Performance of Hybrid Methods for Large-Scale Unconstrained Optimization as Applied to Models of Proteins

B. DAS,[1] H. MEIROVITCH,[1] I. M. NAVON[2]

[1]*Center for Computational Biology and Bioinformatics and Department of Molecular Genetics and Biochemistry, University of Pittsburgh School of Medicine, 200 Lothrop Street, BST 1058W, Pittsburgh, Pennsylvania 15261*

[2]*Department of Mathematics and Computational Science and Information Technology, Florida State University, Tallahassee, Florida 32306*

Received 18 October 2002; Accepted 22 January 2003

Abstract: Energy minimization plays an important role in structure determination and analysis of proteins, peptides, and other organic molecules; therefore, development of efficient minimization algorithms is important. Recently, Morales and Nocedal developed hybrid methods for large-scale unconstrained optimization that interlace iterations of the limited-memory BFGS method (L-BFGS) and the Hessian-free Newton method (Computat Opt Appl 2002, 21, 143–154). We test the performance of this approach as compared to those of the L-BFGS algorithm of Liu and Nocedal and the truncated Newton (TN) with automatic preconditioner of Nash, as applied to the protein bovine pancreatic trypsin inhibitor (BPTI) and a loop of the protein ribonuclease A. These systems are described by the all-atom AMBER force field with a dielectric constant $\epsilon = 1$ and a distance-dependent dielectric function $\epsilon = 2r$, where r is the distance between two atoms. It is shown that for the optimal parameters the hybrid approach is typically two times more efficient in terms of CPU time and function/gradient calculations than the two other methods. The advantage of the hybrid approach increases as the electrostatic interactions become stronger, that is, in going from $\epsilon = 2r$ to $\epsilon = 1$, which leads to a more rugged and probably more nonlinear potential energy surface. However, no general rule that defines the optimal parameters has been found and their determination requires a relatively large number of trial-and-error calculations for each problem.

© 2003 Wiley Periodicals, Inc.   J Comput Chem 24: 1222–1231, 2003

Key words: energy minimization; proteins; loops; hybrid method; truncated Newton; dielectric constant; force field

Introduction

The interatomic interactions of biomolecules such as proteins and nucleic acids are usually described by an empirical potential energy function (force field), which depends on the structure (geometry) of the molecule and typically leads to a "rugged" energy surface consisting of a tremendous number of local minima.[1] Identifying the lowest-energy minima, in particular the *global* minimum, is the goal of protein folding studies, where the energy, rather than the free energy, is accepted as an *approximate* criterion of stability. A more rigorous criterion is minimum harmonic free energy, F^{har}, where F^{har} is obtained at an energy minimum from the harmonic entropy, S^{har}, which is proportional to the determinant of the Hessian, the matrix of second derivatives of the energy with respect to the molecular coordinates.[2–6] Calculation of the Hessian at a minimum is also an essential part of a normal-mode analysis.[7]

The above discussion already demonstrates the importance of energy minimization in computational structural biology and the need for developing efficient minimization algorithms. The common algorithms, such as conjugate gradients or Newton methods, are of a local character, that is, they drive an initial molecular structure to the closest energy minimum rather than to the global one. However, the applicability of these methods is much wider because most of the *global* optimization procedures, including our local torsional deformation (LTD) method for cyclic molecules,[8,9] are based on a large number of local energy minimizations (see,

Correspondence to: H. Meirovitch; e-mail: hagaim@pitt.edu

Contract/grant sponsor: National Institutes of Health; contract/grant number: R01GM61916

Contract/grant sponsor: National Science Foundation; contract/grant number: ATM-0201808 (I.M.N.)

e.g., refs. 10–18). Therefore, attempting to optimize LTD, we tested in a previous preliminary study several minimization algorithms and found the limited-memory BFGS (L-BFGS)[19] to be the most efficient for a force field with an implicit solvation model[9] [see eqs. (1) and (2) below] .

The experience gained thus far from treating various problems, in particular large-scale unconstrained minimizations,[20–23] is that truncated Newton (TN)[24–27] and L-BFGS are powerful optimization methods that are more efficient than other techniques (see also refs. 28–30). TN tends to blend the rapid (quadratic) convergence rate of the classic Newton method with feasible storage and computational requirements. The L-BFGS algorithm is simple to implement because it does not require knowledge of the sparsity structure of the Hessian or knowledge of the separability of the objective function. Further, the amount of storage needed can be controlled by the user. It has been found that, in general, TN performs better than L-BFGS for functions that are nearly quadratic, while for highly nonlinear functions L-BFGS outperforms TN.[31]

These aspects and others are discussed in an excellent review on minimization methods by Schlick,[32] who together with Fogelson also programmed their own TN algorithm and included it in the package TNPACK.[26,27] This package enables the user to supply a sparse preconditioning matrix that transfers the Hessian into a matrix with more clustered eigenvalues, which in turn enhances convergence. This implementation of TN differs from that of Nash,[25] which uses automatic preconditioning and has been applied to a variety of problems with considerable success; the latter has the advantage of easy portability because the preconditioner does not have to be tailored to the specific problems at hand. In her review, Schlick presents systematic efficiency comparisons between several algorithms applied to the molecule deoxycytine (87 variables) and to clusters of water molecules. For the former system, TN with preconditioning is found to be the most efficient, requiring ≈ 2 times less CPU time than L-BFGS with preconditioning, while for the water clusters the picture is more complex.

Derreumaux et al.[33] tested the efficiency of TN as applied to peptides and proteins modeled by the CHARMM force field[34] using an updated version of TNPACK. In this implementation, the preconditioner is based on the short-range interactions, that is, the bond stretching and bending, and the torsional potentials. It is shown that for several molecules of sizes $n = 12$–1299 atoms TNPACK with preconditioning outperforms the steepest-descent, nonlinear conjugate gradient, adapted basis Newton–Raphson, and Newton–Raphson algorithms installed in the CHARMM package.[33] More recently, Xie and Schlick showed that for molecules of sizes $n = 22$–2030 atoms TNPACK requires less CPU time than both CG and L-BFGS and reaches low gradient norms.[35,36]

Because the performance of minimization algorithms depends to a large extent on the system and the cost function used,[37] we have carried out recently[38] a systematic performance study of the algorithms, L-BFGS of Liu and Nocedal,[19] TN with automatic preconditioner of Nash,[25] and the nonlinear conjugate gradients (CG) of Shanno and Phua.[39] These algorithms were applied to penta- and heptacyclic peptides and the 58-residue protein bovine pancreatic trypsin inhibitor (BPTI) modeled by two energy functions. One is the GROMOS87 united atoms force field,[40] which takes into account the intramolecular interactions only; the second function is defined by the GROMOS potential energy and an implicit solvation term based on the accessible surface area of each atom and its solvation parameter [see eqs. (1) and (2)]. With the GROMOS energy alone the performance of TN with respect to the CPU time was found to be 1.2–2 times better than that of both L-BFGS and CG; on the other hand, for the solvation model L-BFGS outperforms TN by a factor of 1.5–2.5 and CG by a larger factor, in accord with our preliminary studies.[9] These results were also analyzed in light of theories developed by Nash and Nocedal[31] and Axelsson and Lindskog,[41,42] which rely on the eigenvalues and other quantities derived from the Hessian. Our study has been the first where such an analysis has been applied to optimization problems of biomacromolecules.

