

# **Performance of Nonlinear Finite-Difference Poisson-Boltzmann Solvers**

Qin Cai,<sup>1,2</sup> Meng-Juei Hsieh,<sup>2</sup> Jun Wang,<sup>2</sup> and Ray Luo<sup>1,2,\*</sup>

1. Department of Biomedical Engineering
  2. Department of Molecular Biology and Biochemistry
- University of California, Irvine, CA 92697

\* Please send correspondence to R. Luo. email: rluo@uci.edu; fax: (949) 824-9551.

## **Abstract**

We implemented and optimized seven finite-difference solvers for the full nonlinear Poisson-Boltzmann equation in biomolecular applications, including four relaxation methods, one conjugate gradient method, and two inexact Newton methods. The performance of the seven solvers was extensively evaluated with a large number of nucleic acids and proteins. Worth noting is the inexact Newton method in our analysis. We investigated the role of linear solvers in its performance by incorporating the incomplete Cholesky conjugate gradient and the geometric multigrid into its inner linear loop. We tailored and optimized both linear solvers for faster convergence rate. In addition, we explored strategies to optimize the successive over-relaxation method to reduce its convergence failures without too much sacrifice in its convergence rate. Specifically we attempted to adaptively change the relaxation parameter and to utilize the damping strategy from the inexact Newton method to improve the successive over-relaxation method. Our analysis shows that the nonlinear methods accompanied with a functional-assisted strategy, such as the conjugate gradient method and the inexact Newton method, can guarantee convergence in the tested molecules. Especially the inexact Newton method exhibits impressive performance when it is combined with highly efficient linear solvers that are tailored for its special requirement

## Introduction

Electrostatic interaction plays a key role in determining the structure and function of biomolecules.<sup>1-14</sup> However, modeling of electrostatic interaction in biomolecules remains to be a serious computational challenge. The difficulty in modeling a biomolecular system resides in its high dimensionality, especially when explicit solvents are used. Explicit solvents can provide a realistic description of the solution system but require expensive computational resources for biomolecules of interest. In contrast, implicit solvent representation reduces the system degrees of freedom by capturing the average or continuum behavior of the solvent. To model the electrostatic interaction in the salt water solution, the Poisson-Boltzmann equation (PBE) has been widely used:

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

where  $\varepsilon$  is the dielectric constant,  $\phi$  is the electrostatic potential,  $\rho_0$  is the solute charge density,  $\lambda$  is the ion-exclusion function with values of 0 within the Stern layer and 1 outside the Stern layer,  $e$  is the unit charge,  $z_i$  is the valence of ion type  $i$ ,  $c_i$  is the number density of ion type  $i$ ,  $k_B$  is the Boltzmann constant, and  $T$  is the absolute temperature.

For a solution with symmetric 1:1 salt, eqn. (1) can be simplified to

$$\nabla \cdot \varepsilon \nabla \phi = -4\pi\rho_0 + \lambda \frac{\varepsilon_{out} \kappa^2}{C} \sinh(C\phi), \quad (2)$$

where  $\kappa^2 = \frac{8\pi e^2 I}{\varepsilon_{out} k_B T}$  and  $C = \frac{ez}{k_B T}$ . Here “out” denotes the outside solvent,  $I$  represents

the ionic strength of the solution and  $I = z^2 c$ . If the electrostatic potential is weak and the

ionic strength is low, the nonlinear PBE can be simplified to the linearized form<sup>15</sup>

$$\nabla \cdot \varepsilon \nabla \phi = -4\pi\rho_0 + \lambda \varepsilon_{out} \kappa^2 \phi. \quad (3)$$

The linearized PBE is easier to solve but it is not very accurate in modeling highly charged biomolecules, such as nucleic acids, while the nonlinear PBE predictions have been shown to yield good agreement with experiments and explicit ion simulations.<sup>16-20</sup> Solution of the nonlinear PBE has attracted much attention in the past. Just as linear PBE solvers, these methods can also be grouped into three categories according to how the PBE is discretized, that is, the finite difference method (FDM),<sup>17,18,21-27</sup> the finite element method (FEM),<sup>28-34</sup> and the boundary element method (BEM).<sup>35-37</sup> A combination of the FDM and the BEM<sup>38</sup> was also reported. Some of these methods have been incorporated into the widely used PB programs, including Delphi,<sup>22,27</sup> UHBD,<sup>23</sup> PBEQ,<sup>22</sup> and APBS.<sup>28,29</sup> This study intends to evaluate the existing nonlinear FDM solvers and explore strategies to improve their performance.

After the nonlinear PBE is discretized with the FDM, a nonlinear system is generated as follows

$$\begin{aligned} & \varepsilon_{i-1,j,k}^x [\phi_{i,j,k} - \phi_{i-1,j,k}] + \varepsilon_{i,j,k}^x [\phi_{i,j,k} - \phi_{i+1,j,k}] \\ & \varepsilon_{i,j-1,k}^y [\phi_{i,j,k} - \phi_{i,j-1,k}] + \varepsilon_{i,j,k}^y [\phi_{i,j,k} - \phi_{i,j+1,k}] \\ & \varepsilon_{i,j,k-1}^z [\phi_{i,j,k} - \phi_{i,j,k-1}] + \varepsilon_{i,j,k}^z [\phi_{i,j,k} - \phi_{i,j,k+1}] \quad , \\ & + \lambda \frac{\varepsilon_{out} \kappa^2 h^2}{C} \sinh(C\phi_{i,j,k}) = 4\pi q_{i,j,k} / h \end{aligned} \quad (4)$$

where  $i, j$ , and  $k$  are the grid indexes along  $x, y$  and  $z$  axes, respectively.  $\varepsilon_{i,j,k}^x$  is the dielectric

constant between grids  $(i, j, k)$  and  $(i+1, j, k)$ .  $\varepsilon_{i,j,k}^y$  and  $\varepsilon_{i,j,k}^z$  are defined similarly.  $h$  is the grid spacing in each dimension.  $\phi_{i,j,k}$  is the potential at  $(i, j, k)$ .  $q_{i,j,k}$  is the total charge within the cubic volume centered at  $(i, j, k)$ . The nonlinear system can then be denoted as

$$A\phi + N(\phi) = b, \quad (5)$$

where  $A$  is the coefficient matrix for the linear part of the PBE, which is a positive-definite matrix,  $\phi$  is the potential vector,  $b$  is the free charge vector, and  $N(\bullet)$  denotes the nonlinear term in the PBE. The discretized form of PBE can be solved by several numerical methods, such as the nonlinear relaxation methods,<sup>17,18,21,22,25-27</sup> the nonlinear conjugate gradient method,<sup>23</sup> the nonlinear multigrid method,<sup>24</sup> and the inexact Newton method.<sup>16</sup> The relaxation methods, extended from classical linear methods such as Gauss-Seidel and successive over-relaxation, were first attempted to solve the FDM version of the nonlinear PBE.<sup>18,22</sup> However, the convergence of such methods cannot be guaranteed.<sup>16</sup> The multigrid method was also attempted,<sup>24</sup> but it may diverge on certain applications.<sup>16</sup> More robust approaches, such as the conjugate gradient method<sup>23</sup> and the inexact Newton method<sup>16</sup> were also reported for biomolecular applications. The conjugate gradient method is very slow due to the considerable evaluations of the exponential function. The inexact Newton method is very attractive and is proven to converge.<sup>16</sup> More importantly, the inexact Newton method can be combined with the highly efficient linear FDM solvers to yield highly efficient methods.

