J Mol Model (2011) 17:1985–1996
DOI 10.1007/s00894-010-0904-4

ORIGINAL PAPER

Exploring a coarse-grained distributive strategy for finite-difference Poisson–Boltzmann calculations

Meng-Juei Hsieh · Ray Luo

Received: 14 September 2010 / Accepted: 14 November 2010 / Published online: 3 December 2010

Abstract We have implemented and evaluated a coarse-grained distributive method for finite-difference Poisson–Boltzmann (FDPB) calculations of large biomolecular systems. This method is based on the electrostatic focusing principle of decomposing a large fine-grid FDPB calculation into multiple independent FDPB calculations, each of which focuses on only a small and a specific portion (block) of the large fine grid. We first analyzed the impact of the focusing approximation upon the accuracy of the numerical reaction field energies and found that a reasonable relative accuracy of 10^{-3} can be achieved when the buffering space is set to be 16 grid points and the block dimension is set to be at least $(1/6)^3$ of the fine-grid dimension, as in the one-block focusing method. The impact upon efficiency of the use of buffering space to maintain enough accuracy was also studied. It was found that an "optimal" multi-block dimension exists for a given computer hardware setup, and this dimension is more or less independent of the solute geometries. A parallel version of thedistributive focusing method was also implemented. Given the proper settings, the distributive method was able to achieve respectable parallel efficiency with tested biomolecular systems on a loosely connected computer cluster.

Keywords Finite difference · Poisson–Boltzmann · Electrostatic focusing · Distributive computing · Domain decomposition

M.-J. Hsieh · R. Luo
Department of Molecular Biology and Biochemistry,
University of California,
Irvine, CA 92697-3900, USA

R. Luo (✉)
Department of Biomedical Engineering, University of California,
Irvine, CA 92697-3900, USA
e-mail: rluo@uci.edu

Introduction

The Poisson–Boltzmann theory has become well established for the modeling of electrostatic solvation interactions in biomolecules [1–14]. In this approach, a molecular solute is approximated as a continuous cavity with a low dielectric constant, for example with values in the range of 1–4. The solvent is approximated as a continuous medium with a different dielectric constant solvent, for example with a value of ~80 for water. The cavity contains point charges at atomic centers obtained from a molecular mechanics representation of the solute. When mobile ions are present, the ion distribution is also modeled in a mean field fashion (assumed to obey the Boltzmann distribution). With such a continuum approximation of the electrostatic interactions, the electrostatic potential ϕ is characterized by the Poisson–Boltzmann equation (PBE):

$$\nabla \cdot \varepsilon \nabla \phi = -4\pi\rho_0 - 4\pi\lambda \sum_i e z_i c_i \exp(-e z_i \phi / k_B T), \qquad (1)$$

where ε is the dielectric constant, ρ_0 is the solute charge density, λ is the Stern layer masking function, ez_i is the charge of ion type i, c_i is the bulk number density of ion type i, k_B is the Boltzmann constant, and T is the absolute temperature. If the electrostatic field is weak and ionic strength is low, the PBE can be linearized as [15]

$$\nabla \cdot \varepsilon \nabla \phi = -4\pi\rho_0 + \lambda\kappa^2\phi. \qquad (2)$$

Here $\kappa^2 = 4\pi e^2 \sum_i z_i^2 c_i / k_B T$. Since the early 1980s, considerable efforts have been devoted to solving the PBE numerically for biomolecular applications. Many software packages that can solve either the linear or the nonlinear PBE have been released, such as Delphi [16–18], UHBD [19–22], MEAD [23, 24], the CHARMM PBEQ module

[25, 26], the Jaguar PB module [27], APBS [28, 29], and the Amber PBSA module [30–35].

Typical numerical methods involve the discretization of the partial differential equation (PDE) into a system of linear or nonlinear equations and then the solution of the linear or nonlinear system with an iterative approach. The most commonly used discretization approach is the finite-difference method [16–19, 21, 24, 26, 30, 36–38]. In this method, the physical properties of the solution, such as atomic charges and dielectric constants, are mapped onto rectangular grids, and a discrete approximation to the governing PDE is produced. The second approach is the finite-element method [27–29, 39–41], which approximates the potential with the superposition of a set of basis functions. A linear or nonlinear system for the coefficients produced by the weak formulation has to be solved. The third approach is the boundary-element method [42–55]. One advantage of applying the boundary element approach lies in the fact that the dimension of the linear or nonlinear system is much smaller than that from the finite-difference scheme due to the different dimensionality (3-D vs. 2-D). However, the corresponding system is much denser.

In this study, we focus our effort on the sparse linear systems from the finite-difference discretization of the linear PBE. These methods are often termed finite-difference Poisson–Boltzmann (FDPB) methods. The FDPB method converts the PBE to a system of linear equations $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, where the electrostatic potential ($\boldsymbol{x}$) on every grid point is unknown, the charge distribution ($\boldsymbol{b}$) on the grid point is the source, and the dielectric constant and salt-related terms are wrapped into the coefficient matrix $\mathbf{A}$, which is a seven-banded symmetric matrix [36]. Its numerical solvers can be categorized into direct methods and iterative methods. Due to the large number of unknowns in biomolecular applications, it is not very practical to use direct methods. Therefore, most solvers are iterative in nature for biomolecular applications.

The iterative methods can be loosely grouped into two types: stationary methods and Krylov subspace methods. The partition is not very strict because many stationary methods can be used during preconditioning for the Krylov subspace methods [56]. To implement stationary methods such as Jacobi, successive over-relaxation, and Gauss–Seidel for distributive computing environments, the unknowns are usually reordered with multi-coloring [57] or multi-splitting approaches [58].

Possible strategies for implementing Krylov subspace methods in distributive computing environments are domain decompositions and distributive preconditioners. The principle of domain decompositions is to divide the problem domain into smaller subdomains, solve the subdomains independently, and merge the subdomain solutions in a self-consistent manner. We can decompose a problem domain into either overlapping or nonoverlapping subdomains with the domain decomposition methods for the linear systems from FDPB. An overlapping domain decomposition method normally takes the form of the multiplicative Schwarz method; for nonoverlapping domain decomposition, the Schur method is usually used [59].

Most prior efforts to adapt Krylov subspace methods to distributive computing environments focus on the preconditioning step [60]. These efforts aim at either extracting the data/instruction independence from a serial method or modifying or approximating a serial method to increase the independence [61]. Take the widely used incomplete factorization methods [62] for example. We cannot simply convert the algorithm into a distributive method because of the data dependence in the forward and backward substitutions [61]. Modifying or approximating the methods normally leads to slower convergence because there is a trade-off between the data locality needed for parallelism and the data global dependence needed for fast convergence. Therefore, a reduction in CPU time is not always guaranteed in distributive computing environments [63]. Three basic strategies for incomplete factorization methods such as incomplete Cholesky conjugate gradient (ICCG) are: reordering, series expansion, and domain decomposition [61]. Some common practices for reordering include hyperplane ordering (to change the order of computation) and multi-color ordering (to reorder the unknowns). The hyperplane ordering method can be implemented efficiently for good parallel performance [61, 64–67]. Multi-coloring methods like red–black ordering were proven to result in slower convergence [68]. However, the use of more colors was reported to lead to a good balance between convergence and data/instruction independence [69, 70]. More recent studies of ordering strategies show that a "blocked" red–black ordering method can reduce the cost of synchronization or communication [71]. Series expansions were also attempted to approximate the preconditioners with truncated expansions that can be evaluated more efficiently without data dependence. The idea is to split and truncate the expansions but keep enough terms to ensure that the convergence rate is not degraded too much. The truncated Neumann expansion [72] and other polynomial expansions [73] are often used to approximate the preconditioners. Interestingly, the domain-decomposition idea can also be used to implement preconditioning without data dependence. Aside from decomposing the grid then applying independent incomplete factorization for distributive processing, preconditioners can also be domain-decomposed algebraically [74–76].

