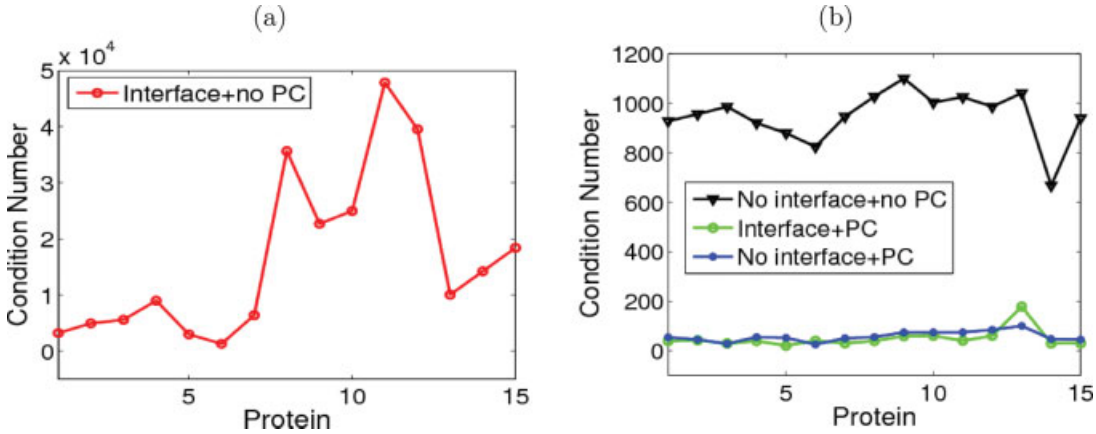

Figure 2. Condition numbers over 15 proteins (PDB IDs from protein 1 to protein 15: 1ajj, 1vii, 1cbn, 1bbl, 1fca, 1sh1, 1vjw, 1fxd, 1bpi, 1a2s, 1frd, 1svr, 1a63, 2erl, and 2pde). (a) Condition numbers for unpreconditioned (unPCed) MIBPB matrices; (b) Comparisons of condition numbers under three settings.

positive definite. The MIB matrices generally have larger condition numbers and require more CPU time.[75–77]

By using of PCs, the magnitudes of condition numbers of MIBPB matrices are significantly reduced to less than one hundred, as shown in the circle plot of Figure 2(b). The triangle plot in Figure 2(b) gives the condition number magnitudes of the matrices from standard FDMs without PC, revealing the huge differences among different treatments. The circle and dot plots are condition number magnitudes for matrices with PC, from both the MIB scheme and the FDM, respectively. Interestingly, it can be concluded that the condition number magnitudes of two schemes are reduced to almost the same level by using the PCs. We can safely say that the difficulty of solving the linear system generated from the interface-based MIBPB scheme is actually comparable with that from standard FD discretization. Under almost the same numerical efficiency, MIB scheme and DNM are able to obtain higher accuracy because all the local geometry information of the molecular surface has been taken into account.

Quantitatively, for a specific KS solver, such as GMRES, the iteration numbers, and computing time of linear systems for seven proteins are listed in Table 1. It is well-known that condition numbers can only be mathematically estimated for large matrices, then the listed condition numbers calculated by PETSc solvers may not be exact. Despite this fact, we can still have a sense from the numbers how the PC significantly reduces the difficulties of solving the linear systems.

As stated earlier, two KS iterative methods, the stabilized BiCG and the GMRES, are associated with three types of PCs, JAC, BJAC, and ILU. Table 2 compares the effect of combinations of these KS solvers and PCs. For different preconditioning strategies, as the ways of counting iteration numbers are different, only the iteration time for each combination is listed in the table. Sample proteins of various sizes are presented in this table, from small size (less than 1000 atoms), middle size (1000–3000 atoms) to large size (around 8000 atoms). It can be concluded that the GMRES performs slightly better than the stabilized BiCG does for small-sized proteins but stabilized BiCG take the lead in middle- and big-sized proteins. Among the three kinds of PCs, the BJAC and the ILU almost have the same effects and are slightly better than the JAC. Therefore, the combination of stabilized BiCG and BJAC is recommended and set as the default option in the MIBPB package.

As indicated at the beginning, all the mathematical algorithms and techniques are enforced to maintain the high-order convergence of the MIBPB solver. Table 3 lists the numerical evidence of the second-order convergence through a set of given protein

Table 1. Iteration Numbers and CPU Time for the Discretization Matrices of Proteins.

Proteins		Preconditioned iteration			Unpreconditioned iteration		
ID	Atoms	Number	Time	Condition number	Number	Time	Condition number
1mbg	903	19	0.3	40	5404	54	118900
1r69	997	18	0.3	40	5400	58	250400
1bor	832	20	0.3	30	2152	23	138850
1vii	596	17	0.2	42	3963	28	4963
1fxd	824	19	0.3	39	7084	80	35637
2erl	573	17	0.2	29	4858	36	14223
1a2s	1272	23	0.6	61	10000	156	24981

Table 2. Iteration Time for Different Combinations of KS Solvers and Preconditioners.

Protein		BiCG			GMRES		
ID	Atom	BJAC	ILU	JAC	BJAC	ILU	JAC
1ajj	519	0.24	0.24	0.26	0.23	0.23	0.38
1vjw	828	0.29	0.29	0.35	0.26	0.26	0.44
1a2s	1272	0.56	0.56	0.51	0.57	0.56	0.84
1a7m	2809	1.69	1.67	1.77	1.93	1.91	2.85
1f6w	8243	3.90	3.88	4.48	4.70	4.65	7.19

Table 3. Convergence Test of MIBPB Solver with a Set of Proteins.

Proteins ID	Error $h = 1.0$ Å	$h = 0.5$ Å	Order	Error $h = 0.25$ Å	Order
1ajj	6.52E–2	1.13E–2	2.52	1.76E–3	2.68
1a23	1.026E1	1.72E–1	2.57	2.74E–2	2.65
1b4l	1.19E–1	1.25E–2	3.25	2.07E–3	2.59
1bbl	1.32E–1	1.86E–2	2.82	1.81E–3	3.36
1bor	9.44E–2	1.31E–2	2.84	1.97E–3	2.73
1fca	1.20E–1	1.20E–2	3.31	1.78E–3	2.76
1frd	7.93E–2	1.24E–2	2.67	2.02E–3	2.61
1fxd	7.66E–2	1.19E–2	2.68	2.00E–3	2.57
1hpt	8.05E–2	1.37E–2	2.50	1.77E–3	2.90
1mbg	1.35E–1	1.08E–2	3.64	1.69E–3	2.67
1neq	8.52E–2	1.27E–2	2.74	1.83E–3	2.79
1r69	7.95E–2	1.15E–2	2.39	1.92E–3	2.96
1svr	7.94E–2	1.17E–2	2.21	1.94E–3	3.13
1uxc	7.55E–2	1.27E–2	2.57	2.02E–3	2.65
1vjw	7.22E–2	1.20E–2	2.58	2.23E–3	2.43
2pde	1.12E–1	1.64E–2	2.77	5.46E–3	1.58

surfaces, atomic coordinates, radii, and charges, where protein surfaces are generated by MSMS, and the standard CHARMM force field parameters are used. A special analytical solution was designed and given in ref. 77 for the convergence order check of all proteins. In this table, the numerical error is defined as $\|u_h^{\text{num}} - u^{\text{exact}}\|_{L_\infty}$, where u_h^{num} is the numerical solution of the PBE at grid resolution h, whereas u^{exact} is the designed exact solution. The numerical experiments are implemented under resolutions $h = 1.0$ Å, 0.5 Å, and 0.25 Å. The numerical error is supposed to be reduced by four times as the grid size is halved and this is clearly demonstrated in the table.

