Numerical Solution of the Nonlinear Poisson-Boltzmann Equation: Developing More Robust and Efficient Methods

Michael J. Holst
Department of Applied Mathematics and CRPC
California Institute of Technology 217-50
Pasadena, CA USA 91125

Faisal Saied
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL USA 61801

We present a robust and efficient numerical method for solution of the nonlinear Poisson-Boltzmann equation arising in molecular biophysics. The equation is discretized with the box method, and solution of the discrete equations is accomplished with a global inexact-Newton method, combined with linear multilevel techniques we have described in a paper appearing previously in this journal. A detailed analysis of the resulting method is presented, with comparisons to other methods that have been proposed in the literature, including the classical nonlinear multigrid method, the nonlinear conjugate gradient method, and nonlinear relaxation methods such as successive over-relaxation. Both theoretical and numerical evidence suggests that this method will converge in the case of molecules for which many of the existing methods will not. In addition, for problems which the other methods are able to solve, numerical experiments show that the new method is substantially more efficient, and the superiority of this method grows with the problem size. The method is easy to implement once a linear multilevel solver is available, and can also easily be used in conjunction with linear methods other than multigrid.

INTRODUCTION

In this paper, we consider numerical solution of the nonlinear Poisson-Boltzmann equation (PBE), the fundamental equation arising in the Debye-Hückel theory [1] of continuum molecular eletrostatics. In the case of a $1:1$ electrolyte, this equation can be written as

$$-\nabla\cdot(\epsilon(\mathbf{r})\nabla u(\mathbf{r})) + \bar{\kappa}^2(\mathbf{r})\sinh(u(\mathbf{r})) = \frac{4\pi e_c^2}{k_B T}\sum_{i=1}^{N_m} z_i\delta(\mathbf{r}-\mathbf{r}_i),\ u(\infty)=0, \quad (1)$$

a three-dimensional second order nonlinear partial differential equation governing the dimensionless electrostatic potential $u(\mathbf{r}) = e_c\Phi(\mathbf{r})/k_B^{-1}T^{-1}$, where $\Phi(\mathbf{r})$ is the electrostatic potential at a field position $\mathbf{r}$. The importance of this equation for modeling biomolecules is well-established; more detailed discussions of the use of the Poisson-Boltzmann equation may be found in the survey articles of Briggs and McCammon [2] and Sharp and Honig [3].

In the equation above, the coefficient $\epsilon(\mathbf{r})$ jumps by more than an order of magnitude across the interface between the molecule and surrounding solvent. The modified Debye-Hückel parameter $\bar{\kappa}$, proportional to the ionic strength of the solution, is discontinuous at the interface between the solvent region and an ion exclusion layer surrounding the molecular surface. The molecule itself is represented by N_m point charges $q_i = z_i e_c$ at positions $\mathbf{r}_i$, yielding the delta functions in (1), and the constants e_c, k_B, and T represent the charge of an electron, Boltzmann's constant, and the absolute temperature. Equation (1) is referred to as the *nonlinear Poisson-Boltzmann equation*, and it is often approximated by the *linearized Poisson-Boltzmann equation*, obtained by taking $\sinh(u(\mathbf{r})) \approx u(\mathbf{r})$ when $u(\mathbf{r}) << 1$.

In the nonlinear case, equation (1) presents severe numerical difficulties due to the rapid exponential nonlinearities, discontinuous coefficients, delta functions, and infinite domain. In this article, we present a survey of the numerical methods currently employed in the biophysics and biochemistry communities for the linearized and nonlinear Poisson-Boltzmann equations. We then propose an alternative numerical method, which is a combination of global inexact-Newton methods and multilevel methods. A detailed analysis of this method is presented, with comparisons to other methods that have been proposed in the literature, including the classical nonlinear multigrid method, the nonlinear conjugate gradient method, and nonlinear relaxation methods such as successive over-relaxation. Both theoretical and numerical evidence shows that the global inexact-Newton-multilevel method is superior to all other methods currently in use. In particular, this method is more robust (it converges in cases where the other methods fail) and substantially more efficient (the advantage of this method grows with the problem size). The method is easy to implement once a linear multilevel solver is available, and can also easily be used in conjunction with linear methods other than multigrid (the robustness is maintained, although the efficiency may be less).

Outline

We begin by discussing the form of the algebraic equations which are produced by standard discretization methods applied to equation (1). We then briefly review the methods that have been proposed and used for the linearized Poisson-Boltzmann equation, including the linear multilevel method we have presented in a previous paper. We then discuss in a little more detail the methods recently proposed in the literature for the nonlinear Poisson-Boltzmann equation, and present inexact-Newton methods as alternatives. We formulate an algorithm based on the combination of global inexact-Newton methods and our linear multilevel method, and

we state and prove two simple conditions which guarantee that the resulting inexact-Newton-multilevel method is globally convergent. Some test problems are then formulated, including some more difficult problems involving superoxide dismutase (SOD) and tRNA. Numerical experiments are then presented, showing that in fact the Newton-multilevel approach is both more robust and orders of magnitude more efficient than existing methods.

Due to the length of the paper, we also give a more detailed outline of the material below.

DISCRETIZING THE PBE

The infinite domain of equation (1) is often truncated to a finite domain $\Omega \subset \mathbb{R}^3$ with boundary Γ, and boundary conditions on Γ are provided by a known analytical solution; detailed discussions appear in Tanford [4] and in references [5, 6]. The equation then becomes:

$$-\nabla \cdot (\epsilon \nabla u) + \bar{\kappa} \sinh(u) = f \text{ in } \Omega \subset \mathbb{R}^3, \qquad (2)$$

$$u = g \text{ on } \Gamma,$$

where the source term in equation (1) has been denoted as the generic function f. The functions ϵ and $\bar{\kappa}$ may be only piecewise continuous functions on Ω, although we assume that the coefficient discontinuities are regular, and can be identified during the discretization process. In particular, to discretize (2) accurately, the domain Ω must be divided into discrete elements such that the discontinuities always lie along element boundaries, and never within an element. While this is not completely possible, it is important to achieve this as much as possible due to discrete approximation theory considerations (cf. the texts by Varga [7] and Strang and Fix [8]).

We begin by partitioning the domain Ω into the finite elements or volumes τ^j, such that:

- $\Omega \equiv \bigcup_{j=1}^{M} \tau^j$, where the elements τ^j are for example hexahedra or tetrahedra.
- The discontinuities in the coefficients $\{\epsilon, \bar{\kappa}, f\}$ are taken to lie along the boundaries of τ^j.
- The union of the (4 or 8) corners of the τ^j form the *nodes* $\mathbf{x}^i$ in the resulting *mesh* of nodes.
- The set $\{\tau^{j;i}\} \equiv \{\tau^j : \mathbf{x}^i \in \tau^j\}$ consists of all elements $\tau^{j;i}$ having $\mathbf{x}^i$ as a corner.
- Define $\tau^{(i)} \equiv \bigcup_j \tau^{j;i} \equiv \{\bigcup_j \tau^j : \mathbf{x}^i \in \tau^j\}$ to be the "box" around $\mathbf{x}^i$ formed by union of the $\tau^{j;i}$.
- We require continuity of u and $\bar{\mathbf{a}} \nabla u \cdot \mathbf{n}$ across the regions having different values of $\bar{\mathbf{a}}$.

The *box* (*integral*, *finite volume*) *method* has been one of the standard approaches for discretizing two- and three-dimensional interface problems occurring in reactor physics and reservoir simulation [7, 9]; similar methods are used in computational fluid dynamics. The motivation for these methods has been the attempt to enforce conservation of certain physical quantities in the discretization process.

The box-method approach

We begin by integrating (2) over an arbitrary $\tau^{(i)}$. The resulting equation is:

$$\int_{\tau^{(i)}} \left(-\nabla \cdot (\epsilon \nabla u) + \bar{\kappa} \sinh(u) \; d\mathbf{x} - f\right) \; d\mathbf{x} = 0.$$

Recalling the definition of $\tau^{(i)}$ as the union over j of the $\tau^{j;i}$, and employing the divergence theorem for the first term in the equation above, yields:

$$-\sum_j \int_{\partial\tau^{j;i}} (\epsilon\nabla u)\cdot\mathbf{n}\; ds + \sum_j \int_{\tau^{j;i}} (\bar{\kappa}\sinh(u) - f)\; d\mathbf{x} = 0,$$

where $\partial\tau^{j;i}$ is the boundary of $\tau^{j;i}$ and $\mathbf{n}$ is the unit normal. Note that all interior surface integrals in the first term vanish, since $\epsilon\nabla u\cdot\mathbf{n}$ must be continuous across the interfaces. We are left with:

$$-\int_{\partial\tau^{(i)}} (\epsilon\nabla u)\cdot\mathbf{n}\; ds + \sum_j \int_{\tau^{j;i}} (\bar{\kappa}\sinh(u) - f)\; d\mathbf{x} = 0,$$

where $\partial\tau^{(i)}$ denotes the boundary of $\tau^{(i)}$.

Since this last relationship holds exactly in each $\tau^{(i)}$, we can use this last equation to develop an approximation at the nodes $\mathbf{x}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ at the "centers" of the $\tau^{(i)}$ by employing quadrature rules and difference formulas. In particular, the volume integrals in the second two terms can be approximated with quadrature rules. Similarly, the surface integrals required to evaluate the first term can be approximated with quadrature rules, where ∇u is replaced with an approximation. Error estimates can be obtained from difference and quadrature formulas [7], or more generally by analyzing the box-method as a special Petrov-Galerkin finite element method [10, 11].

Non-uniform Cartesian meshes

The methods we develop in this paper for solving nonlinear algebraic equations can be applied quite generally to the equations arising from box or finite element method discretizations of nonlinear elliptic equations on very general non-Cartesian meshes. However, a few of the algebraic multilevel techniques we employ (discussed in more detail later in the paper and in [5, 6]) require *logically* Cartesian meshes. Note that *logically Cartesian* requires only that each mesh point have a nearest neighbor structure as for uniform Cartesian meshes, although the mesh lines need not be uniformly spaced, and the mesh axes need not be orthogonal, or even consist of straight lines at all. A logically Cartesian mesh is then clearly much more general than a *non-uniform Cartesian* mesh, which imposes the additional restriction that the axes consist of orthogonal straight lines. An example of a three-dimensional non-uniform Cartesian mesh, employing Chebyshev spacings around a centrally refined region, is shown in Figure 1.

Our *implementations* of the methods described in this paper, with which we will present numerical experiments in the sections to follow, are designed to allow a non-uniform Cartesian mesh to be employed, such as the mesh in Figure 1. The implementations treat all meshes as general non-uniform Cartesian meshes; no uniform mesh simplifications are employed. When possible, we will employ an adapted non-uniform Cartesian mesh, in order to discretize the given elliptic equation as accurately as possible near coefficient interfaces (for example, the nonlinear jump discontinuity test problem appearing later in the paper). Some of the numerical examples we consider, employing actual molecular data, require discretization on a uniform Cartesian mesh, since the sources of data currently available to us produce information only on uniform meshes (this is discussed in more detail later in the paper).

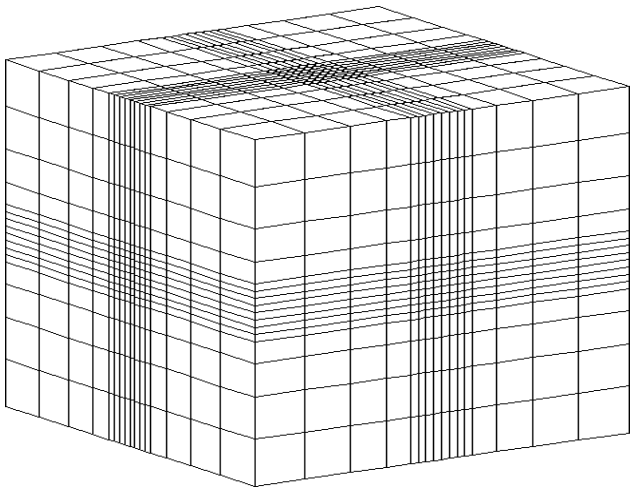

Figure 1: A 3D non-uniform Cartesian mesh.

In this section we will therefore describe the box method in the case of a non-uniform Cartesian mesh, so that the τ^j appearing above are hexahedral elements, whose six sides are parallel to the coordinate axes. By restricting our discussion to elements which are non-uniform Cartesian, the spatial mesh may be characterized by the nodal points

$$\mathbf{x} = (x, y, z) \text{ such that } \left\{ \begin{array}{l} x \in \{x_0, x_1, \ldots, x_{I+1}\} \\ y \in \{y_0, y_1, \ldots, y_{J+1}\} \\ z \in \{z_0, z_1, \ldots, z_{K+1}\} \end{array} \right\}.$$

Any such mesh point we denote as $\mathbf{x}_{ijk} = (x_i, y_j, z_k)$, and we define the *mesh spacings* as

$$h_i = x_{i+1} - x_i, \qquad h_j = y_{j+1} - y_j, \qquad h_k = z_{k+1} - z_k,$$

which are not required to be equal or uniform.

To each mesh point $\mathbf{x}_{ijk} = (x_i, y_j, z_k)$, we associate the closed three-dimensional hexahedral region $\tau^{(ijk)}$ "centered" at $\mathbf{x}_{ijk}$, defined by

$$x \in \left[x_i - \frac{h_{i-1}}{2}, x_i + \frac{h_i}{2}\right], \quad y \in \left[y_j - \frac{h_{j-1}}{2}, y_j + \frac{h_j}{2}\right],$$

$$z \in \left[y_k - \frac{h_{k-1}}{2}, z_k + \frac{h_k}{2} \right].$$

Integrating (2) over $\tau^{(ijk)}$ for each mesh-point $\mathbf{x}_{ijk}$ and employing the divergence theorem as above yields as before:

$$-\int_{\partial\tau^{(ijk)}} (\epsilon \nabla u) \cdot \mathbf{n} \; ds + \int_{\tau^{(ijk)}} (\bar{\kappa} \sinh(u) - f) \; d\mathbf{x} = 0.$$

The volume integrals are approximated with quadrature:

$$\int_{\tau^{(ijk)}} p \; d\mathbf{x} \approx meas(\tau^{(ijk)}) p_{ijk},$$

where $p_{ijk} = p(\mathbf{x}_{ijk})$, and where the volume of $\tau^{(ijk)}$ is

$$meas(\tau^{(ijk)}) = \left[\frac{(h_{i-1} + h_i)(h_{j-1} + h_j)(h_{k-1} + h_k)}{8} \right].$$

Since ϵ is a scalar, the surface integral reduces to:

$$\int_{\partial\tau^{(ijk)}} \epsilon (u_x + u_y + u_z) \cdot \mathbf{n} \; ds.$$

This integral reduces further to six two-dimensional plane integrals on the six faces of the $\tau^{(ijk)}$, and are approximated with the analogous two-dimensional rule, after approximating the partial derivatives with centered differences. Introducing the notation $p_{i-1/2,j,k} = p(x_i - h_{i-1}/2, y_j, z_k)$, and $p_{i+1/2,j,k} = p(x_i + h_i/2, y_j, z_k)$, the resulting discrete equations can be written as:

$$\epsilon_{i-1/2,j,k} \left(\frac{u_{ijk} - u_{i-1,j,k}}{h_{i-1}} \right) \frac{(h_{j-1} + h_j)(h_{k-1} + h_k)}{4}$$

$$+\epsilon_{i+1/2,j,k} \left(\frac{u_{ijk} - u_{i+1,j,k}}{h_i} \right) \frac{(h_{j-1} + h_j)(h_{k-1} + h_k)}{4}$$

$$+\epsilon_{i,j-1/2,k} \left(\frac{u_{ijk} - u_{i,j-1,k}}{h_{j-1}} \right) \frac{(h_{i-1} + h_i)(h_{k-1} + h_k)}{4}$$

$$+\epsilon_{i,j+1/2,k} \left(\frac{u_{ijk} - u_{i,j+1,k}}{h_j} \right) \frac{(h_{i-1} + h_i)(h_{k-1} + h_k)}{4}$$

$$+\epsilon_{i,j,k-1/2} \left(\frac{u_{ijk} - u_{i,j,k-1}}{h_{k-1}} \right) \frac{(h_{i-1} + h_i)(h_{j-1} + h_j)}{4}$$

$$+\epsilon_{i,j,k+1/2} \left(\frac{u_{ijk} - u_{i,j,k+1}}{h_k} \right) \frac{(h_{i-1} + h_i)(h_{j-1} + h_j)}{4}$$

$$+meas(\tau^{(ijk)}) \left(\bar{\kappa}_{ijk} \sinh(u_{ijk}) - f_{ijk} \right) = 0.$$

We have one such nonlinear algebraic equation for each u_{ijk} approximating $u(\mathbf{x}_{ijk})$ at the nodes:

$$\{\mathbf{x}_{ijk};\; i = 0, \ldots, I+1; j = 0, \ldots, J+1; k = 0, \ldots, K+1\}.$$

This set of equations represents the nonlinear algebraic system which we consider for the remainder of the paper.

