Remark on Algorithm 702—The Updated Truncated Newton Minimization Package

Dexuan Xie and Tamar Schlick
New York University

A truncated Newton minimization package, TNPACK, was described in ACM Transactions on Mathematical Software (Vol. 18, pp. 46–111) in 1992. Modifications to enhance performance, especially for large-scale minimization of molecular potential functions, are described here. They involve three program segments of TNPACK: negative curvature test, modified Cholesky factorization, and line-search stopping rule.

Categories and Subject Descriptors: G.1.6 [**Numerical Analysis**]: Optimization–nonlinear programming; G.4 [**Mathematics of Computing**]: Mathematical Software; J.2 [**Computer Applications**]: Life and Medical Sciences

General Terms: Algorithms, Performance

Additional Key Words and Phrases: truncated Newton method, molecular potential minimization, indefinite preconditioner, modified Cholesky factorization

1. INTRODUCTION

In 1992, Schlick and Fogelson [1992a, 1992b] described a FORTRAN package of subprograms for unconstrained minimization problems in *ACM TOMS* (Vol. 18, pp. 46–111) known as TNPACK. In 1994, TNPACK was adapted by Derreumaux et al. [1994] for the widely-used molecular mechanics and dynamics program CHARMM, and shown to be an efficient tool for the minimization of molecular potential functions in comparison to other available minimizers. Recently, we have examined practical issues of efficiency and implemented modifications to the TNPACK version of CHARMM to improve reliability and enhance convergence for large-scale complex nonlinear problems [Xie and Schlick 1999a; 1999b].

This note summarizes the modifications made in TNPACK and provides details regarding program usage. The basic changes involve the following program segments: (1) negative curvature test, (2) modified Cholesky (MC) factorization, and (3) line search algorithm. These modifications have been analyzed theoretically and numerically in recent works [Xie and Schlick 1999a; 1999b].

In Section 2 we introduce the algorithm and describe the modified negative curvature test and a strong negative curvature test. In Section 3 we describe our MC method, termed "UMC" for unconventional MC, as implemented in TNPACK. In

Address: Departments of Chemistry, Mathematics, and Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, (dexuan@cims.nyu.edu, schlick@nyu.edu). This work was supported in part by the National Science Foundation. T. Schlick is an investigator of the Howard Hughes Medical Institute. ACM Transactions on Mathematical Software, Vol. 25, 1999

Section 4, we present the more lenient stopping rule for the line search and a modification of the associated trial value. The reader should consult Schlick and Fogelson [1992a, 1992b] for general user instructions. Section 5 here only summarizes usage details related to our modifications. Appendix A presents the full algorithm of TNPACK, and appendix B includes some numerical examples that show the performance of the updated program. We welcome users to contact us for further information (dexuan@cims.nyu.edu and schlick@nyu.edu).

2. TERMINATION RULES FOR PCG

The truncated Newton (TN) method [Dembo and Steihaug 1983] solves the unconstrained minimization problem $E(X^*) = \min_{X\in\mathcal{D}} E(X)$, where $E(X)$ is a twice continuously differentiable real-valued function in an open set $\mathcal{D}$ of the n-dimensional vector space R^n. TN uses a nested sequence of iterations. A sequence of "outer" solution vectors $\{X^k\}$ can be expressed in the form

$$X^{k+1} = X^k + \lambda_k P^k, \quad k = 0, 1, 2, \ldots,$$

where P^k is a descent direction, λ_k is the steplength, and X^0 is an initial guess. The inner loop involves obtaining P^k by a "truncated" preconditioned conjugate gradient (PCG) loop; each λ_k is then generated by using a line search scheme ([Moré and Thuente 1994], for example). The search vector P^k is obtained from the PCG approximate solution of the Newton equations

$$H(X^k)P = -g(X^k), \tag{1}$$

where $g(X^k)$ and $H(X^k)$ are the gradient and Hessian, respectively, of the objective function E at X^k (often denoted as g_k and H_k for clarity). Important in practice are details regarding the efficient solution approximation of system (1), including handing indefiniteness and avoiding too many iterations when the quadratic model is poor.

Let p_j and d_j be the jth iterate and direction vectors of PCG, respectively. Since the Hessian matrix H_k may be indefinite, a negative curvature test is required to guarantee a descent direction for TN. The original package used the following Test 1A, proposed in Dembo and Steihaug [1983].

Test 1A *(Negative curvature test). If $d_j^T H_k d_j < \delta d_j^T d_j$, where δ is a given small positive number (e.g., $\delta = 10^{-10}$), and d_j is a vector generated in PCG (see algorithm in appendix A), exit with search direction $P^k = p_j$ or d_j (for $j = 1$, set $P^k = -g_k$).*

Since our analysis in Xie and Schlick [1999a] showed that d_j is a poorer choice than p_j as an exit search direction, this option has been removed, leading to the following modified Test 1A′.