Other efforts to solve linear problems include the use of a two- (or more) level grid-based solution such as multigrid methods [77, 78]. Multigrid methods usually consist of the following steps: (1) performing some stages of a basic

method (i.e., one of the iterative solvers) to smooth out the error; (2) restricting the current state of the problem to a subset of coarse grids and solving the resulting smaller problem; (3) interpolating the coarse-grid solution back to the original grid and performing a number of stages of the basic method. These steps are applied recursively to achieve convergence [79]. Many aspects of multigrid algorithms are readily implemented in multiple threads for distributive environments. Indeed, multigrid methods have become popular for solving linear systems arising from PDEs [28, 29, 80, 81].

A seemingly related method is electrostatic focusing in FDPB. The focusing technique is a series of finite-difference runs that are performed with successively finer grids. Each run has boundary conditions that are calculated from the potential map of its predecessor [36]. The focusing technique ensures reasonable accuracy on the finest grid without a huge computational overhead, because only a certain part of the problem domain is solved on the finest grid. Long-range interactions with the rest of the domain outside the focusing region can be accurately represented by the coarser-grid calculations. However, there has been no theoretical justification of focusing when it is applied in the finite-difference method [28], though the error due to the approximation is often extremely small under certain conditions.

A limitation of the electrostatic focusing calculations is that the memory usage of the finest grid for the region of interest may still be unmanageably large in FDPB calculations of large biomolecular systems. It has been reported that the fine grids in the focusing technique can be divided into multiple smaller blocks for distributive computation [82]. This idea, termed "multi-block" focusing below, is similar to the domain decomposition concept, but the boundaries of the subdomains are neither conjugated nor communicated during computation. It is thus an approximated method. Nevertheless, the method may be used to lower the memory usage of each focusing run and is potentially friendly to distributive computing environments.

In this study, we explored the distributive multi-block focusing technique in order to analyze what it takes to achieve an acceptable accuracy and good performance in numerical FDPB solutions of large biomolecular systems. A highly scalable parallel version of the method is also implemented and incorporated into the Amber/PBSA program [30–35]. In the following, we summarize our algorithm design and implementation and evaluate its performance using several challenging test cases. We conclude with a brief discussion of potentially more accurate numerical algorithms for future development in the field of distributive FDPB calculations of large biomolecular systems.

Methods

Finite-difference Poisson–Boltzmann method

In our implementation of the FDPB method, a uniform cubic grid is used to discretize the linear PBE [Eq. 2]; i.e., the spacing between neighboring grid points is uniformly set to be h. The grid points are labeled (i, j, k), i=1,....xm, j=1,....ym, and k=1,....zm, where xm, ym and zm are the numbers of grid points along the x, y and z axes, respectively. Electrostatic potential (φ) and atomic charge (q) are mapped onto the grid points. The dielectric constant is defined at the midpoint between any two neighboring grid points: $\varepsilon_i(i, j, k)$ denotes the dielectric constant between grids (i, j, k) and $(i+1, j, k)$, and $\varepsilon_j(i, j, k)$ and $\varepsilon_k(i, j, k)$ are used similarly. Finally, κ^2 is used to absorb all related coefficients in the Boltzmann term. The discretized linear PBE can then be written as

$$\begin{aligned}
&-h^{-2}\varepsilon_i(i-1,j,k)[\phi(i-1,j,k)-\phi(i,j,k)]\\
&-h^{-2}\varepsilon_i(i,j,k)\quad[\phi(i+1,j,k)-\phi(i,j,k)]\\
&-h^{-2}\varepsilon_j(i,j-1,k)[\phi(i,j-1,k)-\phi(i,j,k)]\\
&-h^{-2}\varepsilon_j(i,j,k)\quad[\phi(i,j+1,k)-\phi(i,j,k)]\\
&-h^{-2}\varepsilon_k(i,j,k-1)[\phi(i,j,k-1)-\phi(i,j,k)]\\
&-h^{-2}\varepsilon_k(i,j,k)\quad[\phi(i,j,k+1)-\phi(i,j,k)]\\
&+\quad\kappa^2\qquad\phi(i,j,k)\qquad\qquad=h^{-3}q(i,j,k),
\end{aligned}\tag{3}$$

which is a linear system of $xmymzm$ equations, usually denoted $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ (with the electrostatic potential as the unknown $\boldsymbol{x}$, the charge distribution as the source $\boldsymbol{b}$, and the dielectric constant and salt-related terms wrapped into a seven-banded symmetric coefficient matrix $\mathbf{A}$).

One-block versus multi-block electrostatic focusing

As mentioned in the "Introduction," electrostatic focusing can be used to obtain an accurate solution in a specific region of a problem domain. For the sake of simplicity of presentation, we utilized only a two-level focusing construct in this study; i.e., only two grids (coarse and fine) were used in all test calculations discussed below. The first step in the electrostatic focusing method is a coarse-grid FDPB calculation spanning the entire problem domain. The problem domain is often set to be large enough to secure a good free-boundary condition. The coarse-grid solution of the potential is then used to define the boundary condition (i.e., the boundary potential) for the fine grid covering only the region of interest. Here the boundary potential is computed with the trilinear weighted interpolation method for a good balance of accuracy and efficiency [31].

Again, as we mentioned in the "Introduction," there often is memory limitation in electrostatic focusing calcu-

lations. For example, an electrostatic focusing calculation for the complete 70S ribosome requires ~16.6 GB memory to achieve a resolution of 0.5 Å at the fine-grid level. To overcome this limitation, the fine-grid region can be divided into multiple subdomains or blocks to be solved independently, via the so-called divide-and-conquer strategy. Subsequent multiple blocks can be made small enough to be processed sequentially on a personal workstation or in a distributive manner on networked workstations or computing nodes.

In the following we term the extended electrostatic focusing method the "multi-block" electrostatic focusing method, and the classical electrostatic focusing method the "one-block" electrostatic focusing method. One noticeable feature of the multi-block electrostatic focusing method is that the adjacent blocks overlap (see Fig. 1) in order to maintain good overall calculation accuracy. The overlapping region is termed the buffering space in this study. Its influence upon the calculation accuracy is analyzed in detail below.