The algebraic equations

After using the Dirichlet boundary data from (2), only equations for the interior nodes remain:

$$\{\mathbf{x}_{ijk};\; i = 1, \ldots, I;\; j = 1, \ldots, J;\; k = 1, \ldots, K\}.$$

We denote the total number of unknowns in the system of equations as $n = I \cdot J \cdot K$, and it is convenient to consider a vector-oriented ordering of the unknowns. For the non-uniform Cartesian mesh we have described, the *natural ordering* is defined as:

$$\mathbf{x}_p = \mathbf{x}_{ijk}, \qquad p = (k-1) \cdot I \cdot J + (j-1) \cdot I + i,$$

where

$$i = 1, \ldots, I; \quad j = 1, \ldots, J; \quad k = 1, \ldots, K,$$

which defines a one-to-one mapping between $\mathbf{x}_p$ and $\mathbf{x}_{ijk}$, and defines $\mathbf{x}_p$ for $p = 1, \ldots, n$. Employing the natural ordering of the meshpoints to order the unknowns u_i in the vector u yields a single nonlinear algebraic system of equations of the form:

$$Au + N(u) - f = 0, \tag{3}$$

where the vector f consists of components $meas(\tau^{(i)}) f_i$ for each of the mesh points $\mathbf{x}_i$, and the function $N(u)$ is a nonlinearity with "diagonal form", in that $N(u) = (N_1(u), \ldots, N_n(u))^T$, with $N_i(u) = N_i(u_i)$. Here, $N_i(u_i) = meas(\tau^{(i)}) \bar{\kappa}(\mathbf{x}_i) \sinh(u_i)$. In the linear case, $N_i(u_i) = meas(\tau^{(i)}) \bar{\kappa}(\mathbf{x}_i) u_i$.

The natural ordering of the unknowns u_i gives rise to a matrix A representing the linear part of (3) which is seven-banded and block-tridiagonal. This banded structure in the case of non-uniform Cartesian meshes allows for very efficient implementations of iterative methods for numerical solution of the discrete linear and nonlinear equations; the seven-banded form is depicted in Figure 2 for a $3 \times 3 \times 3$ non-uniform Cartesian mesh.

It is not difficult to show [5, 6] that the matrix A in (3) arising from box method discretization on a non-uniform Cartesian mesh is symmetric positive definite (SPD). This symmetry property holds for both box and finite element method discretizations on very general meshes, and is a consequence of the fact that the original differential operator is formally self-adjoint. Note that simple finite differences *do not* naturally produce symmetric matrices from self-adjoint differential operators except for fully uniform Cartesian meshes, unless very special care is taken.

Much more difficult questions, which we will not consider further here, are the well-posedness of the full nonlinear system (3), as well as the original continuous problem (2). It can be shown that both (3) and (2) have unique solutions depending continuously on the problem data. These and other technical questions are addressed quite fully in references [5, 6].

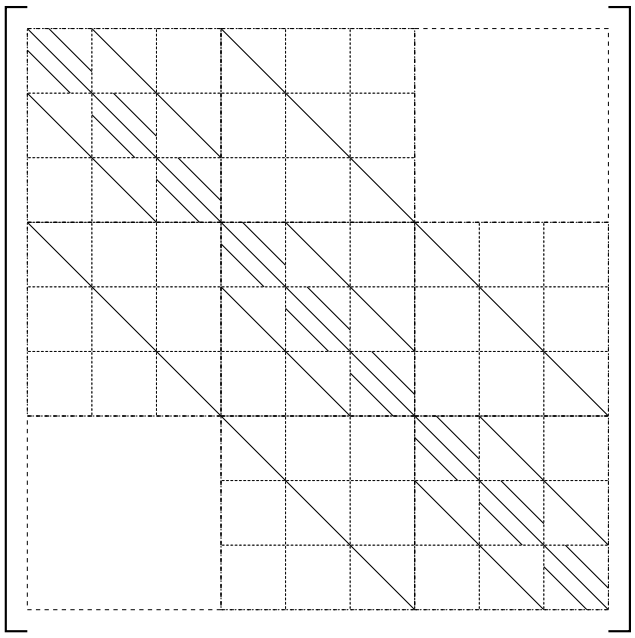

Figure 2: Banded matrix produced by the box-method.

LINEARIZED PBE METHODS

If the nonlinear term in equation (3) is zero, $N_i(u_i) \equiv 0$, or if it is linear, $N_i(u_i) = h^2\bar{\kappa}(\mathbf{x}_i)u_i$, in which case the term can be added to the diagonal of the matrix A in (3), then we are faced with linear algebraic equations:

$$Au = f, \tag{4}$$

where the matrix A is symmetric positive definite (SPD). The matrix A is a linear operator mapping $\mathbb{R}^n$ into $\mathbb{R}^n$, the space $\mathbb{R}^n$ being a linear space equipped with an inner-product $(\cdot,\cdot)$ inducing a norm $\|\cdot\|$ defined as follows:

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

Since the matrix A is SPD, it defines a second inner-product and norm:

$$(u,v)_A = (Au,v), \qquad \|u\|_A = (u,u)_A^{1/2}, \qquad \forall u, v \in \mathbb{R}^n.$$

While our purpose here is not to discuss the mathematical structure of $\mathbb{R}^n$, the importance of either norm which we may associate with $\mathbb{R}^n$ (and hence the inner-product which induces the particular norm) is that the norm defines a *metric* or *distance* function on the space $\mathbb{R}^n$, which allows us to measure the distance between points in $\mathbb{R}^n$. Equipped with only the inner-product and norm on $\mathbb{R}^n$, one can establish simple conditions for linear and nonlinear iterative methods to guarantee certain desirable convergence properties.

Classical linear methods

Linear iteration methods for solving the equation (4) for the unknown u can be thought of as having the form:

Algorithm 1 *(Basic Linear Method for $Au = f$)*

$$u^{i+1} = u^i + B(f - Au^i) = (I - BA)u^i + Bf,$$

where B is an SPD matrix approximating A^{-1} in some sense, and where the method begins with some initial "guess" at the true solution u, namely u^0. Subtracting the above equation from the following identity for the true solution u:

$$u = u - BAu + Bf,$$

yields an equation for the error $e^i = u - u^i$ at each iteration of the method:

$$e^{i+1} = (I - BA)e^i = \cdots = (I - BA)^{i+1}e^0. \tag{5}$$

The convergence of Algorithm 1, which refers to the question of whether $u^i \to u$ (or equivalently $e^i \to 0$) as $i \to \infty$, is determined completely by the spectral radius (the eigenvalue of largest magnitude) of the error propagation operator:

$$E = I - BA,$$

which we denote as $\rho(E)$.

Theorem 1 *The condition $\rho(E) < 1$ is necessary and sufficient for convergence of Algorithm 1.*

Proof. See for example Theorem 7.1.1 in Ortega [12]. □

If λ is an eigenvalue of E, then since $|\lambda|\|u\| = \|\lambda u\| = \|Eu\| \le \|E\|\,\|u\|$ for any norm $\|\cdot\|$, it follows that $\rho(E) \le \|E\|$ for all norms $\|\cdot\|$ (equality holds if and only if E is symmetric with respect the inner-product defining $\|\cdot\|$). Therefore, $\|E\| < 1$ and $\|E\|_A < 1$ are both sufficient conditions for convergence of Algorithm 1. In fact, it is the norm of the error propagation operator which will bound the reduction of the error at each iteration, which follows from (5):

$$\|e^{i+1}\|_A \le \|E\|_A \|e^i\|_A \le \|E\|_A^{i+1} \|e^0\|_A. \tag{6}$$

The spectral radius $\rho(E)$ of the error propagator E is called the *convergence factor* for Algorithm 1, whereas the norm of the error propagator $\|E\|$ is referred to as the *contraction number* (with respect to the particular choice of norm $\|\cdot\|$).

We mention now some classical linear iterations for discrete elliptic equations $Au = f$, where A is an SPD matrix. Since A is SPD, we may write $A = D - L - L^T$, and where D is a diagonal matrix and L a strictly lower-triangular matrix. Some of the classical variations of Algorithm 1 take as $B \approx A^{-1}$ the following:

(1) Richardson: $B = \lambda_{\max}^{-1}(A)$
(2) Jacobi: $B = D^{-1}$
(3) Gauss-Seidel: $B = (D - L)^{-1}$
(4) SOR: $B = \omega(D - \omega L)^{-1}$

Consider the case of the Poisson equation with zero Dirichlet boundary conditions discretized with the box-method on a uniform mesh with m mesh-points in each mesh direction ($n = m^3$) and mesh spacing $h = 1/(m + 1)$. This is equation (2) with $\epsilon \equiv 1$ and $\bar{\kappa} \equiv 0$. In this case, the eigenvalues of both A and the error propagation matrix E can be determined analytically, allowing for an analysis of the convergence rates of the Richardson, Jacobi, Gauss-Seidel, and SOR iterations:

(1) Richardson: $\rho(E) = 1 - O(h^2)$
(2) Jacobi: $\rho(E) = 1 - O(h^2)$
(3) Gauss-Seidel: $\rho(E) = 1 - O(h^2)$
(4) SOR: $\rho(E) = 1 - O(h)$

The same dependence on h is exhibited for one- and two-dimensional problems. Therein lies the fundamental problem with all classical relaxation methods: as $h \to 0$, then for the classical methods $\rho(E) \to 1$, so that the methods converge more and more slowly as the problem size is increased. This same behavior is also demonstrated for discretized forms of more general equations on more general meshes, such as those considered in this paper.

In the paper of Nicholls and Honig [13], an adaptive SOR procedure is developed for the linearized Poisson-Boltzmann equation, employing a power method to estimate the largest eigenvalue of the Jacobi iteration matrix, which enables estimation of the optimal relaxation parameter for SOR using Young's formula (page 110 in Varga [7]). The eigenvalue estimation technique employed is similar to the power method approach discussed on page 284 in Varga [7]. In the implementation of the method in the computer program DELPHI, several additional techniques are employed to increase the efficiency of the method. In particular, a red/black ordering is employed allowing for vectorization, and array-oriented data structures (as opposed to three-dimensional grid data structures) are employed to maximize vector lengths. The implementation is also specialized to the linearized Poisson-Boltzmann equation, with constants hard-coded into the loops rather than loaded as vectors to reduce vector loads.

In several recent papers [5, 6, 14], we considered an SOR method provided with the optimal relaxation parameter, implemented with a red/black ordering and array oriented data structures, yielding maximal vector lengths and very high performance on both the Convex C240 and the Cray Y-MP. In detailed comparisons with specially designed linear multilevel methods (which we will briefly review in a moment), experiments indicated that the linear multilevel methods were superior to the relaxation methods such as SOR, and the superiority grew with the problem size.

Linear conjugate gradient methods

The conjugate gradient (CG) method was developed by Hestenes and Stiefel [15] for linear systems with symmetric positive definite operators A. It is common to *precondition* the linear system by the SPD *preconditioning operator* $B \approx A^{-1}$, in which case the generalized or preconditioned conjugate gradient method [16] results. Our purpose in this section is to briefly examine the algorithm and its contraction properties. The *Omin* [17] implementation of the CG method has the form:

Algorithm 2 *(Preconditioned CG)*

Let $u^0 \in \mathbb{R}^n$ *be given.*
$r^0 = f - Au^0, \quad s^0 = Br^0, \quad p^0 = s^0.$
Do $i = 0, 1, \ldots$ *until convergence:*
$\alpha_i = (r^i, s^i)/(Ap^i, p^i)$
$u^{i+1} = u^i + \alpha_i p^i$
$r^{i+1} = r^i - \alpha_i A p^i$
$s^{i+1} = Br^{i+1}$
$\beta_{i+1} = (r^{i+1}, s^{i+1})/(r^i, s^i)$
$p^{i+1} = s^{i+1} + \beta_{i+1} p^i$
End do.

The algorithm can be shown to converge in n steps since the preconditioned operator BA is A-SPD [17]. Note that if $B = I$, then this algorithm is exactly the Hestenes and Stiefel algorithm. It can be shown (see for example references [5, 6] for a more complete discussion and additional references) that the error in the conjugate gradient method contracts according to the following formula:

$$\|e^{i+1}\|_A \leq 2 \left(\frac{\sqrt{\kappa_A(BA)} - 1}{\sqrt{\kappa_A(BA)} + 1} \right)^{i+1} \|e^0\|_A,$$

where the *generalized* or A-condition number of the matrix BA is defined as the quantity

$$\kappa_A(BA) = \|BA\|_A \|(BA)^{-1}\|_A = \frac{\lambda_{\max}(BA)}{\lambda_{\min}(BA)}.$$

It is not difficult to show (cf. references [5, 6]) that the spectral radius of a linear method defined by a SPD B, provided with an optimal relaxation parameter, is given by:

$$\delta_{\text{opt}} = 1 - \frac{2}{1 + \kappa_A(BA)}, \tag{7}$$

whereas the CG contraction is bounded by:

$$\delta_{\text{cg}} = 1 - \frac{2}{1 + \sqrt{\kappa_A(BA)}}. \tag{8}$$

Assuming $B \neq A^{-1}$, we always have $\kappa_A(BA) > 1$, so we must have that $\delta_{\text{cg}} < \delta_{\text{opt}} \leq \delta$, where δ is the contraction rate of the linear method defined by B. This

implies that it always pays in terms of an improved contraction number to use the conjugate gradient method to accelerate a linear method; the question remains of course whether the additional computational labor involved will be amortized by the improvement.

Unfortunately, the convergence rates of both linear methods and conjugate gradient methods depend on the condition number $\kappa_A(BA)$. When B is defined by the standard linear methods or other approaches such as incomplete factorizations, it is not difficult to show that $\kappa_A(BA)$ grows with the problem size, sometimes quite rapidly, which results in the contraction rates (7) and (8) worsening (approaching 1) as the problem size is increased. Multilevel methods were created to solve exactly this problem; in many cases, it can be shown that $\kappa_A(BA)$ remains bounded, independent of the problem size, when the operator B is defined by a multigrid algorithm.

The application of conjugate gradient methods to the Poisson-Boltzmann equation is discussed by Davis and McCammon [18], including comparisons with some classical iterative methods such as SOR. The conclusions of their study were that conjugate gradient methods were substantially more efficient than relaxation methods including SOR, and that incomplete factorizations were effective preconditioning techniques for the linearized Poisson-Boltzmann equation. We showed in several recent papers [5, 6, 14] that in fact for the problem sizes typically considered, the advantage of conjugate gradient methods over SOR is not so clear if an efficient SOR procedure is implemented, and if a near optimal parameter is available. Of course, if larger problem sizes are consider, then the superior complexity properties of the conjugate gradient methods in three-dimensions (cf. references [5, 6] for a detailed discussion) will eventually yield a more efficient technique than SOR.