Test 1A′ *(Modified Negative curvature test). If $d_j^T H_k d_j < \delta d_j^T d_j$, where δ is a given small positive number (e.g., $\delta = 10^{-10}$), exit with search direction $P^k = p_j$ (for $j = 1$, set $P^k = -g_k$).*

In Xie and Schlick [1999a], we proved that if $d_j^T H_k d_j > 0$ for $j = 1, 2, \ldots, l$, then all p_j with $2 \le j \le l+1$ are descent directions and satisfy

$$g_k^T p_{l+1} < \cdots < g_k^T p_j < g_k^T p_{j-1} < \cdots < g_k^T p_2 < 0. \tag{2}$$

In practice, however, (2) does not necessarily hold due to computer rounding errors. Hence, to guarantee a good descent direction in finite precision arithmetic, we also add the following Test 2A, called the "strong negative curvature test", as an alternative [Xie and Schlick 1999a].

Test 2A (Strong negative curvature test). If $g_k^T p_{j+1} > g_k^T p_j$, exit with search direction $P^k = p_j$ (for $j = 1$, set $P^k = -g_k$).

Theoretically, Test 2A is equivalent to Test 1A$'$, but in practice may lead to different performance. This is because Test 1A$'$ may not guarantee a descent direction or a "good" descent direction in the sense of (2) in finite precision arithmetic while Test 2A can do so. Numerical results and analyses [Xie and Schlick 1999a] show that Test 2A can improve the performance of TNPACK in comparison to Test 1A$'$. We let the user choose the test in an option list of input parameters to TNPACK, and leave Test 2A as the default.

The above tests halt the inner loop in case of negative curvature or non-monotonic improvements in the search vectors. In addition, TN methods also exit the inner loop when the residual of (1), in which H_k may be replaced by a related matrix, is sufficiently small. Test 1B below serves this purpose. It is the same as in the previous version.

Test 1B (Truncation test). Let r_j be the residual vector satisfying $r_j = -g_k - H_k p_j$. If $\|r_j\| \le \min\{c_r/k, \|g_k\|\}$, where c_r is a given positive number (e.g., $c_r = 0.5$), and $\|\cdot\|$ is the standard Euclidean norm divided by $\sqrt{n}$, exit with search direction $P^k = p_j$.

The inner PCG loop is also halted when the number of iterations at each inner loop exceeds IT_{PCG}, a maximum number of allowable PCG iterations for each inner loop. Our previously recommended value for IT_{PCG} was n, but we found that a smaller value, such as 40, works better when $n \gg 40$ [Xie and Schlick 1999a].

3. AN UNCONVENTIONAL MODIFIED CHOLESKY (UMC) FACTORIZATION

In TNPACK, a symmetric sparse approximation to the Hessian, namely the preconditioner M_k, is used to accelerate convergence of the inner CG loop. This matrix is not necessarily positive definite when chosen according to a physical subdivision of energy components, as in molecular applications [Schlick 1993]. Thus, a sparse modified Cholesky (MC) factorization based on the Yale Sparse Matrix Package [Gill and Murray 1974; Eisenstat et al. 1981; Eisenstat et al. 1982; Schlick and Fogelson 1992a] is used to construct a modified preconditioner $\widetilde{M_k}$, where $\widetilde{M_k} = M_k + E_k$, and E_k is a nonnegative diagonal matrix.

Our numerical experience with biomolecular-structure optimization showed [Xie and Schlick 1999a] that various MC schemes [Cheng and Higham 1998; Gill and Murray 1974; Gill et al. 1981; Schnabel and Eskow 1990] can produce very large modifications to M_k that can slow down the overall convergence of the method. To overcome this difficulty, we proposed the UMC [Xie and Schlick 1999a] summarized below.

Let L be a unit lower-triangular matrix and D and E diagonal matrices. We write these matrices as: $L = (l_{ij})_{n\times n}$, $D = \text{diag}(d_1, d_2, \cdots, d_n)$, and $E = \text{diag}(e_1, e_2, \cdots, e_n)$. For a given symmetric matrix $M = (m_{ij})_{n\times n}$ and a given parameter τ, UMC generates a factorization $LDL^T = \widetilde{M} = M + E$ with diagonals d_j $(j = 1, 2, \ldots, n)$ of D defined by

$$d_j = \begin{cases} \max\{\widetilde{d}_j, \theta^2/\beta^2\} & \text{when } \widetilde{d}_j > \delta, \\ \delta & \text{when } |\widetilde{d}_j| \le \delta, \\ \min\{\widetilde{d}_j, -\theta^2/\beta^2\} & \text{when } \widetilde{d}_j < -\delta, \end{cases}$$

where $\widetilde{d}_j = m_{jj} - \sum_{k=1}^{j-1} l_{jk}c_{jk} + \tau$, $c_{ij} = l_{ij}d_j$, $\theta = \max_{j+1\le i\le n} |c_{ij}|$, $\delta = \max(\varepsilon, \xi\varepsilon)$, ξ is the largest magnitude of an element of M, ε is a small positive number (we set $\varepsilon = 10^{-6}$), and $\beta^2 = \xi/\sqrt{n(n-1)}$, an *a priori* choice of β in our UMC.

