MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **58** (2007) 209-213

ISSN 0340 - 6253

Two experiments with the Newton-Krylov algorithm for Poisson Boltzmann equation

Sanjay Kumar Khattri
Stord Haugesund University College, Norway
sanjay.khattri@hsh.no
ans.hsh.no/home/skk

(Received February 16, 2007)

Abstract: In this article, we are analysing the Newton Krylov algorithm for solving the nonlinear Poisson Boltzmann equation. We are analysing impact of the tolerance of the linear solver on the convergence of the nonlinear solver. It is known that the Newton's algorithm does not converge for all initial guesses. We are presenting two techniques for making initial guesses. In the first technique, initial guess is made by solving the linear part of the nonlinear equation and neglecting the nonlinear part. While in the second technique, the initial guess is generated by solving the corresponding linearised equation. Here, nonlinear part is replaced by the equivalent linear part.

1 Introduction

The past fifty years has seen considerable advance in solving linear systems. Krylov subspace method is the result of the huge effort by the researchers during the last century. It is one among the ten best algorithms of the 20th century. There exists optimal linear solvers [12]. But, still there exists no optimal nonlinear solver or the one that we know of. Our research is in the field of optimal solution of nonlinear equations generated by the discretization of the Poisson Boltzmann equation.

Discretization of the non-linear Poisson-Boltzmann Equation equation results in a system of non-linear algebraic equations with symmetric Jacobian. The Newton algorithm is the most useful tool for solving non-linear equations. It consists of solving a series of linear system of equations (Jacobian system). The Poisson-Boltzmann equation arising in molecular bio-physics is given as [3, 4, 5, 6, 7, 8, 9, 10, 11, 1]

$$\nabla(-\mathrm{K}(\mathbf{r})\nabla p) + f(p) = s(x,y) \qquad \textbf{in} \quad \Omega \tag{1}$$

$$p(x,y) = p^{\mathrm{D}} \qquad \textbf{on} \quad \partial\Omega_{\mathrm{D}} \tag{2}$$

$$g(x,y) = (-\mathrm{K}\nabla p)\cdot\hat{\mathbf{n}} \qquad \textbf{on} \quad \partial\Omega_{\mathrm{N}} \tag{3}$$

Here, Ω is a polyhedral domain in $\mathbb{R}^{\mathrm{d}}$ (d = 2, 3), the source function $s(x,y)$ is assumed to be in $L^2(\Omega)$, and the medium property K is uniformly positive. The property K can be discontinuous in space. In the equations (2) and (3), $\partial\Omega_{\mathrm{D}}$ and $\partial\Omega_{\mathrm{N}}$ represent Dirichlet and Neumann part of the boundary, respectively. $f(p)$ represents nonlinear part of the equation. In biophysics literature, the medium properties K is referred to as the permitivity [9, 5, 6, 7, 8, 10]. It takes the values of the appropriate dielectric constants in the different regions of the domain Ω. The equations (1), (2) and (3) models a wide variety of processes with practical applications. For example, pattern formation in biology, viscous fluid flow phenomena, chemical reactions, biomolecule electrostatics and crystal growth.

The Poisson-Boltzmann equation (PBE) one of the most popular continuum models for describing electrostatic interactions between molecular solutes in salty and aqueous media. The importance of the equation (1) for modelling biomolecules is well established [3, 4, 5, and references therein]. The continuum electrostatics plays an important role in several areas of biomolecular simulation. For example, diffusional processes to determine ligand-protein and protein-protein binding kinetics, implicit solvent molecular dynamics of biomolecules, solvation and binding energy calculations to determine ligand-protein and protein-protein equilibrium binding constants, aid in rational drug design, and biomolecular titration studies. Equation (1) is also part of many molecular dynamics simulators. See the references [9, 11, 13, 14] for studying electorstatic interactions.

We are using Newton-Krylov algorithm for solving the discrete nonlinear system of equations formed by the finite volume method [2]. Since initial guess is very important for the convergence of Newton's algorithm. Thus, for making initial guess, we are using two approaches. In the first approach, we are solving the linear part of the equation (1), i.e., $\nabla(-\mathrm{K}\nabla p) = s$ and using the solution as an initial guess for the Newton's iteration. In the second approach, we are solving the corresponding linearised equation ($\nabla(-\mathrm{K}\nabla p)+\overline{f(p)} = s$) for making an initial guess. Here, $\overline{f(p)}$ is the linear representation of the nonlinear part $f(p)$.