Recently, Morales and Nocedal[43] developed a hybrid method that consists of interlacing in a dynamic way the L-BFGS method with the Hessian-free Newton (HFN). The limited-memory matrix [see eqs. (8)–(10) below] plays a dual role of preconditioning the inner conjugate gradient iteration in the HFN method as well as providing the initial approximation of the inverse of the Hessian matrix in the L-BFGS iteration. In this way information gathered by each method improves the performance of the other without increasing the computational cost. The hybrid method alleviates the shortcomings of both L-BFGS and HFN. HFN normally requires fewer iterations than L-BFGS to reach the solution, but the effort invested in one iteration can be high and the curvature information gathered in the process is lost after the iteration is completed. L-BFGS, on the other hand, performs inexpensive iterations, with poorer curvature information—a process that can become slow on ill-conditioned problems. Indeed, Alekseev and Navon[44] found the hybrid method to be the best performer as tested on cost functionals related to inverse problems in fluid dynamics (see also ref. 45).

Because of the potential of the hybrid method, in this article we study its performance as applied to the protein BPTI and a loop of the protein ribonuclease (RNase) A modeled by the all-atom AMBER force field[46] (implemented in the program TINKER[47]). This potential function is defined with a dielectric constant $\epsilon = 1$ and with a distance-dependent dielectric function $\epsilon = 2r$, where r is the distance between two atoms. The electrostatic interactions for $\epsilon = 1$ are strong, leading to a rugged potential energy surface that becomes flatter for $\epsilon = 2r$ due to the weaker electrostatic interactions.[48] Therefore, the nonlinearity in the potential function is expected to decrease in going from $\epsilon = 1$ to $\epsilon = 2r$ and it is of interest to study the effect of this property on the performance of various minimization procedures. As in our previous article,[38] we also test the solvation model discussed above. For comparison, these systems are also studied by the L-BFGS algorithm of Liu and Nocedal and the TN algorithm with automatic preconditioner of Nash. The performance of the algorithms is compared with respect to the CPU time and the number of energy/gradient calculations.

Theory and Methods

Protein Models and Energy Functions

Two systems are studied. One is the 58-residue protein BPTI that consists of 878 atoms (including all the hydrogens), where the variables treated in the minimizations are the corresponding 878 × 3 = 2634 Cartesian coordinates; the initial 3D structure for energy minimization is the crystal structure denoted 8pti in the Protein Data Bank (PDB) to which hydrogen atoms have been added. The second system is the 8-residue loop (64–71), Ala-Cys-Lys-Asn-Gly-Gln-Thr-Asn (108 atoms) of RNase A (1860 atoms), where the initial structure for energy minimization is 1rat.pdb with added hydrogens. In this case only the loop is allowed to move during the minimization while the coordinates of the rest of the protein (the template) are held fixed at their X-ray values. More accurately, only 614 atoms are included in the template, that is, any nonloop atom with a distance smaller than 10 Å from at least one loop atom (in the initial loop structure), together with all the other atoms pertaining to the same residue. Thus, the number of Cartesian variables in the loop–protein system is 108 × 3 = 324. The calculations (including the addition of hydrogens) were performed with the molecular mechanics/dynamic program TINKER,[47] where the various optimizers have been implemented as subroutines.

The potential energy E_{AMBER} of these systems is defined by the all-atom AMBER force field[46] (provided by the program TINKER) consisting of harmonic bond stretching potentials (to maintain the connectivity of the polymer chain), harmonic bond bending potentials, torsional potentials that depend on dihedral angles, ϕ, and nonbonded 6-12 Lennard–Jones and electrostatic interactions (between charges and partial charges q_i),

$$E_{\text{AMBER}} = \sum_{\text{bonds}} K_r(r - r_{\text{eq}})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{\text{eq}})^2 + \sum_{\text{dihedrals}} \frac{V_n}{2}[1 + \cos(n\phi - \gamma)] + \sum_{i<j}\left[\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{\epsilon r_{ij}}\right] \quad (1)$$

r and θ are the actual values of the bond lengths and angles and r_{eq} and θ_{eq} are their equilibrium values, respectively. r_{ij} is the distance between atoms i and j and K_r, K_θ, V_n, n, γ, A_{ij}, *and* B_{ij} are constants. We study the dielectric constant $\epsilon = 1$ and a distance-dependent dielectric function $\epsilon = 2r_{ij}$, which mimics the screening of the electrostatic interactions by the surrounding water. For BPTI the nonbonded interactions are calculated between every pair of atoms without applying cutoffs. For the loop, on the other hand, only the loop–loop and loop–template interactions are considered, while the template–template interactions are ignored. For both molecules the amino acid residues Arg, Lys, His, Asp, and Glu are charged. It should be pointed out that distance-dependent dielectric functions $\epsilon = mr_{ij}$ with $m \geq 1$ have been commonly added to force fields to partially model solvent effects (see ref. 48 and references cited therein); therefore, it is of interest to test minimization procedures applied to such force fields. Still, better implicit solvation models are available that include in addition to $\epsilon = mr_{ij}$, also the Born energy and the hydrophobic interaction.[48] A simplified potential of this category, E_{tot}, is studied here for the loop.

Thus, the total potential energy E_{tot} is based on the force field energy and a solvation term, E_{solv},

$$E_{\text{tot}} = E_{\text{AMBER}} + E_{\text{solv}} = E_{\text{AMBER}} + \sum_i \sigma_i A_i \quad (2)$$

A_i is the (conformational-dependent) solvent-accessible surface area (SASA) of atom i and σ_i is the corresponding atomic solvation parameter (ASP) derived by Das and Meirovitch for protein loops.[48] The SASA is defined as the surface traced by the center of a spherical probe of 1.4 Å (mimicking a water molecule) as it is rolled over the surface of the protein; This area is calculated analytically with the program MSEED,[49] which is based on a modification of the analytic equations presented by Connoly[50] and Richmond.[51] Use is made of the global Gauss–Bonet formula, which describes the closed boundary of a regular region bounded by simple, piece-wise regular curves. The program provides analytic derivatives of the SASA with respect to the Cartesian coordinates, which are required by the present minimizers. One problem with E_{tot} is the possible occurrence of discontinuities in the gradient of A_i and the existence of saddle points. This might stop the minimization process close to a local minimum, when the contributions to the gradient from all the components are small. In fact, gradient norms of only up to 10^{-3} kcal/(mol · Å) have been found to be attainable with E_{tot}.