In recent papers [5, 6, 14], we also considered several more advanced preconditioners than considered in by Davis and McCammon [18], including methods developed by van der Vorst and others [19], which employ special orderings to improve vectorization during the back substitutions. We presented experiments with a preconditioned conjugate gradient method (implemented so as to yield maximal vector lengths and high performance), provided with four different preconditioners: (1) diagonal scaling; (2) an incomplete Cholesky factorization (the method for which Davis and McCammon present results [18]); (3) the same factorization but with a *plane-diagonal-wise ordering* [19] allowing for some vectorization of the backsolves; and (4) a vectorized *modified* incomplete Cholesky factorization [19] with modification parameter $\alpha = 0.95$, which has an improved convergence rate over standard ICCG. Experiments indicated that the linear multilevel methods were superior to all of the conjugate gradient methods, and the superiority grew with the problem size.

Linear multilevel methods

Multilevel (or *multigrid*) methods are highly efficient numerical techniques for solving the algebraic equations arising from the discretization of partial differential equations. These methods were developed in direct response to the deficiencies of the classical iterations and conjugate gradient methods discussed in the previous sections. Some of the early fundamental papers are are due to Brandt [20] and Hackbusch [21], and a comprehensive analysis of the many different aspects of these methods is given in the text by Hackbusch [22].

Consider the nested sequence of finite-dimensional spaces $\mathbb{R}^{n_1} \subset \mathbb{R}^{n_2} \subset \cdots \subset \mathbb{R}^{n_J} \equiv \mathbb{R}^n$. To formulate a multigrid method, we require prolongation operators I_{k-1}^k mapping $\mathbb{R}^{n_{k-1}}$ into $\mathbb{R}^{n_k}$, restriction operators I_k^{k-1} mapping $\mathbb{R}^{n_k}$ into $\mathbb{R}^{n_{k-1}}$, and coarse space problems $A_k u_k = f_k$, where A_k maps $\mathbb{R}^{n_k}$ into itself. We also require *smoothing* operators $R_k \approx A_k^{-1}$. The prolongation typically corresponds to an interpolation, and the restriction is taken as a multiple of the transpose, $I_k^{k-1} = cI_{k-1}^k$. We begin with the problem $Au = f$ in the finest space $\mathbb{R}^n$, and in each space $\mathbb{R}^{n_k}$ we must somehow construct the approximating coarse system $A_k u_k = f_k$ of fewer dimensions, the smoothing operators $R_k \approx A_k^{-1}$, and the transfer operators I_k^{k-1} and I_{k-1}^k relating adjacent spaces.

If we can construct the various operators mentioned above, then the multilevel or multigrid algorithm can be stated in a very simple recursive fashion. For the linear system $Au = f$ in the finest space $\mathbb{R}^n$, the algorithm returns the approximate solution u^{i+1} after one iteration of the method applied to the initial approximate u^i.

Algorithm 3 *(Symmetric Multilevel Method)*

$$u^{i+1} = ML(J, u^i, f)$$

where $u_k^1 = ML(k, u_k^0, f_k)$ *is defined recursively:*

IF $(k = 1)$ *THEN:*
(1) Direct solve: $u_1^1 = A_1^{-1} f_1$.
ELSE:
(1) Pre-smooth: $w_k = u_k^0 + R_k^T(f_k - A_k u_k^0)$.
(2) Correction: $v_k = w_k + I_{k-1}^k\{ML(k-1, 0, I_k^{k-1}[f_k - A_k w_k])\}$
(3) Post-smooth: $u_k^1 = v_k + R_k(f_k - A_k v_k)$.
END.

The transpose R_k^T of the post-smoothing operator R_k is used for the pre-smoothing operator because it can be shown that the resulting operator B defined implicitly by the multigrid algorithm is symmetric; in other words, multigrid can be viewed as the basic linear method (1),

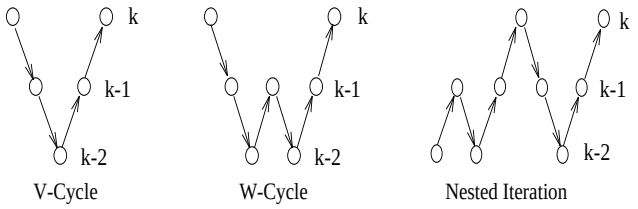

Figure 3: Various multilevel algorithms.

where the symmetric operator B is only defined implicitly. Therefore, the multigrid algorithm can also be used as a preconditioner for the conjugate gradient method, even though B is not explicitly available.

The procedure just outlined involving correcting with the coarse space once each iteration is referred to as the *V-cycle* [20]. Another variation is termed the *W-cycle*, in which two coarse space corrections are performed per level at each iteration. More generally, the *p-cycle* would involve p coarse space corrections per level at each iteration for some integer $p \geq 1$. The *full multigrid method* [20] or *nested iteration technique* [22] begins with the coarse space, prolongates the solution to a finer space, performs a p-cycle, and repeats the process, until a p-cycle is performed on the finest level. The methods can be depicted as in Figure 3.

Various techniques have been proposed for constructing the coarse problems $A_k u_k = f_k$. We mention in particular the references [23, 24, 25, 26], in which most of the techniques currently in use, including those discussed below, were first discovered and developed. Note that a simple discretization of the same differential equation, but on coarser meshes, is effective only in the case of smooth coefficients. In the presence of discontinuous coefficients, convergent multigrid methods can be constructed only if special care is taken in the construction of the coarse space subproblems $A_k u_k = f_k$, and in the construction of the transfer operators I_{k-1}^k and I_k^{k-1}. The effectiveness of coefficient averaging techniques, applied to the linearized Poisson-Boltzmann equation, is discussed in detail in Holst and Saied [14], and also in references [5, 6]. A more robust approach is to algebraically enforce the *variational conditions*

$$A_{k-1} = I_k^{k-1} A_k I_{k-1}^k, \qquad I_k^{k-1} = (I_{k-1}^k)^T. \tag{9}$$

This technique is described in detail and applied to the linearized Poisson-Boltzmann equation and related problems in references [5, 6]. While it is quite computationally complex to impose (9) algebraically (the expressions for which require a symbolic manipulator such as MAPLE), the advantage of this approach can be demonstrated both theoretically and numerically [5, 6]. Note however that for the linearized Poisson-Boltzmann equation, the less expensive coefficient averaging approach has been shown to be sufficient to produce very efficient multigrid methods for nearly all test problems we have encountered, and for the purposes of this paper, we will consider mainly methods based on coefficient averaging, as presented in Holst and Saied [14].

In an earlier paper [14], we presented results for the linearized Poisson-Boltzmann equation for a single multilevel method, which was selected from several multilevel methods as the most efficient; several different multilevel methods for a more difficult jump discontinuity test problem are compared in references [5, 6]. The particular multilevel method chosen in Holst and Saied [14] for the linearized Poisson-Boltzmann equation was constructed from the following components (discussed in detail in Holst and Saied [14] and in references [5, 6]).

A harmonic coefficient averaging technique was used to create coefficients for the coarser mesh problems, and a standard box method was used to discretize the problem on the coarse mesh using the averaged coefficients. Operator-based prolongation was also employed. The pre- and post-smoothing operators which we employed corresponded to red/black Gauss-Seidel iterations, where each smoothing step consisted of ν sweeps, with each sweep consisting of one sub-sweep with the red points followed by one sub-sweep with the black points. A *variable v-cycle* [27] approach to accelerating multilevel convergence was employed, so that the number of pre- and post-smoothing sweeps changes on each level; in our implementation, the number of pre- and post-smoothing sweeps at level k was given by $\nu = 2^{J-k}$, so that one pre- and post-smoothing was performed on the finest level $k = J$, and $\nu = 2^{J-1}$ sweeps on the coarsest level $k = 1$, with the number increasing geometrically on coarser levels. The coarse problem was solved with the Hestenes-Stiefel conjugate gradient method (Algorithm 2 with $B = I$). It was demonstrated in Holst and Saied [14] that this multilevel methods was substantially more efficient than relaxation and conjugate gradient methods for the linearized Poisson-Boltzmann equation and similar problems.

NONLINEAR PBE METHODS

Studies of numerical solution techniques for the nonlinear Poisson-Boltzmann equation have employed nonlinear Gauss-Seidel methods [28], nonlinear SOR methods [13], nonlinear conjugate gradient methods [29], and more recently, nonlinear multigrid methods [5, 6, 30]. Therefore, we will focus on these methods for the comparisons to inexact-Newton-multilevel methods in following sections. We first briefly describe these methods, and then discuss what results were obtained with these methods for the nonlinear Poisson-Boltzmann equation.

Nonlinear relaxation methods

The classical linear methods discussed earlier, such as Gauss-Seidel and SOR, can be extended in the obvious way to nonlinear algebraic equations of the form (3). In each case, the method can be viewed as a fixed-point iteration:

$$u^{n+1} = G(u^n).$$

Of course, implementations of these methods, which we refer to as nonlinear Gauss-Seidel and nonlinear SOR methods, now require the solution of a sequence of one-dimensional nonlinear problems for each unknown in one step of the method. Since the one-dimensional nonlinear problems are often solved with Newton's method, these methods are also referred to as Gauss-Seidel-Newton and SOR-Newton methods, meaning that the Gauss-Seidel or SOR iteration is the main or outer iteration, whereas the inner iteration is performed by Newton's method.

The convergence properties of these types of methods, as well as a myriad of variations and related methods, are discussed in detail in Ortega and Rheinboldt [31]. Note, however, that the same difficulty arising in the linear case also arises here: as the problem size is increased (the mesh size is reduced), these methods converge more and more slowly. As a result, we will consider alternative methods in a moment, such as nonlinear conjugate gradient methods, nonlinear multilevel methods, and finally inexact-Newton methods.

Nonlinear Gauss-Seidel is used by Allison et al. [28] for the nonlinear Poisson-Boltzmann equation, where a nonlinear Gauss-Seidel procedure is developed for the full nonlinear Poisson-Boltzmann equation, employing a continuation method to handle the numerical difficulties created by the exponential nonlinearity. Polynomial approximations of the exponential function are employed, and the degree of the polynomial is continued from degree one (linearized Poisson-Boltzmann equation) to degree nineteen. At each continuation step, the nonlinear Poisson-Boltzmann equation employing the weaker nonlinearity is solved with nonlinear Gauss-Seidel iteration. The final degree nineteen solution is then used as an initial approximation for the full exponential nonlinear Poisson-Boltzmann equation, and nonlinear Gauss-Seidel is used to resolve the final solution. This procedure, while perhaps one of the first numerical solutions produced for the full nonlinear problem, is extremely time-consuming.

An improvement is, as in the linear case, to employ a nonlinear SOR iteration. The procedure works very well for the nonlinear Poisson-Boltzmann equation in many situations and is extremely efficient [13]; unfortunately, there are cases where the iteration diverges [32, 13]. In particular, it is noted on page 443 of Nicholls and Honig [13] that if the potential in the solvent (where the exponential term is evaluated) passes a threshold value of seven or eight, then the nonlinear SOR method they propose diverges. We will present some experiments with a nonlinear SOR iteration, provided with an experimentally determined near optimal relaxation parameter, and implemented with a red/black ordering and array oriented data structures for high performance.

Nonlinear conjugate gradient methods

Let A be an SPD matrix, $B(\cdot)$ a nonlinear mapping from $\mathbb{R}^n$ into $\mathbb{R}$, and let $(\cdot,\cdot)$ denote an inner-product in $\mathbb{R}^n$. The following minimization problem:

$$\text{Find } u \in \mathbb{R}^n \text{ such that } J(u) = \min_{v\in\mathbb{R}^n} J(v),$$

where

$$J(u) = \frac{1}{2}(Au,u) + B(u) - (f,u),$$

is equivalent to the associated zero-point problem:

$$\text{Find } u \in \mathbb{R}^n \text{ such that } F(u) = Au + N(u) - f = 0,$$

where $N(u) = B'(u)$; this follows by simply differentiating $J(u)$ to obtain the gradient mapping $F(\cdot)$ associated with $J(\cdot)$. We will assume here that both problems are uniquely solvable. A more detailed discussion of convex functionals and their related gradient mappings can be found in references [5, 6].

An effective approach for solving the zero-point problem, by exploiting the connection with the minimization problem, is the *Fletcher-Reeves* version [33] of the nonlinear conjugate gradient method, which takes the form:

Algorithm 4 *(Fletcher-Reeves Nonlinear CG)*

Let $u^0 \in \mathbb{R}^n$ *be given.*
$r^0 = f - N(u^0) - Au^0, \quad p^0 = r^0.$
Do $i = 0, 1, \ldots$ *until convergence:*
  $\alpha_i =$ *(see below)*
  $u^{i+1} = u^i + \alpha_i p^i$
  $r^{i+1} = r^i + N(u^i) - N(u^{i+1}) - \alpha_i A p^i$
  $\beta_{i+1} = (r^{i+1}, r^{i+1})/(r^i, r^i)$
  $p^{i+1} = r^{i+1} + \beta_{i+1} p^i$
End do.

The directions p^i are computed from the previous direction and the new residual, and the steplength α_i is chosen to minimize the associated functional $J(\cdot)$ in the direction p^i. In other words, α_i is chosen to minimize $J(u^i + \alpha_i p^i)$, which is equivalent to solving the one-dimensional zero-point problem:

$$\frac{dJ(u^i + \alpha_i p^i)}{d\alpha_i} = 0.$$

Given the form of $J(\cdot)$ above, we have that

$$\begin{aligned} J(u^i + \alpha_i p^i) \quad = \quad & \frac{1}{2}(A(u^i + \alpha_i p^i), u^i + \alpha_i p^i) \\ & + B(u^i + \alpha_i p^i) - (f, u^i + \alpha_i p^i) \end{aligned}$$

A simple differentiation with respect to α_i (and some simplification) gives:

$$\frac{dJ(u^i+\alpha_i p^i)}{d\alpha_i} = \alpha_i(Ap^i,p^i)-(r^i,p^i) +(N(u^i+\alpha_i p^i)-N(u^i),p^i),$$

where $r^i = f - N(u^i) - Au^i$ is the nonlinear residual. The second derivative with respect to α_i will be useful also, which is easily seen to be:

$$\frac{d^2J(u^i+\alpha_i p^i)}{d\alpha_i^2} = (Ap^i,p^i)+(N'(u^i+\alpha_i p^i)p^i,p^i).$$

Now, Newton's method for solving the zero-point problem for α_i takes the form:

$$\alpha_i^{m+1} = \alpha_i^m - \delta^m$$

where

$$\delta^m = \frac{dJ(u^i+\alpha_i^m p^i)/d\alpha_i}{d^2J(u^i+\alpha_i^m p^i)/d\alpha_i^2}$$

$$= \frac{\alpha_i^m(Ap^i,p^i)-(r^i,p^i)+(N(u^i+\alpha_i^m p^i)-N(u^i),p^i)}{(Ap^i,p^i)+(N'(u^i+\alpha_i^m p^i)p^i,p^i)}.$$

The quantities (Ap^i,p^i) and (r^i,p^i) can be computed once at the start of each line search for α_i, each requiring an inner-product (Ap^i is available from the CG iteration). Each Newton iteration for the new α_i^{m+1} then requires evaluation of the nonlinear term $N(u^i+\alpha_i^m p^i)$ and inner-product with p^i, as well as evaluation of the derivative mapping $N'(u^i+\alpha_i p^i)$, application to p^i, followed by inner-product with p^i.

In the case that $N(\cdot)$ arises from the discretization of a nonlinear partial differential equation and is of *diagonal form*, meaning that the j-th component function of the vector $N(\cdot)$ is a function of only the j-th component of the vector of nodal values u, or $N_j(u) = N_j(u_j)$, then the resulting Jacobian matrix $N'(\cdot)$ of $N(\cdot)$ is a diagonal matrix. This situation occurs with box-method discretizations of the nonlinear Poisson-Boltzmann equation and similar equations. As a result, computing the term $(N'(u^i+\alpha_i p^i)p^i,p^i)$ can be performed with fewer operations than two inner-products.

The total cost for each Newton iteration (beyond the first) is then evaluation of $N(\cdot)$ and $N'(\cdot)$, and something less than three inner-products. Therefore, the line search can be performed fairly inexpensively in certain situations. If alternative methods are used to solve the one-dimensional problem defining α_i, then evaluation of the Jacobian matrix can be avoided altogether, although the Jacobian matrix is cheaply computable in the particular applications we are interested in here.