The above-mentioned tests are carried out in conjunction with the PETSc software package, whose installation may not be so straightforward. An alternative is to use the SLATEC, which is easier to implement and also includes tens of linear system solvers with different PCs. To compare the performance of the PETSc and the SLATEC, we show the computation time of ten methods in the SLATEC for five proteins, whose atom number varies from 500 to 8000 in Table 4. All methods are listed in the form: PC/solver. Here GS, DS, BiGS, and OM represent the Gauss-Seidel, the diagonal scaling, the biconjugate gradient squared method, and the orthomin sparse iterative method, respectively. The combination of the ILU/BiCG is used in the PETSc. From the table, it can be seen that the iteration time of the PETSc is slightly shorter than that of most solvers in the SLATEC for small-sized proteins. The last column of the table lists the averaged CPU time for the PETSc and solvers in SLATEC. The averaged time, which in some sense could reflect the abilities of solvers for proteins in various sizes, is the sum of the CPU time for each corresponding protein and weighted by the atom number. By checking the averaged CPU time one can generally conclude that the ILU/BiCG of the PETSC takes less iteration time than the SLATEC schemes do. Moreover, according to our experience, the PETSc is more stable than the SLATEC for large proteins. However, the SLATEC can be easily incorporated in the MIBPB package. Whereas, the PETSc needs to be preinstalled by the user as discussed below.

Usage Illustration and Application

Work Flow of the MIBPB Package

The MIBPB solver package incorporates with two packages to accomplish the electrostatic potential calculation. First, molecular structures are prepared via Python software package PDB2PQR (http://pdb2pqr.sourceforge.net/): it accomplishes many common tasks of preparing structures for continuum electrostatic calculations, such as adding a limited number of missing heavy atoms to biomolecular structures, determining side-chain pK_as, placing missing hydrogens, so forth. Users can either submit the protein PDB index to the online server (http://pdb2pqr.sourceforge.net/) or download the executable file to prepare the molecular structure.

Once the molecular structure is prepared, the computational domain Ω will be automatically generated based on the coordinates of the protein atoms: first a smallest cuboid that contains the protein will be calculated and then each length of the cuboid is symmetrically extend at two ends by 5 to 10 Å, depending on the protein size. This strategy usually used in many FDMs is verified to be reasonable in practices and also the extension of the cuboid can be customized easily. The larger size of Ω is of course closer to real biological situation. However, the solution of the PBE is not sensitive to this change while the computational cost will be increased.

Additionally, the geometry of the molecular surface used in the MIB scheme is generated by the MSMS (http://www.scripps.edu/ sanner/html/msms_home.html). Given the information of the coordinates and radius of each atom in the molecule, surfaces are generated at given water probe radius in a triangulation form. The intersection of each triangle with the meshing lines and the normal direction extracted from the surface information are key ingredients of the MIBPB scheme. For the MSMS parameter, the water molecule probe radius is recommended as 1.4 and the vertex density is 10. These parameters are enough to generate the molecular surface with good quality, various 3D Cartesian grid resolutions in current use can obtain necessary surface information under this setting.

There are two options for choosing KS solvers and PCs in solving MIBPB matrices. One is to use the SLATEC, which has been incorporated in our MIBPB package. The other way is to use the PETSc. According to our tests, the PETSc is generally more stable and reliable than the SLATEC, particularly for large proteins. It

Table 4. Comparison of CPU Time for the PETSc and the SLATEC Schemes.

Protein ID Atoms	1ajj 519	1vjw 828	1a2s 1272	1a7m 2809	2ade 8344	Averaged CPU time
PETSc	0.235	0.272	0.529	1.729	3.777	2.72
GS/GS	0.866	1.222	2.225	9.512	55.016	35.58
ILU/ILU	0.523	0.883	1.344	5.854	32.479	21.07
DS/BiCG	0.331	0.467	1.041	3.140	14.015	9.27
ILU/BiCG	0.262	0.401	0.701	2.038	7.846	5.27
DS/BiGS	0.243	0.313	0.602	2.900	8.879	6.05
ILU/BiGS	0.187	0.393	0.410	1.433	6.575	4.34
DS/OM	0.206	0.420	0.496	3.338	21.993	14.08
ILU/OM	0.179	0.291	0.389	1.25	5.993	3.95
DS/GMRES	0.417	0.559	0.999	3.856	26.262	16.84
ILU/GMRES	0.198	0.279	0.439	1.615	7.685	5.05

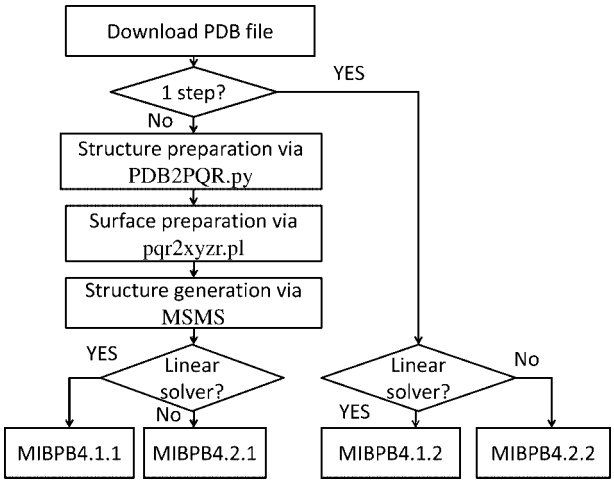

Figure 3. Work flow of the MIBPB package.

needs to be preinstalled by the user if one chooses the PETSc matrix acceleration option.

The current MIBPB package offers half stand-alone solvers in which users have to prepare the molecular structures and generate the surfaces on their own with desired parameters. The package also has one-step solvers, which have integrated all the steps with default parameter settings. Either the half stand-alone or one-step solver is further classified into linear solver and nonlinear solver. Therefore, there are in total four executable MIBPB files in the package. Additionally, two other small auxiliary Perl scripts, the pqr2xyzr.pl and dat2dx.pl are included in the MIBPB package to accomplish the molecular surface preparation and ultimate data visualization. Figure (3) is the work flow of the MIBPB package usage. Users are referred to http://www.math.msu.edu/~wei/MIBPB/ for detailed instructions.

For a clearer demonstration, we use one specific protein example to illustrate the procedure. Protein with ID 1ajj is assumed to have been downloaded from the PDB and saved as file 1ajj.pdb.

1. Prepare the protein structure
   - Input file: 1ajj.pdb
   - Command line: python pqd2pqr.py –ff=CHARMM 1ajj.pdb 1ajj_apbs.pqr
   - Output file: 1ajj_apbs.pqr.
   - Remark: For full usage of pqb2pqr.py, users are referred to the corresponding link.
2. Molecular surface preparation
   - Input file: 1ajj_apbs.pqr
   - Command line: pqr2xyzr 1ajj
   - Output files: 1ajj.xyzr, 1ajj.pqr
   - Remark: 1ajj.xyzr file stores the coordinates and radii of the atoms in the protein, 1ajj.pqr stores the coordinates and partial charges. They are necessary files for the MSMS to generate molecular surfaces.
3. Molecular surface generation
   - Input files: 1ajj.xyzr, 1ajj.pqr
   - Command line: msms -if 1ajj.xyzr -prob 1.4 -de 10 -of 1ajj
   - Output files: 1ajj.vert, 1ajj.face. Now the molecular surface is generated in the triangulation form. The vertices and normal direction of each triangle are saved in these files.
   - Remark: water probe radius and triangulation density are set as default values 1.4 and 10, respectively. They are adjustable parameters.
4. MIBPB implementation
   - Linear solver: mibpb4.1.1 1ajj eps1=1 eps2=80 h=0.5
   - Nonlinear solver: mibpb4.2.1 1ajj eps1=1 eps2=80 kappa=1.0 h=0.5
   - Output file: 1ajj_pbe.dat
   - Remark: Above command lines give the standard format. Parameters are adjustable.