Essentially, UMC produces in a numerically-stable procedure an LDL^T factorization for a matrix M that may be indefinite. Still, the PCG-generated vectors in our context are guaranteed to be directions of descent. Moreover, a user-prescribed parameter τ is set so that near a minimum the preconditioner resulting from UMC is positive definite. If $\tau > |\lambda_{min}(M)|$, where $\lambda_{min}(M)$ is the minimum eigenvalue of M, then $\widetilde{M}$ is positive definite with an error matrix $E = \widetilde{M} - M = \tau I$; otherwise, by similar arguments to those used in Gill and Murray [1974], a worst-case bound of $\|E\|_\infty$ can be derived [Xie and Schlick 1999a].

Note that UMC can produce an indefinite preconditioner $\widetilde{M}$ when $\tau < |\lambda_{min}(M)|$. With the use of singularity and negative curvature tests as shown in Appendix A, the recursive formulas of PCG are well defined for a non-singular preconditioner. Furthermore, the PCG-generated search vectors $\{P^k\}$ of TN are directions of descent (expression (2) above) even when the preconditioner $\widetilde{M_k}$ is indefinite. We found this overall procedure to work much better than that which allows excessively large modifications in MC [Xie and Schlick 1999a].

The choice of τ affects overall performance of the minimization. Our experience over many problems and system sizes suggests that $\tau = 10$ is a reasonable value [Xie and Schlick 1999a]. This is the default value in TNPACK, but users can set τ to other values suitable for their problems.

4. LINE SEARCH MODIFICATIONS

A line search scheme is an iterative algorithm for solving the one-dimensional minimization problem $\min\limits_{\lambda>0} f(\lambda)$, where

$$f(\lambda) = E(X^k + \lambda P^k), \quad \lambda > 0.$$

The line search stopping rule influences the overall efficiency of the minimization method. TNPACK uses the line search algorithm of [Moré and Thuente 1994], which relies on the following stopping rule.

Criterion 1 (Stopping rule for line search). *Terminate the line search when the steplength* $\lambda_k > 0$ *satisfies*

$$E(X^k + \lambda_k P^k) \leq E(X^k) + \alpha \lambda_k g(X^k)^T P^k, \tag{3}$$

and

$$|g(X^k + \lambda_k P^k)^T P^k| \leq \beta |g(X^k)^T P^k|, \tag{4}$$

where α *and* β *are given constants satisfying* $0 < \alpha < \beta < 1$.

Our more lenient stopping rule [Xie and Schlick 1999b] in TNPACK is the following.

Criterion 2 (More lenient stopping rule for line search). *Terminate the line search when the steplength* $\lambda_k > 0$ *satisfies either (3) and*

$$g(X^k + \lambda_k P^k)^T P^k \geq \beta g(X^k)^T P^k, \tag{5}$$

or, alternatively, conditions (3) and

$$g(X^k + \lambda_k P^k)^T P^k \leq (2 - \beta) g(X^k)^T P^k, \tag{6}$$

where α *and* β *are given constants satisfying* $0 < \alpha < \beta < 1$.

Condition (5) is a component of (4), which works only for a strictly convex function f. Condition (6) is introduced to produce a sufficiently large steplength when the function f is not strictly convex; for details, see Xie and Schlick [1999b].

Using the more lenient second criterion above can reduce the number of line search iterations per outer loop step in special cases. It also forces sufficient function decrease according to (3) and guarantees a sufficiently large steplength. Hence, we found it useful in practice for overall efficiency of a minimization procedure for large-scale multivariate functions whose function evaluations are expensive. Moreover, in theory, the global convergence for a descent method using Criterion 2 can be proven in the same way as for Criterion 1 [Dennis and Schnabel 1983]. However, in most of cases, condition (6) may not be encountered, so that performance is overall very similar to the first criterion [Xie and Schlick 1999b]. Thus, we leave Criterion 1 as the default choice in TNPACK and let the user select Criterion 2 if desired.

In addition, we have added a safeguard to the rule determining λ_k at each line search iteration. Instead of defining the line search iterate

$$\lambda^{(j+1)} = \lambda_c,$$

where λ_c is the minimum point of the cubic interpolant [Moré and Thuente 1994], we use:

$$\lambda^{(j+1)} = \max(\lambda_l + \sigma(\lambda_t - \lambda_l), \lambda_c),$$

where σ is a small positive number such as 0.001, and λ_l and λ_t are the two endpoints of the interval generated by the line search algorithm [Moré and Thuente 1994]. This modification avoids ending the line search with a λ_c value that is very small, a case we have encountered in practice. See Xie and Schlick [1999b] for further details.

Table I. The Four New User-Specified Parameters

Parameters	Function
OPLIST(15)	specifies the choice of PCG termination tests in the case of negative curvature: 1: use Test 1A$'$ 2: use Test 2A *The default value is 1*
OPLIST(16)	specifies the choice of line search stopping rules 1: use Criterion 1 2: use Criterion 2 *The default value is 1*
OPLIST(17)	specifies the choice of the MC methods 1: use the standard MC 2: use UMC *The default value is 1*
PLIST(8)	specifies the parameter τ of UMC. The default value is 10.0

5. SOME USAGE CHANGES ON TNPACK

The usage details for the new version of TNPACK are the same as described in Schlick and Fogelson [1992a] except for small changes involving the two work arrays OPLIST and PLIST.