In the current design (Fig. 1), every focusing FDPB calculation depends on the solution of the coarse-grid FDPB calculation. There are two choices with how to proceed in the coarse-grid FDPB calculation if the focusing blocks are distributed to different computing threads: (1) only the master thread is used to solve the coarse-grid FDPB, and the solution is communicated to slave threads before the fine-grid FDPB starts; or (2) the coarse-grid FDPB is solved on all threads before starting the assigned fine-grid FDPB calculations, so the coarse-grid solution is kept local on each thread. Apparently the second strategy is slightly better because it reduces the need for communication among all threads. Of course, the molecular data are still needed on the slave threads before the coarse-grid FDPB calculation, and the final electrostatic energies are communicated back to the master thread after finishing the assigned fine-grid FDPB calculations. The distributive multi-block focusing method can then be summarized as the following pseudo procedure:

1. (a) Discretize the problem domain with the coarse grid. Discretize the region of interest with the fine grid.
   (b) Decompose the fine grid into blocks.
   (c) Distribute blocks to computing threads if multiple threads are deployed.
2. Solve the linear system for the coarse grid.
3. Solve the linear systems for the assigned fine-grid blocks in turn.
4. Collect electrostatic energies from blocks.

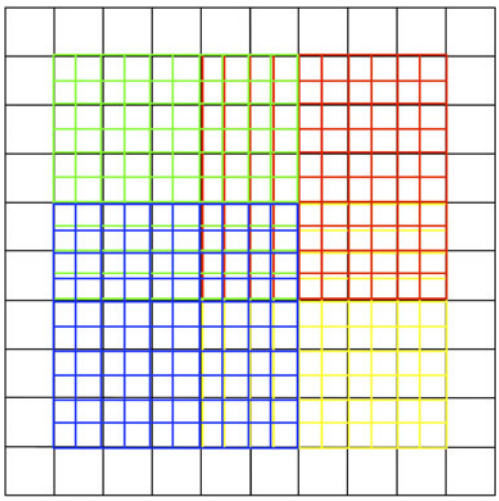

Fig. 1 Four multi-block focusing subdomains (blocks) on a two-dimensional grid. The coarse grid is colored *black*. The fine grid is covered by four overlapping blocks in different colors. The dimension of each fine-grid block is 7×7. The buffering space is 2 grid points thick on all sides of each block. Therefore the actual computed grid number for each block in this example is 11×11

Since the multi-block focusing method is only an approximation to the classical one-block focusing method, it is important to analyze the influence of the approximation on the calculation accuracy while optimizing the algorithm for efficiency. We checked three key data structures in the FDPB algorithm to gain a better understanding of how the approximation may influence accuracy as follows. (1) The coefficient matrix (i.e., the dielectric constant and salt term). It is fairly straightforward to enforce exact numerical consistency between the matrices of the multi-block focusing method and the one-block focusing method if the same grids are used. (2) The constant term (i.e., the atomic charge term). It is also not difficult to enforce numerical consistency between the atomic charge terms in the two focusing methods if the same grids are used. (3) The boundary condition for the fine-grid iteration. This is where the inconsistency may be located; i.e., the boundary grid points are deeply buried inside the solute in the multi-block focusing method, while the boundary grid points are exposed in solvent in the one-block focusing method. Since the boundary potentials are assigned from the coarse-grid solution, the quality of the boundary potentials on the fine grid is clearly different from that in the one-block focusing method, where the boundary grid points are well away from the heterogeneous dielectrics and atomic charges near the solute. In the one-block focusing method, it is a very good approximation to interpolate fine-grid boundary potentials from the coarse-grid potentials. However, in the multi-block focusing method, the distortion might be noticeable when the boundary grid points are buried well inside the solute. Further analysis is certainly necessary before the method is recommended for wide applications.

This is why the buffering space is required in each block to maintain a reasonable accuracy level in the multi-block focusing method. Note that the fine-grid potentials in the

buffering space are neither passed to other computing threads nor used for later energy calculations. Thus, any large error from the distortion of boundary potentials when atomic charges and heterogeneous dielectrics are too close to the edge of the block can be alleviated to a certain degree. Of course, the use of buffering space leads to an extra computing cost in the multi-block focusing method. Thus, on the one hand, the use of smaller fine grid blocks accelerates each FDPB calculation; on the other hand, the use of buffering spaces slows down the overall calculation. In general, there is an optimal block partition scheme that achieves the best efficiency, as shown below in the "Results and discussion."

Distributive implementation of the multi-block electrostatic focusing method

When implementing the multi-block focusing method as a distributive program suitable for multi-thread computing environments, we also paid attention to computing resource usage, particularly in relation to memory management and network communication management. As we mentioned in the "Introduction," one key benefit of the multi-block focusing method is that it reduces the memory needed. A distributive algorithm like the multi-block focusing method undoubtedly divides a large system into smaller and manageable pieces. However, strict memory management is still necessary to achieve the desired benefit (i.e., allocating only the memory needed for the current calculation). Similarly, communication management is also strictly enforced so that only the smallest amount of data required for multi-thread implementation is broadcast. We reduced the communication needs of the distributive multi-block focusing method to several 1-to-N broadcasts of working arrays for initialization, and one N-to-1 broadcast for collecting electrostatic energies in the current implementation.

MICCG solver

The basic iterative solver used in the multi-block focusing method is the MICCG method with the following pseudo program:

```
x_0 := initial guess (normally 0)
r_0 := b - A x_0
solve M r'_0 = r_0
p_0 := r'_0
for i = 0, 1, 2, . . . until convergence, do
      α_i := <r_i, r'_i>/<p_i, A p_i>
      x_{i+1} := x_i + α_i p_i
      r_{i+1} := r_i - α_i A p_i
      if converged then exit
      solve M r'_{i+1} = r_{i+1}
      β_i := <r_{i+1}, r'_{i+1}>/<r_i, r'_i>
      p_{i+1} := r'_{i+1} + β_i p_i
end for
```

where **M** is the preconditioner matrix, ***r*** and ***p*** are auxiliary vectors, and $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ represents the inner product of ***x*** and ***y***. The preconditioner matrix **M** in MICCG can be symbolically written in a factorized form as [83]:

$$\mathbf{M}^{-1} = (\mathbf{E}^{\mathbf{T}} + \mathbf{C}^{\mathbf{T}} + \mathbf{B}^{\mathbf{T}} + \mathbf{D})\,\mathbf{D}^{-1}\,(\mathbf{D} + \mathbf{B} + \mathbf{C} + \mathbf{E}), \quad (4)$$

assuming a 7-band matrix of **A** as

$$\mathbf{A} = \mathbf{E}^{\mathbf{T}} + \mathbf{C}^{\mathbf{T}} + \mathbf{B}^{\mathbf{T}} + \mathbf{diag}(\mathbf{A}) + \mathbf{B} + \mathbf{C} + \mathbf{E}. \quad (5)$$

Here **B** and $\mathbf{B}^{\mathbf{T}}$ are the bands next to the **diag(A)**, **C** and $\mathbf{C}^{\mathbf{T}}$ are *xm* away from the **diag(A)**, and **E** and $\mathbf{E}^{\mathbf{T}}$ are *xmym* away from **diag(A)**. In MICCG, the elements of **D** are computed from a recursive relation as

$$\begin{aligned}\mathbf{D}_{ijk} = \mathbf{diag}(\mathbf{A})_{ijk} &- b_{i-1jk}\left(b_{i-1jk} + \alpha c_{i-1jk} + \alpha e_{i-1jk}\right)/\mathbf{D}_{i-1jk} \\ &- c_{ij-1k}\left(\alpha b_{ij-1k} + c_{ij-1k} + \alpha e_{ij-1k}\right)/\mathbf{D}_{ij-1k} \\ &- e_{ijk-1}\left(\alpha b_{ijk-1} + \alpha c_{ijk-1} + e_{ijk-1}\right)/\mathbf{D}_{ijk-1},\end{aligned} \quad (6)$$

where α was suggested to be 0.95 in the original MICCG implementation [64], while we adopted a value of −0.3 for biomolecules in this study [30].