Work Flow for the Display of the Surface Electrostatic Potential

After the electrostatic potential file is obtained by running the MIBPB solver, we can display it on the molecular surface by using the VMD (http://www.ks.uiuc.edu/Research/vmd/), a molecular visualization program. We are able to visualize the potential distribution on the surface by implementing a file transformation via the Perl script dat2dx.pl. Moreover, by taking the difference of surface electrostatic potentials under different grid resolutions h, we are also able to check the convergence of the solutions and therefore suggest a proper grid resolution for balancing high numerical accuracy and efficiency. The procedure is shown as the following.

1. Visualization file preparation.
   - MIBPB package generates output file [pdbname]_pbe.dat, in which the electrostatic potentials on grid points of the protein–solvent system are stored. Before displaying the electrostatic potential on the molecular surface, one needs to use dat2dx.pl script to transform the data file to the [pdbname].dx file.
   - For example, for protein 1ajj, one gets 1ajj_pbe.dat file from the MIBPB package. Then use the command: dat2dx.pl 1ajj [dcel] [xleft] [xright] [yleft] [yright] [zleft] [zright] where [dcel] is the mesh size (we assume a uniform mesh). Here, [xleft],[xright], [yleft], [yright], and [zleft] and [zright] prescribe the span of computational domain in x, y, and z direction, respectively. Here, xleft, xright, yleft, yright, zleft, and zright should be the same as those used in calculating the potential.
2. Molecular surface drawing
   - Load the PDB data file into the VMD
   - Set drawing parameters in the Graphical Representation window: choose the "Volume" option for coloring method and the "Surf" option for drawing method.
3. Surface electrostatic potential drawing
   - Load the [pdbname].dx format potential file into the VMD. In the molecular file browser window, load [pdbname].dx file for the same protein as that in molecular surface (instead of for new molecular).
   - Set drawing parameters in the same Graphical Representation window as that in the second step. Choose the "Volume" option for coloring method and the "Surf" option for drawing method. Adjust the Color Scale Data Range to see different color effects.

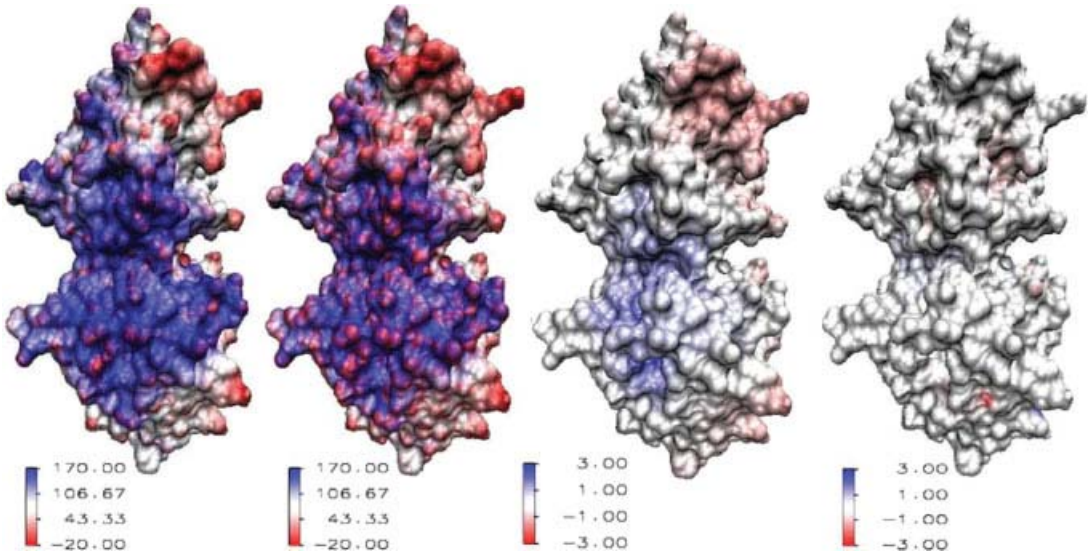

Figure 4. Visualizations of surface electrostatic potentials of protein 1beb. From left to right: (a) Surface electrostatic potential ($I = 0, h = 1.0$ Å); (b): Surface electrostatic potential with the ionic strength $I = 1$ ($h = 1.0$ Å); (c): The difference of surface electrostatic potentials between in an ionic solvent ($I = 1$) and in a water solvent ($I = 0$); (d): The difference of surface electrostatic potentials in water ($I = 0$) between grid meshes $h = 1.0$ Å and $h = 0.5$ Å.

Figure 4 illustrates the visualization of electrostatic potential calculated from the MIBPB package, using protein 1beb as an example. The potentials calculated via both the linear MIBPB solver and the nonlinear MIBPB solver are plotted on the molecular surface via the VMD through the above procedure. Figure 4(a) displays the potential distribution on the surface of protein 1beb when the solvent is water. In this case, the linear MIBPB solver is implemented because $\bar{\kappa}$ is set as zero. Although Figure 4(b) presents the potential distribution when $\bar{\kappa}^2 = 8.48$, in which case the nonlinear MIBPB solver is used. These two calculations are carried out when grid resolution h is taken as 1.0 Å. Figure 4(c) gives the difference of electrostatics in (a) and (b), from which the salt effect on electrostatic distribution may be observed. Figure 4(d) reveals the potential difference in solvent when the calculations are under resolutions $h = 1.0$ Å and 0.5 Å, that is, the error $|\phi_h - \phi_{h/2}|$. It can be found that the error is almost zero around the molecular surface, this fact indicates that at $h = 1.0$ Å, the result is accurate enough so that reducing grid resolution to 0.5 Å does not give too much improvement. Mathematically speaking, the result is almost convergent between mesh size 1.0 and 0.5 Å, which is the recommended grid resolution range in the MIBPB package.

Application to Solvation Energy Calculations

One of the most important applications of the PBE model is solvation energy calculations for the protein–solvent systems. In this section, solvation energies of 28 proteins are calculated and compared with popular PBE solvers to examine the feasibility, usefulness, and robustness of the linear solver in the MIBPB package. These proteins have a wide range of numbers of atoms, from around 500 up to 10,000. The corresponding spatial dimensions extend from about 30 Å × 30 Å × 30 Å to almost 100 Å × 100 Å × 100 Å. Among these calculations, the dielectric constant is set to 1 for proteins and 80 for the solvent. The ion strength $\bar{\kappa}$ is set to zero because no ion is considered for the moment.

The calculation of electrostatic solvation energy ΔG_{elec} is to sum all the fixed charges $\{q_i\}$ of the solute in the solvent, weighted by the reaction field potential $\phi_{\text{rf}}(\mathbf{x})$:

$$\Delta G_{\text{elec}} = \frac{1}{2}\sum_i q_i \phi_{\text{rf}}(\mathbf{x}_i) \tag{11}$$

where $\mathbf{x}_i$ is the position of each charge. Based on the continuum electrostatics, the reaction field potential is the difference between the electrostatic potential in the solvent environment $\phi_s(x)$ and the reference electrostatic potential $\phi_{\text{ref}}(\mathbf{x})$, that is, $\phi_{\text{rf}}(\mathbf{x}) = \phi_s(\mathbf{x}) - \phi_{\text{ref}}(\mathbf{x})$. Here, $\phi_{\text{rf}}(\mathbf{x})$ can be computed by solving the PBE twice with corresponding settings. Specifically, $\phi_{\text{ref}}(\mathbf{x})$ is calculated by setting a uniform dielectric constant in the whole computational domain, whereas $\phi_s(\mathbf{x})$ is calculated by setting the dielectric constants for solute and solvent differently. Therefore, $\phi_{\text{ref}}(\mathbf{x})$ can be obtained by the standard linear PB equation with the Dirichlet boundary condition via the standard finite difference or FFT methods but $\phi_s(\mathbf{x})$ is solved by using the MIBPB algorithm.