The arrays OPLIST and PLIST (of dimension 20) are required in TNPACK to specify options and parameters for minimization. Only 14 entries of OPLIST and 7 of PLIST have been used in the original TNPACK, respectively; see Schlick and Fogelson [1992a]. In particular, entry OPLIST(12) specifies the choice of exit directions in the case of negative curvature for PCG, with five options provided in the original program. Since the PCG direction d_j defines a poor search direction for TN and the preconditioner M_k may be indefinite, we only use the option OPLIST(12) = 1 (i.e., Test 1A$'$) now; the other four options have been deleted. As a result, the user's input value for OPLIST(12) does not matter.

We also introduced four new user-specified parameters, OPLIST(15), OPLIST(16), OPLIST(17), and PLIST(8) to specify the PCG termination test, the line search stopping rule, the modified Cholesky factorization, and UMC threshold parameters, respectively (see Table I).

Like other parameters, sample values are produced in subroutine SETLIS, and they can be modified in the driver interface between the SETLIS and TNMIN calls.

APPENDIX

A. THE TRUNCATED NEWTON ALGORITHM

The new algorithmic components are marked by asterisks. These include the line search scheme, UMC factorization, and negative curvature test (steps (2)–(4) of the inner loop). Default parameter settings are indicated. The indices k and i are used respectively to denote outer loop and inner loop iterates.

Outer Loop of the Truncated Newton Method

(1) *Initialization*

—Set $k = 0$ and evaluate $E(X^0)$ and $g(X^0)$ corresponding to initial guess X^0, where E is the objective function to be minimized,

—If $||g(X^0)|| < 10^{-8} \max(1, ||X^0||)$, where $||\cdot||$ is the standard Euclidean norm divided by $\sqrt{n}$, exit algorithm; otherwise, continue to step 2.

(2) *Preparation for UMC*

—Evaluate the preconditioning matrix M_k.

—Determine the sparsity pattern of M_k. The upper triangle of M_k is stored in a compressed row format, and the pattern is specified by two integer arrays that serve as row and column pointers [Schlick and Fogelson 1992a].

—Compute the *symbolic factorization* LDL^T of M_k, that is, the sparsity structure of the factor L.

—Evaluate the Hessian matrix H_k.

(3) Inner Loop

Compute search vector P^k by the linear preconditioned conjugate gradient method for solving the Newton equations $H_k P = -g(X^k)$ *approximately* based on the UMC factorization of M_k (see below).

(4)* *Line search*

—Compute a steplength λ by safeguarded cubic and quadratic interpolation [Moré and Thuente 1994; Xie and Schlick 1999b] so that $X^{k+1} = X^k + \lambda P^k$ satisfies either (A1) and (A2a) or (A1) and (A2b), where:

$$E(X^{k+1}) \leq E(X^k) + \alpha \lambda g(X^k)^T P^k \ , \tag{A1}$$

$$g(X^{k+1})^T P^k \geq \beta g(X^k)^T P^k \ , \tag{A2a}$$

$$g(X^{k+1})^T P^k \leq (2 - \beta) g(X^k)^T P^k \tag{A2b}$$

with $\alpha = 10^{-4}$ and $\beta = 0.9$.

(5) *Convergence tests*

—Check the following inequalities:

$$E(X^{k+1}) - E(X^k) < \epsilon_f (1 + |E(X^{k+1})|) \ , \tag{A3a}$$

$$||X^{k+1} - X^k|| < \sqrt{\epsilon_f}\ (1 + ||X^{k+1}||)/100 \ , \tag{A3b}$$

$$||g(X^{k+1})|| < \epsilon_f^{1/3} (1 + |E(X^{k+1})|) \ , \tag{A3c}$$

$$||g(X^{k+1})|| < \epsilon_g (1 + |E(X^{k+1})|) \ , \tag{A3d}$$

where $\epsilon_f = 10^{-10}$ and $\epsilon_g = 10^{-8}$.

—If conditions {A3a,A3b,A3c} or (A3d) are satisfied, *exit* algorithm.

(6) *Preparation for the next Newton step*

—Compute the new preconditioner M_{k+1} by using the pattern determined originally.
—Evaluate the new Hessian matrix H_{k+1}.
—Set $k \leftarrow k+1$ and go to step 3.

Inner Loop k of the Truncated Newton Method (Step 3 above)

The sequence $\{p_i\}$ below represents the PCG vectors used to construct P^k in step 3 of Outer loop. We omit the Newton subscript k from g, H, and M and the superscript k from P for clarity. In step (4)*, either Test 2A or Test 1A′ can be used.