Other computation details

Four biomolecular systems with nontrivial sizes were used as test cases in this study. These are: the 5S ribosomal RNA of the 70S ribosome (chain B of PDB id: 2J01, denoted rRNA), the p53 DNA binding domain tetramer in complex with DNA (PDB id: 2AC0, denoted p53 DBD), the GROEL heptamer (PDB id: 1SX3, denoted GROEL), and the complete 70S ribosome (PDB id: 2J01+2J02, denoted the 70S ribosome). For the 70S ribosome, the missing residues in the terminal regions were ignored and the gaps within protein subunits were rebuilt with MODELLER [84]. The all-atom models of these systems were built according to the Amber force field, with tRNA and metal ion parameters obtained from the Amber contributed parameters database (http://www.pharmacy.manchester.ac.uk/bryce/amber). All models were subsequently energy-minimized with the Amber molecular dynamics program SANDER. More detailed information on the tested biomolecular systems is listed in Table 1.

In this study, the dielectric constants of the internal and external regions were set to 1 and 80, respectively. The ionic strength was set to be 200 mM for rRNA and the 70S ribosome, and 150 mM for p53 DBD and GROEL. The coarse-grid spacing was 2 Å and the fine-grid spacing was 0.5 Å, respectively, if not otherwise specified. The convergence criterion for finite-difference iteration was set to 10^{-9}, and a two-level focusing technique was used for all the calculations, as described above. The boundary potentials for the coarse-grid FDPB calculation were set to zero.

Table 1 Details for the tested systems. The solute dimensions were estimated by determining the smallest rectangular bounding box that the solute could possibly fit in

Systems	Number of atoms	Solute dimensions (Å^3)
rRNA	3846	86×76×77
p53 DBD	13813	86×84×115
GROEL	54922	148×149×80
70S ribosome	244687	220×265×258

This is only a reasonable approximation for the free space boundary condition when the finite-difference grid boundary is far away from the solute. Thus, the dimension of the coarse grids was set to be at least twice that of the solute to secure a good boundary condition. The fine-grid boundary potentials were interpolated using the coarse-grid potentials. All other computational details can be found in our previous publications on the FDPB method [30–35].

The electrostatic energy was computed as

$$\Delta G_{elec} = \frac{1}{2}\sum_j q_j\phi_j, \tag{7}$$

where j runs over all charged atoms. The electrostatic potential φ_j at atomic charge q_j was computed by solving the linear PBE as in Eq. 2. The reaction field energy was computed as the difference between the electrostatic energy in the solvent dielectric and the electrostatic energy in the uniform solute dielectric. All reported energy values were averaged from the 64 individual computations with random grid origins. The relative accuracies were measured as relative deviations between the mean energy from multi-block focusing FDPB calculations and the mean energy from nonfocusing FDPB calculations. The PBSA module in the Amber 9 package [30–35] was used to implement the multi-block focusing method. LAM/MPI (http://www.lam-mpi.org) was used for parallel implementation. All testing runs were conducted on an Intel Pentium 4 cluster or an Intel Xeon cluster. All data communications were routed with a private gigabit ethernet network, which is separated from the administrative network.

Results and discussion

To assess the accuracy and performance of the distributive multi-block focusing method for FDPB, we first analyzed several factors that are likely to impact the agreement between the multi-block focusing and nonfocusing FDPB calculations. This was followed by our analysis of the efficiency of the distributive method on both single-thread and multi-thread platforms.

Accuracy considerations

To achieve excellent overall numerical quality in the total electrostatic energy with the focusing method, the grid boundary should be set far from the surface of the solute. Otherwise, the potentials on atoms near the grid boundary are often distorted, leading to inaccurate total electrostatic energy. However, the focusing method is also used where only a portion of the solute (i.e., an enzyme active site) is of interest. In such applications, the focusing grid (i.e., the fine grid used here) is defined well within the solute interior. Apparently, for this strategy to be reliable, the focusing grid should be set large enough to cover the region of interest with enough buffering space so that all atoms of interest are located well within the boundary of the focusing grid.

The same consideration also applies to the multi-block focusing calculations, since the block boundaries are well inside the solute to partition the domain of interest into same-sized blocks. Due to this limitation, a thicker buffering space should reduce the numerical inconsistency between the multi-block focusing and nonfocusing FDPB calculations. In addition, a multi-block focusing calculation with larger blocks generally agrees better with the nonfocusing calculation, since there are few boundaries when larger blocks are used. In the following, two selected systems (rRNA and GROEL) were used to assess the influence of both buffering space and block dimension upon the accuracy of the new method.

Figure 2 shows that smaller overall errors can be achieved given a thicker buffering space and/or larger

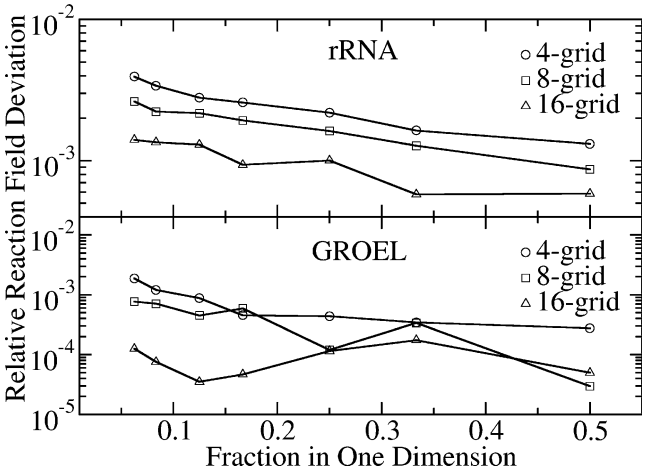

Fig. 2 Relative deviation in reaction field energy with respect to buffering spaces and block sizes. Relative deviation is computed as the difference between the multi-block focusing FDPB calculation and the nonfocusing FDPB computation, and is averaged over 64 runs with random grid offsets. The block size (per dimension) is shown as a fraction of the fine grid, as in the classical one-block focusing calculation

blocks, consistent with our initial estimation above. Specifically, a buffering space of 16 grid points consistently yields good accuracy for all tested block dimensions. Overall, a reasonable relative accuracy of 10^{-3} can be achieved when the buffering space is set to be 16 grid points and the block dimension is set to be at least $(1/6)^3$ of the fine grid in the classical one-block focusing method. This accuracy level is quite acceptable considering that the FDPB convergence error is usually around 10^{-3} at the grid spacing of 0.5 Å often used in biomolecular applications [Wang J, Luo R (2010) Reducing grid-dependence in finite-difference Poisson–Boltzmann calculations, submitted].