Despite the early introduction of various nonlinear PBE solvers to biomolecular applications, a comparative and extensive analysis of these solvers is still in need. Such an

analysis can guide future development and application of nonlinear PBE solvers. In this study we implemented, evaluated, and improved if possible seven nonlinear PBE solvers in the FDM scheme. In the following, the tested algorithms of the nonlinear PBE solvers are first summarized. This is followed by a comprehensive analysis of their convergence and performance with a large number of high-quality crystal structures of DNAs, RNAs, and proteins.

## Methods

The discretized PBE, eqn. (5), is often solved iteratively in the following form

$$\phi^{t+1} = \phi^t + \delta\phi^t, \quad (6)$$

where  $\delta\phi^t$  is a update of  $\phi^t$  at the  $t$ -th iteration. The conjugate gradient method follows a minimization strategy. It first intends to find an update  $\delta\phi^t$ , which is  $A$ -conjugate to all previous updates if the nonlinear term is eliminated. Once  $\delta\phi^t$  is obtained at the  $t$ -th iteration, a line search along the direction  $\delta\phi^t$  is conducted to minimize a pre-defined functional. In contrast, the inexact Newton method and the relaxation method are both derivatives of the root-finding Newton method for nonlinear functions, which uses the first-order Taylor expansion of a nonlinear function,  $g(\phi) = A\phi + N(\phi) - b$ , to obtain an appropriate update  $\delta\phi^t$ . The inexact Newton method requires  $\delta\phi^t$  to be scaled to descend a functional that is closely related to that used in the conjugate gradient method.

### Conjugate Gradient Algorithm

Luty et al first explored to use the nonlinear conjugate gradient (CG) method to solve the nonlinear PBE.<sup>23</sup> The nonlinear conjugate gradient is derived from the linear conjugate gradient method in a straightforward fashion. The CG method tries to solve a minimization problem. The pre-defined functional  $G(\phi)$  to be minimized is the integral form of  $g(\phi)$ :

$$\min_{\phi} \left\{ G(\phi) : G(\phi) = \frac{1}{2} \phi A \phi + \Delta \Pi - b \phi \right\}. \quad (7)$$

Here  $\Delta \Pi = \sum_{i,j,k} \int N(\phi_{i,j,k}) d\phi_{i,j,k}$ . Therefore, the stationary point of  $G(\phi)$  is also the solution of  $g(\phi) = 0$ . In the Fletcher-Reeves version of the CG method,  $\delta\phi^t$  is computed in the following way:<sup>39</sup>

$$\delta\phi^t = -g(\phi^t) + \beta^t \delta\phi^{t-1} \quad (8)$$

where  $\beta^t = \frac{(g(\phi^t), g(\phi^t))}{(g(\phi^{t-1}), g(\phi^{t-1}))}$ , which is used to enforce the  $A$ -conjugacy between  $\delta\phi^t$  and all previous updates in the linear case.  $(a, b)$  denotes the dot product of two vectors  $a$  and  $b$ . Note that  $\delta\phi^t$  is scaled so that eqn. (6) becomes

$$\phi^{t+1} = \phi^t + \alpha^t \delta\phi^t \quad (9)$$

where  $\alpha^t$  is the scaling factor. By tuning  $\alpha^t$ , the functional  $G(\phi)$  is minimized. The pseudo-code for the nonlinear CG algorithm is given as follows

For  $i, j, k$  from 1 to  $xm, ym, zm$

$$\begin{aligned} \varepsilon_{i,j,k} &= \varepsilon_{i-1,j,k}^x + \varepsilon_{i,j-1,k}^y + \varepsilon_{i,j,k-1}^z + \varepsilon_{i,j,k}^x + \varepsilon_{i,j,k}^y + \varepsilon_{i,j,k}^z \\ g_{i,j,k}^0 &= \varepsilon_{i,j,k} \phi_{i,j,k}^0 - \varepsilon_{i-1,j,k}^x \phi_{i-1,j,k}^0 - \varepsilon_{i,j-1,k}^y \phi_{i,j-1,k}^0 - \varepsilon_{i,j,k-1}^z \phi_{i,j,k-1}^0 \\ &\quad - \varepsilon_{i,j,k}^x \phi_{i+1,j,k}^0 - \varepsilon_{i,j,k}^y \phi_{i,j+1,k}^0 - \varepsilon_{i,j,k}^z \phi_{i,j,k+1}^0 - 4\pi q_{i,j,k} / h \end{aligned}$$

```


$$\delta\phi_{i,j,k}^0 = -g_{i,j,k}^0$$

End for i, j, k
Do until convergence
  For i, j, k from 1 to xm, ym, zm
    
$$\sigma_{i,j,k} = \varepsilon_{i,j,k} \delta\phi_{i,j,k}^t - \varepsilon_{i-1,j,k}^x \delta\phi_{i-1,j,k}^t - \varepsilon_{i,j-1,k}^y \delta\phi_{i,j-1,k}^t - \varepsilon_{i,j,k-1}^z \delta\phi_{i,j,k-1}^t$$

    
$$- \varepsilon_{i,j,k}^x \delta\phi_{i+1,j,k}^t - \varepsilon_{i,j,k}^y \delta\phi_{i,j+1,k}^t - \varepsilon_{i,j,k}^z \delta\phi_{i,j,k+1}^t$$

  End for i, j, k
  Do until convergence
    Solve  $(\delta\phi^t, g^t) + (\delta\phi^t, N(\phi^t + \alpha^t \delta\phi^t) - N(\phi^t)) + \alpha^t (\delta\phi^t, \sigma) = 0$ 
    for  $\alpha^t$  using Newton's root-finding method
  End do
  For i, j, k from 1 to xm, ym, zm
    
$$\phi_{i,j,k}^{t+1} = \phi_{i,j,k}^t + \alpha^t \delta\phi_{i,j,k}^t$$

    
$$g_{i,j,k}^{t+1} = g_{i,j,k}^t + N(\phi_{i,j,k}^{t+1}) - N(\phi_{i,j,k}^t) + \alpha^t \sigma$$

  End for i, j, k
  
$$\beta^{t+1} = (g^{t+1}, g^{t+1}) / (g^t, g^t)$$

  For i, j, k from 1 to xm, ym, zm
    
$$\delta\phi_{i,j,k}^{t+1} = -g_{i,j,k}^{t+1} + \beta^{t+1} \delta\phi_{i,j,k}^t$$

  End for i, j, k
  
$$t = t + 1$$

End do

```

where the superscript 0 represents the initial value and the superscript  $t$  represents the value at the  $t$ -th iteration. Here the inner Newton iterations are employed to find the scaling factor  $\alpha^t$  that minimizes  $G(\phi)$ . Luty et al showed that the nonlinear CG solver was about four times slower than the linear CG solver with otherwise identical conditions.<sup>23</sup>