Note that if the nonlinear term $N(\cdot)$ is absent, then the zero-point problem is linear and the associated energy functional is quadratic:

$$F(u) = Au - f = 0, \qquad J(u) = \frac{1}{2}(Au,u)-(f,u).$$

In this case, the Fletcher-Reeves CG algorithm reduces to exactly the Hestenes-Stiefel [15] linear conjugate gradient algorithm (Algorithm 2 discussed earlier, with the preconditioner $B = I$). The exact solution to the linear problem $Au = f$, as well as to the associated minimization problem, can be reached in no more than n steps, where n is the dimension of the space $\mathbb{R}^n$ (see Theorem 8.6.1 in Ortega and Rheinboldt [31]). The calculation of the steplength α_i no longer requires the iterative solution of a one-dimensional minimization problem with Newton's method, since:

$$\frac{dJ(u^i+\alpha_i p^i)}{d\alpha_i} = \alpha_i(Ap^i,p^i)-(r^i,p^i) = 0$$

yields an explicit expression for the α_i which minimizes the functional J in the direction p^i:

$$\alpha_i = \frac{(r^i,p^i)}{(Ap^i,p^i)}.$$

In the recent paper of Luty et. al [29], a nonlinear conjugate gradient method is applied to the nonlinear Poisson-Boltzmann equation. The conclusions of their study were that the Fletcher-Reeves variant of the nonlinear conjugate gradient method, which is the natural extension of the Hestenes-Stiefel algorithm they had employed for the linearized Poisson-Boltzmann equation in an earlier study [18], was an effective technique for the nonlinear Poisson-Boltzmann equation. We note that it is remarked on page 1117 of the paper by Luty et al. [29] that solution time for the nonlinear conjugate gradient method on the full nonlinear problem is five times greater than for the linear method applied to the linearized problem. We will present experiments with the standard Fletcher-Reeves nonlinear conjugate gradient method, Algorithm 4, which they employed. Our implementation is aggressively optimized for high performance.

Nonlinear multilevel methods

Fully nonlinear multilevel methods were developed originally by Brandt [20] and Hackbusch [34]. These methods attempt to avoid Newton-linearization by accelerating nonlinear relaxation methods with multiple coarse problems. We are again concerned with the problem:

$$F(u) = Au + N(u) - f = 0.$$

Let us introduce the notation $M(\cdot) = A + N(\cdot)$, which yields the equivalent problem:

$$M(u) = f.$$

Consider a nested sequence of finite-dimensional spaces $\mathbb{R}^{n_1} \subset \mathbb{R}^{n_2} \subset \cdots \subset \mathbb{R}^{n_J} \equiv \mathbb{R}^n$, where $\mathbb{R}^{n_J}$ is the finest space and $\mathbb{R}^{n_1}$ the coarsest space, each space being connected to the others via prolongation and restriction operators, exactly as in the linear case described earlier. The *full approximation scheme* [20] or the *nonlinear multigrid method* [22] has the following form:

Algorithm 5 *(Nonlinear Multilevel Method)*

$$u^{i+1} = NML(J, u^i, f)$$

where $u_k^1 = NML(k, u_k^0, f_k)$ *is defined recursively:*

IF $(k = 1)$ *THEN:*
(1) Solve directly: $u_1^1 = M_1^{-1}(f_1)$.
ELSE:
(1) Restriction: $u_{k-1} = I_k^{k-1} u_k^0$,
$r_{k-1} = I_k^{k-1}(f_k - M_k(u_k^0))$.
(2) Coarse source term:
$f_{k-1} = M_{k-1}(u_{k-1}) - r_{k-1}$.
(3) Coarse problem:
$w_{k-1} = u_{k-1} - NML(k-1, u_{k-1}, f_{k-1})$.
(4) Prolongation: $w_k = I_{k-1}^k w_{k-1}$.
(5) Damping parameter: $\lambda =$ *(see below)*.
(6) Correction: $v_k = u_k^0 + \lambda w_k$.
(7) Post-smoothing: $u_k^1 = S_k(v_k, f_k)$.
END.

The practical aspects of this algorithm and variations are discussed by Brandt [20], and a convergence theory has been developed by Hackbusch [22], and more recently by Hackbusch and Reusken [35, 36, 37, 38].

Note that we have introduced a damping parameter λ in the coarse space correction step of Algorithm 5. In fact, without this damping parameter, the algorithm fails for difficult problems such as those with exponential or rapid nonlinearities. To explain how the damping parameter is chosen, we refer back to our discussion of nonlinear conjugate gradient methods. We begin with the following energy functional:

$$J_k(u_k) = \frac{1}{2}(A_k u_k, u_k)_k + B_k(u_k) - (f_k, u_k)_k.$$

As we have seen, the resulting minimization problem:

$$\text{Find } u_k \in \mathbb{R}^{n_k} \text{ such that } J_k(u_k) = \min_{v_k \in \mathbb{R}^{n_k}} J_k(v_k)$$

is equivalent to the associated zero-point problem:

$$\text{Find } u_k \in \mathbb{R}^{n_k} \text{ such that } F_k(u_k) = 0,$$

where $F_k(u_k) = A_k u_k + N_k(u_k) - f_k = 0$, and where $N_k(u_k) = B_k'(u_k)$. In other words, $F_k(\cdot)$ is a gradient mapping of the associated energy functional $J_k(\cdot)$, where we assume that both problems above are uniquely solvable.

In Hackbusch and Reusken [36], it is shown under certain conditions that the prolongated coarse space correction $w_k = I_{k-1}^k w_{k-1}$ is a descent direction for the functional $J_k(\cdot)$, meaning that there exists some $\lambda > 0$ such that

$$J_k(u_k + \lambda w_k) < J_k(u_k).$$

In other words, the nonlinear multigrid method can be made globally convergent if a damping parameter λ is found for each coarse grid correction. We can find such a λ by minimizing $J_k(\cdot)$ along the descent direction w_k, which is equivalent to solving the following one-dimensional problem:

$$\frac{dJ(u_k + \lambda w_k)}{d\lambda} = 0.$$

As in the discussion of the nonlinear conjugate gradient method, the one-dimensional problem can be solved with Newton's method:

$$\lambda^{m+1} = \lambda^m - \frac{\mathbf{X}}{\mathbf{Y}},$$

where (exactly as for the nonlinear CG method)

$$\begin{aligned} \mathbf{X} &= \lambda^m (A_k w_k, w_k)_k - (r_k, w_k)_k + (N_k(u_k + \lambda^m w_k) \\ &\quad - N_k(u_k), w_k)_k, \\ \mathbf{Y} &= (A_k w_k, w_k)_k + (N_k'(u_k + \lambda^m w_k) w_k, w_k)_k. \end{aligned}$$

Now, recall that the "direction" from the coarse space correction has the form: $w_k = I_{k-1}^k w_{k-1}$. The expressions for $\mathbf{X}$ and $\mathbf{Y}$ then take the form:

$$\begin{aligned} \mathbf{X} &= \lambda^m (A_k I_{k-1}^k w_{k-1}, I_{k-1}^k w_{k-1})_k - (r_k, I_{k-1}^k w_{k-1})_k \\ &\quad + (N_k(u_k + \lambda^m I_{k-1}^k w_{k-1}) - N_k(u_k), I_{k-1}^k w_{k-1})_k, \\ \mathbf{Y} &= (A_k I_{k-1}^k w_{k-1}, I_{k-1}^k w_{k-1})_k \\ &\quad + (N_k'(u_k + \lambda^m I_{k-1}^k w_{k-1}) I_{k-1}^k w_{k-1}, I_{k-1}^k w_{k-1})_k. \end{aligned}$$

It is not difficult to show [36] that certain finite element discretizations of the nonlinear elliptic problem we are considering, on two successively refined meshes, satisfy the following so-called *nonlinear variational conditions*:

$$A_{k-1} + N_{k-1}(\cdot) = I_k^{k-1} A_k I_{k-1}^k + I_k^{k-1} N_k(I_{k-1}^k \cdot),$$
$$I_k^{k-1} = (I_{k-1}^k)^T. \tag{10}$$

As in the linear case, these conditions are usually required [35, 36] to show theoretical convergence results about nonlinear multilevel methods. Unfortunately, unlike the linear case, there does not appear to be a way

to enforce these conditions algebraically (at least for the strictly nonlinear term $N_k(\cdot)$) in an efficient way. Therefore, if we employ discretization methods other than finite element methods, or cannot approximate the integrals accurately (such as if discontinuities occur within elements on coarser levels) for assembling the discrete nonlinear system, then the variational conditions will be violated. There is then no guarantee that the coarse grid correction is a descent direction. In other words, in the presence of coefficient discontinuities and/or non-finite element discretizations, the nonlinear multigrid method may not converge, and may not be a fully reliable, robust method.

Nonlinear multigrid methods have been considered by Oberoi and Allewell [30] and in references [5, 6] for the nonlinear Poisson-Boltzmann equation. For simple Poisson-Boltzmann equation problems, it has been shown to be an efficient method, and appears to demonstrate $O(n)$ complexity as does linear multigrid [5, 6, 30]. However, experiments performed elsewhere (see references [5, 6]) and below indicate that even a quite sophisticated implementation of nonlinear multigrid may diverge for difficult problems such as the nonlinear Poisson-Boltzmann equation with complex, large, or highly charged molecules. The inexact-Newton-multilevel methods we propose in the next section overcome these difficulties, and converge even for the most difficult problems.

The method we employ for our numerical experiments below is the nonlinear multilevel method presented earlier as Algorithm 5. All components required for this nonlinear method are as in the linear harmonically averaged multilevel method described in Holst and Saied [14] and in Holst [5, 6], except for the following required modifications. The pre- and post-smoothing iterations correspond to nonlinear Gauss-Seidel, where each smoothing step consisting of ν sweeps; as in the linear case, we employ a variable v-cycle so that ν increases as coarser levels are reached. Nonlinear operator-based prolongation [5, 6] is also employed for nested iteration; otherwise, linear operator-based prolongation is used. The coarse problem is solved with the nonlinear conjugate gradient method, and a damping parameter, as described earlier is required; otherwise, the method does not converge for rapid nonlinearities such as those present in the nonlinear Poisson-Boltzmann equation.

INEXACT-NEWTON METHODS

Given the nonlinear operator $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$, a generalization of the classical one-dimensional Newton's method for solving the problem $F(u) = 0$ is as follows:

$$F'(u^n)v^n = -F(u^n) \tag{11}$$

$$u^{n+1} = u^n + v^n, \tag{12}$$

where $F'(u^n)$ is the Jacobian matrix of partial derivatives:

$$F'(u) = \nabla F(u)^T = \left[\frac{\partial F_i(u)}{\partial u_j}\right],$$

where $F(u) = (F_1(u), \ldots, F_m(u))^T$, and where the function $u = (u_1, \ldots, u_n)^T$. In other words, the Newton iteration is simply a special fixed-point iteration:

$$u^{n+1} = G(u^n) = u^n - F'(u^n)^{-1}F(u^n). \tag{13}$$

There are several variations of the standard (or *full*) Newton iteration (11)–(12) commonly used for nonlinear algebraic equations which we mention briefly. A *quasi*-Newton method refers to a method which uses an approximation to the true Jacobian matrix for solving the Newton equations. A *truncated*-Newton method uses the true Jacobian matrix in the Newton iteration, but solves the Jacobian system only approximately, using an iterative linear solver in which the iteration is stopped early or *truncated*. *Inexact-* or *approximate*-Newton methods refers to all of these types of methods collectively, where in the most general case an approximate Newton direction is produced in some unspecified fashion.

The inexact-Newton approach is of interest for the nonlinear Poisson-Boltzmann equation for the following reasons. First, in the case of problems such as the nonlinear Poisson-Boltzmann equation, which consist of a leading linear term plus a nonlinear term which does not depend on derivatives of the solution, the nonlinear algebraic equations generated by discretization have the form:

$$F(u) = Au + N(u) - f = 0.$$

The matrix A is SPD, and the nonlinear term $N(\cdot)$ is often simple, and in fact is often *diagonal*, meaning that the j-th component of the vector function $N(u)$ is a function of only the j-th entry of the vector u, or $N_j(u) = N_j(u_j)$; this occurs for example in the case of a box-method discretization of the Poisson-Boltzmann equation and similar equations. Further, it is often the case that the derivative $N'(\cdot)$ of the nonlinear term $N(\cdot)$, which will be a diagonal matrix due to the fact that $N(\cdot)$ is of diagonal form, can be computed (often analytically) at low expense. If this is the case, then the entire true Jacobian matrix is available at low cost, since :

$$F'(u) = A + N'(u).$$

For the nonlinear Poisson-Boltzmann equation, we have that $N_i(u) = N_i(u_i) = meas(\tau^{(i)})\bar{\kappa}(\mathbf{x}_i)\sinh(u_i)$, so that the contribution to the Jacobian can be computed analytically:

$$N_i'(u) = N_i'(u_i) = meas(\tau^{(i)})\bar{\kappa}(\mathbf{x}_i)\cosh(u_i).$$

A second reason for our interest in the inexact-Newton approach is that the efficient multilevel methods for the linearized Poisson-Boltzmann equation [5, 6, 14] can be used effectively for the Jacobian systems; this is because the Jacobian $F'(u)$ is essentially the linearized Poisson-Boltzmann operator, where only the diagonal Helmholtz-like term $N'(\cdot)$ changes from one Newton iteration to the next. Our fast linear multilevel methods should be effective as inexact Jacobian system solvers, and this has been demonstrated numerically in earlier papers [5, 6, 14] and will be again later in this paper.

The hope is that solving the Jacobian systems only approximately (requiring perhaps a few more Newton iterations due to the inexactness of the Newton direction), using a fast linear multilevel method, will be less costly in terms of execution time than employing a full Newton method (requiring fewer Newton iterations since the direction is exact), and solving the Jacobian systems exactly at each iteration. We will see that this is the case later in the paper, and in fact the inexact approach may be substantially more efficient than the full Newton approach.

However, there are two important considerations when using an inexact Newton method. First, how "inexactly" can one solve the Jacobian system and still converge at a desirably fast rate, and how can one enforce global convergence properties for the overall Newton iteration, so that the method will be robust. We state more precisely, and then answer, both of these questions in the next two sections, and then present the resulting global inexact-Newton-multilevel method in the third section.

Inexactness and superlinear convergence

Let $u \in \mathbb{R}^n$. A sequence $\{u^n\}$ is said to *converge strongly* to u if $\lim_{n\to\infty} \|u - u^n\| = 0$. There are three basic important notions regarding the *rate* of convergence of a sequence of iterates produce by Newton's method, and we state them below as definitions.

Definition 1 *The sequence $\{u^n\}$ converges Q-linearly to u if there exists $c \in [0,1)$ and $\bar{n} \geq 0$ such that for $n \geq \bar{n}$,*

$$\|u - u^{n+1}\| \leq c\|u - u^n\|.$$

Definition 2 *The sequence $\{u^n\}$ converges Q-superlinearly to u if there exists $\{c_n\}$ such that $c_n \to 0$ and:*

$$\|u - u^{n+1}\| \leq c_n\|u - u^n\|.$$

Definition 3 *The sequence $\{u^n\}$ converges at rate Q-order(p) to u if there exists $p > 1$, $c \geq 0$, $\bar{n} \geq 0$ such that for $n \geq \bar{n}$,*

$$\|u - u^{n+1}\| \leq c\|u - u^n\|^p.$$

The following notion of continuity is also necessary.

Definition 4 *The mapping $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ is called Hölder-continuous on D with constant γ and exponent p if there exists $\gamma \geq 0$ and $p \in (0,1]$ such that*

$$\|F(u) - F(v)\| \leq \gamma\|u - v\|^p \quad \forall u, v \in D \subset \mathcal{H}.$$

If $p = 1$, then F is called uniformly Lipschitz-continuous on D, with Lipschitz constant γ. If in addition $\gamma < 1$, then F is called a contraction mapping with contraction constant γ.