(1) *Initialization*
—Set $i = 1$, $p_1 = 0$, and $r_1 = -g$.
—Set the parameter η_k controlling the accuracy of the computed search vector: $\eta_k = \min\{c_r/k \ , \|g\|\}$, where $c_r \leq 1$ (we use $c_r = 0.5$).
—Set the parameter IT_{PCG} as the maximum number of allowable PCG iterations for each inner loop (default $IT_{PCG} = 40$).

(2)* *The unconventional modified Cholesky (UMC) factorization*
—Perform the *numerical factorization* of M by UMC with a chosen parameter τ (default $\tau = 10$). The resulting modification $\widetilde{M}$ of M is used as the preconditioner of PCG, and is stored in the same sparse row format used for M.
—Solve for z_i in $\widetilde{M} z_i = r_i$.
—Set $d_i = z_i$.

(3)* *Singularity test*
—Compute the matrix/vector product $q_i = H d_i$.
—If $|r_i^T z_i| \leq \zeta(r_i^T r_i)$ or $|d_i^T q_i| \leq \zeta$ (e.g., $\zeta = 10^{-15}$),
exit PCG loop with $P = p_i$ (for $i = 1$, set $P = -g$).

(4)* *Negative curvature test*: implement one of the following two tests.
—Test 1A′:
If $d_i^T q_i \leq \delta(d_i^T d_i)$ (e.g., $\delta = 10^{-10}$),
exit inner loop with $P = p_i$ (for $i = 1$, set $P = -g$);
else update α_i and p_{i+1} as in (A4), and continue to step 5.
—Test 2A: Update the quantities

$$\alpha_i = r_i^T z_i \,/\, d_i^T q_i, \quad \text{and} \quad p_{i+1} = p_i + \alpha_i d_i. \tag{A4}$$

If $g^T p_{i+1} \geq g^T p_i - \zeta$,
exit inner loop with $P = p_i$ (for $i = 1$, set $P = -g$);
else continue to step 5.

(5) *Truncation test*
—Compute $r_{i+1} = r_i - \alpha_i q_i$.
—If $\|r_{i+1}\| \leq \eta_k \|g\|$ or $i + 1 > IT_{PCG}$,
exit inner loop with search direction $P = p_{i+1}$;
else continue to step 6.

(6) *Continuation of PCG*
—Solve for z_{i+1} in $\widetilde{M} z_{i+1} = r_{i+1}$.

—Update the quantities

$$\beta_i = r_{i+1}^T z_{i+1} \,/\, r_i^T z_i, \text{ and } d_{i+1} = z_{i+1} + \beta_i d_i$$

—Set $i \leftarrow i + 1$, and go to step 3.

B. IMPLEMENTATION EXAMPLES

Three implementation examples are included with the TNPACK software to illustrate the performance to users. The objective functions are familiar optimization test problems from Algorithm 566 by Moré, Garbow, and Hillstrom [1981a, 1981b]. The second-derivative routines for the 18 test functions of Alg. 566 can be obtained from the supplementary Hessian package, HESFCN, developed by Averbukh, Figueroa, and Schlick [1992, 1994]. We used all default starting points specified in Alg. 566 for the 18 test functions. The preconditioner for PCG was chosen as the diagonal of the Hessian matrix H_k for all problems.

Included among the 18 test functions are the extended Rosenbrock function with an even integer n

$$f(X) = \sum_{j=1,3,5,\ldots,n-1} \left\{(1 - x_j)^2 + 100(x_{j+1} - x_j^2)^2\right\},$$

and the trigonometric function

$$f(X) = \sum_{j=1}^{n} \left(n - \sum_{i=1}^{n} \cos x_i + j(1 - \cos x_j) - \sin x_j \right)^2 .$$

Detailed descriptions of TNPACK for these two functions have given in Schlick and Fogelson [1992b]. We include them as two separate implementation examples with the TNPACK code to let users easily test TNPACK options. We used the following starting points X^0 for the Rosenbrock and trigonometric functions

$$X^0 = (-1.2-\cos 1, 1+\cos 1, -1.2-\cos 3, 1+\cos 3, \ldots, -1.2-\cos(n-1), 1+\cos(n-1))$$

and

$$X^0 = (1/n + 0.2\cos 1, 1/n + 0.2\cos 2, 1/n + 0.2\cos 3, \ldots, 1/n + 0.2\cos n),$$

respectively. Note that they are different from the default starting points given in Alg. 566 for these two functions. To test non-diagonal preconditioners and the reordering option of TNPACK, we defined the following preconditioner M_k for the trigonometric function: elements m_{ii} are set to the ith diagonal element of H_k for $1 \le i \le n$; off-diagonals $m_{1,n-1} = m_{n-1,1} = 0.1$, $m_{1,n} = m_{n,1} = -0.1$, and other elements are zero. We set $\tau = 0.5$ for the trigonometric function for better performance while the default value of τ was used for Rosenbrock function.