### Inexact Newton Method

The inexact Newton method starts from the standard Newton method. The first-order

Taylor expansion of  $g(\phi)$  at  $\phi = \phi'$  gives

$$g(\phi' + \delta\phi') = g(\phi') + \left[ g'(\phi) \Big|_{\phi'} \right] \delta\phi' = g(\phi') + \left[ A + N'(\phi') \right] \delta\phi', \quad (10)$$

where  $N'(\phi)$  is the Jacobian matrix of the vector  $N(\phi)$ , and  $N'(\phi)$  is a diagonal matrix in this case. The ideal  $\delta\phi'$  would make the new  $\phi'^{+1}$  the root of  $g(\phi) = 0$ . Thus we have

$$\left[ A + N'(\phi') \right] \delta\phi' = -g(\phi'). \quad (11)$$

In eqn. (11), the inverse of  $\left[ A + N'(\phi') \right]$  is difficult to compute and the corresponding  $\delta\phi'$  cannot be obtained within a few iterations, but it is actually unnecessary to solve eqn. (11) precisely for  $\delta\phi'$ . Eqn. (11) is solved iteratively to the extent that a pre-defined functional  $f(\phi)$  is ensured to decrease in the update direction  $\delta\phi'$ , or in other words, a descent direction of  $f(\phi)$  is found. Although the integral of  $g(\phi)$  in the above nonlinear CG algorithm is a natural choice for the functional,<sup>23</sup> a simpler form is also effective,<sup>16</sup>

$$\min_{\phi} \left\{ f(\phi) : f(\phi) = \frac{1}{2} g(\phi)^T \cdot g(\phi) \right\} \quad (12)$$

It has been proven that if the following condition is satisfied

$$\left\| \left[ A + N'(\phi') \right] \delta\phi' + g(\phi') \right\| < \|g(\phi')\|, \quad (13)$$

a descent direction of  $f(\phi)$  can always be obtained.<sup>16</sup> Next a line search along the descent direction is conducted to assure  $f(\phi'^{+1}) < f(\phi')$ . There are various ways to solve eqn. (11) inexactly, such as the multigrid (MG) method, the incomplete Cholesky conjugate gradient (ICCG) method, and the successive over-relaxation (SOR) method. In summary, the NT

algorithm can be written as follows.

```

Do until convergence
  Calculate the nonlinear PB residual  $g(\phi')$ .
  Calculate the Jacobian matrix  $N'(\phi')$ .
  Let  $A^L(\phi') = A(\phi') + N'(\phi')$ .
  Iteratively solve a linear problem  $A^L(\phi')\delta\phi' = -g(\phi')$  for  $\delta\phi'$ 
    until  $\|A^L(\phi')\delta\phi' + g(\phi')\| < \|g(\phi')\|$ 
  Conduct line search along the direction  $\delta\phi'$ , i.e. looking
    for  $\alpha'$ , to satisfy  $\|f(\phi' + \alpha'\delta\phi')\| < \|f(\phi')\|$ 
End do

```

The two NT solvers tested in this study are combined with ICCG (NT-ICCG) and MG (NT-MG), respectively, which are employed to solve the inner linear problem for updating the potentials. The ICCG method is an optimized version by Luo et al.<sup>40</sup> In the MG method, we applied a four-level v-cycle implementation,<sup>41</sup> where the restriction and prolongation were realized with a three-dimensional seven-banded version of Alcouffe's algorithm.<sup>42,43</sup> We employed the SOR method for the MG pre-smoothing and post-smoothing on fine grids, and also for solving the linear problem on the coarsest grids. The relaxation parameter was set as 1.5 on fine grids and 1.9 on the coarsest grids. Both the pre-smoothing and post-smoothing use five SOR steps. Because eqn. (11) is solved inexactly, we adopted this simple and fast algorithm which would be otherwise unstable and unsuitable to solve a normal linear problem with tight convergence criterion.<sup>41</sup> Specifically the convergence criterion for the NT-MG method was set as  $\| [A + N'(\phi')] \delta\phi' + g(\phi') \| < \| g(\phi') \|$ , and the convergence criterion for

the NT-ICCG method was set as  $\| [A + N'(\phi')] \delta\phi' + g(\phi') \| < 0.1 \times \| g(\phi') \|$ .

### Relaxation Algorithms

Unlike the inexact Newton method, a nonlinear relaxation method uses a matrix  $B$  that is approximate to  $[A + N'(\phi')]^{-1}$  in eqn. (11). Specifically, for the nonlinear SOR method,

$$B = \omega [D + \omega L + N'(\phi')]^{-1} . \quad (14)$$

For the nonlinear Jacobi method,

$$B = [D + N'(\phi')]^{-1} . \quad (15)$$

For the nonlinear Gauss-Seidel (GS) method,

$$B = [D + L + N'(\phi')]^{-1} . \quad (16)$$

In eqn. (14)–(16),  $D$  is a diagonal matrix,  $L$  is a strictly lower triangular matrix, so that  $A = D + L + L^T$ , where  $L^T$  denotes the transpose of  $L$ . Each of the above approximate matrices can be easily inverted and the update ( $\delta\phi'$ ) can be obtained by forward substitutions.

The nonlinear relaxation algorithms are similar to their linear counterparts. The unknowns are updated iteratively in a main loop. At each step, however, a nonlinear term, either a sinh/cosh function<sup>22,25-27</sup> or a polynomial,<sup>17,18,21</sup> has to be evaluated on every grid. A typical nonlinear relaxation algorithm for the PBE can be summarized as the following pseudo-code.

Do until convergence

```

For i, j, k from 1 to xm, ym, zm
  
$$\sigma = \varepsilon_{i-1,j,k}^x \phi_{i-1,j,k}^{t+1} + \varepsilon_{i,j-1,k}^y \phi_{i,j-1,k}^{t+1} + \varepsilon_{i,j,k-1}^z \phi_{i,j,k-1}^{t+1}$$

  
$$+ \varepsilon_{i,j,k}^x \phi_{i+1,j,k}^t + \varepsilon_{i,j,k}^y \phi_{i,j+1,k}^t + \varepsilon_{i,j,k}^z \phi_{i,j,k+1}^t$$

  
$$\varepsilon_{i,j,k} = \varepsilon_{i-1,j,k}^x + \varepsilon_{i,j-1,k}^y + \varepsilon_{i,j,k-1}^z + \varepsilon_{i,j,k}^x + \varepsilon_{i,j,k}^y + \varepsilon_{i,j,k}^z$$

  
$$\phi_{i,j,k}^{t+1} = (1-\omega) \phi_{i,j,k}^t + \frac{\omega(\sigma + 4\pi q_{i,j,k}/h)}{\varepsilon_{i,j,k} + N'(\phi_{i,j,k}^t)}$$

End for i, j, k
t = t + 1
End do