If an initial approximation is close enough to the true solution u, then under certain conditions it can be shown that [39] that a full Newton's method will converge, and do so Q-superlinearly. The convergence rate will not be so advantageous if an inexact Newton method is employed. However, it can be shown that the convergence behavior of these inexact-Newton methods is similar to the standard Newton's method, and Newton-Kantorovich-like theorems can be established (see Chapter 18 of Kantorovich and Akilov [39] and below).

In particular, Quasi-Newton methods are studied in Dennis and Moré [40], and a "characterization" theorem is established for the sequence of approximate Jacobian systems. This theorem establishes sufficient conditions on the sequence $\{B_i\}$, where $B_i \approx F'$, to ensure superlinear convergence of a quasi-Newton method. An interesting result which they obtained is that the "consistency" condition is not required, meaning that the sequence $\{B_i\}$ need not converge to the true Jacobian $F'(\cdot)$ at the root of the equation $F(u) = 0$, and superlinear convergence can still be obtained. In a later paper [41], this characterization theorem is rephrased in a geometric form, showing essentially that the full or true Newton step must be approached, asymptotically, in both length and direction, to attain superlinear convergence in a quasi-Newton iteration.

Inexact-Newton methods are studied directly by Dembo et al. [42]. Their motivation is the use of iterative solution methods for approximate solution of the true Jacobian systems. They establish conditions on the accuracy of the inexact Jacobian solves at each Newton iteration which will ensure superlinear convergence. The inexact-Newton method is analyzed in the form:

$$\begin{aligned} F'(u^n)v^n &= -F(u^n) + r^n, \qquad \frac{\|r^n\|}{\|F(u^n)\|} \leq \eta_n, \\ u^{n+1} &= u^n + v^n. \end{aligned}$$

In other words, the quantity r^n, which is simply the residual of the Jacobian linear system, indicates the inexactness allowed in the approximate linear solve, and is exactly what one would monitor in a linear iterative solver. It is established that if the *forcing sequence* $\eta_n < 1$ for all n, then the above method is locally convergent. Their main result is the following theorem.

Theorem 2 *Assume there exists a unique u such that $F(u) = 0$, that the inexact-Newton iterates $\{u^n\}$ converge to u, and that both $F(\cdot)$ and $F'(\cdot)$ are sufficiently smooth. Then:*

1. *The convergence is superlinear if:* $\lim_{n\to\infty} \eta_n = 0$.
2. *The convergence is at least Q-order(1+p) if $F'(u)$ is Hölder continuous with exponent p, and*

$$\eta_n = O(\|F(u^n)\|^p), \text{ as } n \to \infty.$$

Proof. See Dembo et al. [42]. □

As a result of this theorem, they suggest the tolerance rule:

$$\eta_n = \min\left\{\frac{1}{2}, C\|F(u^n)\|^p\right\}, \quad 0 < p \leq 1, \tag{14}$$

which guarantees Q-order convergence of at least $1 + p$.

Inexactness and global convergence

As noted in the previous section, Newton-like methods converge if the initial approximation is "close" to the solution; different convergence theorems require different notions of closeness. If the initial approximation is close enough to the solution, then superlinear or Q-order(p) convergence occurs. However, the fact that these theorems require a good initial approximation is also indicated in practice: it is well known that Newton's method will converge slowly or fail to converge at all if the initial approximation is not good enough.

On the other hand, methods such as those used for unconstrained minimization can be considered to be "globally" convergent methods, although their convergence rates are often extremely poor. One approach to improving the robustness of a Newton iteration without loosing the favorable convergence properties close to the solution is to combine the iteration with a global minimization method. In other words, we can attempt to force global convergence of Newton's method by requiring that:

$$\|F(u^{n+1})\| < \|F(u^n)\|,$$

meaning that we require a decrease in the value of the function at each iteration. But this is exactly what global minimization methods, such as the nonlinear conjugate gradient method, attempt to achieve: progress toward the solution at each step.

More formally, we wish to define a minimization problem, such that the solution of the zero-point problem we are interested in also solves the associated minimization problem. Let us define the following two problems:

P1: Find $u \in \mathbb{R}^n$ such that $F(u) = 0$.
P2: Find $u \in \mathbb{R}^n$ such that $J(u) = \min_{v\in\mathbb{R}^n} J(v)$,

where $J(\cdot)$ is a *functional*, mapping $\mathbb{R}^n$ into $\mathbb{R}$. We assume that Problem 2 has been defined so that the unique solution to Problem 1 is also the unique solution to Problem 2; note that in general, there may not exist a *natural* functional $J(\cdot)$ for a given $F(\cdot)$, although we will see in a moment that it is always possible to construct an appropriate functional $J(\cdot)$.

A *descent direction* for the functional $J(\cdot)$ at the point u is any direction v such that the directional derivative of $J(\cdot)$ at u in the direction v is negative, or $(J'(u), v) < 0$, where $(\cdot,\cdot)$ is an inner-product in $\mathbb{R}^n$, and $J'(\cdot)$ is the derivative of the functional $J(\cdot)$:

$$J'(u) = \nabla J(u)^T = \left(\frac{J(u)}{\partial u_1}, \ldots, \frac{J(u)}{\partial u_n}\right)^T.$$

If v is a descent direction, then it is not difficult to show (Theorem 8.2.1 in Ortega and Rheinboldt [31]) there exists $\lambda > 0$ such that:

$$J(u + \lambda v) < J(u). \tag{15}$$

This follows from a generalized Taylor expansion (cf. page 255 in Kesavan [43]), since

$$J(u + \lambda v) = J(u) + \lambda(J'(u), v) + O(\lambda^2).$$

If λ is sufficiently small and $(J'(u), v) < 0$ holds (v is a descent direction), then clearly $J(u + \lambda v) < J(u)$. In other words, if a descent direction can be found at the current solution u^n, then an improved solution u^{n+1} can be found for some steplength in the descent direction v; i.e., by performing a one-dimensional line search for λ until (15) is satisfied.

Therefore, if we can show that the Newton direction is a descent direction, then performing a one-dimensional line search in the Newton direction will always guarantee progress toward the solution. In the case that we define the functional as:

$$J(u) = \frac{1}{2}\|F(u)\|^2 = \frac{1}{2}(F(u), F(u)),$$

we can show that the Newton direction is a descent direction. While the following result is easy to show for $\mathbb{R}^n$, it is also true in the general case of a Hilbert space [5, 6] when $\|\cdot\| = (\cdot,\cdot)^{1/2}$:

$$J'(u) = F'(u)^T F(u).$$

The Newton direction at u is simply $v = -F'(u)^{-1}F(u)$, so if $F(u) \neq 0$, then:

$$\begin{aligned}(J'(u), v) &= -(F'(u)^T F(u), F'(u)^{-1}F(u)) \\ &= -(F(u), F(u)) < 0.\end{aligned}$$

Therefore, the Newton direction is always a descent direction for this particular choice of $J(\cdot)$, and by the introduction of the damping parameter λ, the Newton iteration can be made globally convergent in the above sense.

Consider now the *inexact* Newton method; since only the *exact* Newton direction is known to be a descent direction, we have no assurance that the inexact direction will give descent, so the global properties gained by a damping parameter are lost. We can still attempt to introduce the damping parameter λ as before, so that the resulting algorithm for solving $F(u) = 0$ is:

Algorithm 6 *(Damped-Inexact-Newton Method)*

$$\begin{aligned} F'(u^n)v^n &= -F(u^n) + r^n, \qquad \frac{\|r^n\|}{\|F(u^n)\|} \leq \eta_n, \\ u^{n+1} &= u^n + \lambda_n v^n, \end{aligned}$$

where we have left as unspecified how "large" the residual r^n is allowed to be, and how the damping parameters λ_n are chosen.

The following theorem from references [5, 6] gives a necessary and sufficient condition on the residual r^n of the Jacobian system system for the resulting inexact Newton direction to be a descent direction. This will allow us to use the damping parameter to achieve global convergence properties in the inexact-Newton algorithm.

Theorem 3 *Inexact-Newton (Algorithm 6) yields a descent direction v at u if and only if the residual of the Jacobian system, $r = F'(u)v + F(u)$, satisfies:*

$$(F(u), r) < (F(u), F(u)).$$

Proof. We remarked earlier that an equivalent minimization problem (appropriate for Newton's method) to associate with the zero point problem $F(u) = 0$ is given by $\min_{u\in\mathbb{R}^n} J(u)$, where $J(u) = (F(u), F(u))/2$. We also noted that the derivative of $J(u)$ can be written as $J'(u) = F'(u)^T F(u)$. Now, the direction v is a descent direction for $J(u)$ if and only if $(J'(u), v) < 0$. The exact Newton direction is $v = -F'(u)^{-1}F(u)$, and as shown earlier is always a descent direction. Consider now the inexact direction satisfying:

$$F'(u)v = -F(u) + r, \qquad \text{or} \qquad v = F'(u)^{-1}[r - F(u)].$$

This inexact direction is a descent direction if and only if:

$$\begin{aligned} (J'(u), v) &= (F'(u)^T F(u), F'(u)^{-1}[r - F(u)]) \\ &= (F(u), r - F(u)) \\ &= (F(u), r) - (F(u), F(u)) \\ &< 0, \end{aligned}$$

which is true if and only if the residual of the Jacobian system r satisfies:

$$(F(u), r) < (F(u), F(u)).$$

□

This leads to the following simple sufficient condition for descent.

Corollary 4 *Inexact Newton (Algorithm 6) yields a descent direction v at the point u if the residual of the Jacobian system, $r = F'(u)v + F(u)$, satisfies:*

$$\|r\| < \|F(u)\|.$$

Proof. From the proof of Theorem 3 we have:

$$\begin{aligned} (J'(u), v) &= (F(u), r) - (F(u), F(u)) \\ &\leq \|F(u)\|\|r\| - \|F(u)\|^2, \end{aligned}$$

where we have employed the Cauchy-Schwarz inequality. Therefore, if $\|r\| < \|F(u)\|$, then the rightmost term is clearly negative (unless $F(u) = 0$), so that v is a descent direction. □

The sufficient condition presented as Corollary 4 appears in references [5, 6], and also as a lemma in [44]. Note that most stopping criteria for the Newton iteration involve evaluating $F(\cdot)$ at the previous Newton iterate u^n. The quantity $F(u^n)$ will have been computed during the computation of the previous Newton iterate u^n, and the tolerance for u^{n+1} which guarantees descent requires $(F(u^n), r) < (F(u^n), F(u^n))$ by Theorem 3. This involves only the inner-product of r and $F(u^n)$, so that enforcing this tolerance requires only an additional inner-product during the Jacobian linear system solve, which for n unknowns introduces an additional n multiplies and n additions. In fact, a scheme may be employed in which only a residual tolerance requirement for superlinear convergence is checked until an iteration is reached in which it is satisfied. At this point, the descent direction tolerance requirement can be checked, and additional iterations will proceed with this descent stopping criterion until it too is satisfied. If the linear solver reduces the norm of the residual monotonically (such as any of the linear methods discussed earlier), then the first stopping criterion need not be checked again.

In other words, this adaptive Jacobian system stopping criterion, enforcing a tolerance on the residual for local superlinear convergence *and* ensuring a descent direction at each Newton iteration, can be implemented at the same computational cost as a simple check on the norm of the residual of the Jacobian system.

Alternatively, the sufficient condition given in Corollary 4 may be employed at no additional cost, since only the residual norm need be computed, which is also required to insure superlinear convergence using Theorem 2.

Inexact-Newton-MG for the PBE

Discretization of the nonlinear Poisson-Boltzmann equation (1) with the box-method discussed earlier produces

a set of n nonlinear algebraic equations in n unknowns of the form (3), which we repeat here:

$$F(u) = Au + N(u) - f = 0.$$

The "holy grail" for this problem is an algorithm which (1) always converges, and (2) has optimal complexity, which in this case means $O(n)$.

As we have just seen, the inexact-Newton method can be made essentially globally convergent with the introduction of a damping parameter. In addition, close to the root, inexact-Newton has at least superlinear convergence properties thanks to Theorem 2. If a method with linear convergence properties is used to solve the Jacobian systems at each Newton iteration, and the complexity of the linear solver is the dominant cost of each Newton iteration, then the complexity properties of the linear method will determine the complexity of the resulting Newton iteration asymptotically. With an efficient inexact solver such as a multilevel method for the early damped iterations, employing a more stringent tolerance for the later iterations as the root is approached, a very efficient yet robust nonlinear iteration should result; in fact, if the linear method behaves as $O(n)$, then a superlinearly-convergent nonlinear iteration should as well.

The idea here, motivated by the work of Bank and Rose [45, 46], is to combine the robust damped inexact-Newton methods with the fast linear multilevel solvers developed by Holst and Saied [14] and Holst [5, 6] for the inexact Jacobian system solves. Combination with linear multilevel iterative methods for the semiconductor problem has been considered by Bank and Rose [46], along with questions of complexity. In a paper of Bank and Rose [45], an analysis of inexact-Newton methods is performed, where a damping parameter has been introduced. A quite sophisticated algorithm GLOBAL is constructed, enforcing both global and superlinear convergence properties; the sufficient descent condition established above is implicitly imbedded in their algorithm GLOBAL.

We propose the following alternative globally convergent inexact-Newton algorithm which is easy to understand and implement, based on the simple necessary and sufficient descent conditions established in the previous section.

Algorithm 7 *(Damped-Inexact-Newton method)*

(1) Inexact solve: $F'(u^n)v^n = -F(u^n) + r^n$,
$TST(r^n) = TRUE$,
(2) Correction: $u^{n+1} = u^n + \lambda_n v^n$,

where λ_n *and* $TST(r^n)$ *are defined as:*

$TST(r^n)$: *IF:* $\|r^n\| \le C\|F(u^n)\|^{p+1},\ C, p > 0$,
(local Q-order(1+p) convergence)
AND: $(F(u^n), r^n) < (F(u^n), F(u^n))$
(guaranteed descent)
THEN: $TST \equiv TRUE$
ELSE: $TST \equiv FALSE$.

λ_n: $\|F(u^n + \lambda_n v^n)\| \le \|F(u^n)\|$ *by line search; Always possible if* $TST(r^n) = TRUE$. *The full inexact-Newton step* ($\lambda_n = 1$) *is always tried first.*

An alternative less expensive $TST(r^n)$ *is as follows:*

$TST(r^n)$: *IF:* $\|r^n\| \le C\|F(u^n)\|^{p+1},\ C, p > 0$,
(local Q-order(1+p) convergence)
AND: $\|r^n\| < \|F(u^n)\|$
(guaranteed descent)
THEN: $TST \equiv TRUE$
ELSE: $TST \equiv FALSE$.

In Algorithm 7, the second condition in the first $TST(\cdot)$ procedure is the necessary and sufficient condition for the inexact-Newton direction to be a descent direction, established in Theorem 3. The second condition in the alternate $TST(\cdot)$ procedure is the weaker sufficient condition established in Corollary 4. Note that, in early iterations when Q-order(1+p) for $p > 0$ is not to be expected, just satisfying one of the descent conditions is (necessary and) sufficient for progress toward the solution. Algorithm 7 decouples the descent and superlinear convergence conditions, and would allow for the use of only the weakest possible test of $(F(u^n), r^n) < (F(u^n), F(u^n))$ far from the solution, ensuring progress toward the solution with the least amount of work per Newton step.

Note also that the Q-order(1+p) condition

$$\|r^n\| \le C\|F(u^n)\|^{p+1}$$

does *not* guarantee a descent direction, so that it is indeed important to satisfy the descent condition separately. The Q-order(1+p) condition *will* impose descent if

$$C\|F(u^n)\|^{p+1} < \|F(u^n)\|,$$

which does not always hold. If one is close to the solution, so that $\|F(u^n)\| < 1$, and if $C \le 1$, then the Q-order(1+p) condition will imply descent. By this last comment, we see that if $\|F(u^n)\| < 1$ and $C \le 1$, then the full inexact-Newton step is a descent direction, and since we attempt this step first, we see that our algorithm reduces to the algorithm studied by Dembo et al. [42] near the solution; therefore, Theorem 2 above applies to Algorithm 7 near the solution without modification.