The performance of the MIBPB method for calculating solvation energies has already been examined in our previous work.[77] It has been shown that the MIBPB solver has high accuracy and good convergence order because of the use of interface treatments but has relatively low numerical efficiency because of the absence of appropriate matrix acceleration techniques. The MIBPB matrix requires longer CPU time to solve. The Krylov theory and associated PCs discussed in this work make the MIBPB solver more efficient. Here, the new results are presented for various proteins.

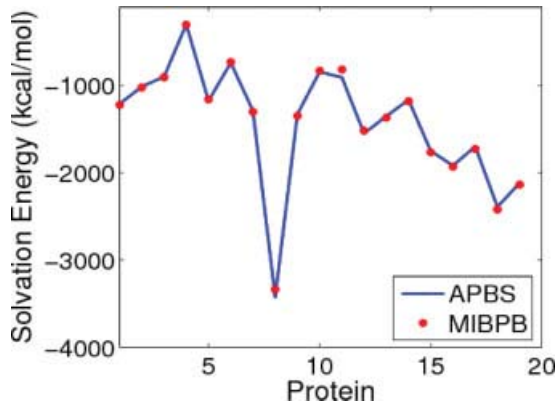

Figure 5. Comparison of solvation energies of proteins (From protein 1 to 19: 1ajj,1bbl,1vii,1cbn,2pde,1sh1,1fca,1fxd,1vjw,1bor,1hpt,1bpi, 1mbg,1r69,1neq,1a2s,1svr,1a63,1a7m) calculated by using the MIBPB and the APBS methods. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Figure 5 gives the comparison of the calculated solvation energies from the MIBPB and the APBS packages. It is seen that the solvation energies calculated from the MIBPB agree very well with those from the APBS. The mesh sizes of $h = 1$ Å is used in the MIBPB and $h = 0.25$ Å is used in the APBS methods, respectively. The reader is referred to ref. 77 for a more detailed comparison among the MIBPB, the APBS, and the PBEQ methods.

Once the preconditioning techniques are applied, the required CPU time is significantly reduced. Figure 6 illustrates the differences of the CPU time needed to calculate solvation energies for 14 moderately large proteins at three different grid resolutions. The solid lines are the CPU time for preconditioned (PCed) systems and dashed lines are for unpreconditioned (unPCed) systems. It can be concluded that at each grid resolution, PCs can save more than half of the overall CPU time.

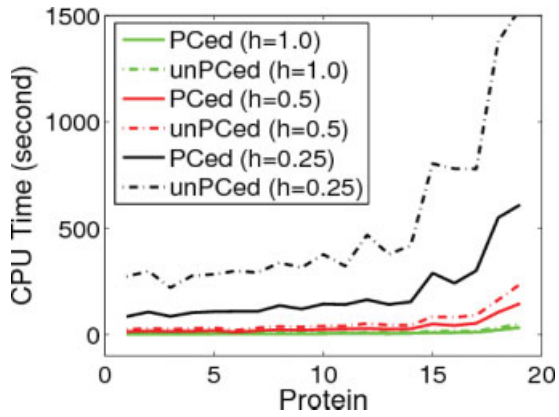

Figure 6. Comparison of CPU time of preconditioned (PCed) and unpreconditioned (unPCed) MIBPB methods for 19 proteins (from protein 1 to 19: 1ajj,1bbl,1vii,1cbn,2pde,1sh1,1fca,1fxd,1vjw,1bor,1hpt,1bpi, 1mbg,1r69,1neq,1a2s,1svr,1a63,1a7m). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Table 5. Solvation Energies and CPU Time For Proteins.

	Solvation energy (kcal/mol)				CPU time (second)			
	MIBPB+KS			MIBPB	MIBPB+KS			MIBPB
h (Å)	1.0	0.5	0.25	0.25	1.0	0.5	0.25	0.25
1ajj	−1141.9	−1136.3	−1136.6	−1137.2	2.3	15	85	273
2pde	−826.2	−819.9	−817.2	−820.9	3.4	17	108	283
1vii	−914.6	−901.5	−902.8	−901.2	2.8	15	86	221
1cbn	−311.1	−303.6	−303.7	−303.8	2.9	16	102	277
1bor	−858.8	−854.3	−857.9	−853.7	4.5	24	143	377
1bbl	−998.3	−986.9	−988.7	−986.8	2.8	16	106	298
1fca	−1215.5	−1199.9	−1200.0	−1200.1	3.5	19	109	292
1uxc	−1157.6	−1138.1	−1139.1	−1138.7	4.2	21	127	347
1sh1	−755.7	−728.0	−751.4	−753.3	3.3	12	109	300
1mbg	−1368.4	−1349.8	−1352.4	−1346.1	4.8	25	142	378
1ptq	−893.5	−871.8	−872.2	−873.1	3.9	22	133	376
1vjw	−1250.9	−1236.9	−1236.9	−1237.9	4.1	26	120	315
1fxd	−3309.7	−3299.7	−3301.6	−3300.0	4.2	31	138	338
1r69	−1111.0	−1086.5	−1087.9	−1089.5	5.5	32	154	419
1hpt	−827.3	−810.9	−812.7	−814.3	4.9	26	141	322
1bpi	−1320.8	−1298.9	−1301.3	−1301.9	5.4	50	164	469
1a2s	−1928.8	−1913.1	−1913.6	−1913.5	9.6	47	242	780
1frd	2879.8	−2851.4	−2856.3	−2851.9	10.8	51	284	707
1svr	−1741.6	−1709.8	−1710.7	−1711.2	11.1	57	301	779
1neq	−1765.6	−1729.1	−1732.7	−1730.1	9.1	50	289	804
1a63	−2420.8	−2371.2	−2370.2	−2373.5	22	113	550	1376
2erl	−964.2	−948.2	−949.3	−948.8	2.3	15	101	276

Table 5 lists the results for the tested proteins at different grid resolutions and compares the values with the original MIBPB-III scheme in terms of solvation energies and CPU time. For each protein case from different grid resolutions, the CPU time increases in nonuniform pattern from less than 10 s for $h = 1.0$ Å, several tens of seconds for $h = 0.5$ Å, to a few hundreds of seconds for $h = 0.25$ Å. Note that there is an eight-time increase in the number of unknowns when the mesh size is halved. The increase in the CPU time is roughly linear with the increase in the number of unknowns.

It is found that at resolution of 0.25 Å, the results from the MIBPB + KS and from the original MIBPB-III have less than 0.1% disagreement. This is due to the use of different convergence norms in the KS solvers and the regular solver. The solvation energy calculations show a correct convergence tendency. The values from

Table 6. Solvation Energies (kcal/mol) and CPU Time (second) for Large Proteins.

Protein		MIBPB ($h = 1.0$ Å)	
ID	Atoms	Solvation energy	CPU time (s)
1cbg	7838	−5659.9	181
1c4k	11439	−9901.9	398
1e24	7776	−9506.4	231
1f6w	8243	−5611.2	225
1po7	7796	−5471.2	206

resolutions of 0.25 Å and 0.5 Å are pretty close, whereas calculations at $h = 0.25$ Å cost much more CPU time. Therefore, we can conclude that grid resolution between 0.5 Å and 1.0 Å is sufficient for the calculation and can guarantee the accuracy.

Table 6 shows the robustness and efficiency of the MIBPB package for calculating solvation energies of large proteins, which exceed 7000 atoms. For time efficiency, all the calculations are carried out at the grid resolution of $h = 1.0$ Å. Note that the reliability of these solvation free energies has been cross-checked with other popular PB solvers. The reported CPU time can be used as a reference.