```

Here  $\omega$  is the relaxation parameter,  $\omega=1$  corresponds to the nonlinear GS method, and  $1 < \omega < 2$  corresponds to the nonlinear SOR method.  $N'(\phi_{i,j,k}^t)$  is a diagonal element of the matrix  $N'(\phi^t)$ <sup>25</sup> or a corresponding approximate expression.<sup>17,18,21,22</sup> The above procedure works well for the nonlinear PBE in many situations but there are cases where the iteration diverges.<sup>22</sup> The convergence failures can be reduced by optimizing the relaxation parameter  $\omega$  and adding the nonlinearity gradually. For example, in the Delphi program, the nonlinear term is added to the PBE by 5% each time and the optimal  $\omega$  is estimated adaptively based on the average nonlinearity across the whole space.<sup>27</sup>

In this study, the nonlinear SOR solver uses a constant high-value  $\omega$ , i.e.,  $\omega=1.9$ . The optimal relaxation parameter for the linear SOR method is between 1.9 and 1.95, depending on the structures.<sup>41</sup> We chose  $\omega=1.9$  because it gives a reasonable balance between convergence rate and convergence failure among tested molecules. Reducing  $\omega$  further can lead to fewer convergence failures but also lower convergence rate. For example,  $\omega=1.8$  reduces convergence failures by 24% but simultaneously reduces the convergence rate by 49%. Instead of optimizing  $\omega$ , we implemented two different strategies to reduce the

convergence failures of SOR. In the first revised SOR, termed the adaptive SOR (ASOR) method, we initially use the high-value  $\omega$  and then gradually lower it if the norm of the residual starts to increase. As will be shown below, ASOR can reduce the convergence failures of the original SOR method and retains its overall convergence rate. The second modified SOR method combines SOR with the same line search used in the two NT methods after  $\delta\phi'$  is calculated. The “damped” SOR (DSOR) method can also improve the convergence efficiently, though neither can guarantee convergence as will be shown below.

### **Simulation Details**

We implemented seven nonlinear PBE solvers in the PBSA program of the AMBER 10 package,<sup>44</sup> including one implementation of GS, three implementations of SOR, one implementation of CG, and two implementations of NT. The relaxation solvers and the conjugate gradient solver all solve the corresponding linear PBE first and utilize the solution as the initial guess of the solution of the nonlinear PBE.

The dielectric constant was set to 1 within the molecular interior and it was set to 80 within the solvent. The solvent probe was set to be 1.5 Å to compute the solvent excluded surface that was used as the solute/solvent dielectric boundary. The ion probe was set to be 2.0 Å to compute the ion accessible surface that was used as the interface between the Stern layer and the bulk ion accessible solvent region. The finite-difference grid spacing was set as 0.5 Å. The ratio between the longest dimension of the finite-difference grid and that of the solute was set as 1.5. The convergence criterion for the nonlinear system was set to be  $10^{-6}$  and the ionic strength was set to 150 mM if they are not mentioned otherwise. All floating

point data were set in double precision to be consistent with the rest of the Amber 10 package.

We initially collected 588 high-resolution (at least 2 Å ) nucleic-acid structures with sequence diversity more than 30% from the Protein Data Bank. We first removed all ligand molecules and those structures with non-natural nucleotides unsupported by the Amber force field. Often the unsupported nucleotides are located in the terminal regions, so that the remaining structures can still be used if the terminal regions are deleted. Finally the test set includes 364 nucleic acids. Hydrogen atoms were added in LEAP of the Amber 10 package.<sup>44</sup> These molecules were assigned the charges of Cornell et al<sup>45</sup> and the radii of Tan et al.<sup>46</sup> The atom numbers of the nucleic acids range from 250 to 5,569, and the grid numbers of the nucleic acids range from 313,551 to 15,218,175. The PDB codes of the nucleic acids are given in the Appendix.

The performance statistics of the seven solvers was collected on a computer cluster of 80 nodes with 1GB memory of 3.0GHz P4 CPUs. For some methods, calculations on the ten largest molecules in the test of 364 nucleic acids require more than 1GB memory, so they were left out in the overall analysis. Next, we tested the two NT solvers with the 22 largest nucleic acids (>2,000 atoms) in the test set, with the ten largest nucleic acids included, on a server node with 8GB memory. We also tested the two NT solvers with the 26 largest proteins (>4,000 atoms) from the Amber test set.<sup>44</sup> Finally, we analyzed the effects of different salt properties on the performance of the two NT solvers.

## **Results and Discussion**

## **Idealized System Test**

We first tested the nonlinear solvers with an idealized system, i.e., a single ion with radius of 1 Å and multiple charges in the salt-water solution. In this simple system, the grid spacing was set to be 0.25 Å. We compared the numerical solutions of the seven solvers with the solutions obtained from the predictor-corrector Adams method in Mathematica 6.0 under different conditions. Here different charges for the single ion and different ion concentrations were used. Figure 1 shows the results with different charges for the single ion while the ion concentration is 500 mM, and Figure 2 shows the results under different ion concentrations while the charge of the single ion is 2 e. Note that the solutions of all seven tested numerical solvers are represented by the same symbols due to their virtually identical numerical values. Both figures demonstrate excellent agreements between the seven implemented solvers and the standard numerical method packaged in Mathematica for the tested systems.

## **Solver Convergence Statistics**

We studied the convergence of all the solvers for the 354-nucleic-acid test set. The convergence statistics are shown in Table 1. Three out of the seven solvers can converge within 10,000 steps in all test cases, i.e., the CG method and the two NT methods, each of which makes use of a functional-assisted strategy. The original SOR method fails in most cases but is noticeably improved if  $\omega$  is adaptively modified: 76% failed cases in the original SOR method converge with the ASOR method. After a damped step size is applied in the DSOR method, 88% failed cases in the original SOR method converge.

Since we first solve the corresponding linear problem in the nonlinear SOR solvers, the nonlinearity is weakened. This strategy was found to improve the convergence. Moreover, lower-value  $\omega$ , or even under-relaxation, can obtain better convergence than high-value  $\omega$ .<sup>22,27</sup> For example, in our test, the GS method converges in more test cases than the original SOR method in spite of its much lower convergence rate. The minimal  $\omega$  in the ASOR method is equal to 1, so it is expected that the ASOR method has the same number of convergence failures as the GS method, but it converges much faster because over-relaxation is initially used. The DSOR method conducts a line search to descend  $f(\phi)$  in eqn. (12) but still cannot guarantee convergence because the direction is not necessarily a descent direction. Note that for those test cases that the original SOR method cannot converge, the DSOR method shows better convergence rate than the ASOR method.

One distinctive difference between the NT-ICCG method and the NT-MG method is that a tighter convergence criterion for eqn. (11) can accelerate the NT-ICCG method while probably having a negative effect on the convergence rate of the NT-MG method. The reason is that the current algorithm in the NT-MG method is suitable only for inexactly solving eqn. (11) and the linear MG becomes unstable when exact solution of eqn. (11) is requested. In contrast, a tighter convergence criterion for eqn. (11) can decrease a great number of the Newton iterations and at the same time add only a little time to each iteration in the NT-ICCG method.