Algorithm efficiency on serial platforms

Apparently there is an extra cost just to maintain enough accuracy by using buffering space. Thus, one issue is how to minimize the extra cost while maintaining enough accuracy. In general, multi-block focusing calculations with larger blocks waste less time due to the existence of fewer block boundaries. Of course, the larger blocks have to fit in the physical memory without accessing virtual memory for the calculation to be efficient at all. Even if the larger blocks do fit into the physical memory, in general it takes more time to compute the resulted linear systems due to the larger memory requirement of the FDPB solvers. That is why an "optimal" multi-block size may exist for a given computer hardware setup, and the size is more or less independent of the actual solute or grid geometry, as illustrated in Fig. 3, which shows that the optimal block dimension is ~73^3 for the tested hardware, so this is used in the further testing of the method below. More interestingly, multi-block focusing calculations can be even faster than the single-block focusing calculations on a single CPU when the "optimal" multi-block size is used. For example, the multi-block focusing calculations with the optimal block size in Fig. 3 are about 20~30% faster than the single-block focusing calculations for both geometry settings.

Algorithm efficiency on multi-thread platforms

Due to the distributive nature of the multi-block focusing method, a multi-thread version of the algorithm can be implemented in a straightforward fashion to distribute the fine-grid blocks over multiple computing threads. In this implementation, each thread has all of the necessary data to process the assigned blocks once the coarse-grid FDPB starts (i.e., each thread processes the coarse-grid FDPB redundantly. Thus, there is no cross-thread communication once the coarse-grid FDPB computation starts. Note that this is more beneficial than processing the coarse-grid FDPB computation on the master thread and communicating all data to the slave threads. Given such a setting, there still are other issues that may impact the performance of the multi-block focusing method, as discussed below.

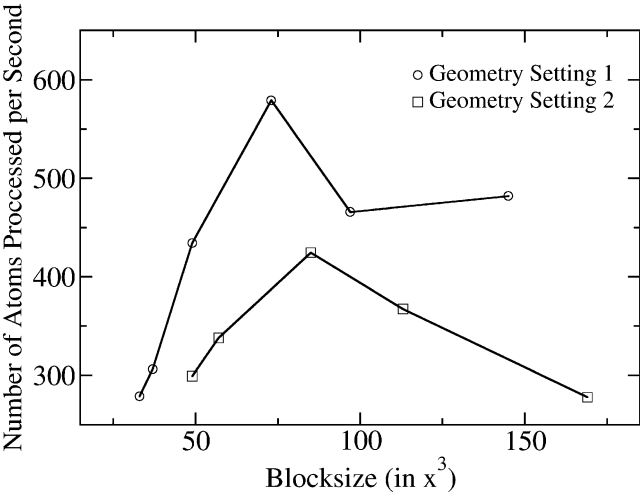

Fig. 3 Fine-grid performance defined as number of atoms processed per second versus block size in the 70S ribosome for a distributive multi-block focusing computation. Two different geometry settings were used to divide the fine grid into multiple blocks in order to analyze the existence of an optimal block size for each grid geometry. It is possible to observe an optimal block size that yields maximum efficiency in both tested geometry settings

Nonparallel overhead

The multi-block focusing method is meant to divide an unmanageably large system into manageable smaller pieces that can be handled by the computer hardware at hand. However, there is nondistributive overhead; i.e., the coarse-grid FDPB that is used to assign boundary grid potential for fine-grid FDPB calculations. Apparently, the computational cost of the coarse-grid FDPB calculation should be less than or at least comparable to individual fine-grid FDPB calculations. Otherwise the coarse-grid FDPB calculation would become the performance bottleneck. This scenario can easily be avoided by using a very coarse FDPB grid. For example, the default coarse grid is eight times as coarse as the fine FDPB grid in the PBSA program. In addition, it is also possible to use more than two levels of FDPB grids if a finer-coarse FDPB grid is desired, a common practice in focusing calculations.

Cache size issue

Another issue of importance to parallel performance is the cache size, because the FDPB runs are memory-

intensive jobs. It is rare that a single FDPB run can fit in the cache memory of a typical machine. This may reduce efficiency on SMP parallel platforms due to the cache memory competition among different threads. We compared the performance of a four-thread calculation on two platforms: one node with two dual-core Intel Xeon processors, and four nodes with identical processors but with only one thread on each node. The p53 DBD and 70S ribosome were used to measure the fine-grid speedups on the two different communication platforms. The physical memory size was set to be larger than the size of each thread, so that no virtual memory was utilized. Table 2 shows that the parallel computation performance over a network of multiple nodes is much better than that over an SMP node of multiple processors, indicating the existence of cache competition.

To confirm the above finding, we analyzed the computational times of several multi-block focusing runs with much smaller grid dimensions on the same hardware. Indeed, if we set the solver memory usage of each thread to less than 2 MB (i.e., half of the cache size per processor), it is possible to observe a noticeable speedup for the SMP parallel run (Table 3). Of course, parallel efficiency deterioration already starts to occur at ~1.8 MB due to other components that may not be 100% swapped out of the cache memory (Table 3).

Of course, larger cache memory does alleviate the limitation of an SMP parallel setup, but it is unlikely to eliminate the limitation because of the large memory usages for typical biomolecular applications. For example, the memory usage is ~146 MB for the block dimension (73^3) used in our current implementation. Thus, our recommendation is to avoid running more threads than the number of physical processors on the same nodes if the cache memory is shared between the different cores on the same physical processor.

Table 2 Fine-grid parallel speedups for the p53 DBD and 70S ribosome with the SMP and TCP/IP methods, respectively

Communication method (system)	2 threads	4 threads
SMP(p53 DBD)	1.411	1.878
TCP/IP(p53 DBD)	1.880	3.826
SMP(70S ribosome)	1.538	1.721
TCP/IP(70S ribosome)	2.123	4.016

The speedups are represented as the ratios of the parallel to the serial computation throughputs. Four identical machines (Dell PE1950III) with two Intel Xeon 5140 processors with two cores were used in the comparison. There were two cores on each processor, which shared 4 MB of shared cache memory

Table 3 Dependence of parallel efficiency upon the solver memory size and number of SMP threads

Memory size	1 thread	2 threads	4 threads
1452 kB	100%	99.94%	99.05%
1819 kB	100%	98.77%	94.41%
2243 kB	100%	95.76%	84.07%
2728 kB	100%	90.42%	66.72%

Solver memory sizes were measured as the total size of working arrays. See Table 2 for the tested hardware. Interestingly, it is still possible to observe degraded parallel efficiency for two-thread jobs with memory sizes of between 2 and 4 MB. This is because the operating system sometimes schedules both threads to the same processors

Load balance

The fine-grid FDPB performance also depends on how the computational load is distributed over threads. One reason for this is that the number of blocks is not always divisible by the number of threads: some threads are assigned more blocks than others. The second reason is related to the heterogeneity of fine-grid blocks, as shown in Fig. 4, which shows that the distribution of per-block computation times is highly scattered for the two tested systems. Indeed, the variation in computation times can be up to 200%.