Application to Salt Effects on Protein–Protein Binding

In this section, the ability of the MIBPB package to solve the nonlinear PBE is tested by checking solvent salt effect on protein–protein binding. The nonlinear PBE has had considerable success in describing biomelocular electrostatics with salt effects on the binding of ligands, peptides and proteins to nucleic acids, membranes, and proteins. The binding free energies reflect the nonspecific salt dependence of the formation of macromolecular complex and the measurement is the binding affinity. Some experimental data are available, and the binding affinity is calculated as the ratio between salt dependent binding energy $\Delta\Delta G_{\rm el}(I)$ at a specific salt strength I, and the natural logarithm of I. In this work, we have implemented the nonlinear version of the PBE solver in the MIBPB package. Simulation results are obtained by varying the ionic strengths.

The binding energy ($\Delta G_{\rm el}$) has several components, whereas the one related to electrostatics is the difference of the electrostatic free energies of the complex and each of its free molecules[81]

$$\Delta G_{\rm el}(I) = G_{\rm el}^{\rm AB}(I) - G_{\rm el}^{\rm A}(I) - G_{\rm el}^{\rm B}(I), \tag{12}$$

where $G_{\rm el}^{\rm AB}(I)$, $G_{\rm el}^{\rm A}(I)$, and $G_{\rm el}^{\rm B}(I)$ represent the electrostatic free energies of the complex AB, component A and component B, respectively, at a given ionic strength I.

The electrostatic free energy can be further split into three components

$$G_{\rm el}(I) = G_{\rm coul} + G_{\rm rxn} + G_{\rm salt}(I), \tag{13}$$

where $G_{\rm coul}$ is the Coulomb energy calculated in a homogeneous medium, $G_{\rm rxn}$ is the corrected reaction field energy, and $G_{\rm salt}(I)$ is the electrostatic energy contributed by mobile ions. Among the three terms in eq. (13), only $G_{\rm salt}(I)$ is salt dependent. Thus, the salt dependence of the binding free energy $\Delta\Delta G_{\rm el}(I)$ is electrostatic component of the binding energy in eq. (12) calculated at some salt strength I minus the one calculated at the zero salt concentration[81]

$$\begin{aligned}\Delta\Delta G_{\rm el}(I) &= \Delta G_{\rm el}(I) - \Delta G_{\rm el}(I=0)\\ &= \left\{G_{\rm el}^{\rm AB}(I) - G_{\rm el}^{\rm AB}(I=0)\right\}\\ &\quad - \left\{G_{\rm el}^{\rm A}(I) - G_{\rm el}^{\rm A}(I=0)\right\}\\ &\quad - \left\{G_{\rm el}^{\rm B}(I) - G_{\rm el}^{\rm B}(I=0)\right\},\end{aligned} \tag{14}$$

where various energy terms are calculated at different ionic strengths by using the MIBPB package.

To verify our nonlinear solver, one heterodimeric and one homodimeric complexes are studied in this work. The experiments on these two protein complexes can be found in refs. 82, 83 and biological features (1emv and 1beb) are listed in Table 7. The first four columns describe the properties of proteins and the last two columns are the slopes (binding affinity) of the lines in Figure 7. It can be seen in a quantitative view that the slopes obtained from experiments and simulations are very close to each other. The calculations were performed assuming that all Arg, Asp, Glu, and Lys residues are ionized in both free and bound states. The results are obtained with dielectric constants of 2 and 80 for the solute and the continuum solvent, respectively. The parameter $\overline{\kappa}^2$ is determined by:

$$\overline{\kappa}^2 = \left(\frac{8\pi^2 N_a e_c^2}{1000 k_B T}\right) I \tag{15}$$

where e_c, k_BT are the same as those defined in eq. (1), N_a is the Avogadro's number. After a simple derivation, $\overline{\kappa}^2$ is given by

$$\overline{\kappa}^2 = 8.486902807\ \text{Å}^{-2} I \tag{16}$$

for $T = 298$ K. Here the ion strength I is in the unit of mole.

Figure 7 depicts the experimental and calculated salt dependence of the binding free energies $\Delta\Delta G_{\rm el}(I)$ for the two complexes. The $\Delta\Delta G_{\rm el}(I)$ are plotted against the logarithm of the salt strength I. The salt dependence is assumed as in a linear pattern; therefore, the least square fitting line is applied to calculate the binding affinity, which is the slope of the line. It should be explained that experimental data dots for $\Delta\Delta G_{\rm el}(I)$ are read from the graphs in ref. 81, whereas the fitting line slope is explicitly given based on the experimental data with error bars. The diamond points and solid line are experimental data and the corresponding fitting line, respectively. The circle points and dashed line are numerical stimulations.

Table 7. Binding Affinity.

Complex	PDB ID	Complex charge	Surface area (Å^2)	Charge of the free monomers	Experimental data	MIBPB ($h = 1.0$ Å)
E9Dnase-Im9 (10) (B-A)	1emv	−3	1465	B = +5; A = −8	2.17	2.42
Lactoglobulin dimer (57)(A-B)	1beb	+26	1167	A = B = +13	−1.62	−1.90

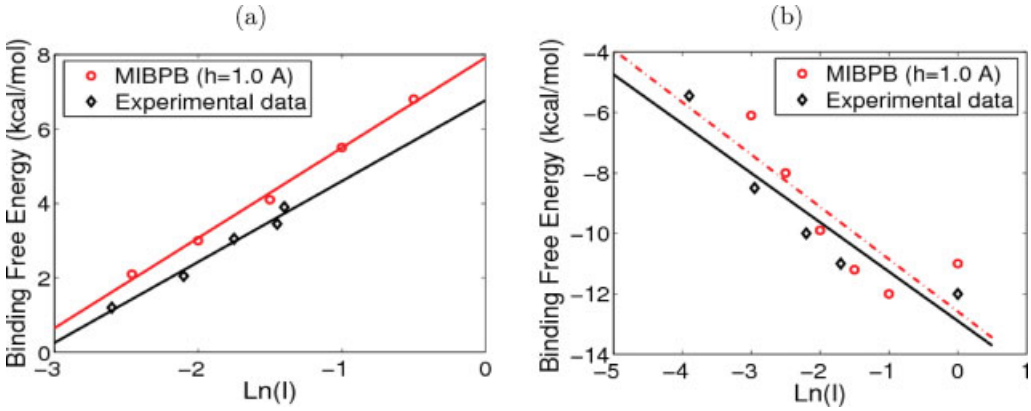

Figure 7. (a) Binding affinity of 1emv; (b) Binding affinity of 1beb. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

In the homo-dimeric complex, the experimentally observed binding free energies decrease with the increase of ionic strength, whereas for the hetero-diemric complex, the experimental measurement had detected an increase. Our results obtained from the MIBPB package reproduced these observations. The calculated magnitudes of the slopes of the salt dependence are in quite good agreement with experimental results within the range of errors, as the fitting lines are almost parallel. The discrepancies between the experimental data and simulation results are also expected: the PBE, no matter in linear or full nonlinear form, only gives the ideal situation of the solute–solvent system but many details, such as the "protein conformation change, pK_a shifts upon complexation or possible ionization states," are lacking.[81] Despite these approximation, the application of PBE for static protein structures is suggested in general by these good agreements with experimental data.

Conclusions

This article introduces the MIBPB software package and its work flow for practical applications in biochemistry, biophysics, and structural biology. Two applications, the solvation free energy calculation and the salt dependence binding affinity calculation, are carried out to justify the robustness, accuracy, and efficiency of solving the linear and nonlinear PBEs. The results of solvation free energy calculations are compared with those from a traditional PBE solver and the results of binding affinity are compared with experimental data. The MIBPB solver is verified to be highly accurate — by far the only existing second-order accurate method for solving the Poisson–Boltzmann (PB) equation with discontinuous dielectric constants, singular charge sources, and geometric singularities from the molecular surfaces of biomolecules. More specifically, the MIBPB has built in advanced interface techniques that are able to deal with discontinuous solvent–solute interfaces and geometric singularities of molecular surfaces. The Dirichlet to Neumann mapping, or Green's function approach, has been developed in the MIBPB solver to analytically resolve the singular point charges. Consequently, the MIBPB solver is able to deliver high accuracy at relatively coarse meshes. The MIBPB solver provides reliable electrostatic potentials at the mesh of about 1 Å, whereas traditional methods have to resort to about 0.25 Å mesh resolution to achieve a similar level of reliability.