### **Timing and Memory Requirement of Inexact Newton Methods**

In the following we focus on the two NT methods because of their significant higher

efficiency and robustness. First, we examined the timing and memory requirement of the two NT methods in solving the PB equations for the test set of 354 nucleic acids. We utilized the APBS finite-difference solver as a reference in this round. Note that all three solvers were compiled and tested under conditions as identical as possible. Specifically the exactly same discretized nonlinear problems were solved. Figure 3 shows that the memory requirement and solver time of the three solvers are similar for smaller molecules, but their difference becomes obvious when the grid number increases. On average, the APBS solver requires the most memory and the NT-ICCG consumes the most time. In addition, both the memory and the timing trends for NT-MG are linear over the grid number. The memory trend for NG-ICCG is linear, but its timing trend only remains to be linear for smaller test cases and becomes nonlinear for larger test cases. It is clear that NT-MG needs less memory and less time than NT-ICCG and the APBS solver for all the tested molecules.

Table 2 and Table 3 list the solver time of the two NT methods for the 22 largest molecules in the test set of 364 nucleic acids and the 26 largest proteins in the Amber test set, respectively. Since tested proteins are more compact than tested nucleic acids, the average grid numbers is similar for the two sets of molecules. It is apparent that the solver times are also comparable between the two sets of molecules, i.e., the solver efficiency mostly depends on the grid number. More importantly, for these largest tested molecules, NT-MG uses only a third of the time of NT-ICCG. Note that it uses about a half of the time of NT-ICCG in the test with 354 nucleic acids. This observation is consistent with the intrinsic advantage of the MG method on large systems.

### **Performance of Inexact Newton Methods versus Convergence Criterion**

Next the performance of the two NT methods under a variety of convergence criteria ranging from  $10^{-1}$  to  $10^{-9}$  was examined. Five nucleic acids of different sizes were selected as test cases in this round, which are 1SGS (1074 atoms), 1JRN (1564 atoms), 1U8D (2145 atoms), 1EHZ (2509 atoms) and 2GWQ (3128 atoms). Figure 4 shows that both methods can improve convergence without any steep jump in the solver time, indicating a constantly smooth convergence behavior. Specifically a linear relationship exists between the solver time and the logarithm of the convergence criterion. The slope, however, increases with the complexity and size of tested molecules.

### **Performance of Inexact Newton Methods versus Ion Valence**

Now we analyze the performance of the two NT methods when the ion valence is changed. While the symmetry of the ions is preserved, the ion valence is chosen to be 1, 2 and 3 respectively in this test. The sample in this test consists five large nucleic acids, which are 1EHZ (2509 atoms), 3BNN (2696 atoms), 1NUV (3112 atoms), 2GWQ (3128 atoms), 3D2V (4970 atoms). Figure 5 shows the trend of the solver times of the two NT methods versus the ion valence. The average solver times of both the NT-MG method and the NT-ICCG method only increase slightly with the ion valence, indicating both NT solvers are stable over different ion valences in the salt water solution.

### **Performance of Inexact Newton Methods versus Ion Concentration**

Finally, the effect of the ion concentration was studied and the results are shown in Figure

6. Three different ion concentrations were tested, including 150 mM, 500 mM, and 1000 mM. Different from the analysis on the ion valence, only NT-MG exhibits stable performance. Regardless of ion concentration, the average solver time of NT-MG for the five molecules is more or less constant. On the contrary, the average solver time of NT-ICCG depends on the ion concentration: the solver uses about one-quarter less solver time when the ion concentration is high ( $\geq 500$  mM). This observation indicates that NT-MG is probably more stable than NT-ICCG under different salt conditions in biomolecular applications.

## **Conclusion**

We implemented and optimized seven finite-difference solvers for the nonlinear Poisson-Boltzmann equation, including four relaxation methods, one CG method and two NT methods. We tested the performance of the seven solvers with a large number of nucleic acids and proteins, with special attentions given to the robust NT algorithm. We investigated the role of linear solvers in its performance by incorporating ICCG and MG into the algorithm. Specifically, a four-level v-cycle was applied in the MG method, where the restriction and prolongation were realized with a three-dimensional seven-band version of Alcouffe's algorithm. In addition, the SOR method was applied for the multigrid pre-smoothing and post-smoothing on fine grids, and also for solving the linear problem on the coarsest grids. We adopted this simple and fast algorithm because the inner linear problem of an NT method does not need to be solved exactly. On the contrary, to accelerate the convergence of our implementation of NT-ICCG, we tightened the convergence criterion of the inner linear solver loop, which would cause our implementation of NT-MG to be unstable. In addition, we

explored strategies to optimize the SOR method to reduce its convergence failures without too much sacrifice in its convergence rate. In the ASOR method,  $\omega$  was designed to decrease when the norm of the residual stops reducing. This method reduces the convergence failures by 76% and retains much of the overall convergence rate of the original SOR method with a high-value  $\omega$ . In the DSOR method, the damping strategy from the NT method was utilized to optimize the search step length and was found to reduce the convergence failures by 88%. Overall our results show that the nonlinear methods accompanied with a functional-assisted strategy can guarantee convergence, such as the CG method and the NT method. Especially the NT method exhibits impressive performance when it is combined with highly efficient linear solvers.

Finally it is instructive to discuss future directions in the optimization of nonlinear solvers for biomolecular applications. First point charges are widely used to represent atomic charge density distributions in current biomolecular models. Unfortunately this practice introduces singularity into the right hand side of PBE. The presence of charge singularity results in discontinuity in the electrostatic potential with large error when the finite-difference method is used. We have developed a new formulation to remove the charge singularity in the linear finite-difference solvers.<sup>47</sup> Given the current implementations of the nonlinear solvers, we are in a position to investigate the effect of charge singularity on the performance of the nonlinear finite-difference solvers. In addition, we plan to extend our analysis and optimization of the nonlinear finite-difference solvers in the context of molecular dynamics simulations. It is expected the efficiency of the nonlinear solvers to be improved in molecular dynamics simulations just as in our prior analysis of the linear solvers.<sup>41</sup> However, further

development is necessary to fully take advantage of the potential update nature in molecular dynamics to achieve computational efficiency high enough for routine biomolecular applications.

## **Acknowledgements**

We thank Drs. Hong-Kai Zhao (UC Irvine) and Zhi-Lin Li (NC State) for exciting discussions. This work is supported in part by NIH (GM069620 & GM079383).

## **Appendix**

### **Test Set of 364 Nucleic Acids**

100D, 109D, 110D, 118D, 126D, 127D, 131D, 137D, 138D, 151D, 152D, 157D, 158D, 160D, 161D, 165D, 181D, 182D, 184D, 190D, 191D, 192D, 196D, 198D, 1A2E, 1BD1, 1BNA, 1CSL, 1D10, 1D11, 1D12, 1D13, 1D15, 1D23, 1D32, 1D36, 1D37, 1D38, 1D39, 1D43, 1D44, 1D45, 1D46, 1D48, 1D49, 1D54, 1D56, 1D57, 1D58, 1D63, 1D67, 1D78, 1D79, 1D88, 1D8G, 1D8X, 1D96, 1DA0, 1DA9, 1DC0, 1DCG, 1DJ6, 1DL8, 1DN8, 1DNO, 1DNS, 1DNT, 1DNX, 1DNZ, 1DOU, 1DQH, 1EHV, 1EHZ, 1EN3, 1EN8, 1EN9, 1ENE, 1ENN, 1EVP, 1EVV, 1F27, 1FD5, 1FDG, 1FMQ, 1FMS, 1FN2, 1FQ2, 1FTD, 1G4Q, 1I0T, 1I1P, 1I2Y, 1I7J, 1ICG, 1ICK, 1ID9, 1IDW, 1IH1, 1IHA, 1IKK, 1IMR, 1IMS, 1JGR, 1JO2, 1JRN, 1JTL, 1K9G, 1KCI, 1KD3, 1KD4, 1KD5, 1L1H, 1L2X, 1L4J, 1LJX, 1M69, 1M6F, 1M6G, 1M6R, 1M77, 1MF5, 1MSY, 1NLC, 1NQS, 1NT8, 1NUJ, 1NUV, 1NVN, 1NVY, 1O0K, 1OFX, 1OSU, 1P20, 1P4Y, 1P4Z, 1P79, 1PFE, 1PJG, 1PJO, 1Q96, 1Q9A, 1QCU, 1QYK, 1QYL, 1R68, 1RQY, 1RXB, 1S23, 1S2R, 1SGS, 1SK5, 1T0E, 1U8D, 1UB8, 1UE4, 1V9G,