Thus, a simple sequential partitioning of blocks does not guarantee a balanced load for good performance. To improve load balancing, a randomized partition was used to assign blocks to computing threads. Comparisons of the randomized and sequential partition on the test cases of the p53 DBD and 70S ribosome are shown in Fig. 5. A noticeable benefit can be observed from the randomized partition: the scalability of the algorithm clearly improves

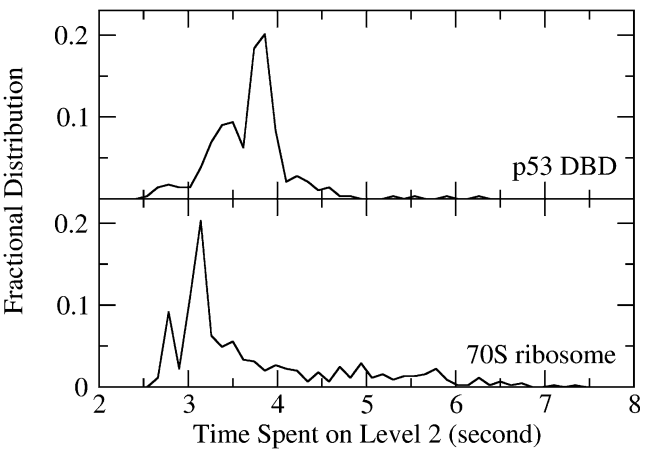

Fig. 4 Distribution of computation times for the fine-grid blocks for the p53 DBD and 70S ribosome, respectively

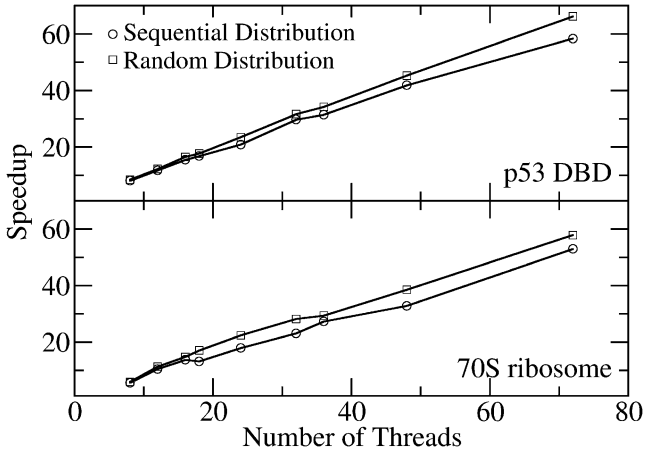

Fig. 5 Comparison of randomized and sequential partitioning in terms of fine-grid parallel speedups for p53 DBD (288 blocks) and 70S ribosome (512 blocks)

when the randomized partition is used. Finally, the overall performance for both tested systems can be found in Fig. 6, which shows a more or less linear speedup over the number of threads used.

Conclusions and future directions

We have implemented and evaluated an approximate distributive method for FDPB calculations of large biomolecular systems. This method is based on the electrostatic focusing principle of decomposing a large fine-grid FDPB calculation into multiple independent FDPB calculations, each of which focuses on only a small and specific portion or block of the large fine grid.

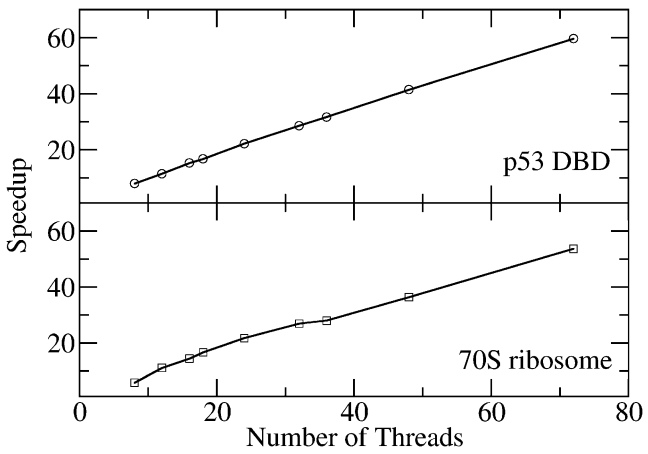

Fig. 6 Overall parallel scaling in the multi-block focusing calculations for p53 DBD (288 blocks) and 70S ribosome (512 blocks), respectively. The scaling of parallel computing is represented by the speedups versus the number of threads used in the computation

We analyzed the impact of the approximation upon the accuracy of the numerical reaction field energies. It was found that a reasonable relative accuracy of 10^{-3} can be achieved when the buffering space is set to be 16 grid points and the block dimension is set to be at least $(1/6)^3$ of the fine-grid dimension, as in the classical one-block focusing method. The impact upon the efficiency of using buffering space to maintain enough accuracy was also studied. It was found that "optimal" multi-block dimensions exist for a given computer hardware setup, and that the dimensions are more or less independent of the actual FDPB grid geometry. A multi-thread version of the distributive multi-block focusing method was also implemented, and its efficiency was analyzed. Interestingly, a noticeable degradation of parallel efficiency was observed on the SMP parallel platforms. This can be attributed to the competition of cache memory. We also evaluated the effect of load balancing on the parallel efficiency. It was found that a random shuffling of blocks over computing threads can be used to improve the scalability of the method on parallel platforms.

Finally, it is interesting to discuss future directions for the distributive computation of FDPB, especially for the MICCG-related FDPB solvers. As reviewed, there are several competing methods for the MICCG solvers: hyperplane ordering [61, 64–67], multi-color ordering [69, 70], and blocked red–black ordering [71]. The beauty of grouping grid points into hyperplanes is that all grid points in a hyperplane can be computed independently of each other. More interestingly, the computation order in the hyperplane ordering is equivalent to that of the natural ordering, which is crucial for the fast convergence of MICCG methods. However, this method is probably attractive only on shared memory parallel platforms, due to the high level of data dependency. The multi-color ordering was proposed as an improvement of the red–black ordering, in order to strike a balance between data independence and convergence [85]. In this strategy, unknowns of the same color have no data relationship to each other, so that the substitutions in MICCG can be independently computed. In the blocked red–black ordering, grid points are labeled in blocks of colors of both red and black. The natural ordering is used within each block. Since blocks with the same color do not have a data dependency, these blocks can be independently processed. Doing so reduces data transfer frequencies between blocks of different colors, especially when the block size is large [71]. Due to the low data dependency between blocks, this method should be suitable for distributed parallel platforms. We are implementing and analyzing all these strategies to analyze their scaling for the distributive computing of realistic biomolecular applications.

Acknowledgments This work is supported in part by the National Institutes of Health (GM079383 & GM093040).