Note that due to the special form of the nonlinear operator arising in the discrete nonlinear Poisson-Boltzmann equation, the damping step can be implemented in a surprisingly efficient manner. During the

one-dimensional line search for the parameter λ_n, we continually check for satisfaction of the inequality:

$$\|F(u^n + \lambda_n v^n)\| < \|F(u^n)\|.$$

The term on the right is available from the previous Newton iteration. The term on the left, although it might appear to involve computing the full nonlinear residual, in fact can avoid the operator-vector product contributed by the linear term. Simply note that

$$F(u^n + \lambda_n v^n) = A[u^n + \lambda_n v^n] + N(u^n + \lambda_n v^n) - f$$
$$= [Au^n - f] + \lambda_n[Av^n] + N(u^n + \lambda_n v^n).$$

The term $[Au^n - f]$ is available from the previous Newton iteration, and $[Av^n]$ need be computed only once at each Newton step. Computing $F(u^n + \lambda_n v^n)$ for each damping step beyond the first requires only the "saxpy" operation $[Au^n - f] + \lambda_n[Av^n]$ for the new damping parameter λ_n, and evaluation of the nonlinear term at the new damped solution, $N(u^n + \lambda_n v^n)$.

For the numerical comparisons with other methods, we employ Algorithm 7, taking $p = 1$ and $C = 1.0 \times 10^{-2}$ in the procedure $TST(\cdot)$, using the less expensive, sufficient descent condition form of the $TST(\cdot)$ procedure. The Jacobian system is solved inexactly at each step to the residual tolerance specified by $TST(\cdot)$ by employing the linear multilevel we designed for the linearized Poisson-Boltzmann equation [5, 6, 14]. The damping parameters λ_n are selected by a standard line search technique. The result is an extremely robust and efficient numerical method for the nonlinear Poisson-Boltzmann equation, as we will see shortly.

SOME TEST PROBLEMS

We describe briefly the nonlinear Poisson-Boltzmann equation test problems which we use to numerically evaluate and compare the methods which have been proposed for the nonlinear Poisson-Boltzmann equation. We also describe a test problem which has a rapid nonlinearity and very large jump discontinuities in the coefficients, which will be used to evaluate some of the multilevel techniques.

The nonlinear PBE

Consider a very broad range of possible temperatures $T \in [200K, 400K]$, a broad range of possible ionic strengths $I_s \in [0, 10]$, and the following representative polygonal domain:

$$\Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}],$$

where the diameter of Ω is on the order of 10 $\mathring{\text{A}}$ to 500 $\mathring{\text{A}}$. We assume that the set of discrete charges $\{\mathbf{x}_1, \ldots, \mathbf{x}_{N_m}\}$ representing the molecule lie well within the domain, and hence far from the boundary Γ of Ω. The nonlinear Poisson-Boltzmann equation for the dimensionless potential $u(\mathbf{x})$ then has the form:

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

$$u(\mathbf{x}) = \frac{C}{4\pi\epsilon_w} \sum_{i=1}^{N_m} [z_i e^{-\bar{\kappa}(\mathbf{x})|\mathbf{x}-\mathbf{x}_i|/\sqrt{\epsilon_w}}]/|\mathbf{x} - \mathbf{x}_i|, \quad \text{on } \Gamma,$$

where $C = 4\pi e_c^2 / k_B^{-1} T^{-1}$, and $\epsilon_w = 80$. We have employed one of the known analytical solutions for the linearized problem to obtain the boundary condition on Γ appearing above. As remarked earlier, this is commonly done; see for example Tanford [4] or in references [5, 6] for more detailed discussions of analytical solutions.

It is easy to show [5, 6] that the problem coefficients satisfy the following bounds for the given temperature and ionic strength ranges:

1. $2 \le \epsilon(\mathbf{x}) \le 80$.
2. $0 \le \bar{\kappa}^2(\mathbf{x}) \le 127.0$.
3. $5249.0 \le C \le 10500.0$.
4. $-1 \le z_i \le 1$.

The nonlinear Poisson-Boltzmann problem will then be completely defined by specifying the following quantities:

- $\mathbf{x}_{\min}$, $\mathbf{x}_{\max}$, $\mathbf{y}_{\min}$, $\mathbf{y}_{\max}$, $\mathbf{z}_{\min}$, $\mathbf{z}_{\max}$; the domain.
- $\epsilon(\mathbf{x})$; the electrostatic molecular surface.
- $\bar{\kappa}(\mathbf{x})$; defined by ionic strength and exclusion layer.
- C; constant depending only on the temperature T.
- $\{\mathbf{x}_1, \ldots, \mathbf{x}_{N_m}\}$; the charge locations.
- $\{z_1, \ldots, z_{N_m}\}$; the associated fractional charges.

For all of our molecule test problems, we use $T = 298$ which determines the constant C; this is a common parameter setting for these types of problems. The domain geometry will be defined by the particular molecule, as well as the parameters $\epsilon(\mathbf{x})$ and $\bar{\kappa}(\mathbf{x})$, although we must specify also the ionic strength I_s to completely determine $\bar{\kappa}(\mathbf{x})$. The charge locations and corresponding fractional charges will also be determined by the particular molecule.

The test data is taken from the Brookhaven protein databank, with the help of the DELPHI and UHBD software packages (described below). The test molecules chosen for our study of the nonlinear Poisson-Boltzmann equation are the following:

- Acetamide (CH_3CONH_2) at 1.0 molar, a small molecule (few angstroms in diameter).
- Crambin at 0.001 molar, a medium size molecule.
- tRNA at 0.2 molar, a large highly charged molecule creating numerical difficulties.
- SOD at 0.1 molar, a large enzyme currently undergoing intensive study in the biophysics community.

Brookhaven data and existing software

We have connected the software implementations of our methods to both the DELPHI and UHBD electrostatics programs, and we will use data provided by these packages. The DELPHI package was developed in the laboratory of Dr. B. Honig at Columbia University, and the UHBD package was developed in the laboratory of Dr. J. A. McCammon at the University of Houston. These codes are designed to begin with a protein data bank (pdb) file description of the protein or enzyme in question, obtained from the protein data bank at Brookhaven National Laboratory. The pdb files contain the coordinates of all of the atoms in a particular structure, obtained from X-ray crystallography pictures of the structure. The UHBD and DELPHI programs begin with the atom coordinates, and then construct both the electrostatic surface and the exclusion layer by moving a probe around the molecule which has the radius of a representative ion. We remark that quite sophisticated algorithms are now being employed for surfacing [32].

Both UHBD and DELPHI are designed around Cartesian meshes (both implementations are actually restricted to uniform Cartesian meshes), and the electrostatic surface and exclusion layer information are represented as three-dimensional discrete grid functions $\epsilon_h(\mathbf{x})$ and $\bar{\kappa}_h(\mathbf{x})$. The mesh function $\bar{\kappa}_h(\mathbf{x})$ is produced at the same mesh-points where the unknowns $u_h(\mathbf{x})$ are located, whereas the mesh function $\epsilon_h(\mathbf{x})$ is produced at half-mesh-points in each coordinate direction as needed for the box-method discretization we described earlier in the paper (also employed in both UHBD and DELPHI). The atoms themselves, which will most likely not lie on a Cartesian mesh, must be mapped to the Cartesian coordinates, and their corresponding charges distributed to the neighboring mesh points. Several approaches are possible; a trilinear interpolation approach is taken in both packages.

Note that the selection of the domain completely determines the boundary conditions for a given problem, as we have specified the boundary function $g(\mathbf{x})$ above. Several different approaches have been proposed to approximate $g(\mathbf{x})$, since it is clear that to evaluate $g(\mathbf{x})$ at each boundary point of the three-dimensional domain will require all pair-wise interactions of the charges and

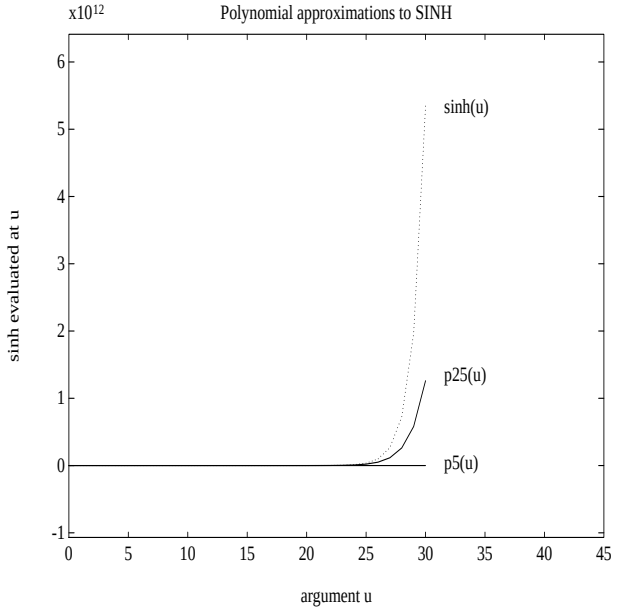

Figure 4: Accuracy of polynomial approx. to sinh.

the boundary points; efficient versions are offered as options for example in UHBD, all of which appear to give similarly good approximations of the true boundary condition $u(\infty) = 0$ (when the molecule is taken to lie well within the domain Ω). In both UHBD and DELPHI, the problem domain Ω is typically constructed around the selected molecule so that no more than thirty percent of Ω in each coordinate direction is taken up by the molecule, which is centered in the domain. The linearized analytical solution used for the boundary condition function $g(\mathbf{x})$ above, and employed in both DELPHI and UHBD, appears to give very good approximation of the true boundary conditions in most situations.

Polynomial nonlinear forms of the PBE

It has been common in the literature to use low-degree polynomial approximations to the hyperbolic sine function, avoiding the difficulties which occur with the exponential terms in the true sinh function. For example, in the paper of Jayaram et al. [47], three term polynomials are used. However, Figure 4 illustrates how poor such approximations are in situations (which frequently occur) when the argument becomes on the order of 10 or more. Note that the units on the vertical axis are 1×10^{12}. In the figure, the true hyperbolic function is plotted with the dotted line; polynomial approximations of degree five and twenty-five are plotted with the solid lines. It seems clear that the full exponential terms must be included in the nonlinear equation in these situ-

ations, which occur even in the case of lysozyme [32]. In some sense it is a mute point, since our global inexact-Newton-multilevel methods control the numerical problems of the exponential nonlinearity well, and for implementation reasons (the intrinsic exponential functions are much faster than a loop which evaluates a polynomial) the polynomial nonlinearity solution actually takes longer to compute numerically with our methods (and other methods, when they converge for the exponential case) than the full exponential case. Therefore, we will consider only the more correct exponential model.

A nonlinear jump discontinuity problem

The following test problem will be used to explore the convergence behavior of the multilevel methods. The domain is the unit cube:

$$\Omega = [0,1] \times [0,1] \times [0,1].$$

The nonlinear equation has the form:

$$-\nabla \cdot (\bar{\mathbf{a}}(\mathbf{x}) \nabla u(\mathbf{x})) + b(\mathbf{x}, u(\mathbf{x})) = f(\mathbf{x}) \text{ in } \Omega \subset \mathbb{R}^3,$$

$$u(\mathbf{x}) = g(\mathbf{x}) \text{ on } \Gamma. \tag{16}$$

where the coefficients in equation (16) are taken to be:

1. $\bar{\mathbf{a}} : \Omega \mapsto \mathbf{L}(\mathbb{R}^3),\ a_{ij}(\mathbf{x}) = \delta_{ij}\epsilon(\mathbf{x}),\ 1 \le \epsilon(\mathbf{x}) \le 10^3.$
2. $b : \Omega \times \mathbb{R} \mapsto \mathbb{R},\ b(\mathbf{x}, u(\mathbf{x})) = \lambda e^{u(\mathbf{x})},\ \lambda \ge 0.$
3. $f : \Omega \mapsto \mathbb{R},\ -1 \le f(\mathbf{x}) \le 1.$
4. $g : \Gamma \mapsto \mathbb{R},\ g(\mathbf{x}) = 0.$

We will construct $\epsilon(\mathbf{x})$ to be piecewise constant, taking one value in a subdomain $\Omega_1 \subset \Omega$, and a second value in the region $\Omega \backslash \Omega_1$, so that $\epsilon(\mathbf{x})$ is defined as follows:

$$\epsilon(\mathbf{x}) = \left\{ \begin{array}{l} 1 \le \epsilon_1 \le 1.0 \times 10^3 \text{ if } \mathbf{x} \in \Omega_1, \\ 1 \le \epsilon_2 \le 1.0 \times 10^3 \text{ if } \mathbf{x} \in \Omega \backslash \Omega_1. \end{array} \right\}$$

We will take ϵ_1 and ϵ_2 to be quite different in magnitude, so that their ratio:

$$D = \frac{\epsilon_1}{\epsilon_2}$$

will be 10^3 or 10^{-3}, similar to the nonlinear Poisson-Boltzmann equation. (Additional experiments were performed by Holst [5, 6], taking D to be be as large as 10^8 or as small as 10^{-8}, and the resulting convergence behavior of the various methods was analyzed in detail.) We define the subdomain $\Omega_1 \subset \Omega$ to consist of the following two smaller cubes:

$$\Omega_1 = [0.25, 0.50] \times [0.25, 0.50] \times [0.25, 0.50]$$

$$\cup\ [0.50, 0.75] \times [0.50, 0.50] \times [0.50, 0.75].$$

Table 1: Nonlinear PBE methods.

Method	Description
DINMH	damped-inexact-Newton-MH
DFNMH	damped-full-Newton-MH
NMH	nonlinear MH
NCG	nonlinear CG (Fletcher-Reeves)
NSOR	nonlinear SOR (1-D Newton)
NGS	nonlinear Gauss-Seidel (1-D Newton)

For this simple problem, it would of course be possible to construct all coarse meshes as needed for the multilevel methods to align with Ω_1; this would not possible with problems such as the nonlinear Poisson-Boltzmann equation and a complex molecule. Therefore, since we wish to simulate the case that the discontinuities in $\epsilon(\mathbf{x})$ cannot be resolved on coarser meshes, the multiple levels of tessellations of Ω into discrete meshes Ω_k are constructed so that the discontinuities in $\epsilon(\mathbf{x})$ lie along mesh lines *only on the finest mesh*. In addition, we employ a non-uniform Cartesian mesh, as shown in Figure 1, which attempts to provide a more accurate description of the discontinuity interface on the finest mesh.

Note that if $\epsilon_1 = \epsilon_2 \equiv 1$, then problem (16) with the above coefficients is the Bratu problem (see page 432 in Davis [48] for information about this interesting problem) on the unit cube.

NUMERICAL COMPARISONS

The global inexact-Newton-multilevel method presented earlier is investigated numerically when applied to the nonlinear Poisson-Boltzmann equation and to a nonlinear test problem with large jump discontinuities in the coefficients and exponential nonlinearity. A detailed comparison to other methods is presented, including comparisons to the classical nonlinear multigrid method, the nonlinear conjugate gradient method, and nonlinear relaxation methods such as SOR. Our results indicate that the two multilevel-based methods are superior to the relaxation and conjugate gradient methods, and that the advantage of the multilevel-based methods grows with the problem size. In addition, experiments indicate that the inexact Newton-multilevel approach is the most efficient and robust method for the test problems, and in particular is both more efficient and more robust than the nonlinear multigrid method.

Table 1 provides a key to the plots and tables to follow. For reference, at times we also will refer to the linear methods in Table 2. Unless otherwise indicated, all data in the plots and tables to follow *include* the preprocessing costs incurred by the various methods. In

Table 2: Some linearized PBE methods.

Method	Description
MH	harmonically averaged MG [14]
MICCG	modified incomplete Cholesky CG [14]
DSCG	diagonally preconditioned CG [14]

other words, the multilevel methods times include the additional time required to set up the problem on coarse grids. This gives a complete and fair assessment of the total time required to reach the solution.