1VAQ, 1VJ4, 1VS2, 1VZK, 1WOE, 1WQY, 1XA2, 1XCS, 1XCU, 1XJX, 1XJY, 1XPE,  
1XVK, 1XVN, 1XVR, 1Z3F, 1Z8V, 1ZCI, 1ZEV, 1ZEX, 1ZEY, 1ZEZ, 1ZF0, 1ZF1, 1ZF2,  
1ZF3, 1ZF4, 1ZF5, 1ZF6, 1ZF7, 1ZF8, 1ZF9, 1ZFA, 1ZFB, 1ZFC, 1ZFF, 1ZFG, 1ZNA,  
1ZPH, 1ZPI, 200D, 212D, 215D, 220D, 221D, 222D, 224D, 232D, 234D, 235D, 236D, 240D,  
241D, 243D, 244D, 245D, 248D, 251D, 255D, 258D, 259D, 260D, 272D, 276D, 279D,  
284D, 288D, 292D, 293D, 295D, 2A43, 2A7E, 2ADW, 2AVH, 2B0K, 2B1B, 2B2B, 2B3E,  
2D47, 2D94, 2D95, 2DCG, 2DES, 2DYW, 2DZ7, 2EES, 2EET, 2EEU, 2EEV, 2F8W, 2G32,  
2G3S, 2G9C, 2GB9, 2GPM, 2GQ4, 2GQ5, 2GQ6, 2GQ7, 2GVR, 2GW0, 2GWA, 2GWQ,  
2GYX, 2HBN, 2HTO, 2I2I, 2I5A, 2IE1, 2O1I, 2O4F, 2OE5, 2OE8, 2OIY, 2OKS, 2PKV,  
2PL4, 2PL8, 2PLB, 2PLO, 2PWT, 2Q1R, 2QEK, 2R22, 2V6W, 2V7R, 2VAL, 2Z75, 307D,  
308D, 310D, 312D, 314D, 315D, 317D, 331D, 332D, 334D, 336D, 348D, 349D, 351D,  
352D, 354D, 355D, 360D, 362D, 368D, 369D, 370D, 371D, 377D, 385D, 386D, 393D,  
394D, 395D, 396D, 397D, 398D, 399D, 3BNN, 3C2J, 3C44, 3CGP, 3CGS, 3CJZ, 3CZW,  
3D0M, 3D2V, 3DIL, 3DNB, 3ERU, 3EUM, 413D, 414D, 420D, 423D, 428D, 431D, 432D,  
434D, 435D, 437D, 439D, 440D, 441D, 442D, 443D, 452D, 453D, 455D, 463D, 465D,  
466D, 472D, 473D, 476D, 477D, 479D, 480D, 482D, 483D, 485D, 5DNB, 7BNA, 9BNA,  
9DNA.

## References

- 1 M. E. Davis and J. A. McCammon, *Chemical Reviews* **90** (3), 509 (1990).
- 2 K. A. Sharp, *Current Opinion in Structural Biology* **4** (2), 234 (1994).
- 3 M. K. Gilson, *Current Opinion in Structural Biology* **5** (2), 216 (1995).
- 4 B. Honig and A. Nicholls, *Science* **268** (5214), 1144 (1995).
- 5 B. Roux and T. Simonson, *Biophysical Chemistry* **78** (1-2), 1 (1999).
- 6 C. J. Cramer and D. G. Truhlar, *Chemical Reviews* **99** (8), 2161 (1999).
- 7 D. Bashford and D. A. Case, *Annual Review Of Physical Chemistry* **51**, 129 (2000).
- 8 N. A. Baker, *Current Opinion in Structural Biology* **15** (2), 137 (2005).
- 9 J. H. Chen, W. P. Im, and C. L. Brooks, *Journal of the American Chemical Society* **128** (11), 3728 (2006).
- 10 M. Feig, J. Chocholousova, and S. Tanizaki, *Theoretical Chemistry Accounts* **116** (1-3), 194 (2006).
- 11 W. Im, J. H. Chen, and C. L. Brooks, in *Peptide Solvation and H-Bonds* (Elsevier Academic Press Inc, San Diego, 2006), Vol. 72, pp. 173.
- 12 P. Koehl, *Current Opinion in Structural Biology* **16** (2), 142 (2006).
- 13 B. Z. Lu, Y. C. Zhou, M. J. Holst, and J. A. McCammon, *Communications in*

- Computational Physics **3** (5), 973 (2008).
- <sup>14</sup> J. Wang, C. H. Tan, Y. H. Tan, Q. Lu, and R. Luo, *Communications in Computational Physics* **3** (5), 1010 (2008).
- <sup>15</sup> T. L. Hill, *An Introduction to Statistical Thermodynamics*. (Dover Publications, Inc., New York, 1986).
- <sup>16</sup> M. J. Holst and F. Saied, *Journal of Computational Chemistry* **16** (3), 337 (1995).
- <sup>17</sup> K. A. Sharp and B. Honig, *Journal of Physical Chemistry* **94** (19), 7684 (1990).
- <sup>18</sup> B. Jayaram, K. A. Sharp, and B. Honig, *Biopolymers* **28** (5), 975 (1989).
- <sup>19</sup> N. V. Prabhu, M. Panda, Q. Y. Yang, and K. A. Sharp, *Journal of Computational Chemistry* **29** (7), 1113 (2008).
- <sup>20</sup> X. Ye, Q. Cai, W. Yang, and R. Luo, *Biophysical Journal*, doi:10.1016/j.bpj.2009.05.012 (2009).
- <sup>21</sup> S. A. Allison, J. J. Sines, and A. Wierzbicki, *Journal of Physical Chemistry* **93** (15), 5819 (1989).
- <sup>22</sup> A. Nicholls and B. Honig, *Journal of Computational Chemistry* **12** (4), 435 (1991).
- <sup>23</sup> B. A. Luty, M. E. Davis, and J. A. McCammon, *Journal of Computational Chemistry* **13** (9), 1114 (1992).
- <sup>24</sup> H. Oberoi and N. M. Allewell, *Biophysical Journal* **65** (1), 48 (1993).