Description of the Algorithms

In this work we test implementations of the L-BFGS Version VA15 in the Harwell library,[19] the TN method described by Nash,[25] and the hybrid method of Morales and Nocedal.[43] A brief description of the major components of each algorithm is given below. While we do not apply the conjugate gradient algorithm in this study, it is part of the L-BFGS method; therefore, for completeness we describe it as well.[39]

For a molecule of n atoms we use the following notations: $f_k = f(\mathbf{x}_k)$ denotes the potential energy function E [eqs. (1) and (2)], where $\mathbf{x}_k$ is the $3n$ vector of the Cartesian coordinates at the kth iteration. $\mathbf{g}_k = g(\mathbf{x}_k) = \nabla f_k$ is the gradient vector of size $3n$, and $H_k = \nabla^2 f_k$ is the $3n \times 3n$ symmetrical Hessian matrix of the second partial derivatives of f with respect to the coordinates. In all the algorithms the new iterate is calculated from

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \quad (3)$$

where $\mathbf{p}_k$ is the descent direction vector and α_k is the step length. Iterations are terminated when

$$\|\mathbf{g}_k\| < 10^{-6}\max(1, \|\mathbf{x}_k\|) \quad (4)$$

where $\|\cdot\|$ denotes the Euclidian norm. The necessary changes in the programs were made to ensure that all three algorithms use the same termination criterion. Also, the three methods use the same line search, which is based on cubic interpolation and is subject to the so-called *strong Wolf conditions*,[52]

$$f(\mathbf{x}_k) - f(\mathbf{x}_k + \alpha_k\mathbf{p}_k) \geq -\mu\alpha_k\mathbf{p}_k^T\mathbf{g}_k$$
$$|\mathbf{g}(\mathbf{x}_k + \alpha_k\mathbf{p}_k)^T\mathbf{p}_k| \leq \eta|\mathbf{g}_k^T\mathbf{p}_k| \quad (5)$$

where the superscript T denotes transpose, $0 < \mu < \eta < 1$ and their values are given below.

Nonlinear Conjugate Gradient Algorithm

CG uses the analytic derivatives of *f*, defined by $\mathbf{g}_k$. A step along the current negative gradient vector is taken in the first iteration; successive directions are constructed so that they form a set of mutually conjugate vectors with respect to the Hessian. At each step, the new iterate is calculated from eq. (3) and the search directions are expressed recursively as

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1} \tag{6}$$

Calculation of β_k with the algorithm incorporated in CONMIN is described by Shanno.[53] Automatic restarting is used to preserve a linear convergence rate. For restart iterations, the search direction $\alpha_k = 1$. On the other hand, for nonrestart iteration

$$\alpha_{k+1} = \frac{\alpha_k \mathbf{g}_k^T \mathbf{p}_k}{\mathbf{g}_{k+1}^T \mathbf{p}_{k-1}} \tag{7}$$

Limited-Memory BFGS Algorithm

The L-BFGS method is an adaptation of the BFGS method to large problems, achieved by changing the Hessian update of the latter. Thus, in BFGS[54,55] eq. (3) is used with an approximation $\tilde{H}_k$ to the inverse Hessian that is updated by

$$\tilde{H}_{k+1} = V_k^T \tilde{H}_k V_k + \rho_k \mathbf{s}_k \mathbf{s}_k^T \tag{8}$$

where $V_k = I - \rho_k \mathbf{y}_k \mathbf{s}_k^T$, $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$, $\boldsymbol{\rho}_k = 1/(\mathbf{y}_k^T \mathbf{s}_k)$, and I is the identity matrix. The search direction is given by

$$\mathbf{p}_{k+1} = -\tilde{H}_{k+1} \mathbf{g}_{k+1} \tag{9}$$

In L-BFGS, instead of forming the matrices $\tilde{H}_k$ explicitly (which would require a large memory for a large problem) one only stores the vectors $\mathbf{s}_k$ and $\mathbf{y}_k$ obtained in the last m iterations, which define $\tilde{H}_k$ implicitly; a cyclical procedure is used to retain the latest vectors and discard the oldest ones. Thus, after the first m iterations, eq. (8) becomes

$$\begin{aligned}\tilde{H}_{k+1} &= (V_k^T \cdots V_{k-m}^T)\tilde{H}_{k+1}^0 (V_{k-m} \cdots V_k)\\ &+ \rho_{k-m}(V_k^T \cdots V_{k-m+1}^T)\mathbf{s}_{k-m}\mathbf{s}_{k-m}^T\\ &\times (V_{k-m-1}^T \cdots V_k)\\ &+ \rho_{k-m-1}(V_k^T \cdots V_{k-m+2}^T)\mathbf{s}_{k-m+1}\mathbf{s}_{k-m+1}^T\\ &\times (V_{k-m+2}^T \cdots V_k)\\ &\vdots\\ &+ \rho_k \mathbf{s}_k \mathbf{s}_k^T \end{aligned} \tag{10}$$

with the initial guess $\tilde{H}_{k+1}^0$, which is the sparse matrix,

$$\tilde{H}_{k+1}^0 = \frac{\mathbf{y}_k^T \mathbf{s}_k}{\mathbf{y}_k^T \mathbf{y}_k} I \tag{11}$$

Many previous studies have shown that typically $3 \leq m \leq 7$, where $m > 7$ does not improve the performance of L-BFGS.

Truncated Newton Algorithm

In TN, a search direction is computed by finding an *approximate* solution to the Newton equations,

$$H_k \mathbf{p}_k = -\mathbf{g}_k \tag{12}$$

The use of an approximate search direction is justified because an exact solution of the Newton equation at a point far from the minimum is unnecessary and computationally wasteful in the framework of a basic descent method. Thus, for each *outer* iteration [eq. (12)], there is an *inner* iteration loop making use of the conjugate gradient method that computes this approximate direction, $\mathbf{p}_k$.

The conjugate gradient inner algorithm is preconditioned by a scaled two-step limited-memory BFGS method with Powell's restarting strategy used to reset the preconditioner periodically. A detailed description of the preconditioner may be found in ref. 56. The Hessian vector product $H_k \mathbf{v}$ for a given $\mathbf{v}$ required by the inner conjugate gradient algorithm is obtained by a finite difference approximation,

$$H_k \mathbf{v} \approx [\mathbf{g}(\mathbf{x}_k + h\mathbf{v}) - \mathbf{g}(\mathbf{x}_k)]/h \tag{13}$$

A major issue is how to adequately choose h;[22] in this work we use $h = \sqrt{\epsilon}(1 + \|\mathbf{x}_k\|)$, where ϵ is the machine precision. The inner algorithm is terminated using the quadratic truncation test, which monitors a sufficient decrease of the quadratic model $q_k = \mathbf{p}_k^T H_k \mathbf{p}_k/2 + \mathbf{p}_k^T \mathbf{g}_k$:

$$(1 - q_k^{i-1}/q_k^i) \leq c_q/i \tag{14}$$

where i is the counter for the inner iteration and c_q is a constant, $0 < c_q \leq 1$, which will be specified later. The inner algorithm is also terminated if an imposed upper limit on the number of inner iterations, M, is reached or when a loss of positive definiteness is detected in the Hessian (i.e., when $\mathbf{v}^T H_k \mathbf{v} < 10^{-12}$). TN methods can be extended to more general problems, which are not convex in much the same way as Newton's method (see ref. 57).