For finite volume discretization of the nonlinear equations and the Newton algorithm for the poisson boltzmann equations the following reference are recommended [1, 2].

2 Newton Krylov algorithm

For using Newton's algorithm equation (1) is discretized in the residual form $\nabla \cdot (-\mathrm{K}\,\nabla p) + f(p) - s = 0$ [2]. This discretization results in a vector of nonlinear equations $\mathbf{A}(\mathbf{p})$. This vector is given as

$$\mathbf{A}(\mathbf{p}) = \begin{pmatrix} A_1(\mathbf{p}) \\ A_2(\mathbf{p}) \\ \vdots \\ A_n(\mathbf{p}) \end{pmatrix}.$$

We are interested in finding the vector $\mathbf{p}$ which makes the operator $\mathbf{A}$ vanish.

The nonlinear algebraic operator $\mathbf{A}(\mathbf{p})$ can be expanded by the Taylor's series around an initial guess $\mathbf{p}_0$.

$$\mathbf{A}(\mathbf{p}) = \mathbf{A}(\mathbf{p_0}) + \boldsymbol{J}(\mathbf{p_0})\triangle\mathbf{p} + HOT \tag{4}$$

Here, $\boldsymbol{J}$ is the Jacobian matrix, and $\triangle\mathbf{p}$ is $(\mathbf{p} - \mathbf{p_0})$. The Jacobian $\boldsymbol{J}$ is $n \times n$ matrix. The Jacobian $\boldsymbol{J}$ in the equation (4) is given as

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial A_1}{\partial p_1} & \frac{\partial A_1}{\partial p_2} & \cdots & \frac{\partial A_1}{\partial p_n} \\ \frac{\partial A_2}{\partial p_1} & \frac{\partial A_2}{\partial p_2} & \cdots & \frac{\partial A_2}{\partial p_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial A_n}{\partial p_1} & \frac{\partial A_n}{\partial p_2} & \cdots & \frac{\partial A_n}{\partial p_n} \end{pmatrix}$$

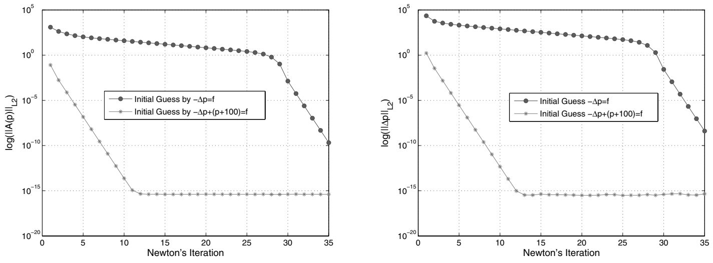
It should be noted that Jacobian $\boldsymbol{J}$ is symmetric and will be positive definite and diagonal dominant for positive non-linearity ($f > 0$). We are interested in finding the zeroth of the non-linear function $\mathbf{A}(\mathbf{p})$. Setting equation (4) equal to zero and neglecting higher order terms.

$$\boldsymbol{J}(\mathbf{p_0})\triangle\mathbf{p} = -\mathbf{A}(\mathbf{p_0}) \tag{5}$$

The linear system (5) is the basis for the Newton's algorithm for finding the zeroth of the non-linear function $\mathbf{A}(\mathbf{p})$. The linear system (5) is solved by the CG algorithm [12]. The pseudo code is presented in the Algorithm 1. We have implemented the algorithm in C^{++}.