We also include below numerical examples for large-scale molecular potential functions. We consider two molecular systems: alanine dipeptide (22 atoms) and the protein BPTI (568 atoms). The widely-used molecular mechanics and dynamics program CHARMM [Brooks et al. 1983] (version 24b1) was used to compute the potential energy functions and their derivatives; parameter files for the energy

Table II. TNPACK Performance for Rosenbrock and Trigonometric Functions

TNPACK	Final Energy	Final $\|\|g\|\|$	TN (PCG) Itns.	E Evals.	CPU Time
Rosenbrock function ($n = 1000$)					
Modified	4.3512×10^{-18}	2.82×10^{-9}	28 (500)	45	0.208 sec.
Original	7.3024×10^{-24}	2.52×10^{-12}	50 (744)	117	0.302
Trigonometric function ($n = 1000$)					
Modified	1.1215×10^{-13}	9.43×10^{-9}	21 (73)	23	8.045 sec.
Original	1.3833×10^{-17}	1.06×10^{-10}	27 (74)	41	8.488

Table III. TNPACK Performance for the 18 Test Functions of Algorithm 566

Prob	TNPACK	Final Energy	Final $\|\|g\|\|$	TN (PCG) Itns.	E Evals.
1	Modified	1.7884×10^{-19}	3.35×10^{-9}	16 (41)	19
	Original	2.8711×10^{-32}	6.67×10^{-16}	17 (40)	23
2	Modified	3.2182×10^{-14}	1.22×10^{-9}	271 (948)	295
	Original	2.4268×10^{-1}	3.05×10^{-7}	1274 (6781)	2963
3	Modified	1.1279×10^{-8}	6.74×10^{-9}	2 (3)	3
	Original	1.1279×10^{-8}	5.60×10^{-11}	2 (4)	3
4	Modified	7.6372×10^{-6}	1.28×10^{-5}	36 (53)	52
	Original	2.9769×10^{-7}	1.28×10^{-7}	107 (182)	167
5	Modified	5.6077×10^{-13}	1.85×10^{-8}	14 (29)	20
	Original	1.0454×10^{-18}	3.43×10^{-10}	16 (34)	20
6	Modified	3.2357×10^{-22}	8.04×10^{-11}	9 (14)	10
	Original	1.9639×10^{-18}	6.27×10^{-9}	9 (14)	10
7	Modified	4.7140×10^{-1}	1.19×10^{-9}	9 (16)	10
	Original	4.7140×10^{-1}	7.52×10^{-15}	9 (19)	10
8	Modified	1.5179×10^{-5}	3.49×10^{-8}	45 (101)	56
	Original	1.5179×10^{-5}	3.43×10^{-9}	52 (96)	64
9	Modified	3.200×10^{-6}	4.46×10^{-8}	9 (17)	13
	Original	3.1981×10^{-6}	1.85×10^{-10}	35 (95)	44
10	Modified	1.9722×10^{-31}	6.28×10^{-10}	4 (5)	14
	Original	5.4210×10^{-20}	7.09×10^{-10}	4 (5)	10
11	Modified	8.5822×10^{4}	8.96×10^{-3}	10 (27)	11
	Original	8.5822×10^{4}	7.22×10^{-3}	10 (27)	11
12	Modified	7.9990×10^{-11}	1.40×10^{-7}	29 (53)	39
	Original	Line search failed at 3rd TN after 30 line search iterations			
13	Modified	2.5737×10^{-3}	1.10×10^{-12}	9 (24)	11
	Original	2.5737×10^{-3}	7.79×10^{-9}	8 (21)	11
14	Modified	1.3433×10^{-20}	9.08×10^{-11}	28 (49)	34
	Original	1.4800×10^{-25}	2.97×10^{-13}	33 (57)	43
15	Modified	1.4061×10^{-12}	3.34×10^{-9}	22 (80)	23
	Original	7.3082×10^{-13}	3.13×10^{-9}	21 (75)	22
16	Modified	2.0461×10^{-21}	5.97×10^{-11}	9 (14)	11
	Original	7.9394×10^{-24}	2.37×10^{-12}	9 (16)	12
17	Modified	1.5576×10^{-19}	1.11×10^{-9}	94 (341)	100
	Original	4.3014×10^{-23}	1.43×10^{-10}	52 (175)	64
18	Modified	3.3521×10^{-25}	1.33×10^{-12}	7 (11)	9
	Original	2.9041×10^{-17}	9.18×10^{-9}	6 (9)	11

Table IV. Performance of TNPACK for Alanine Dipeptide (66 Variables) Minimization Using the Standard Modified Cholesky Factorization (MC) versus Our UMC with $\tau = 10$

	Final E	Final $\|\|g\|\|$	TN (PCG) Itns.	E Evals	CPU Time
MC	−15.245	3.27×10^{-10}	2386 (2738)	5939	9.07 sec.
UMC	−15.245	1.31×10^{-10}	30 (227)	45	0.92

Table V. Comparison of TNPACK with Other Two CHARMM Minimizers and LM-BFGS for BPTI (1704 Variables)