To understand the Hessian free Newton (HFN) method, let us consider an inexact Newton-type iteration for solving the problem, $\mathbf{x}^+ = \mathbf{x} + \alpha\mathbf{p}$, where α is a step length and the search direction, $\mathbf{p}$, is an approximate minimizer of the quadratic model, $q(\mathbf{p}) = f(\mathbf{x}) + \mathbf{p}^T g(\mathbf{x}) + (1/2)\mathbf{p}^T B \mathbf{p}$. Here, $\mathbf{g}$ denotes the gradient of the objective function f and B is a symmetrical and positive definite matrix. The approximate minimization of the quadratic q is performed by the CG method. It is assumed that if negative curvature is encountered the CG iteration terminates immediately without exploring this negative curvature direction. A Hessian-free inexact Newton method is a particular instance of this method in which B is intended to be the Hessian, $\Delta^2 f(\mathbf{x})$, but is not computed explicitly. All products of the form B are either approximated by finite differences, $B\mathbf{v} \approx [\mathbf{g}(\mathbf{x} + h\mathbf{v}) - \mathbf{g}(\mathbf{x})]/h$, where h is a small parameter, or are computed by automatic differentiation techniques. There is no consensus on the best termination test for the CG iteration, and various rules are used in practice.[35,36,58]

Hybrid Method

The hybrid method consists of interlacing in a dynamic way the L-BFGS method with the HFN discussed above. The limited-memory matrix plays a dual role of preconditioning the inner conjugate gradient iteration in the HFN method as well as providing the initial approximation of the inverse of the Hessian matrix in the L-BFGS iteration. In this way information gathered by each method improves the performance of the other without increasing the computational cost.

The hybrid method alleviates the shortcomings of both L-BFGS and HFN. HFN requires fewer iterations than L-BFGS to reach the solution, while the computational cost of each iteration is high and the curvature information gathered in the process is lost after the iteration has been completed. L-BFGS, on the other hand, performs inexpensive iterations, with poorer curvature information—a process that can become slow on ill-conditioned problems.

Algorithmically, implementation of the hybrid method includes two limited-memory matrices: one that is used to precondition the current CG (PCG) iteration and a second that is built in the course of the PCG iteration and will define the preconditioner for the next outer iteration. A detailed description of the preconditioning process is given in refs. 45 and 60. In the hybrid method that will be tested below, l steps of the L-BFGS method are alternated with t steps of the HFN method, where the choice of l and t will be discussed below. We illustrate this as

$$[l^*(\text{L-BFGS}) \xrightarrow{\tilde{H}(m)} t^*(\text{HFN}(\text{PCG})) \xrightarrow{\tilde{H}(m)} \text{repeat}],$$

where $\tilde{H}(m)$ is a limited-memory matrix that approximates the inverse of the Hessian matrix [eq. (10)] and m denotes the number of correction pairs stored. The L-BFGS cycle starts from the initial unit or a weighted unit matrix, $\tilde{H}(m)$ is updated using the most recent m pairs, and the matrix obtained at the last L-BFGS cycle is used to precondition the first of the t HFN iterations. In the remaining $t - 1$ iterations the limited-memory matrix $\tilde{H}(m)$ is updated using information generated by the inner preconditioned CG (PCG) iteration and it is used to precondition the next HFN iteration. At the end of the HFN steps the most current $\tilde{H}(m)$ matrix is used as the initial matrix in a new cycle of L-BFG steps.

The hybrid algorithm is described below:

```
Choose a starting point $\mathbf{x}$, the memory parameter $m$, and an initial
    choice of the length $l$ of the L-BFGS cycle; set method ←'L-
    BFGS'; first ←.true.
While a convergence test is not satisfied:
Repeat
compute $\mathbf{p}$: call PCG ($B$, $\mathbf{p}$, $\mathbf{g}$, method, status, $maxcg$);
compute $\alpha$: call LNSRCH ( $\alpha$ );
compute $\mathbf{x}^+ = \mathbf{x} + \alpha\mathbf{p}$;
store $\mathbf{s} = \mathbf{x}^+ - \mathbf{x}$ and $\mathbf{y} = \mathbf{g}^+ - \mathbf{g}$;
call ADJUST ( $l$, $t$, $\alpha$, method, status, first, $maxcg$ );
End repeat
End while.
```

The vectors **s** and **y** are used to update the limited memory matrix. The parameter "method" can have the values "L-BFGS" or "HFN" and "*maxcg*" determines the maximum number of CG iterations allowed. This procedure returns a search direction **p** and the value of "status," which is used by procedure ADJUST to modify the lengths l and t of the L-BFGS and HFN cycles. The procedure LNSRCH is a standard backtracking line search routine enforcing the Wolfe conditions [eq. (5)]. A more detailed description of procedures PCG and ADJUST is provided in Morales and Nocedal.[43]

Initial Tuning of the Algorithms

In ref. 38, part of the parameters were tuned as applied to several peptides and BPTI modeled by the GROMOS force field. The optimal values of these parameters were found to be equal to the default values, which are also used here because of the similarity between the systems studied and the functional forms of the AMBER and GROMOS force fields. For eq. (5) they are $\mu = 10^{-4}$ for L-BFGS and TN and the hybrid method, $\eta = 0.25$ for L-BFGS, and $\eta = 0.9$ for TN. For L-BFGS we checked several values of m and provide in the tables the values that led to the shortest and longest minimization times. Unlike ref. 38, where only $m = 1$ could be used for BPTI (with GROMOS), no problems were encountered here to apply L-BFGS($m > 1$) to BPTI; correspondingly, using CHARMM, Xie and Schlick applied successfully L-BFGS with $m > 1$ to BPTI and the larger protein lysozyme.[35]

For TN, the default value, $c_q = 0.5$, is the best for the quadratic truncation test [eq. (14)]. For this algorithm we also used $M = \max[N/2,50]$ and $h = \sqrt{\epsilon}\,(1 + \|\mathbf{x}_k\|)$. All the calculations were carried out in double precision on a workstation equipped with a Digital Alpha 21264 (500-MHz) processor. The machine precision is $\epsilon = 10^{-14}$.

Results and Discussion

The initial 3D structures (the atomic Cartesian coordinates) for energy minimization of BPTI (8pti) and RNase (irat) were taken from the PDB. Hydrogen atoms were added to these structures using the program TINKER,[47] which was also used for calculating the potential energy. The three optimization programs (minimizers)—L-BFGS, TN of Nash, and the hybrid method—were implemented within TINKER as subroutines.

Two tables are presented for each model, for a dielectric constant $\epsilon = 1$ (to be distinguished from ϵ, the machine precision) and a distance-dependent dielectric function $2r_{ij}$. For the hybrid method a relatively large number of minimizations (80–100) were performed for each model based on different combinations of the integer parameters *maxcg*, l, and t, where *maxcg* is the maximum number of conjugate gradient iterations and l and t are the initial number of iterations of L-BFGS and HFN, respectively. If $l = 0$ the code would run a pure HFN method, while for $t = 0$ the standard L-BFGS method will be performed. All the calculations with the hybrid method used an L-BFGS parameter $m = 8$. For each model two sets of results (out of ≈100 minimizations) are presented in the tables, those that required the shortest

Table 1. Minimization Results for the RNase Loop Using Dielectric Constant $\epsilon = 1$.

maxcg or *m*	*l*	*t*	CPU (s)	No. functions/gradients
Hybrid method				
Shortest minimizations (CPU)				
50	200	10	70	836
40	200	10	71	855
60	200	10	74	886
5	200	10	75	897
7	30	20	76	922
7	200	10	78	941
15	200	10	79	947
Longest minimizations (CPU)				
5	1	10	163	1955
5	50	30	123	1489
30	30	10	136	1637
40	150	10	122	1465
80	100	20	130	1566
L-BFGS				
8			111	1336
5			133	1593
TN-Nash			140	1672

Initial energy $E = -144.3$ kcal/mol; final minimized energy $E = -286.43377$. *maxcg* is the maximum number of CG iterations allowed. *t* and *l* are the numbers of HFN and L-BFGS iterations, respectively. A large number of minimizations (~100) were performed for different sets of parameters *maxcg*, *t*, and *l*. The fastest and slowest minimizations with respect to CPU time appear in shortest minimizations and longest minimizations, respectively. For L-BFGS the fastest and slowest minimizations are provided, where *m* is the number of iterations considered in the L-BFGS procedure. At the minimum, $\|\mathbf{x}_k\| = 654.3$ Å; $\|\mathbf{g}_k\| = 8.946 \times 10^{-3}$ kcal/(mol Å).

and longest CPU time, respectively; we also provide the number of times the function and gradients were calculated. For L-BFGS seven minimizations were carried out with $4 \leq m \leq 10$, but only two results are presented in tables for the *m* values that led to the minimizations with shortest and longest CPU times; only one result carried out with TN is displayed. The common tolerance [eq. (4)] is 10^{-6}.