In this work, we further equip the MIBPB solver with advanced KS techniques to accelerate the speed of the convergence of solving linear equation systems originated from the MIBPB discretization. The performances of various solver-PC combinations have been carefully examined through mathematical analysis and numerical experiments. Dramatic reductions in condition numbers are found when appropriate PCs are used. Upon the use of appropriate combinations of PCs and solvers, significant reductions in the CPU time are found in solving the PB equation for large proteins. Both the PETSc and the SLATEC are employed in the present MIBPB package to speed up the convergence rate of the iterations of the linear systems. The PETSc package is found to be more reliable and efficient. In the present work, the structure preparation of proteins is conducted via the PDB2PQR software package, whereas the MSMS software package is used for the molecular surface generation.

Additionally, the nonlinear MIBPB solver has been developed in this work. This is achieved via the standard inexact Newton method, assisted by the KS acceleration techniques. The present nonlinear MIBPB solver has been tested and applied to the salt dependence analysis of protein–protein binding interactions. Our results of binding affinities are compared with experimental data.

Appendix A: Linear Equation Systems and MIBPB Matrix

A system of linear algebraic equations is formed after discretizing the PB model

$$L_h u_h = f_h \tag{A1}$$

where L_h is a real nonsingular n by n matrix under grid spacing h, u_h is the numerical solution vector and f_h is the source term vector. The matrix L_h is viewed as a linear operator mapping $\mathbb{R}^n$ into $\mathbb{R}^n$, the space $\mathbb{R}^n$ being a linear space equipped with an inner-product $(\cdot,\cdot)$ inducing a norm $\|\cdot\|$ defined as follows

$$(u, v) = \sum_{i=1}^{n} u_i v_i, \quad \|u\| = (u, u)^{1/2}, \forall u, v \in \mathbb{R}^n.$$

where u_i represents the i-th component of the vector u.

Generally, systems of linear algebraic equations are commonly solved by using direct methods and iterative methods. Direct methods, such as Gaussian elimination and LU decomposition, work for general matrices with arbitrary structure but require large computer memory. Therefore, they are not computationally efficient, and hence unsuitable for solving the 3D PB model of biomolecules, even for small proteins.

Some of the iterative methods, such as Richardson, Jacobi, Gauss-Seidel, and SOR iterations, also work well for general structured matrices but they are barely used due to the reduced robustness for large protein system. The classic linear iteration methods for solving eq. (A1) can be viewed as the following form

$$u_h^{j+1} = u_h^j - BL_h u_h^j + Bf_h, \tag{A2}$$

where B is matrix approximating L_h^{-1} in some sense. Different construction of matrix B results in a different iterative method. The necessary and sufficient condition for the convergence of algorithm (A2) is that the spectrum ρ of the error propagation operator must be smaller than 1, that is, $E = I_h - BL_h$ and $\rho(E) < 1$,[84] where I_h is the identity operator associated with the grid resolution h. The smaller value of $\rho(E)$ indicates the better convergence of the method. The spectra of this family of iteration methods can be expressed as $\rho(E) = 1 - O(h^2)$, which implies that as grid spacing gets smaller, these methods converge more and more slowly. This property severely restricts the wide applications of these methods for large linear systems.

The conjugate gradient (CG) method is a very efficient iterative method if the matrix is symmetric and positive definite. Actually, it is the main workhorse of most popular PBE solvers, because the matrices from the standard FDMs or FEMs satisfy these good properties. The multigrid (MG) method is an accelerating technique and can be applied in combination with any of commonly used solvers. Using a hierarchy of discretizations, MG shifts the computation between coarser and finer grids by extrapolation and restriction, and thus accelerates the convergence. It is almost the fastest accelerating technique known so far and applied in many popular PB solvers, such as the APBS.

Unfortunately, the matrix L_h from the MIB can barely take advantages from the described methods due to its notoriously complicated structure. For the discretization of the Laplace operator in the PBE by standard FDMs, every grid point except the boundary ones takes the following form:

$$\begin{aligned} -\nabla \cdot (\epsilon \nabla u)|_{\mathbf{x}=x_i,y_j,z_k} &= c_0 u_{i,j,k} + c_1 u_{i-1,j,k} + c_2 u_{i+1,j,k} \\ &+ c_3 u_{i,j-1,k} + c_4 u_{i,j+1,k} \\ &+ c_5 u_{i,j,k-1} + c_6 u_{i,j,k+1} \end{aligned} \tag{A3}$$

where i, j, and k represent the discretization indices along the x, y, and z directions, respectively. The coefficients $c_m, m = 0, 1, \ldots 6$ only depend ϵ and grid spacing h. The symmetric structure of eq. (A3) and the facts $\sum_{m=0}^{6} c_m = 0$ and $c_1 = c_2 = c_3 = c_4 = c_5 = c_6$ make the whole matrix symmetric and positive definite.

However, as the MIB scheme takes into account the interface treatment and at all the irregular grid points near the interface, discetizations are modified. For the simplest case, assume that only one fictitious point is needed and without the loss of generality, the modification is in the form:

$$\begin{aligned} -\nabla \cdot (\epsilon \nabla u)|_{\mathbf{x}=x_i,y_j,z_k} &= c_0 u_{i,j,k} + c_1 f^* + c_2 u_{i+1,j,k} \\ &+ c_3 u_{i,j-1,k} + c_4 u_{i,j+1,k} \\ &+ c_5 u_{i,j,k-1} + c_6 u_{i,j,k+1} \end{aligned} \tag{A4}$$

Note that the fictitious value f^* is used in eq. (A4) for the smooth extension of the function. The fictitious value f^* can further be expanded as the linear combination of the unknown function values.

$$f^* = \sum_{m=1}^{M} \tilde{c}_m u_{i'_m,j'_m,k'_m} \tag{A5}$$

where $u_{i'_m,j'_m,k'_m}$ is the nearby function values around $u_{i,j,k}$, $\tilde{c_m}, m = 1, 2, \ldots, M$ are the corresponding coefficients. Usually, $M = 10$ in second order MIB scheme for a smooth interface but could be bigger for interface with singularities. The choice of $u_{i'_m,j'_m,k'_m}$ and calculation of $\tilde{c}_m$ totally depend on the local information of the interface. The introduction of the fictitious values gives high accuracy for the interface problems but also ruins the good properties, such as symmetry and positive-definiteness of the overall matrix.

To solve the matrices generated from the MIB scheme, the direct methods and regular iterative methods will be ruled out from the beginning because of the poor convergence for huge systems. The CG method also does not work because the unpredictably general matrix structures. Meanwhile, the direct application of the multigrid method, which is an important accelerating technique, also has a potential problem because of the shift of irregular point locations during grid refinement cycles. Reference 85 showed the poor behavior of the algebraic multigrid method (AMD) and proposed a new multigrid scheme based on the local interface problem but the interpolation operator at the interface will cost much extra work.

Therefore, we put more emphasis on looking for suitable solvers and accelerating techniques in the KS theory. Stabilized BiCG and GMRES are two examples in KS methods, which deal with the general nonsingular matrix that does not have to be symmetric and positive definite. In the following section, the KS methods and their analysis are briefly introduced. Different types of PCs are associated to the KS solvers. These combinations are tested in "Preconditioner Accelerated MIBPB Solvers" section to achieve the optimal convergence rate for solving the linearized or nonlinear MIBPB matrices.