An initial approximation of zero was taken to start each method, and each method used a stopping criteria based on the norm of the nonlinear function:

$$\|F(u^i)\| < \text{ TOL } = 1.0e-9,$$

where u^i represents the i^{th} iterate, and $F(\cdot)$ is the discrete nonlinear algebraic operator for the equation $F(u) = 0$ which we are trying to solve. Of course, this is not the most appropriate stopping criteria for nonlinear iterations (more appropriate stopping tests are discussed in detail in references [5, 6]), but for our test problems this test does indicate well when the solution is approached, and it is the best approach for comparing different methods since it guarantees that each method is producing a solution of the same quality.

We remark that it was required to perform all computations in double precision; this is necessitated by the rapid nonlinearities present in the equations, which result an extreme loss in precision. Note that calculations in double precision are more costly than single precision calculations, and so the execution times reported here for some of the methods will be somewhat longer than some of the single precision times reported in earlier papers [5, 6, 14]

Timing figures on the Convex C240 and the Convex C3 were obtained from the system timing routine `getrusage`. A more detailed performance analysis on several more sequential as well as some parallel machines can be found in references [5, 6].

Results for acetamide

Figure 6 compares the methods in Table 1 for the acetamide problem. For this problem, all of the methods converge, and the two multilevel-based algorithms are superior. The nonlinear conjugate gradient and nonlinear SOR methods have comparable performance. The method DINMH is extremely efficient, representing an improvement of more than a factor of fifty over the nonlinear SOR and nonlinear conjugate gradient methods, and a factor of ten over the nonlinear multigrid method NMH.

Results for crambin

Figure 5 compares the methods in Table 1 for the crambin problem. Again, all of the methods converge, and the two multilevel-based algorithms are superior. The nonlinear conjugate gradient shows superiority to the nonlinear relaxation methods. The method DINMH is again extremely efficient, representing an improvement of more than a factor of fifty over the nonlinear conjugate gradient method, and a factor of ten over the nonlinear multigrid method NMH.

Results for tRNA

We included this test problem because it appears to cause severe difficulties for other methods which have been tried; in fact, the nonlinear SOR procedure proposed by Nicholls and Honig [13] was known to diverge for this problem [32]. However, we note that their method was not a true SOR-Newton iteration, and was instead a fixed-point iteration based on a certain splitting of the operator (see page 443 in Nicholls and Honig [13]). When a true SOR-Newton iteration is employed, the method converges for this problem. Figure 7 shows the relative performance of the various methods. Again, the method DINMH is the most efficient by far of the methods presented, representing a factor of fifty improvement over the next best method.

Note that for this problem, the nonlinear multilevel method diverges, even with linesearch for a damping parameter. Since we do not enforce the nonlinear variational conditions exactly, as outlined earlier and in more detail in references [5, 6], we have no guarantee that the coarse level correction is a descent direction, and so this method is not a global method; this particular test problem illustrates this fact. It also shows that nonlinear multigrid method NMH is not only less robust than DINMH, but also less reliable than NSOR and NCG.

Results for SOD

Figure 8 shows only two methods applied to the superoxide dismutase (SOD) test problem: the method DINMH applied to the full nonlinear Poisson-Boltzmann equation; and the linear DSCG method applied to the linearized Poisson-Boltzmann equation. All other nonlinear methods studied here diverged for this test problem. Again, the method DINMH converges very rapidly, and the superlinear convergence is clearly visible.

We have included the plot of the linear method DSCG to show clearly that the DINMH method, solving the full nonlinear problem, is more than a factor of two times more efficient than one of the best available methods in the literature for only the linearized problem.

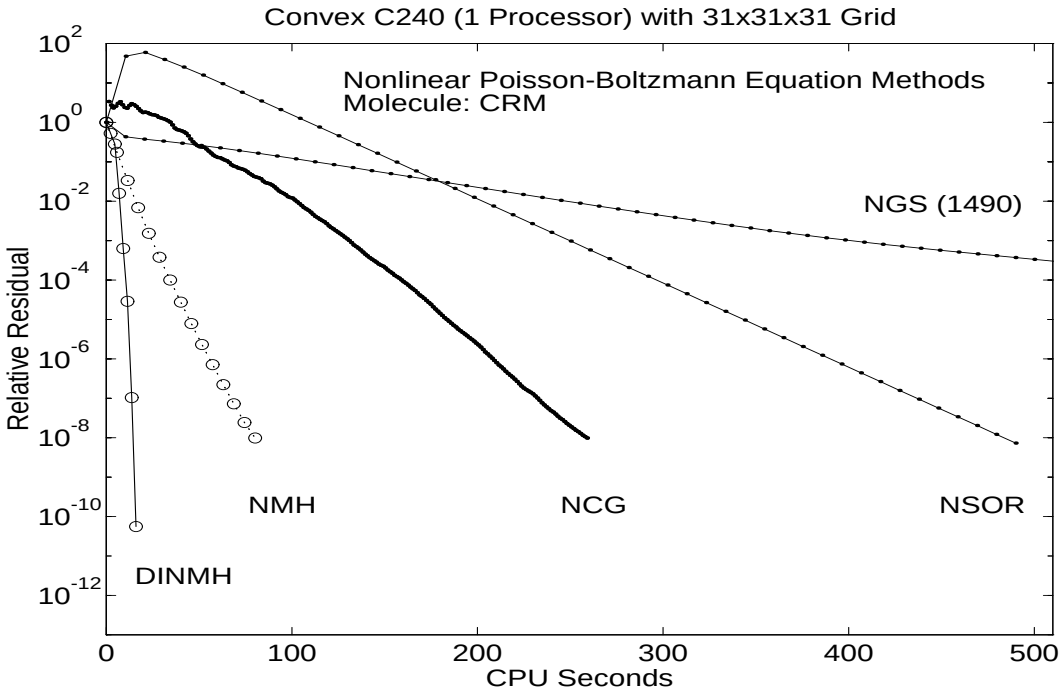

Figure 5: Comparison of various methods for the nonlinear crambin problem.

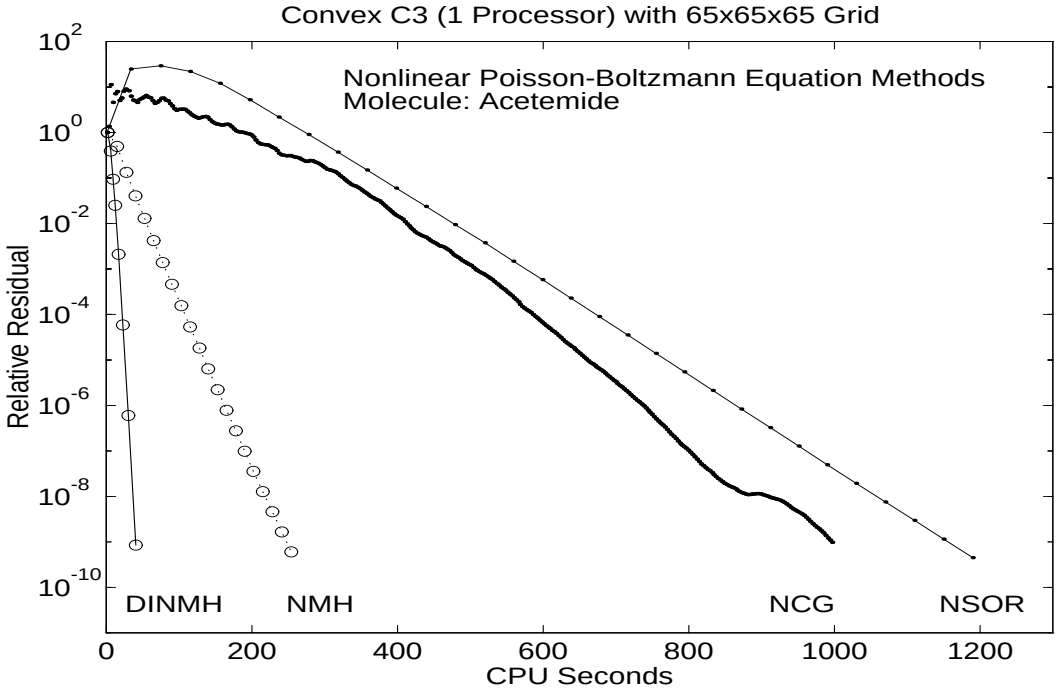

Figure 6: Comparison of various methods for the nonlinear acetamide problem.

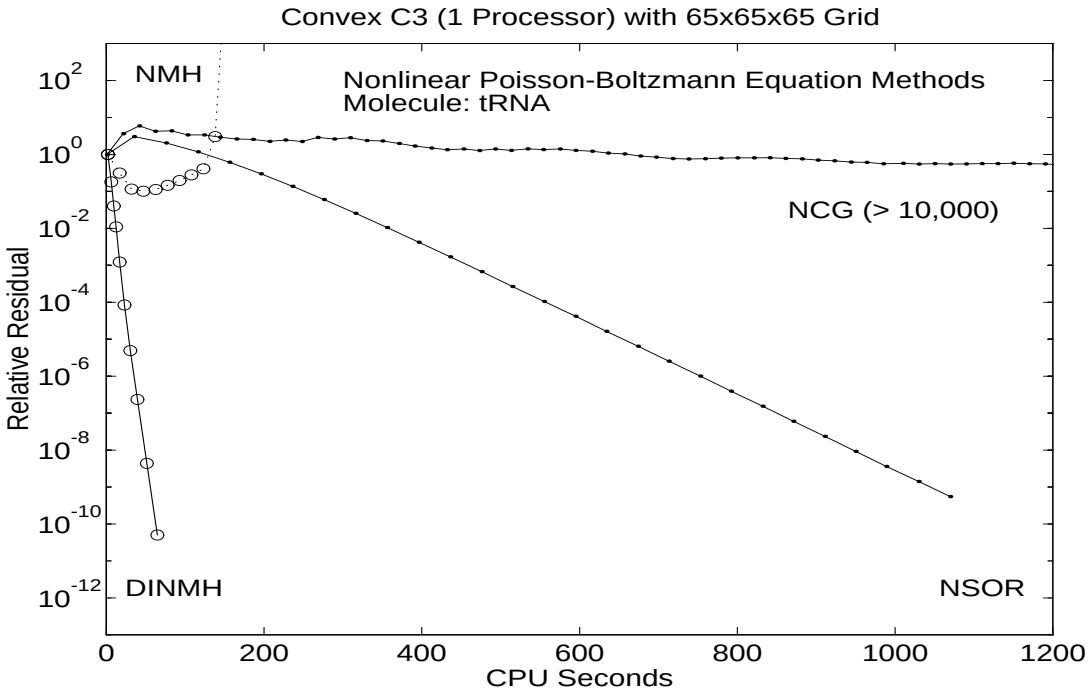

Figure 7: Comparison of various methods for the nonlinear tRNA problem.

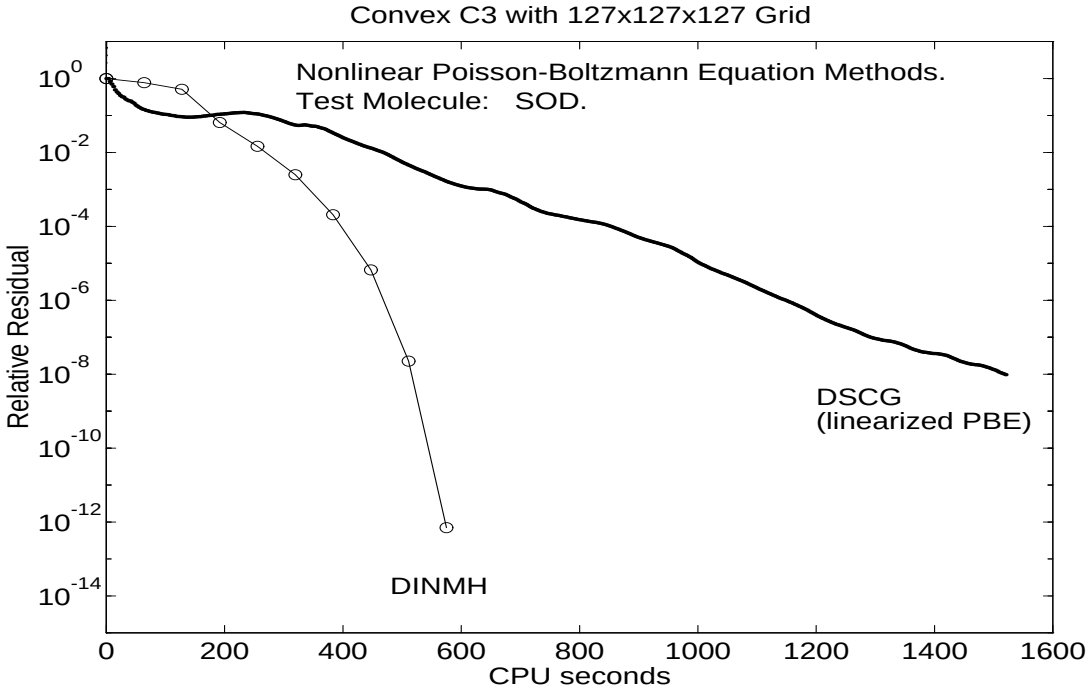

Figure 8: Comparison of various methods for the nonlinear SOD problem.

Jump discontinuity problem results

Figure 9 shows the behavior of the five methods in Table 1 when applied to the jump discontinuity dest problem, with $D = \epsilon_1/\epsilon_2 = 10^{-3}$. The three multilevel-based methods are substantially superior to the nonlinear relaxation and conjugate gradient methods. More interestingly, the comparison between the full Newton method (DFNMH) and the inexact Newton method (DINMH) shows at least a factor of four improvement gained by employing the inexactness strategy outlined earlier (and can be found in more detail in references [5, 6]).

Figure 10 shows the first 200 CPU seconds of Figure 9 expanded to the whole axis. We have included the linear methods MH, MICCG, and DSCG on the plot to illustrate more graphically how efficient the method DINMH is; it requires less than a factor of two times more CPU seconds than the linear method MH for the linearized problem, and is a factor of two times more efficient than the next best *linear* method, MICCG with vectorizable orderings (operating at near peak efficiency on the Convex C3, which is a vector processor).

Storage requirements

We make a few remarks about the storage required for the multilevel methods as well as some of the other methods appearing in this paper. We are faced with the discrete problem of the form:

$$Au + N(u) = f,$$

where A is an $n \times n$ SPD matrix, $N(\cdot)$ is a nonlinear function mapping $\mathbb{R}^n$ into $\mathbb{R}^n$, u is the $n \times 1$ vector of unknowns, and f is the $n \times 1$ vector of source function values. The number of unknowns n is related to the original discrete mesh as $n = I \cdot J \cdot K$, where I, J, and K are the number of mesh-points in each direction of the Cartesian mesh. Employing the box-method on the Cartesian mesh, the matrix A can be represented by seven diagonals, only four of which need be stored in arrays of length n, due to the symmetry of A. The box-method produces "diagonal" nonlinear functions $N(\cdot)$ from the types of nonlinear partial differential equations we consider in this paper, and $N(\cdot)$ can be represented by a single real nonlinear function and a coefficient array of length n. Therefore, simply to store the nonlinear algebraic problem on the finest desired (possibly non-uniform) Cartesian mesh requires approximately $4n+1n+1n+1n = 7n$. The nonlinear iterative algorithms we have considered here require various amounts of additional storage for implementation.

With regard to multilevel methods, since the number of unknowns drops by a factor of eight as one moves to a coarser mesh in three dimensions if standard successively refined non-uniform Cartesian meshes are used, we see that the storage required to represent on all meshes a vector having length n on the finest mesh is:

$$n + \frac{n}{8} + \frac{n}{64} + \cdots = n \cdot \left(\frac{1}{8} + \frac{1}{64} + \cdots \right) \leq \frac{8}{7} \cdot n.$$

We will assume that enough levels are always used so that not only is the coarse problem computational cost negligible, but also the storage requirement (including possibly direct factorization of the matrix) is negligible due to the size of the coarse problem.