In Table 1 results are presented for the loop using a dielectric constant $\epsilon = 1$. The best result for the hybrid method (70-s CPU) required 1.6 and 2 times less computer time than the shortest minimizations obtained by L-BFGS($m = 8$), (111-s CPU) and TN (140-s CPU), respectively. The function/gradient numbers are proportional to the corresponding CPU values. For this model the longest minimization obtained for the hybrid method (163-s CPU) is 2.3 times longer than the shortest one and on average the longest minimizations in the table require ≈1.8 more computer time than the shortest minimizations. The problem is that no clear correlation between the parameters *maxcg*, *l*, and *t* and the efficiency of the minimization has been observed. The suggested value of the parameter *maxcg* is min(*N*/2, 50), where *N* is the number of variables, meaning that in our case *maxcg* ≈50. However, while the best results indeed were obtained with *maxcg* = 40, 50, and 60, some of the worst results were obtained with comparable values *maxcg* = 30, 40, and 80 whereas some of the shortest minimizations were obtained with *maxcg* = 5–15. Still, from the entire study we found that *maxcg* < 5 in most cases leads to relatively long minimizations. Another general conclusion from the entire study is that the results are better for $l > t$ as is already evident from Table 1 (note that the worst result is obtained for $l = 1$ and $t = 10$); still, satisfying this condition does not guarantee an efficient minimization, where $l = 200$ and $t = 0$ have led to the best result, while long minimizations were obtained with the corresponding pairs (150, 10) and (100, 20) (see Table 1).

In Table 2 results are displayed for the same loop, modeled with the AMBER force field with a distance-dependent dielectric function $\epsilon = 2r_{ij}$ rather than $\epsilon = 1$. The weakened electrostatic interactions for $\epsilon = 2r_{ij}$ are expected to lead to a flatter energy surface and hence a decrease in the nonlinearity of the energy function, making the minimization procedure easier. Indeed, the shortest minimizations for the hybrid and L-BFGS methods in Table 2 required slightly less computer time than the corresponding calculations in Table 1. Also, the ratio CPU(L-BFGS)/CPU-(hybrid) calculated for the shortest minimizations decreased from 1.6 ($\epsilon = 1$) to 1.54 ($\epsilon = 2r_{ij}$), while the corresponding ratio for TN decreased more significantly, from 2 (Table 1) to 1.42 (Table 2) due to a strong decrease in the TN minimization time, from 140-s CPU (Table 1) to 98-s CPU (Table 2). Correspondingly, the difference between the average CPU times required for the shortest and longest minimizations of the hybrid and L-BFGS methods is smaller in Table 2 than in Table 1. In Table 2, however, the values

Table 2. Minimization Results for the RNase A Loop Using a Distance-Dependent Dielectric Function, $\epsilon = 2r$.

maxcg or *m*	*l*	*t*	CPU (s)	No. functions/gradients
Hybrid method				
Shortest minimizations				
50	30	20	69	844
80	100	10	70	856
30	100	10	71	874
80	150	10	72	890
70	150	10	72	887
70	100	20	72	892
70	30	20	73	902
40	30	20	73	898
40	50	5	73	898
Longest minimizations				
7	30	10	109	1345
5	1	10	106	1302
10	30	20	105	1296
10	50	30	105	1294
15	5	10	105	1294
L-BFGS				
6			106	1301
8			114	1405
TN-Nash			98	1198

Initial energy $E = 48.2$ kcal/mol; final minimized energy $E = -61.75175$. The parameters are defined in Table 1. $\|\mathbf{x}_k\| = 646.6$ Å; $\|\mathbf{g}_k\| = 8.894 \times 10^{-3}$ kcal/(mol Å).

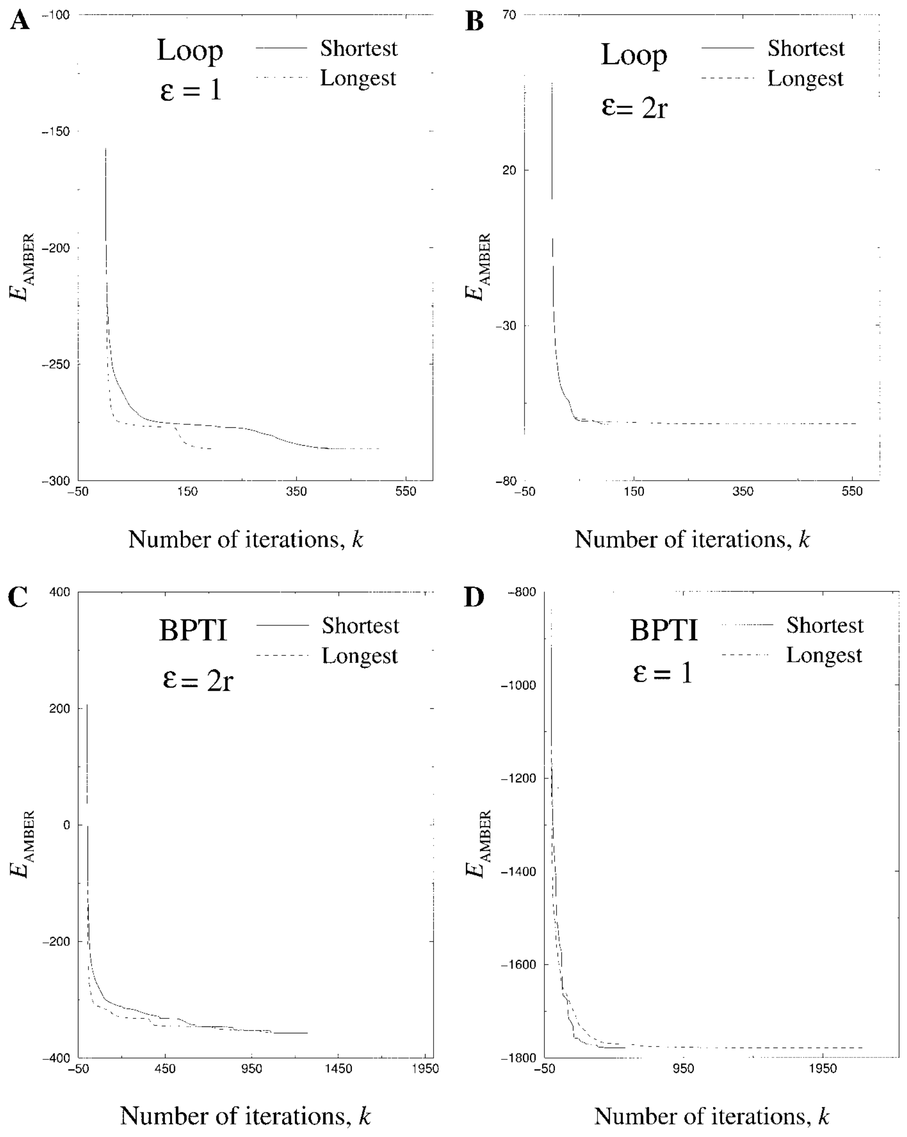

Figure 1. Plots of the AMBER force field energy, $E_{\rm AMBER}$ (eq. (1)), versus the number of iterations k obtained during the shortest and longest minimizations of the loop [(a) and (b), $\epsilon = 1$ and $\epsilon = 2r_{ij}$, respectively] and BPTI [(c) and (d), $\epsilon = 2r_{ij}$, and $\epsilon = 1$, respectively].