- <sup>25</sup> K. E. Forsten, R. E. Kozack, D. A. Lauffenburger, and S. Subramaniam, *Journal of Physical Chemistry* **98** (21), 5580 (1994).
- <sup>26</sup> Z. X. Xiang, Y. Y. Shi, and Y. W. Xu, *Journal of Computational Chemistry* **16** (2), 200 (1995).
- <sup>27</sup> W. Rocchia, E. Alexov, and B. Honig, *Journal of Physical Chemistry B* **105** (28), 6507 (2001).
- <sup>28</sup> M. Holst, N. Baker, and F. Wang, *Journal of Computational Chemistry* **21** (15), 1319 (2000).
- <sup>29</sup> N. Baker, M. Holst, and F. Wang, *Journal of Computational Chemistry* **21** (15), 1343 (2000).
- <sup>30</sup> A. I. Shestakov, J. L. Milovich, and A. Noy, *Journal of Colloid and Interface Science* **247** (1), 62 (2002).
- <sup>31</sup> P. E. Dyshlovenko, *Computer Physics Communications* **147** (1-2), 335 (2002).
- <sup>32</sup> A. Sayyed-Ahmad, K. Tuncay, and P. J. Ortoleva, *Journal of Computational Chemistry* **25** (8), 1068 (2004).
- <sup>33</sup> L. Chen, M. J. Holst, and J. C. Xu, *Siam Journal on Numerical Analysis* **45**, 2298 (2007).
- <sup>34</sup> D. Xie and S. Zhou, *BIT Numerical Mathematics* **47** (4), 853 (2007).

- 35 A. A. Rashin and J. Malinsky, *Journal of Computational Chemistry* **12** (8), 981 (1991).
- 36 Y. N. Vorobjev, J. A. Grant, and H. A. Scheraga, *Journal of the American Chemical Society* **114** (9), 3189 (1992).
- 37 H. X. Zhou, *Journal of Chemical Physics* **100** (4), 3152 (1994).
- 38 A. H. Boschitsch and M. O. Fenley, *Journal of Computational Chemistry* **25** (7), 935 (2004).
- 39 R. Fletcher and C. M. Reeves, *Computer Journal* **7** (2), 149 (1964).
- 40 R. Luo, L. David, and M. K. Gilson, *Journal of Computational Chemistry* **23** (13), 1244 (2002).
- 41 J. Wang and R. Luo, *Journal of Computational Chemistry*, Submitted.
- 42 M. Holst and F. Saied, *Journal of Computational Chemistry* **14** (1), 105 (1993).
- 43 R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter, *Siam Journal on Scientific and Statistical Computing* **2** (4), 430 (1981).
- 44 D. A. Case, T. A. Darden, T. E. Cheatham III, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, M. Crowley, R. C. Walker, W. Zhang, K. M. Merz, B. Wang, S. Hayik, A. Roitberg, G. Seabra, I. Kolossváry, K. F. Wong, F. Paesani, J. Vanicek, X. Wu, S. R. Brozell, T. Steinbrecher, H. Gohlke, L. Yang, C. Tan, J. Mongan, V. Hornak, G. Cui, D. H. Mathews, M. G. Seetin, C. Sagui, V. Babin, and P. A. Kollman, *AMBER 10*,

University of California, San Francisco. (2008).

<sup>45</sup> W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman, *Journal of the American Chemical Society* **117** (19), 5179 (1995).

<sup>46</sup> C. H. Tan, L. J. Yang, and R. Luo, *Journal of Physical Chemistry B* **110** (37), 18680 (2006).

<sup>47</sup> Q. Cai, J. Wang, H. K. Zhao, and R. Luo, *Journal of Chemical Physics* **130** (14), 145101 (2009).

## Tables

**Table 1.** Solver Convergence and Relative Performance Statistics in the Test Set of 354 Nucleic Acids. Relative performance of a solver for a molecule is defined as the CPU time of the solver over the CPU time of the NT-MG solver for the same molecule. Avg Rel: average relative performance over all tested molecules. Unconv: number of convergence failures.

Solver	GS	SOR	ASOR	DSOR	CG	NT-ICCG	NT-MG
Avg Rel	47.10	3.32	3.71	5.86	21.35	1.99	1.00
Unconv	6	25	6	3	0	0	0

**Table 2.** Solver Times (second) of Two Inexact Newton Methods for the 22 Largest Nucleic Acids in the Test Set of 364 Nucleic Acids.  $N_{atom}$ : atom number;  $N_{grid}$ : grid number.

Nucleic Acids	$N_{atom}$	$N_{grid}$	NT-MG	NT-ICCG
1EHZ	2509	7258191	51.71	131.35
1EVV	2509	6650175	46.76	108.97
1I2Y	2124	3820287	18.31	51.83
1NUJ	3112	7498575	53.54	140.65
1NUV	3112	7498575	53.97	142.65
1Q96	2616	4887087	31.08	80.03
1U8D	2145	3285711	20.83	44.12
1ZCI	2448	5980975	34.74	107.87
244D	3120	4956175	34.81	79.97
2DZ7	2032	2597023	17.35	28.18
2EES	2145	3285711	20.43	48.83
2EET	2147	3285711	20.89	46.78
2EEU	2145	3285711	20.71	46.67
2EEV	2147	3285711	20.75	48.43
2G3S	2580	4779775	26.27	77.83
2G9C	2144	3285711	20.67	48.07
2GWQ	3128	6286383	40.55	104.68
352D	3024	4956175	34.75	90.32
3BNN	2696	8078175	57.32	146.12
3D2V	4970	8299375	61.94	164.3
2Z75	4646	11979711	102.47	225.23
3DIL	5569	15218175	126.39	383.26

**Table 3.** Solver Times (second) of Two Inexact Newton Methods for the 26 Largest Proteins in the Amber test set.

Proteins	$N_{atom}$	$N_{grid}$	NT-MG	NT-ICCG
1B8O_A	4348	4779775	29.91	98.38
1C0P_A	5566	7258191	73.34	194.19
1D8V_A	4211	5849375	32.19	119.25
1DCI_A	4281	6204975	40.45	160.39
1DJ0_A	4176	5759775	44.74	115.93
1DS1_A	4916	4869375	45.02	121.3
1E19_A	4874	6384175	34.93	127.91
1E6Q_M	7819	8816751	74.85	193.63
1E6U_A	4966	5180175	49.82	103.8
1EZA_0	4034	4921631	29.07	69.93
1EZO_A	5735	8392815	49.42	147.62
1F24_A	6221	6286383	39.43	108.85
1HZY_A	5092	4869375	30.28	97.17
1IXH_0	4856	5637663	36.70	87.21
1MLA_0	4485	4424175	28.18	81.41
1PA2_A	4441	4779775	30.43	91.16
1QH4_A	5983	6829375	37.72	130.39
1QNR_A	5129	4379375	37.47	91.42
1QOP_B	5895	5233167	41.56	140.93
1QQF_A	4365	4019679	22.28	56.62
1QTW_A	4380	4342767	26.85	85.04
1YUB_0	4168	6768719	37.55	134.5
2CTC_0	4801	4342767	28.59	71.34
2OLB_A	8254	7866207	53.02	148.77
3SIL_0	5804	5359375	38.29	120.5
7A3H_A	4578	3615183	22.25	68.95

## Figure Captions

**Figure 1.** Comparison between the numerical solutions from PBSA and Mathematica for the idealized system under different charges of the single ion. PBSA: numerical solutions in the PBSA program. MATHEMATICA: numerical solutions from the Mathematica program. The charge of the single ion is set as 1 e, 2 e, and 4 e, respectively. Ion concentration is 500mM and ion valence is 1. Only potential in the ion accessible region is plotted.