Appendix B: KS Method and Preconditioning

Suppose u^0 is an initial guess for the solution u in system (A1) and defines the initial residual $r^0 = f - Lu^0$. For notation simplicity,

the subscript h is dropped here. As shown in ref. 86, the KS can be derived from the following projection method. The m^{th} iteration $u^m, m = 1, 2, \ldots$ is of the form

$$u^m \in r^0 + \mathcal{S}_m, \tag{B1}$$

where $\mathcal{S}_m$ is some m-dimensional space, called the search space. Strictly speaking, eq. (B1) is an abused notation, it means that u^m can be decomposed as the residual r^0 and an element in space $\mathcal{S}_m$. Because of m degrees of freedom, a total of m constraints is required to make u^m unique. This is achieved by choosing an m-dimensional space $\mathcal{C}_m$, called the constraint space, and by requiring that the m^{th} residual is orthogonal to that space, that is,

$$r^m = f - Lu^m \in r^0 + L\mathcal{S}_m, \quad r^m \perp \mathcal{C}_m. \tag{B2}$$

Here, the orthogonality is in the sense of the inner product in the Euclidean space.

If the space $\mathcal{S}_m$ is defined as the KS $\mathcal{K}_m(L, r^0)$, that is,

$$\mathcal{S}_m = \mathcal{K}_m(L, r^0) \equiv \text{span}\{r^0, Lr^0, \ldots, L^{m-1}r^0\}, \quad m = 1, 2, \ldots, \tag{B3}$$

then the projection method is the so-called KS method. More specifically, if $\mathcal{C}_m = \mathcal{S}_m$, it is the Galerkin method, which includes the CG method and its generalizations, and if $\mathcal{C}_m = L\mathcal{S}_m$, it yields the GMRES. These are the basic idea of KS methods.

For the convergence analysis, note that conditions (B1) and (B2) imply that the error $u - u^m$ and the residual r^m can be written in the polynomial form

$$u - u^m = p_m(L)(u - u^m), \quad r^m = p_m(L)r^0, \tag{B4}$$

where p_m is a polynomial of degree at most m and with value one at the origin. Reference 86 gives the error bound for KS methods

$$\frac{\|u - u^m\|}{\|u - u^0\|} \leq \min_{p \in \pi_m} \max_k |p(\lambda_k)|, \tag{B5}$$

where π_m denotes the set of polynomials of degree at most m and with value one at the origin, λ_k are the eigenvalues of the matrix L. It can be concluded from eq. (B5) that the convergence behavior of the KS methods is completely determined by their spectra.

However, as indicated in ref. 86, it is always difficult to really evaluate the upper bound. Alternatively, it states that the condition number of the matrix is a criteria which although, only partially reveals the practice convergence behavior but is easier to calculate. For matrix L, the condition number is defined as the ratio of the extreme eigenvalues or spectra

$$\varsigma = \frac{\lambda_{\max}}{\lambda_{\min}}. \tag{B6}$$

Because the rate of the convergence of Krylov projection methods for a particular linear system is strongly dependent on its spectrum, PC is typically used to alter the spectrum and hence accelerate the convergence rate of iterative techniques. PC can be applied to system (A1) by

$$\left(M_L^{-1} L M_R^{-1}\right)(M_R u) = M_L^{-1} f, \tag{B7}$$

where M_L and M_R denote the left and right precondition matrices. Usually, if $M_R = I$, the left preconditioned results and the residual is given by

$$r_L = M_L^{-1} f - M_L^{-1} L u. \tag{B8}$$

Properly preconditioned matrix $M^{-1}L$ may significantly reduce the condition number of L, hence the rate of convergence is accelerated. The commonly used precondition strategies are JAC, block PC and ILU factorization. However, preconditioning a large sparse system is an empirical exercise. Different PCs work better for different kinds of problems. In "Preconditioner Accelerated MIBPB Solvers" section, the combinations of different KS solvers and PCs are investigated, and the rate of convergence is analyzed via the spectra of preconditioned and unpreconditioned matrices.

The accelerated Krylov methods are also crucial for solving the nonlinear PBE. The discretization of the nonlinear PB equation results in a linear equation system of the form

$$F(u) = Lu + N(u) - f = 0, \tag{B9}$$

where the matrix L is still from the MIBPB scheme, the nonlinear term $N(\cdot)$ is diagonal and $N_i(u) = N_i(u_i) = \bar{\kappa}^2 \sinh(u_i)$. The inexact-Newton method is perhaps one of the most efficient ways to solve nonlinear system (B9)

$$F'(u_n)v_n = -F(u_n) + r_n, \quad \frac{\|r_n\|}{\|F(u_n)\|} \leq \eta_n \tag{B10}$$

$$u_{n+1} = u_n + v_n, \tag{B11}$$

where F' is the Jacobian matrix $[\partial F_i(u)/\partial u_j]$ and takes the form $F'(u) = L + N'(u)$. Here, N' is the Jacobian matrix of $N(u)$ and is also diagonal $N_i'(u) = N_i'(u_i) = \bar{\kappa}^2 \cosh(u_i)$.

It is easy to see that the inexact-Newton method is a two-layer iterative algorithm. The correction term v_n in outer iteration (B11) is considered as a rough solution of inner iteration (B10). The scheme converges linearly when η_n, the ratio of the residual r_n between the function value $F(u)$, is less than 1, and converges super-linearly as the sequence η_n has the property that $\lim_{n\to\infty} \eta_n = 0$. In the MIBPB package, accelerated KS methods are applied to inner iteration (B10) to attain fast convergence.

References

1. Roux, B.; Simonson, T. Biophy Chem 1999, 78, 1.
2. Sharp, K. A.; Honig, B. Annu Rev Biophys Biophys Chem 1990, 19, 301.
3. Davis, M. E.; McCammon, J. A. Chem Rev 1990, 94, 509.