of *maxcg* are significantly larger for the shortest minimizations (40–80) than for the longest ones (7–15). Again, while $l > t$ appears to be a necessary condition for getting the shortest minimizations, it is not a sufficient condition as $l > t$ also occurs for the longest minimizations.

As in Table 1, the CPU ratios for the hybrid method minimizations based on different parameters are close to the corresponding ratios obtained from the numbers of the function/gradient calculated. In Figures 1a and 1b the energy, $E_{\rm AMBER}$ (eq. (1)), is plotted against the number of iterations k for $\epsilon = 1$ and $\epsilon = 2r_{ij}$, respectively, whereas in Figures 2a and 2b the corresponding results are presented for $\log(\|\mathbf{g}_k\|/\|\mathbf{g}_0\|$. In both figures the results are displayed for the shortest and longest minimizations. It is interesting to note that while the CPU times and the total number of function/gradient calculations of the shortest minimizations for $\epsilon = 2r_{ij}$ and $\epsilon = 1$ are almost the same the number of iterations is significantly different, 105 and 440, respectively, which is clearly demonstrated in the figures. On the other hand, the corresponding numbers of iterations for the longest minimizations are comparable, 560 and 511 although ≈50% more function/gradient calculations are carried out for $\epsilon = 2r_{ij}$.

The results in Table 3 are for BPTI modeled by AMBER ($\epsilon = 2r_{ij}$). For this molecule the effect of the muted electrostatic interactions is stronger than for the loop. Thus, while the hybrid method still provides the shortest minimizations, the ratios CPU(LBFGS, m = 5)/CPU(hybrid-shortest) = 1.17 and CPU(TN)/CPU(hybrid-shortest) = 1.19 are significantly smaller than the corresponding ratios obtained for the loop in Tables 1 and 2. Correspondingly,

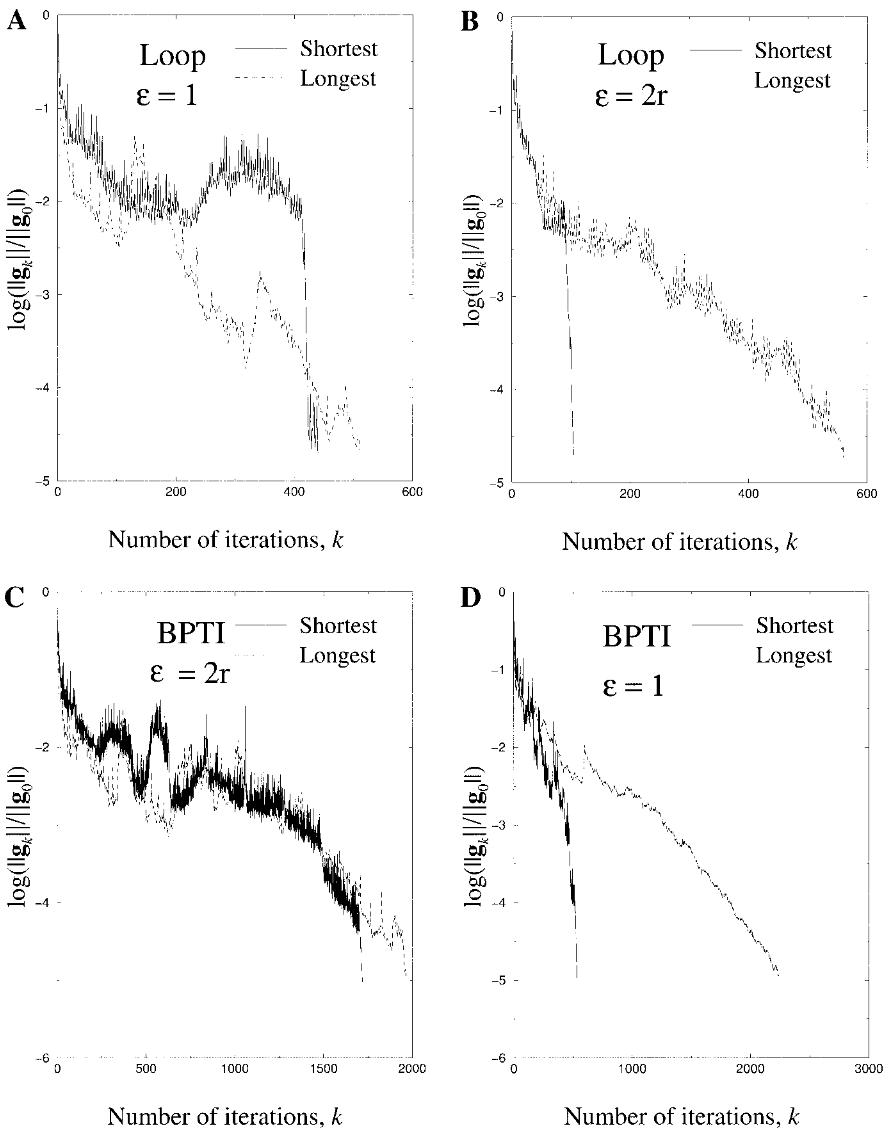

Figure 2. Plots of log($\|\mathbf{g}_k\|/\|\mathbf{g}_0\|$) versus the number of iterations k obtained during the shortest and longest minimizations of the loop [(a) and (b), $\epsilon = 1$ and $\epsilon = 2r_{ij}$, respectively] and BPTI [(c) and (d), $\epsilon = 2r_{ij}$, and $\epsilon = 1$, respectively].

relatively low values, 1.59 and 1.2, are obtained for the ratio CPU(hybrid-longest)/CPU(hybrid-shortest) and the same ratio based on the average CPU times of the sets of shortest and longest minimizations, respectively. The ratios of the function/gradient evaluations are similar to the corresponding CPU ratios, and the behavior of the parameters *maxcg*, *l*, and *t* is similar to that observed in Tables 1 and 2. Figures 1c and 2c (for $\epsilon = 2r_{ij}$) show that both E_{AMBER} and log($\|\mathbf{g}_k\|/\|\mathbf{g}_0\|$ reach the minimum in a smaller number of iterations for the shortest minimization than for the longest one, in accord with the result for CPU(hybrid-longest)/CPU(hybrid-shortest) discussed above.