Algorithm 1: Newton-CG algorithm for Poisson Boltzmann equation.

```
Mesh the domain;
Form the non-linear system, $\mathbf{A}(\mathbf{p})$, on the mesh by FVM;
Find initial guess $\mathbf{p_0}$;
k = 0 ;
while ($k \leq max_{iter}$ or $\|\triangle\mathbf{p}\|_{L_2} \leq tol$ or $\|\mathbf{A}(\mathbf{p})\|_{L_2} \leq tol$) do
    Solve the discrete system $\boldsymbol{J}(\mathbf{p_k})\triangle\mathbf{p} = -\mathbf{A}(\mathbf{p_k})$;
    $\mathbf{p_{k+1}} = \mathbf{p_k} + \triangle\mathbf{p}$;
    $k^{++}$;
end
```



(a) Newton's iteration vs log of $\|\mathbf{A}(\mathbf{p})\|_{L_2}$ on a 50×50 mesh

(b) Newton's iteration vs log of $\|\Delta(\mathbf{p})\|_{L_2}$ on a 50×50 mesh.

Figure 1: Example

3 Numerical Examples

3.1 Example 1

In this example, we are solving equation (6). Here, $\gamma = 1.0$, and the exact solution is given as $p = x\,(x-1)\,y\,(y-1)$. As mentioned, for making an initial guess for Newton's iteration, we are using two approaches. In the first approach, we are solving the linear part $(-\triangle p = f)$ of the nonlinear equation, and in the second approach we are solving the corresponding linearised equation $(-\triangle p + (p+100) = f)$ and using the solution the as the initial guess. The Newton's iteration for both of these initial guesses are reported in the Figure 1. The Figure 1(a) presents the convergence of the residual vector, and the Figure 1(b) presents the convergence of the distance vector.

We are solving the Jacobian system by the preconditioned Conjugate Gradient (ILU preconditioner) with a tolerance of 1×10^{-10}. It is clear from the Figures 1(a) and 1(b) that solving the corresponding linearised equation for the initial guess can make a big difference in computational time and efficiency.

$$-\triangle p + \gamma\,(p+100)\,e^{p} = f \qquad \textbf{in} \quad \Omega \tag{6}$$

$$p(x,y) = p^{\mathrm{D}} \qquad \textbf{on} \quad \partial\Omega_D \tag{7}$$

3.2 Example 2

A Newton's iteration consists solving a series of linear systems. Solving a linear system to a high degree of accuracy can be computationally very expensive. Through this example we demonstrate that it is not required to solve the linear system a very high degree of accuracy. It should be noted that elliptic problems with discontinuous coefficients can produces very ill conditioned linear systems [12]. In this experiment, we do not know the exact solution. We are solving the Poisson Boltzmann equation (8) on $\Omega = (-1, 1)$ with $k = 1.0$. For starting the Newton's iteration, initial guess is made by solving the linearised equation $\nabla(-\epsilon\nabla p) + k\,p = f$.

The computational domain Ω is divided into four equal subdomains based on the medium properties ϵ. In the first and third quadrant, $\epsilon = 100.0$, while into the second and fourth quadrant $\epsilon = 1.0$. The source term f is given as

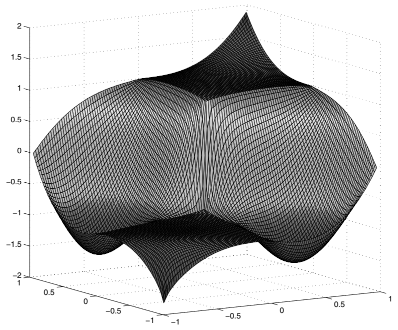
$$f = 2.0\,y\,(y-1) + 2.0\,x\,(x-1) - 100.0(x-1)\,y\,(y-1)\exp(x\,(x-1)\,y\,(y-1))$$

$$-\nabla(\epsilon\nabla p) + k\sinh(p) = f \qquad \textbf{in} \quad \Omega \tag{8}$$

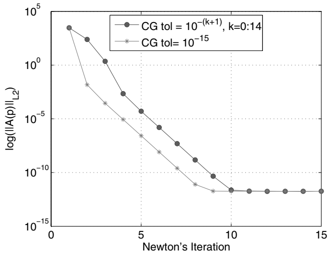
$$p(x, y) = x^3 + y^3 \qquad \textbf{on} \quad \partial\Omega_{\mathrm{D}} \tag{9}$$

Figure 2 presents the outcome of our numerical work. The Figure 2(a) shows the surface plot of the discrete solution. There is a singularity in the solution at origin as it can be expected for the interface problems [2], and the solution belongs in the fractional Sobolev space.