4. Chen, J.; Brooks, C. L., III. Phys Chem Chem Phys 2008, 10, 471.
5. Baker, N. A.; Bashford, D.; Case, D. A. In Implicit solvent electrostatics in biomolecular simulation; Leimkuhler, B.; Chipot, C.; Elber, R.; Laaksonen, A.; Mark, A.; Schlick, T.; Schutte, C.; Skeel, R.; Eds.; Springer: Berlin, Heidelberg, New York, 2006.
6. Dominy, B. N.; Brooks, C. L., III.; J Phys Chem B 1999, 103, 3765.
7. Feig, M.; Im, W.; Brooks, C. L., I.; J Chem Phys 2004, 120, 903.
8. Zhu, J.; Alexov, E.; Honig, B. J Phys Chem B 2005, 109, 3008.
9. Mongan, J.; Simmerling, C.; McCammon, J. A.; Case, D. A.; Onufriev, A. J Chem Theory Comput 2007, 3, 159.
10. Chiba, M.; Fedorov, D. G.; Kitaura, K. J Comput Chem 2008, 29, 2667.
11. Tomasi, J.; Mennucci, B.; Cammi, R. Chem Rev 2005, 105, 2999.
12. Improta, R.; Barone, V.; Scalmani, G.; Frisch, M. J. J Chem Phys 2006, 125, 054103-1.
13. Baker, N. A. Meth Enzymol 2004, 383, 94.
14. Lamm, G. In The Poisson–Boltzmann Equation. Lipkowitz, K. B.; Larter, R.; Cundari, T. R. Eds.; Wiley: Hoboken, NJ, 2003; pp. 147–366.
15. Tsui, V.; Case, D. A. J Am Chem Soc 2000, 122, 2489.
16. Simonson, T. Cur Opin Struct Biol, 2001, 11, 243.
17. Feig, M.; Brooks, C. L., I.; Curr Opin Struct Biol 2004, 14, 217.
18. Beglov, D.; Roux, B. J Chem Phys 1996, 104, 8678.
19. David, L.; Luo, R.; Gilson, M. K. J Comput Chem 2000, 21, 295.
20. Onufriev, A.; Bashford, D.; Case, D. A. J Phys Chem B 2000, 104, 3712.
21. Bashford, D.; Case, D. A. Annu Rev Phys Chem 2000, 51, 129.
22. Tsui, V.; Case, D. A. J Phys Chem B 2001, 105, 11314.
23. Livesay, D. R.; Jambeck, P.; Rojnuckarin, A.; Subramaniam, S. Biochemistry 2003, 42, 3464.
24. Blomberg, N.; Gabdoulline, R. R.; Nilges, M.; Wade, R. C. Proteins 1999, 37, 379.
25. Swanson, J. M. J.; Henchman, R. H.; McCammon, J. A. Biophys J 2004, 86, 67.
26. Massova, I.; Kollman, P. A. J Am Chem Soc 1999, 121, 8133.
27. Nielsen, J. E.; Vriend, G. Proteins 2001, 43, 403.
28. Tang, C. L.; Alexov, E.; Pyle, A. M.; Honig, B. J Mol Biol 2007, 366, 1475.
29. Prabhu, N. V.; Panda, M.; Yang, Q. Y.; Sharp, K. A. J Comput Chem 2008, 29, 1113.
30. Luo, R.; David, L.; Gilson, M. K. J Comput Chem 2002, 23, 1244.
31. Madura, J. D.; Briggs, J. M.; Wade, R. C.; Davis, M. E.; Luty, B. A.; Ilin, A.; Antosiewicz, J.; Gilson, M. K.; Bagheri, B.; Scott, L. R.; McCammon, J. A. Comput Phys Commun 1995, 91, 57.
32. Gabdoulline, R. R.; Wade, R. C. Methods-a Companion Methods Enzymol 1998, 14, 329.
33. Sept, D.; Elcock, A. H.; McCammon, J. A. J Mol Biol 1999, 294, 1181.
34. Cheng, Y.; Suen, J. K.; Radi, Z.; Bond, S. D.; Holst, M. J.; McCammon, J. A. Biophy Chem 2007, 127, 129.
35. Song, Y.; Zhang, Y.; Bajaj, C. L.; Baker, N. A. Biophys J 2004, 87, 1558.
36. Song, Y.; Zhang, Y.; Shen, T.; Bajaj, C. L.; McCammon, J. A.; Baker, N. A. Biophys J 2004, 86, 2017.
37. Petrey, D.; Honig, B. Methods Enzymol 2003, 374, 492.
38. Baker, N. A.; McCammon, J. A. In Electrostatic interactions. Bourne, P.; Weissig, H.; Eds.; Wiley: New York, 2003; pp. 427–440.
39. Kirkwood, J. G. J Comput Phys 1934, 7, 351.
40. Honig, B.; Nicholls, A. Science 1995, 268, 1144.
41. Warwicker, J.; Watson, H. C. J Mol Biol 1982, 157, 671.
42. Im, W.; Beglov, D.; Roux, B. Comput Phys Commun 1998, 111, 59.
43. Baker, N. A.; Sept, D.; Joseph, S.; Holst, M. J.; McCammon, J. A. Proc Natl Acad Sci USA 2001, 98, 10037.
44. Boschitsch, A. H.; Fenley, M. O. J Comput Chem 2004, 25, 935.
45. Zauhar, R. J.; Morgan, R. S. J Mol Biol 1985, 186, 815.
46. Baker, N. A. Curr Opin Struct Biol 2005, 15, 137.
47. Klapper, I.; Hagstrom, R.; Fine, R.; Sharp, K.; Honig, B. Protein 1986, 1, 47.
48. Engles, M.; Gerwert, K; Bashford, D. Biophys Chem 1995, 56, 95.
49. Holst, M.; Baker, N.; Wang, F. J Comput Chem 2000, 21, 1319.
50. Lu, B. Z.; Chen, W. Z.; Wang, C. X.; Xu, X. J. Proteins 2002, 48, 497.
51. Jo, S; Vargyas, M; Vasko-Szedler, J.; Roux, B.; Im, W. Nucl Acids Res 2008, W270, 36.
52. Brooks, B. R.; Bruccoleri, R. E.; Olafson, B. D.; States, D.; Swaminathan, S.; Karplus, M. J Comput Chem 1983, 4, 187.
53. Dong, F.; Vijaykumar, M.; Zhou, H. X. Biophys J 2003, 85, 49.
54. Nina, M.; Im, W.; Roux, B. Biophy Chem 1999, 78, 89.
55. Swanson, J. M. J.; Mongan, J.; McCammon, J. A. J Phys Chem B 2005, 109, 14769.
56. Lee, B.; Richards, F. M. J Mol Biol 1971, 55, 379.
57. Richards, F. M. Annu Rev Biophys Bioeng 1977, 6, 151.
58. Connolly, M. L. J Appl Crystallogr 1983, 16, 548.
59. Sanner, M. F.; Olson, A. J.; Spehner, J. C. Biopolymers 1996, 38, 305.
60. Peskin, C. S. J Comput Phys 1977, 25, 220.
61. Fedkiw, R. P.; Aslam, T.; Merriman, B.; Osher, S. J Comput Phys 1999, 152, 457.
62. Bramble, J.; King, J. Adv Comput Math 1996, 6, 109.
63. Li, Z. L.; Lin, T.; Wu, X. H. Numerische Mathematik 2003, 96, 61.
64. Cai, W.; Deng, S. Z. J Comput Phys 2003, 190, 159.
65. LeVeque, R. J.; Li, Z. L. SIAM J Numer Anal 1994, 31, 1019.
66. Zhao, S.; Wei, G. W. J Comput Phys 2004, 200, 60.
67. Zhou, Y. C.; Zhao, S.; Feig, M.; Wei, G. W. J Comput Phys 2006, 213, 1.
68. Zhou, Y. C.; Wei, G. W. J Comput Phys 2006, 219, 228.
69. Yu, S.; Zhou, Y.; Wei, G. W. J Comput Phys 2007, 224, 729.
70. Yu, S.; Wei, G. W. J Comput Phys 2007, 227, 602.
71. Yu, S. N.; Xiang, Y.; Wei, G. W. Commun Numer Methods Eng 2009, 25, 923.
72. Zhao, S.; Wei, G. W. Int J Numer Meth Eng 2009, 77, 1690.
73. Chen, D.; Wei, G. W.; Cong, X.; Wang, G. Commun Numer Methods Eng 2009, 25, 1137.
74. Zhao, S. IEEE Microwave Wireless Compon Lett 2009, 19, 266.
75. Zhou, Y. C.; Feig, M.; Wei, G. W. J Comput Chem 2008, 29, 87.
76. Yu, S.; Geng, W.; Wei, G. W. J Chem Phys 2007, 126, 244108.
77. Geng, W.; Yu, S.; Wei, G. W. J Phys Chem 2007, 127, 114106.
78. Chern, I.-L.; Liu, J.-G.; Weng, W.-C. Methods Appl Anal 2003, 10, 309.
79. Yu, S. N. Matched interface and boundary (MIB) method for geometric singularities and its application to molecular biology and structural analysis. Dissertation of Michigan State University: Michigan State University, 2007.
80. Holst, M. J. Multilevel Methods for the Poisson-Boltzmann Equation. University of Illinois, Numerical Computing Group: Urbana-Champaign, 1993.
81. Bertonati, C.; Honig, B.; Alexov, E. Biophys J 2007, 92, 1891.
82. Sakurai, K.; Oobatake, M.; Goto, Y. Protein Sci 2001, 10, 2325.
83. Wallis, R.; Moore, G. R.; James, R.; Kleanthous, C. Biochemistry 1995, 34, 13743.
84. Ortega, J. M. Numerical Analysis: A Second Course. Academic Press: NY, 1999.
85. Chen, T.; Strain, J. J Comput Phys 2008, 16, 7503.
86. Liesen, J.; Tichy, P. CGAMM Mitteilungen 2004, 27, 153.