In Table 4 we present results for BPTI obtained with a dielectric constant $\epsilon = 1$. While most of the minimizations have found the same energy minimum (-1778.58 kcal/mol), in some cases lower minima were reached (the lowest is -1917.31 kcal/mol) and in two cases higher minima were obtained. In particular, the TN minimum is -1808.23 and therefore for comparison we include in the "longest minimizations" section of the table results for a hybrid method minimization that reached—1813.84 kcal/mol minimum—the closest to the TN minimum. Again, the shortest minimizations with respect to CPU time obtained by the hybrid method are shorter than those obtained by L-BFGS (a ratio of 941/709 = 1.33 for the smallest CPU results); however, a most significant advantage of the hybrid method is with respect to TN, where the large CPU ratio, 7673/1524 = 5.03, is obtained for TN versus the hybrid method minimization that led to -1813.84 kcal/mol. In other words, the advantage of the hybrid method over L-BFGS and TN is more pronounced for $\epsilon = 1$ than for $\epsilon = 2r_{ij}$.

Table 3. Minimization Results for BPTI Using a Distance-Dependent Dielectric Function, $\epsilon = 2r$.

maxcg or *m*	*l*	*t*	CPU (s)	No. functions/gradients
Hybrid method				
Shortest minimizations				
50	200	10	1157	4768
80	200	10	1163	4798
30	200	10	1163	4780
30	150	10	1164	4804
70	200	10	1178	4859
40	200	10	1181	4872
70	150	10	1191	4913
Longest minimizations				
5	1	10	1841	7613
70	30	20	1428	5909
80	50	20	1381	5700
80	100	20	1355	5599
60	30	20	1353	5582
60	50	30	1321	5448
60	100	50	1299	5371
L-BFGS				
5			1358	5619
6			1485	6144
TN-Nash			1375	5703

Initial energy $E = 207.43$ kcal/mol; final minimized energy $E = -358.03247$. For explanations, see Table 1. $\|\mathbf{x}_k\| = 856.1$ Å; $\|\mathbf{g}_k\| = 8.152 \times 10^{-3}$ kcal/(mol Å).

The average CPU ratio of the longest and shortest minimizations is around 1.4, where the ratio between the worst and best results is 1745/709 = 2.46; similar values are obtained for the corresponding ratios based on the numbers of function/gradient evaluations. As in Tables 1 and 2, for the loop all these ratios are larger than their counterparts obtained from Table 3 using $\epsilon = 2r_{ij}$, which is also in accord with the plots of Figures 1d and 2d for E_{AMBER} and $\log(\|\mathbf{g}_k\|/\|\mathbf{g}_0\|)$, where the difference in the total number of iterations of the shortest and longest minimizations is much larger than the corresponding difference in Figures 1c and 2c. It should be pointed out that the values of *maxcg* for the shortest minimizations are within the range 25–40 while those for the longest minimizations are significantly smaller, ranging from 4–10. The condition $l > t$ appears to be important for obtaining reasonable efficiency, where $l = 1$and $t = 10$ have led to the longest minimization (1642 s), which is significantly larger than the other results for the hybrid method.

It should be pointed out that the quality of a minimized structure can only be determined by comparing it to the experimental crystal structure. These structures are in general slightly different because application of the force field to the crystal structure and adding the hydrogens may result in atomic overlaps, which are removed during the energy minimization. Larger deviations are expected for $\epsilon = 1$ than $\epsilon = 2r_{ij}$ because of the stronger electrostatic interactions caused by $\epsilon = 1$. Indeed, we found that the root mean square deviations between the coordinates of the heavy atoms of the initial and final structures are 1.8 and 0.6 Å for the loop and 1.6 and 1.3 Å for BPTI for $\epsilon = 1$ and $\epsilon = 2r_{ij}$, respectively.

We also carried out minimizations for the loop described by the potential energy E_{tot} [eq. (2)], which includes the solvation energy calculated by SASA and the best-fit set of ASPs defined in ref. 48, using a relatively large tolerance of 10^{-3} [see eq. (4)]. The energy was decreased from -368.848 to -480.167 kcal/mol, but in this case the three methods were found to be approximately equally efficient (≈42-s CPU) and therefore the results are not provided.

In summary, for the two systems studied the hybrid method has been found to be more efficient than both L-BFGS and TN, where the improvement in computer time is by a factor of 1.3–2, and only for BPTI ($\epsilon = 1$) the minimization with TN required ≈5 times more computer time than the best minimization performed with the hybrid method. In particular, the performance of TN(Nash) decreases as the electrostatic interactions increase in going from $\epsilon = 2r_{ij}$ to $\epsilon = 1$, probably due to the increase in the ruggedness of the potential energy surface, which possibly also contains many saddle points. To check whether this inferior performance is a result of a poor preconditioner, or reflects a more general difficulty of TN methods, we carried out several minimizations for the loop with $\epsilon = 1$ using the hybrid method with $l = 0$, that is, applying pure HFN, which is expected to have a more advanced preconditioner

Table 4. Minimization Results for BPTI with a Dielectric Constant, $\epsilon = 1$.

maxcg or *m*	*l*	*t*	CPU (s)	No. functions/gradients
Hybrid method				
Shortest minimizations				
25	30	10	709	2835
30	30	20	743	2970
40	30	30	750	2996
30	100	10	760	3041
25	30	30	763	3049
30	30	13	755	3019
25	25	15	768	3069
25	35	10	780	3124
25	30	20	780	3120
Longest minimizations				
4	1	10	1642	6573
4	10	1	1108	4436
5	10	1	1031	4127
6	10	1	1031	4127
8	10	1	1051	4201
10	10	5	992	3974
7	5	10	1524*	
L-BFGS				
9			941	3761
7			960	3839
TN-Nash			7637*	29709

Initial energy $E = -831.2109$ kcal/mol; final minimized energy $E = -1778.58093$. For explanations, see Table 1. The TN minimum, -1808.23 kcal/mol, is lower than the minimum obtained in most of the other minimizations; therefore, for comparison, we include in the longest minimizations section results for a hybrid method minimization that reached a close minimum, -1813.84 kcal/mol. Above these results appears an asterisk. For the minimum -1778.58093, $\|\mathbf{x}_k\| = 860.5$ Å; $\|\mathbf{g}_k\| = 9.522 \times 10^{-3}$ (kcal/mol Å).

than that of TN(Nash).[45,59] However, the shortest minimization of 137 s (*maxcg* = 50) is close to the 140 s obtained with TN(Nash) whereas using *maxcg* = 5, 20, and 70 required 142, 156, and 161 s, respectively. To further investigate this problem it would be of interest to apply a TN method such as that provided by TNPACK where the preconditioning matrix can be tailored to the specific problem studied.[26,27,33,35,36]

The three methods have shown comparable efficiency for E_{tot}, which is based on surface area calculations. However, determination of the optimal values of the parameters *maxcg*, *l*, and, *t* is not straightforward: It appears to be problem dependent, hence requiring experimentation prior to the "production" minimizations. The present conclusions seem to be general and not tailored to the functions studied; thus, Alekseev and Navon[44] applied the hybrid method to a different set of functions, concluding that the optimal values of *maxcg*, *l*, and, *t* are system dependent.

Acknowledgments

I.M.N. acknowledges the support from NSF Grant ATM-0201808 managed by Dr. Linda Peng, whom he thanks for support.