**Figure 2.** Comparison between the numerical solutions from PBSA and Mathematica for the idealized system under different ion concentrations. Linear: solutions to the linearized PBE. Nonlinear: solutions to the full nonlinear PBE. Ion concentrations are set as 100 mM and 1000 mM, respectively. The charge of the single ion is 2 e and ion valence is 1. Only potential in the ion accessible region is plotted.

**Figure 3.** Scaling of solver CPU times and memory usages versus grid numbers for the two inexact Newton methods and the APBS solver in the test set of 354 nucleic acids.

**Figure 4.** Solver CPU times versus convergence criterion for the two inexact Newton methods. Note that the flat region between adjacent convergence criteria results from the same number of iterations required to converge even if the convergence criteria are different, i.e. the residual reduction is more than the specified convergence criterion reduction.

**Figure 5.** Solver CPU times versus ion valences for the two inexact Newton methods.

**Figure 6.** Solver CPU times versus ion concentrations for the two inexact Newton methods.

## Figures

Figure 1

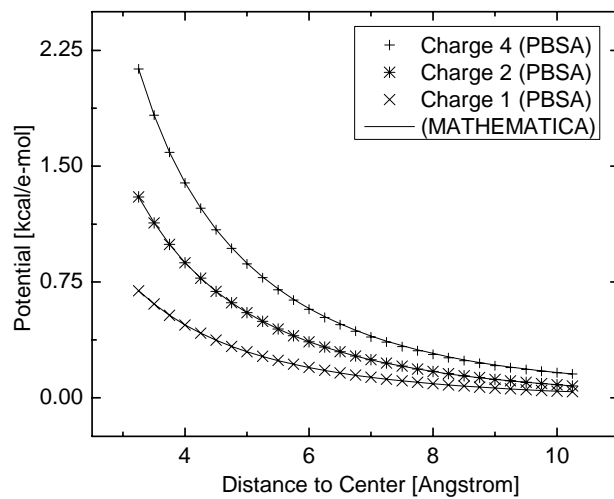


Figure 2

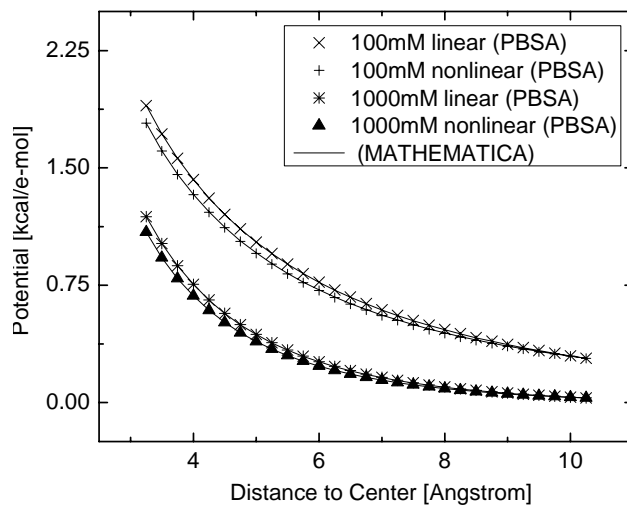


Figure 3

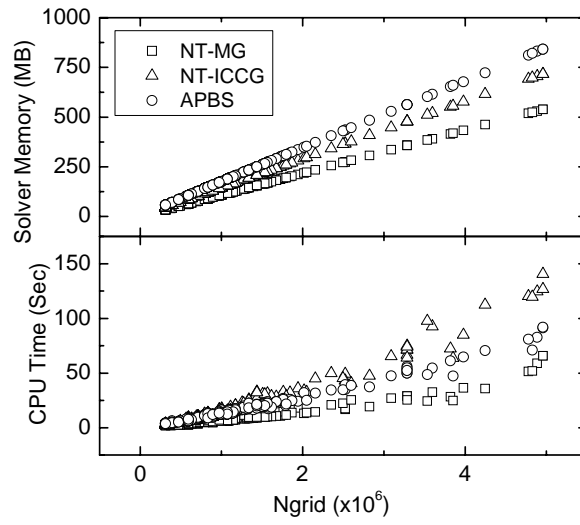


Figure 4

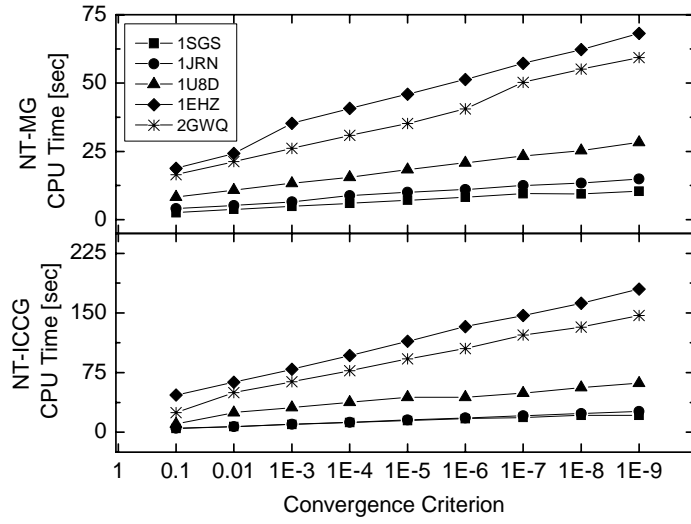


Figure 5

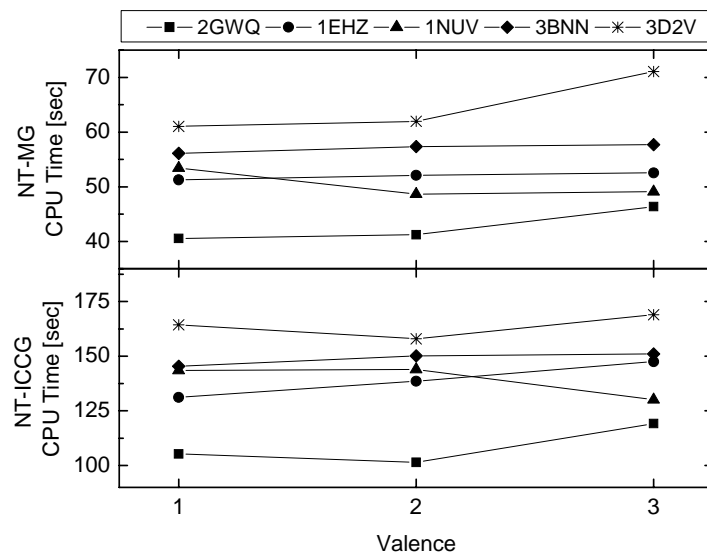


Figure 6