Minimizer	Itns. (PCG Itns.)	Final E	Final $\|\|g\|\|$	E Evals	CPU Time
TNPACK	78 (1604)	−2776.58	1.14×10^{-6}	227	5.75 min.
LM-BFGS	4486	−2792.96	6.3×10^{-5}	4622	12.61
ABNR	8329	−2792.96	8.9×10^{-6}	8330	25.17
CONJ	12469	−2792.93	9.9×10^{-6}	32661	97.8

Table VI. Performance of TNPACK for BPTI Using Criterion 1 (C1) vs. Criterion 2 (C2) in the Line Search

Criterion	Final E	Final $\|\|g\|\|$	TN (PCG) Itns.	E Evals.	CPU Time
C1	−2762.5	3.72×10^{-6}	176 (1387)	537	6.99 min.
C2	−2749.2	7.14×10^{-6}	85 (1029)	250	4.35

functions were used from CHARMM version 19. All nonbonded interactions of the potential energy function were considered, and a distance-dependent dielectric function was used. An effective preconditioner M_k is constructed naturally from the local chemical interactions: bond length, bond angle, and torsional potentials [Schlick and Overton 1987]. More numerical results and discussions can be found in Xie and Schlick [1999a, 1999b], where a CHARMM version of TNPACK was used, and some changes have been made for TNPACK for better performance.

All computations were performed in double precision in serial mode on an SGI Power Challenge L computer with R10000 processors of speed 195 MHZ at New York University. The vector norm $\|\cdot\|$ in tables is the standard Euclidean norm divided by $\sqrt{n}$.

Tables 5 and III display minimization performance of the updated TNPACK using the new options for these test functions, and reports for reference results with the previous TNPACK version using default options. Note that the default values of n in Alg. 566 are small. For example, $n = 3$ for trigonometric function (Problems 13 in Table III) and $n = 2$ for the Rosenbrock function (Problem 14) of Alg. 566. Hence, the CPU times for running TNPACK for the 18 functions of Alg. 566 on the R10000 processor were very small (fraction of a second).

From these two tables we see that the new options perform overall slightly better for all test functions, except for function 17, Wood function. Significant improvements are noted for the second function (Biggs function) and the 12th function (Gulf research and development function). For the second function, a zero value for the minimum is expected, but the older TNPACK version found a larger value. The older TNPACK version failed for function 12 due to rounding errors in the third truncated Newton iterate (after 30 line search iterations) using Criterion 1.

Table VII. A Summary of TNPACK Results

Reference	Test Problems (n)	Main Findings
Schlick and Fogelson [1992b]	(1) Rosenbrock's function (1,000) (2) Trigonometric function (1,000) (3) Potential energy function of deoxycytidine (87) (4) Energy function from models of platelet aggregation (640)	Various TNPACK options and their influence on performance are illustrated.
Zou et al. [1993]	(1) Oceanography problems (7330) (2) Meteorology problems (14,763)	Truncated Newton methods are competitive with L-BFGS, especially for large-scale meteorological problems.
Derreumaux et al. [1994]	Potential energy functions of (1) Alanine Dipeptide (36) (2) N-Methyl-Acetamide (36) (3) Deca-Alanine (198) (4) Mellitin (765) (5) Rubredoxin (1,413) (6) Avian (1,104) (7) A dimer of Insulin (2,892) (8) BPTI (1,740) (9) Lysozyme (3,897)	TNPACK performs significantly better than ABNR when Hessian/vector products are approximated by a finite-difference expression of gradients. Curvature information is important for directing minimization progress, and the use of local structure accelerates convergence of PCG. Very low final gradient norms can be obtained.
Xie and Schlick [1999a]	Potential energy functions of (1) Butane (42) (2) Alanine Dipeptide (66) (3) BPTI (1,704) (4) Lysozyme (6,090)	The UMC and other modifications improve the performance of TNPACK for large-scale molecular systems.

Table 5 compares the performance of TNPACK based on the standard MC [Gill and Murray 1974] to our UMC for alanine dipeptide minimization. Since the preconditioner M_k was indefinite while far away from a local minimum, the standard MC generated an excessively large modification to M_k such that the modified matrix $\widetilde{M}_k$ was a very poor approximation of the Hessian matrix H_k and had a very large condition number. Consequently, TNPACK with the standard MC led to poor performance in comparison to the TNPACK with UMC. We also found that TNPACK with the standard MC did not work for large-scale molecular minimization problems such as BPTI.

Table 5 compares the performance of TNPACK for BPTI minimizations with two other CHARMM minimizers, ABNR (an adopted-basis Newton-Raphson method) and CONJ (a nonlinear conjugate gradient method), as well as LM-BFGS with $u = 5$ stored updates [Liu and Nocedal 1989]. For simplicity, we used the default parameters in CHARMM for ABNR, CONJ and TNPACK. The results in Table 5 show that TNPACK requires less CPU time than the other methods and reaches very low gradient norms. LM-BFGS is the next minimizer in efficiency, but the CPU time of TNPACK is less than that of LM-BFGS by a factor of two or more.