References

1. Vásquez, M.; Némethy, G.; Scheraga, H. A. Chem Rev 1994, 94, 2183.
2. Gibson, K. D.; Scheraga, H. A. Physiol Chem Phys 1969, 1, 109.
3. Gō, N.; Scheraga, H. A. Macromolecules 1970, 3, 178.
4. Hagler, A. T.; Stern, P. S.; Sharon, R.; Becker, J. M.; Naider, F. J Am Chem Soc 1979, 101, 6842.
5. Karplus, M.; Kushik, J. N. Macromolecules 1981, 14, 325.
6. Meirovitch, H.; Meirovitch, E.; Lee, J. J Phys Chem 1995, 99, 4847.
7. Case, D. A. Curr Opin Struct Biol 1994, 4, 285.
8. Baysal, C.; Meirovitch, H. J Phys Chem A 1997, 101, 2185.
9. Baysal, C.; Meirovitch, H. J Am Chem Soc 1998, 120, 800.
10. Chang, G.; Guida, W. C.; Still, W. C. J Am Chem Soc 1989, 111, 4379.
11. Kolossváry, I.; Keseru, G. M. J Comput Chem 2001, 22, 21.
12. Lee, J.; Scheraga, H. A.; Rackovsky, S. J Comput Chem 1997, 18, 1222–1232.
13. Li, Z.; Scheraga, H. A. Proc Natl Acad Sci USA 1987, 84, 6611.
14. Pillardy, J.; Arnautova, Y. A.; Czaplewski, C.; Gibson, K. D.; Scheraga, H. A. Proc Natl Acad Sci USA 2001, 98, 12351.
15. Pillardy, J.; Czaplewski, C.; Liwo, A.; Lee, J.; Ripoll, D. R.; Kazmierkiewicz, R.; Oldziej, S.; Wedemeyer, W. J.; Gibson, K. D.; Arnautova, Y. A.; Saunders, J.; Ye, Y. J.; Scheraga, H. A. Proc Natl Acad Sci USA 2001, 98, 2329.
16. Saunders, M.; Houk, K. N.; Wu,Y.- D.; Still, W. C.; Lipton, M.; Chang, G.; Guida, W. C. J Am Chem Soc 1990, 112, 1419.
17. Saunders, M. J Comput Chem 1991, 12, 645.
18. Vásquez, M.; Scheraga, H. A. Biopolymers 1985, 24, 1437.
19. Liu, D. C.; Nocedal, J. Math Programming 1989, 45, 503.
20. Navon, I. M.; Zou, X.; Berger, M.; Phua, K. H.; Schlick, T.; LeDimet, F. X. In Optimization Technique and Applications, vol. 1; Phua, K. H., Ed.; World Scientific: Singapore, 1992; pp 33.
21. Navon, I. M.; Zou, X.; Berger, M.; Phua, K. H.; Schlick, T.; LeDimet, F. X. In Optimization Technique and Applications, vol. 1; Phua, K. H., Ed.; World Scientific: Singapore, 1992; pp 445.
22. Wang, Z.; Navon, I. M.; Zou, X.; LeDimet, F. X. Comput Opt Appl 1995, 4, 241.
23. Zou, X.; Navon, I. M.; Berger, M.; Phua, K. H.; Schlick, T.; LeDimet, F. X. SIAM J Opt 1993, 3, 582.
24. Dembo, R. S.; Eisenstat, S. C.; Steihaug, T. SIAM J Numer Anal 1982, 19, 400.
25. Nash, S. G. User's Guide for TN/TNBC: FORTRAN Routines for Nonlinear Optimization, Report 397, Mathematical Sciences Department, Johns Hopkins University: Baltimore, MD, 1984.
26. Schlick, T.; Fogelson, A. ACM Trans Math Software 1992, 18, 46.
27. Schlick, T.; Fogelson, A. ACM Trans Math Software 1992, 18, 71.
28. Navon, I. M.; Brown, F. B.; Robertson, D. H. Comput Chem 1990, 14, 305.
29. Robertson, D. H.; Brown, F. B.; Navon, I. M. J Chem Phys 1989, 90, 3221.
30. Wang, Z.; Droegemeier, K.; White, L. Comput Opt Appl 1998, 10, 283.
31. Nash, S. G.; Nocedal, J. SIAM J Opt 1991, 1, 358.
32. Schlick, T. Rev Comput Chem 1992, 3, 1.
33. Derreumaux, P.; Zhang, G.; Schlick, T.; Brooks, B. R. J Comput Chem 1994, 15, 532.
34. Brooks, B. R.; Bruccoleri, R. E.; Olafson, B. D.; Sates, D. J.; Swaminathan, S.; Karplus, M. J Comput Chem 1983, 4, 187.
35. Xie, D.; Schlick, T. SIAM J Opt 1999, 10, 132.
36. Xie, D.; Schlick, T. ACM Trans Math Software 1999, 25, 108.
37. Nocedal, J. J Acta Numer 1991, 199.
38. Baysal, C.; Meirovitch, H.; Navon, I. M. J Comput Chem 1999, 20, 354.
39. Shanno, D. F.; Phua, K. H. ACM Trans Math Software 1980, 6, 618.
40. van Gunsteren, W. F.; Berendsen, H. J. C. Biomos; Nijenborgh AG: Groningen, The Netherlands, 1987.
41. Axelsson, O.; Lindskog, G. Numer Math 1986, 48, 479.
42. Axelsson, O.; Lindskog, G. Numer Math 1986, 48, 499.
43. Morales, J. L.; Nocedal, J. Comput Opt Appl 2002, 21, 143.
44. Alekseev, A. K.; Navon, I. M. Comput Optim Applic, Submitted.
45. Morales, J. L.; Nocedal, J. SIAM J Optim 2000, 10, 1079.
46. Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R.; Merz, K. M., Jr.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A. J Am Chem Soc 1995, 117, 5179.
47. Ponder, J. W. TINKER, Version 3.7; Washington University: St. Louis, 1999.
48. Das, B.; Meirovitch, H. Proteins 2001, 43, 303.
49. Perrot, G.; Cheng, B.; Gibson, K. D.; Palmer, K. A.; Nayeem, A.; Maigret, B.; Scheraga, H. A. J Comput Chem 1992, 13, 1.
50. Connoly, M. L. J Appl Crystallogr 1983, 16, 548.
51. Richmond, T. J. J Mol Biol 1984, 178, 63.
52. Gill, P. E.; Murray, W. Report SOL 79-15; Department of Operation Research, Stanford University: Stanford, CA, 1979.
53. Shanno, D. F. Math Oper Res 1978, 3, 244.
54. Dennis, J. E., Jr.; More, J. J. SIAM Rev 1977, 19, 46.
55. Dennis, J. E., Jr.; Schnabel, R. B. Numerical Methods for Unconstrained Optimization and Nonlinear Equations; Prentice-Hall: Englewood Cliffs, NJ, 1983.
56. Nash, S. G. SIAM J Sci Statist Comput 1985, 6, 599.
57. Nash, S. G. SIAM J Numer Anal 1984, 21, 770.
58. Nash, S. G. SIAM J Sci Statist Comput 1985, 6, 599.
59. Morales, J. L.; Nocedal, J. Algorithm PREQN: Fortran subroutines for preconditioning the conjugate gradient method. Technical Report OTC 99/02; Optimization Technology Center, Northwestern University: Evanston, IL, 1999. Also in ACM Trans Math Software, 2001, 27, 83–91.