References

1. Davis ME, McCammon JA (1990) Electrostatics in biomolecular structure and dynamics. Chem Rev 90:509–521
2. Sharp KA (1994) Electrostatic interactions in macromolecules. Curr Opin Struct Biol 4:234–239
3. Gilson MK (1995) Theory of electrostatic interactions in macromolecules. Curr Opin Struct Biol 5:216–223
4. Honig BH, Nicholls A (1995) Classical electrostatics in biology and chemistry. Science 268:1144–1149
5. Roux B, Simonson T (1999) Implicit solvent models. Biophys Chem 78:1–20
6. Cramer CJ, Truhlar DG (1999) Implicit solvation models: Equilibria, structure, spectra, and dynamics. Chem Rev 99:2161–2200
7. Bashford D, Case DA (2000) Generalized born models of macromolecular solvation effects. Annu Rev Phys Chem 51:129–152
8. Baker NA (2005) Improving implicit solvent simulations: A Poisson-centric view. Curr Opin Struct Biol. doi:10.1016/j.sbi.2005.02.001
9. Chen J, Im W, Brooks CL III (2006) Balancing solvation and intramolecular interactions: toward a consistent generalized born force field. J Am Chem Soc 128:3728–3736
10. Feig M, Chocholousova J, Tanizaki S (2006) Extending the horizon: towards the efficient modeling of large biomolecular complexes in atomic detail. Theor Chem Acc. doi:10.1007/s00214-005-0062-4
11. Im W, Chen J, Brooks CL III (2006) Peptide and protein folding and conformational equilibria: theoretical treatment of electrostatics and hydrogen bonding with implicit solvent models. Adv Protein Chem. doi:10.1016/S0065-3233(05)72007-6
12. Koehl P (2006) Electrostatics calculations: latest methodological advances. Curr Opin Struct Biol. doi:10.1016/j.sbi.2006.03.001
13. Lu BZ, Zhou YC, Holst MJ, McCammon JA (2008) Recent progress in numerical methods for the Poisson–Boltzmann equation in biophysical applications. Commun Comput Phys 3:973–1009
14. Wang J, Tan CH, Tan Y, Lu Q, Luo R (2008) Poisson–Boltzmann solvents in molecular dynamics simulations. Commun Comput Phys 3:1010–1031
15. Hill TL (1986) Dilute electrolyte solutions and plasmas. In: An introduction to statistical thermodynamics. Dover, New York, pp 321–339
16. Gilson MK, Sharp KA, Honig BH (1988) Calculating the electrostatic potential of molecules in solution: method and error assessment. J Comput Chem 9:327–335
17. Nicholls A, Honig BH (1991) A rapid finite-difference algorithm, utilizing successive over-relaxation to solve the Poisson–Boltzmann equation. J Comput Chem 12:435–445
18. Rocchia W, Alexov E, Honig BH (2001) Extending the applicability of the nonlinear Poisson–Boltzmann equation: multiple dielectric constants and multivalent ions. J Phys Chem B 105:6507–6514
19. Davis ME, McCammon JA (1989) Solving the finite difference linearized Poisson–Boltzmann equation: a comparison of relaxation and conjugate gradient methods. J Comput Chem 10:386–391
20. Davis ME, Madura JD, Luty BA, McCammon JA (1991) Electrostatics and diffusion of molecules in solution—simulations with the University of Houston Brownian dynamics program. Comput Phys Commun 62:187–197
21. Luty BA, Davis ME, McCammon JA (1992) Solving the finite-difference non-linear Poisson–Boltzmann equation. J Comput Chem 13:1114–1118
22. Madura JD, Briggs JM, Wade RC, Davis ME, Luty BA, Ilin A, Antosiewicz J, Gilson MK, Bagheri B, Scott LR, McCammon JA (1995) Electrostatics and diffusion of molecules in solution—simulations with the University of Houston Brownian dynamics program. Comput Phys Commun 91:57–95
23. Bashford D, Karplus M (1990) pK_a's of ionizable groups in proteins: atomic detail from a continuum electrostatic model. Biochemistry 29:10219–10225
24. Bashford D (1997) An object-oriented programming suite for electrostatic effects in biological molecules—an experience report on the Mead Project. In: Proc Int Conf Scientific Computing in Object-Oriented Parallel Environments, Marina del Ray, CA, USA, 8–11 Dec 1997. doi:10.1007/3-540-63827-X_66
25. Roux B (1997) Influence of the membrane potential on the free energy of an intrinsic protein. Biophys J 73:2980–2989
26. Im W, Beglov D, Roux B (1998) Continuum solvation model: computation of electrostatic forces from numerical solutions to the Poisson–Boltzmann equation. Comput Phys Commun 111:59–75
27. Cortis CM, Friesner RA (1997) Numerical solution of the Poisson–Boltzmann equation using tetrahedral finite-element meshes. J Comput Chem 18:1591–1608
28. Holst MJ, Baker NA, Wang F (2000) Adaptive multilevel finite element solution of the Poisson–Boltzmann equation. I. Algorithms and examples. J Comput Chem 21:1319–1342
29. Baker NA, Holst MJ, Wang F (2000) Adaptive multilevel finite element solution of the Poisson–Boltzmann equation. II. Refinement at solvent-accessible surfaces in biomolecular systems. J Comput Chem 21:1343–1352
30. Luo R, David L, Gilson MK (2002) Accelerated Poisson–Boltzmann calculations for static and dynamic systems. J Comput Chem 23:1244–1253
31. Lu Q, Luo R (2003) A Poisson–Boltzmann dynamics method with nonperiodic boundary condition. J Chem Phys 119:11035–11047. doi:10.1063/1.1622376
32. Cai Q, Wang J, Zhao HK, Luo R (2009) On removal of charge singularity in Poisson–Boltzmann equation. J Chem Phys. doi:10.1063/1.3099708
33. Cai Q, Hsieh MJ, Wang J, Luo R (2010) Performance of nonlinear finite-difference Poisson–Boltzmann solvers. J Chem Theor Comput. doi:10.1021/Ct900381r
34. Wang J, Tan CH, Chanco E, Luo R (2010) Quantitative analysis of Poisson–Boltzmann implicit solvent in molecular dynamics. Phys Chem Chem Phys. doi:10.1039/B917775b
35. Wang J, Luo R (2010) Assessment of linear finite-difference Poisson–Boltzmann solvers. J Comput Chem. doi:10.1002/Jcc.21456
36. Klapper I, Hagstrom R, Fine R, Honig BH (1986) Focusing of electric fields in the active site of Cu-Zn superoxide dismutase: effects of ionic strength and amino-acid modification. Proteins 1:47–59
37. Holst MJ, Saied F (1993) Multigrid solution of the Poisson–Boltzmann equation. J Comput Chem 14:105–113
38. Forsten KE, Kozack RE, Lauffenburger DA, Subramaniam S (1994) Numerical solution of the nonlinear Poisson–Boltzmann equation for a membrane–electrolyte system. J Phys Chem 98:5580–5586
39. Shestakov AI, Milovich JL, Noy A (2002) Solution of the nonlinear Poisson–Boltzmann equation using pseudo-transient