To show that a significant improvement can be observed when Criterion 2 (C2) of the line search is used, we consider TNPACK for BPTI potential function minimization. Given that the function has many local minima, and different starting position vectors may lead to different minima, we constructed a starting-position vector as below:

$$X^{0,\omega} = X^0 + \Upsilon,$$

where X^0 is the original starting position vector (from an experimental structure), and Υ is a random vector, each component of which is chosen from a uniform distribution between 0 and 1. An initial seed value of one is chosen for the pseudorandom number generator.

Table 5 shows that C2 requires less CPU time to find a minimum than C1.

Finally, Table 5 gives a summary of TNPACK results that appeared in the following four papers: [Schlick and Fogelson 1992b; Zou et al. 1993; Derreumaux et al. 1994; Xie and Schlick 1999a].

REFERENCES

AVERBUKH, V. Z., FIGUEROA, S., AND SCHLICK, T. 1992. *HESFCN — A FORTRAN Package of Hessian Subroutines for Testing Nonlinear Optimization Software.* Technical Report 610, Courant Institute of Math. Sciences, New York University.

AVERBUKH, V. Z., FIGUEROA, S., AND SCHLICK, T. 1994. Remark on algorithm 566. *ACM Trans. Math. Softw. 20*, 3(Sept.), 282–285.

BROOKS, B. R., BRUCCOLERI, R. E., OLAFSON, B. D., STATES, D. J., SWAMINATHAN, S., AND KARPLUS, M. 1983. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comp. Chem. 4*, 187–217.

CHENG, S. H. AND HIGHAM, N. J. 1998. A modified Cholesky algorithm based on a symmetric indefinite factorization. *SIAM J. on Matrix Analysis and Applications 19*, 4, 1097–1110.

DEMBO, R. S. AND STEIHAUG, T. 1983. Truncated-Newton algorithms for large-scale unconstrained optimization. *Math. Prog. 26*, 190–212.

DENNIS, J. E. AND SCHNABEL, R. B. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

DERREUMAUX, P., ZHANG, G., BROOKS, B., AND SCHLICK, T. 1994. A truncated-Newton method adapted for CHARMM and biomolecular applications. *J. Comp. Chem. 15*, 5(May), 532–552.

EISENSTAT, S. C., GURSKY, M. C., SCHULTZ, M. H., AND SHERMAN, A. H. 1982. Yale sparse matrix package, I. The symmetric codes. *Int. J. Numer. Meth. Eng. 18*, 1145–1151.

EISENSTAT, S. C., SCHULTZ, M. H., AND SHERMAN, A. H. 1981. Algorithms and data structures for sparse symmetric Gaussian elimination. *SIAM J. Sci. Statist. Comput. 2*, 225–237.

GILL, P. E. AND MURRAY, W. 1974. Newton-type methods for unconstrained and linearly constrained optimization. *Math. Prog. 28*, 311–350.

GILL, P. E., MURRAY, W., AND WRIGHT, M. H. 1981. *Practical optimization.* Academic Press, London, England.

LIU, D. C. AND NOCEDAL, J. 1989. On the limited memory bfgs method for large-scale optimization. *Math. Prog. 45*, 3(Dec.), 503–528.

MORÉ, J., GARBOW, B. S., AND HILLSTROM, K. E. 1981. Testing unconstrained optimization software. *ACM Transactions on Math. Software. 7*, 17–41.

MORÉ, J., GARBOW, B. S., AND HILLSTROM, K. E. 1981. Algorithm 566: Fortran subroutines for testing unconstrained optimization software. *ACM Transactions on Math. Software. 7*, 136–140.

MORÉ, J. J. AND THUENTE, D. J. 1994. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Software 20*, 3(Sept.), 286–307.

SCHLICK, T. 1993. Modified Cholesky factorizations for sparse preconditioners. *SIAM J. Sci. Comp. 14*, 424–445.

Schlick, T. and Fogelson, A. 1992. TNPACK — A truncated Newton minimization package for large-scale problems: I. Algorithm and usage. *ACM Trans. Math. Softw. 14*, 46–70.

Schlick, T. and Fogelson, A. 1992. TNPACK — A truncated Newton minimization package for large-scale problems: II. Implementation examples. *ACM Trans. Math. Softw. 14*, 71–111.

Schlick, T. and Overton, M. L. 1987. A powerful truncated Newton method for potential energy functions. *J. Comp. Chem. 8*, 1025–1039.

Schnabel, R. B. and Eskow, E. 1990. A new modified Cholesky factorization. *SIAM J. Sci. Statist. Comput. 11*, 1136–1158.

Xie, D. and Schlick, T. 1999. Efficient implementation of the truncated-Newton algorithm for large-scale chemistry applications. *SIAM J. Optim.*, *9*.

Xie, D. and Schlick, T. 1999. A more lenient stopping rule for line search algorithms. Preprint.

Zou, X., Navon, I. M., Berger, M., Phua, K. H., Schlick, T., and Le Dimet, F. X. 1993. Numerical experience with limited-memory quasi-Newton and truncated Newton methods. *SIAM J. Optim. 3*, 582–608.