For checking the impact of the tolerance for the linear solver (5), we performed two experiments till 15 Newton's iteration. In the first instance, we choose a tolerance of 1×10^{-15} for all the 15 iterations, and in the second instance we adaptively defined the tolerance by $10^{-(k+1)}$. Here, k varies from 0 till 14. Figure 2(b) reports the outcome of this experiment. It is clear that the accuracy of the Jacobian solver does not affect the overall convergence of the Newton's algorithm. Such a approach is very practical in solving the nonlinear systems.



(a) Surface plot of the discrete solution. Solution is exhibiting a singularity at $(0, 0)$.



(b) Newton's iteration vs log of $\|A(\mathbf{p})\|_{L_2}$ for different tolerance of the Jacobian solver on a 50×50 mesh

Figure 2: Example 2

4 Conclusions

We have analysed the nonlinear solver for the Poisson Boltzmann equation from two angles initial guess and tolerance of the linear solver. It is shown that

1. Generating the initial guess by solving the corresponding linearised equation (replacing the nonlinear part with the corresponding linear part) is computationally efficient.
2. The nonlinear solver consists of solving a series of linear systems. These linear systems are referred to as the Jacobian systems. Solving the Jacobian systems to a good accuracy is very expensive. It is shown that adapting the tolerance of the Jacobian systems by the Newton iterations result in a computationally efficient approach.

References

[1] S. K. Khattri, Newton-Krylov Algorithm with Adaptive Error Correction for the Poisson-Boltzmann Equation, *MATCH Commun. Math. Comput. Chem.*, **1**, 56 (2006), 197–208.

[2] S. K. Khattri, Nonlinear elliptic problems with the method of finite volumes, *Differential Equations and Nonlinear Mechanics*, 2006.

[3] B. Aksoyw, Adaptive Multilevel Numerical Methods with Applications in Diffusive Biomolecular Reactions, *PhD Thesis, The University of California, San Diego*, (2001).

[4] F. Fogolari, A. Brigo and H. Molinari, The Poisson Boltzmann equation for Biomolecular electrostatics: A Tool for Structural Biology, *J. Mol. Recogn., John Wiley & Sons Ltd.*, **15** (2002) 377–392.

[5] M. Host, R. E. Kozack, F. Saied and S. Subramaniam, Treatment of Electrostatic Effects in Proteins: Multigrid-based Newton Iterative Method for Solution of the Full Nonlinear Poisson-Boltzmann Equation, *Proteins: Structure, Function, and Genetics*, **18**, (1994) 231–245.

[6] M. Holst, R. Kozack, F. Saied, and S. Subramaniam, Protein electrostatics: Rapid multigrid-based Newton algorithm for solution of the full nonlinear Poisson-Boltzmann equation, *J. of Bio. Struct. & Dyn.*, **11**, (1994) 1437–1445.

[7] M. Holst, R. Kozack, F. Saied, and S. Subramaniam, Multigrid-based Newton iterative method for solving the full Nonlinear Poisson-Boltzmann equation, *Biophys. J.*, 66, (1994) A130–A130.

[8] M. Holst, A robust and efficient numerical method for nonlinear protein modeling equations. *Technical Report CRPC-94-9, Applied Mathematics and CRPC, California Institute of Technology,* 1994.

[9] M. Holst and F. Saied, Multigrid solution of the Poisson-Boltzmann equation, *J. Comput. Chem.*, **14**, (1993) 105–113.

[10] M. Holst, MCLite: An Adaptive Multilevel Finite Element MATLAB Package for Scalar Nonlinear Elliptic Equations in the Plane. *UCSD Technical report and guide to the MCLite software package. Available on line at "http://scicomp.ucsd.edu/ mholst/pubs/publications.html".*

[11] M. Holst and F. Saied, Numerical solution of the nonlinear Poisson-Boltzmann equation: Developing more robust and efficient methods, *J. Comput. Chem.,* **16**, (1995) 337–364.

[12] H. A. van der Vorst, Iterative Krylov Methods for Large Linear Systems. *Cambridge monographs on applied and computational mathematics. Cambridge University Press, New York,* 2003.

[13] R. E. Bank, *PLTMG*: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 8.0, *Software Environ. Tools 5, SIAM, Philadelphia,* 1998.

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