Table 3 gives the required storage for a selection of methods. These figures reflect the storage requirements in our implementations; in particular, while the NGS, NSOR, and NCG storage requirements are minimal or close to minimal, the storage requirements for our multilevel methods could be reduced somewhat. To maintain a logically modular structure in our implementations, we have allowed some redundant storage in the implementations. In the methods NMH and DINMH, it is possible to implement the (linear or nonlinear) operator-based prolongation I_{k-1}^k completely in terms of the matrix A (and the nonlinearity $N(\cdot)$), without requiring explicit storage of I_{k-1}^k. This can save $27n/7 \approx 4n$, which makes these methods almost equivalent to NCG in terms of storage requirements, with NMH and DINMH requiring approximately $13.7n$ and $16n$, respectively.

Therefore, as in the case of the linear multilevel methods presented in Holst and Saied [14] and in references [5, 6], not only do the multilevel methods discussed here demonstrate superior complexity properties, we see that they can be implemented with very efficient memory use, requiring the same or only slightly more storage than that required for competing methods such as nonlinear conjugate gradient methods.

CONCLUSIONS

We have shown numerically that the multilevel-based methods discussed in this paper are generally more efficient than existing methods for the nonlinear Poisson-Boltzmann equation for a range of test molecules, and for a difficult test problem with large coefficient discontinuities and rapid nonlinearity. In addition, our results indicate that the damped-inexact-Newton-multilevel approach is not only the most efficient approach for these problems, but is also the most robust of all the methods considered. It converged in all situations, and for the SOD test problem was the only nonlinear method to converge.

Regarding the nonlinear multigrid method, both theoretical and numerical evidence here and elsewhere [5, 6] suggests that without careful use of special techniques for constructing the coarse problems (discussed in Holst and Saied [14] and Holst [5, 6]), and without the use

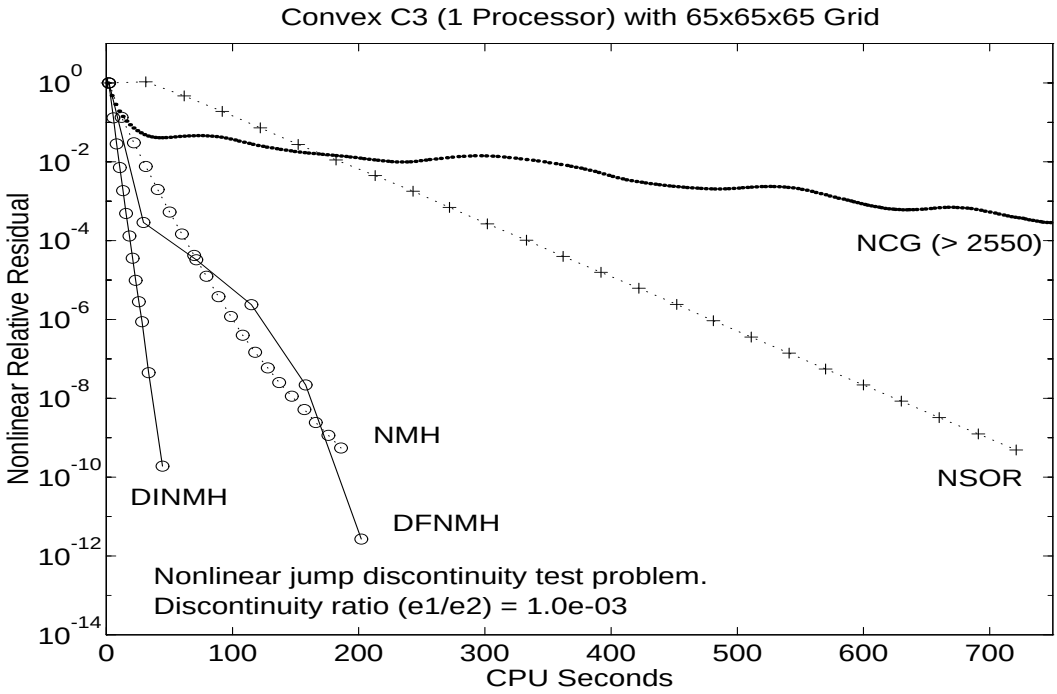

Figure 9: Comparison of methods for the nonlinear discontinuity problem.

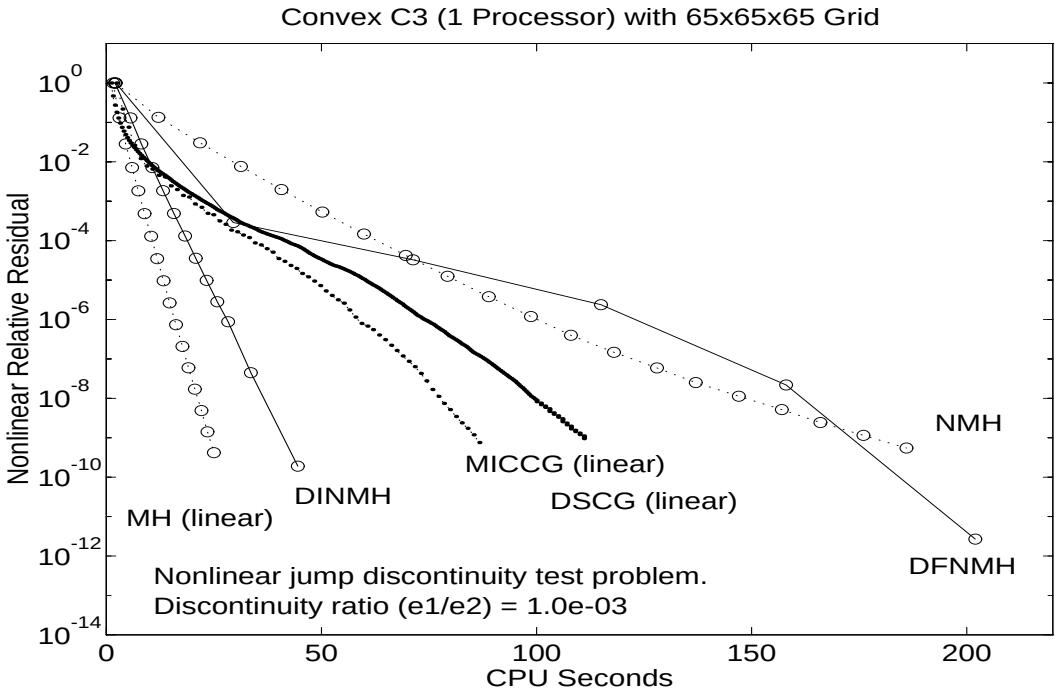

Figure 10: Enlargement of previous figure, with linear methods added.

Table 3: Storage required by various nonlinear (and some linear) elliptic solvers.

Method Name	Storage Requirements						
	A	$N(\cdot)$	u	f	I_{k-1}^{k}	WORK	TOTAL
NGS	$4n$	$1n$	$1n$	$1n$	$0n$	$1n$	$8n$
NSOR	$4n$	$1n$	$1n$	$1n$	$0n$	$1n$	$8n$
NCG	$4n$	$1n$	$1n$	$1n$	$0n$	$6n$	$13n$
NMH	$4n + \frac{4}{7}n$	$1n + \frac{1}{7}n$	$1n + \frac{1}{7}n$	$1n + \frac{1}{7}n$	$\frac{27}{7}n$	$5n + \frac{5}{7}n$	$\approx 17.6n$
DINMH	$4n + \frac{4}{7}n$	$1n + \frac{1}{7}n$	$1n + \frac{1}{7}n$	$1n + \frac{1}{7}n$	$\frac{27}{7}n$	$7n + \frac{7}{7}n$	$\approx 19.9n$

of a damping parameter, nonlinear multigrid methods are inherently non-reliable (may not converge) for problems with exponential nonlinearities and large coefficient discontinuities, such as the nonlinear Poisson-Boltzmann equation and equations occurring in semiconductor modeling.

We believe that this was not observed in the study of Oberoi and Allewell [30] due to the small number of examples considered. In fact, they did not seem to require either the damping parameter or the coefficient averaging techniques built into the method NMH employed here. Our experiments with such a "vanilla" nonlinear multigrid method showed divergence except for the simplest possible molecules (acetamide) at very low ionic strengths ($I_s < 0.0001$) and small mesh sizes ($31 \times 31 \times 31$). To obtain convergence even for the crambin case (Figure 5) required both the harmonic coefficient averaging approach developed in Holst and Saied [14] and the damping parameter discussed earlier in this paper. We are not aware of other techniques for increasing the robustness of nonlinear multigrid, without sacrificing most of the efficiency of the multilevel approach.

The damped inexact-Newton-multilevel method appears to be the most robust of all methods that have been proposed; it converges for all cases we have encountered (this behavior is supported by the theory presented in this paper, which was used to construct the method), and in particular it converges for cases which cause all other proposed methods to fail. In addition, it appears to be substantially more efficient than all other methods that have been proposed (by orders of magnitude for some test problems), and is even more efficient than some of the best existing linear methods producing only a linearized solution.

These considerations demonstrate that the damped inexact-Newton-multilevel method presented in this paper not only makes the nonlinear model completely feasible by providing a very reliable solution technique, but it actually improves on the efficiency of available linear algorithms which are currently used for the less accurate linear model. We remark that initial numerical experiments with larger mesh sizes show that the improvement of the damped-inexact-Newton-multilevel approach over methods grows with the problem size [49].

ACKNOWLEDGMENTS

The author thanks Dr. Saied at the University of Illinois, who supervised much of this work. Thanks are also due to Dr. Kerkhoven, Dr. Saylor, and Dr. Skeel in the Computer Science Department at the University of Illinois, for many helpful discussions on the material in this paper. Dr. Subramaniam and Dr. Kozack of the Beckmann Institute at the University of Illinois provided helpful advice regarding the biophysical aspects of this work. Dr. Nicholls of Columbia University made much effort to supply the author with the molecule data for testing, and made available his experience with this problem and his expertise with the electrostatics program DELPHI, developed in the laboratory of Dr. Honig. Dr. McCammon and his group in the Department of Chemistry at the University of Houston made available the UHBD (University of Houston Brownian Dynamics) software package, which allowed us to access additional data from the Brookhaven protein databank. The National Center for Supercomputer Applications, located at the University of Illinois at Urbana-Champaign, provided access to the Cray Y-MP and the Convex C3 through support by the National Science Foundation. Access to the Convex C240 was provided by the Computing Services Organization, also at the University of Illinois at Urbana-Champaign.

This work was performed at the University of Illinois and at the California Institute of Technology, and was supported in part by Department of Energy Grant No. DOE DE-FG02-91ER25099, and in part by the NSF under Cooperative Agreement No. CCR-9120008.

REFERENCES

[1] P. Debye and E. Hückel, *Physik. Z.*, **24**, 185 (1923).

[2] J. M. Briggs and J. A. McCammon, *Computers in Physics*, **6** (3), 238–243 (1990).

[3] K. A. Sharp and B. Honig, *Annu. Rev. Biophys. Biophys. Chem.*, **19**, 301–332 (1990).

[4] C. Tanford, *Physical Chemistry of Macromolecules*, John Wiley & Sons, New York, NY, 1961.

[5] M. Holst, tech. rep., Department of Applied Mathematics and CRPC, California Institute of Technology, 1994.

[6] M. Holst, PhD thesis, Numerical Computing Group, Department of Computer Science, University of Illinois at Urbana-Champaign, 1993. Also published as Tech. Rep. UIUCDCS-R-03-1821.

[7] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[8] G. Strang and G. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[9] E. L. Wachspress, *Iterative Solution of Elliptic Systems and Applications to the Neutron Diffusion Equations of Reactor Physics*, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[10] R. E. Bank and D. J. Rose, *SIAM J. Numer. Anal.*, **24** (4), 777–787 (1987).

[11] T. Kerkhoven, *SIAM J. Numer. Anal.*, (1992). (To appear).

[12] J. M. Ortega, *Numerical Analysis: A Second Course*, Academic Press, New York, NY, 1972.

[13] A. Nicholls and B. Honig, *J. Comput. Chem.*, **12** (4), 435–445 (1991).

[14] M. Holst and F. Saied, *J. Comput. Chem.*, **14** (1), 105–113 (1993).

[15] M. R. Hestenes and E. Stiefel, *J. Research of NBS*, **49**, 409–435 (1952).

[16] P. Concus, G. H. Golub, and D. P. O'Leary, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, NY, 1976, pp. 309–332.

[17] S. F. Ashby, T. A. Manteuffel, and P. E. Saylor, Tech. Rep. UCRL-98508, Lawrence Livermore National Laboratory, March 1988. To appear in SIAM J. Numer. Anal.

[18] M. E. Davis and J. A. McCammon, *J. Comput. Chem.*, **10** (3), 386–391 (1989).

[19] H. A. van der Vorst, *SIAM J. Sci. Statist. Comput.*, **10** (6), 1174–1185 (1989).

[20] A. Brandt, *Math. Comp.*, **31**, 333–390 (1977).

[21] W. Hackbusch, in Multigrid Methods: Proceedings of Köln-Porz Conference on Multigrid Methods, Lecture notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Berlin, Germany, 1982, Springer-Verlag.

[22] W. Hackbusch, *Multi-grid Methods and Applications*, Springer-Verlag, Berlin, Germany, 1985.

[23] R. E. Alcouffe, A. Brandt, J. E. Dendy, Jr., and J. W. Painter, *SIAM J. Sci. Statist. Comput.*, **2** (4), 430–454 (1981).

[24] J. E. Dendy, Jr. and J. M. Hyman, in Elliptic Problem Solvers, M. Schultz, ed., New York, NY, 1981, Academic Press.

[25] J. E. Dendy, Jr., *J. Comput. Phys.*, **48**, 366–386 (1982).

[26] J. E. Dendy, Jr., *SIAM J. Sci. Statist. Comput.*, **8** (2), 673–685 (1987).

[27] J. H. Bramble and J. E. Pasciak, *Math. Comp.*, **49** (180), 311–329 (1987).

[28] S. A. Allison, J. J. Sines, and A. Wierzbicki, *J. Phys. Chem.*, **93**, 5819–5823 (1989).

[29] B. A. Luty, M. E. Davis, and J. A. McCammon, *J. Comput. Chem.*, **13** (9), 1114–1118 (1992).

[30] H. Oberoi and N. M. Allewell, *Biophysical Journal*, (1993). (To Appear).

[31] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, 1970.

[32] A. Nicholls, *Private communication*, 1993.

[33] R. Fletcher and C. Reeves, *Comput. J.*, **7**, 149–154 (1964).

[34] W. Hackbusch, *Numer. Math.*, **32**, 83–95 (1979).

[35] W. Hackbusch and A. Reusken, in Robust Multigrid Methods, W. Hackbusch, ed., Braunschweig, 1988, Vieweg, pp. 105–113.

[36] W. Hackbusch and A. Reusken, *Numer. Math.*, **55**, 225–246 (1989).

[37] A. Reusken, *Numer. Math.*, **52**, 251–277 (1988).

[38] A. Reusken, *Numer. Math.*, **53**, 663–686 (1988).

[39] L. V. Kantorovich and G. P. Akilov, *Functional Analysis*, Pergamon Press, New York, NY, 1982.

[40] J. E. Dennis, Jr. and J. J. Moré, *Math. Comp.*, **28** (126), 549–560 (1974).

[41] J. E. Dennis, Jr. and J. J. Moré, *Siam Review*, **19** (1), 46–89 (1977).

[42] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *SIAM J. Numer. Anal.*, **19** (2), 400–408 (1982).

[43] S. Kesavan, *Topics in Functional Analysis and Applications*, John Wiley & Sons, Inc., New York, NY, 1989.

[44] S. C. Eisenstat and H. F. Walker, tech. rep., Dept. of Mathematics and Statistics, Utah State University, 1992.

[45] R. E. Bank and D. J. Rose, *Numer. Math.*, **37**, 279–295 (1981).

[46] R. E. Bank and D. J. Rose, *Math. Comp.*, **39** (160), 453–465 (1982).

[47] B. Jayaram, K. A. Sharp, and B. Honig, *Biopolymers*, **28**, 975–993 (1989).

[48] H. T. Davis, *Introduction to Nonlinear Differential and Integral Equations*, Dover Publications, Inc., New York, NY, 1960.

[49] M. Holst, R. Kozack, F. Saied, and S. Subramaniam, *Proteins: Structure, Function, and Genetics*, **18** (3), 231–245 (1994).