continuation and the finite element method. J Colloid Interf Sci. doi:10.1006/jcis.2001.8033
40. Chen L, Holst MJ, Xu J (2007) The finite element approximation of the nonlinear Poisson–Boltzmann equation. SIAM J Numer Anal. doi:10.1137/060675514
41. Xie D, Zhou S (2007) A new minimization protocol for solving nonlinear Poisson–Boltzmann mortar finite element equation. BIT Numer Math. doi:10.1007/s10543-007-0145-9
42. Miertuš S, Scrocco E, Tomasi J (1981) Electrostatic interaction of a solute with a continuum. A direct utilization of ab initio molecular potentials for the prevision of solvent effects. Chem Phys 55:117–129
43. Hoshi H, Sakurai M, Inoue Y, Chûjô R (1987) Medium effects on the molecular electronic structure. I. The formulation of a theory for the estimation of a molecular electronic structure surrounded by an anisotropic medium. J Chem Phys 87:1107–1115
44. Zauhar RJ, Morgan RS (1988) The rigorous computation of the molecular electric potential. J Comput Chem 9:171–187
45. Rashin AA (1990) Hydration phenomena, classical electrostatics, and the boundary element method. J Phys Chem 94:1725–1733
46. Yoon BJ, Lenhoff AM (1990) A boundary element method for molecular electrostatics with electrolyte effects. J Comput Chem 11:1080–1086
47. Juffer AH, Botta EF, van Keulen BAM, van der Ploeg A, Berendsen HJC (1991) The electric potential of a macromolecule in a solvent: a fundamental approach. J Comput Phys 97:144–171
48. Zhou HX (1993) Boundary element solution of macromolecular electrostatics: interaction energy between two proteins. Biophys J 65:955–963
49. Bharadwaj R, Windemuth A, Sridharan S, Honig BH, Nicholls A (1995) The fast multipole boundary-element method for molecular electrostatics—an optimal approach for large systems. J Comput Chem 16:898–913
50. Purisima EO, Nilar SH (1995) A simple yet accurate boundary element method for continuum dielectric calculations. J Comput Chem 16:681–689
51. Liang J, Subramaniam S (1997) Computation of molecular electrostatics with boundary element methods. Biophys J 73:1830–1841
52. Vorobjev YN, Scheraga HA (1997) A fast adaptive multigrid boundary element method for macromolecular electrostatic computations in a solvent. J Comput Chem 18:569–583
53. Totrov M, Abagyan R (2001) Rapid boundary element solvation electrostatics calculations in folding simulations: successful folding of a 23-residue peptide. Biopolymers 60:124–133
54. Boschitsch AH, Fenley MO, Zhou HX (2002) Fast boundary element method for the linear Poisson–Boltzmann equation. J Phys Chem B. doi:10.1021/jp013607q
55. Lu BZ, Cheng XL, Huang JF, McCammon JA (2006) Order *n* algorithm for computation of electrostatic interactions in biomolecular systems. Proc Natl Acad Sci USA. doi:10.1073/pnas.0605166103
56. Strang G (1988) Iterative method for *ax=b*. In: Linear algebra and its application, 3rd edn. Brooks/Cole, Pacific Grove, pp 380–387
57. Evans DJ (1984) Parallel S.O.R. iterative methods. Parallel Comput. doi:10.1016/S0167-8191(84)90380-6
58. O'Leary DP, White RE (1985) Multi-splittings of matrices and parallel solution of linear systems. SIAM J Alg Discr Meth 6:630–640
59. Meurant G (1999) Computer solution of large linear systems. Elsevier, Amsterdam
60. Dongarra JJ, Duff IS, Sorensen DC, Vorst HVD (1991) Iterative solution of sparse linear systems. In: Solving linear systems on vector and shared memory computers. Society for Industrial & Applied Mathematics, Philadelphia, pp 143–190
61. van der Vorst HA, Chan T (1996) Parallel preconditioning for sparse linear equations. Z Angew Math Mech. doi:10.1002/zamm.19960760315
62. Meijerink JA, van der Vorst HA (1977) An iterative solution method for linear-systems of which coefficient matrix is a symmetric m-matrix. Math Comput 31:148–162
63. Chan TF, Kuo CJ, Tong C (1989) Parallel elliptic preconditioners: Fourier analysis and performance on the connection machine. Comput Phys Commun. doi:10.1016/0010-4655(89)90163-X
64. van der Vorst HA (1989) High performance preconditioning. SIAM J Sci Stat Comput 10:1174–1185
65. Traar KP, Mader W, Heinreichsberger O, Selberherr S, Stiftinger M (1990) High performance preconditioning on supercomputers for the 3D device simulator MINIMOS. Supercomputing '90 (Proc 1990 ACM/IEEE Conf on Supercomputing), New York, USA, 12–16 Nov 1990. doi:10.1109/SUPERC.1990.130024
66. Fujino S, Mori M, Takeuchi T (1991) Performance of hyperplane ordering on vector computers. J Comput Appl Math 38:125–136
67. Dongarra JJ, Duff IS, Sorensen DC, van der Vorst HA (1998) Preconditioning and parallel preconditioning. In: Numerical linear algebra for high-performance computers. Society for industrial and Applied Mathematics, Philadelphia, pp 215–230
68. Kuo CJ, Chan TF (1990) Two-color Fourier analysis of iterative algorithms for elliptic problems with red/black ordering. SIAM J Sci Stat Comput 11:767–793
69. Doi S (1991) On parallelism and convergence of incomplete LU factorizations. Appl Numer Math 7:417–436
70. Ma S (2008) A performance comparison of the parallel preconditioners for iterative methods for large sparse linear systems arising from partial differential equations on structured grids. Ieice T Fund Electr. doi:10.1093/ietfec/e91-a.9.2578
71. Iwashita T, Shimasaki M (2002) Block red–black ordering method for parallel processing of ICCG solver. High Perf Comput 2327:297–300. doi:10.1007/3-540-47847-7_16
72. van der Vorst HA (1982) A vectorizable variant of some ICCG methods. SIAM J Sci Stat Comput 3:350–356
73. Bruaset A (1995) A survey of preconditioned iterative methods. Longman, New York
74. Radicati di Brozolo G, Robert Y (1989) Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric linear-systems on a vector multiprocessor. Parallel Comput 11:223–239
75. Chen W, Poirier B (2006) Parallel implementation of efficient preconditioned linear solver for grid-based applications in chemical physics. I. Block Jacobi diagonalization. J Comput Phys. doi:10.1016/j.jcp.2006.04.012
76. Chen W, Poirier B (2006) Parallel implementation of efficient preconditioned linear solver for grid-based applications in chemical physics. II. QMR linear solver. J Comput Phys. doi:10.1016/j.jcp.2006.03.031
77. Brandt A (1973) Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. Proc Third Int Conf on Numerical Methods in Fluid Mechanics, Universities of Paris VI and XI, 3–7 July 1972. doi:10.1007/BFb0118663
78. Brandt A (1977) Multi-level adaptive solutions to boundary-value problems. Math Comput 31:333–390
79. Barrett R, Berry M, Chan TF, Demmel JW, Donato JM, Dongarra JJ, Eijkhout V, Pozo R, Romine C, van der Vorst HA (1994) Templates for the solution of linear systems: building blocks for iterative methods. SIAM, Philadelphia
80. Briggs WL, Henson VE, McCormick SF (2000) A multigrid tutorial, 2nd edn. SIAM, Philadelphia
81. Trottenberg U, Oosterlee CW, Schüller A (2001) Multigrid. Academic, London

82. Baker NA, Sept D, Joseph S, Holst MJ, McCammon JA (2001) Electrostatics of nanosystems: application to microtubules and the ribosome. Proc Natl Acad Sci USA 98:10037–10041
83. van der Vorst HA (1989) ICCG and related methods for 3D problems on vector computers. Comput Phys Commun. doi:10.1016/0010-4655(89)90162-8
84. Fiser A, Do RKG, Săli A (2000) Modeling of loops in protein structures. Protein Sci 9:1753–1773
85. Doi S, Washio T (1999) Ordering strategies and related techniques to overcome the trade-off between parallelism and convergence in incomplete factorizations. Parallel Comput 25:1995–2014