Optimization Methods and Software, Vol. 00, pp. 1–18

Taylor & Francis
Taylor & Francis Group

A MORE LENIENT STOPPING RULE FOR LINE SEARCH ALGORITHMS

DEXUAN XIE[a,*] and TAMAR SCHLICK[b]

[a]*Department of Mathematics and Graduate Program in Scientific Computing, University of Southern Mississippi, Hattiesburg, MS 39406-5045;*
[b]*Departments of Chemistry, Mathematics, and Computer Science, Courant Institute of Mathematical Sciences and the Howard Hughes Medical Institute, New York University, 251 Mercer Street, New York, NY 10012*

(Received 15 October 1998; In final form 12 January 2000)

An iterative univariate minimizer (line search) is often used to generate a steplength in each step of a descent method for minimizing a multivariate function. The line search performance strongly depends on the choice of the stopping rule enforced. This termination criterion and other algorithmic details also affect the overall efficiency of the multivariate minimization procedure. Here we propose a more lenient stopping rule for the line search that is suitable for objective univariate functions that are not necessarily convex in the bracketed search interval. We also describe a remedy to special cases where the minimum point of the cubic interpolant constructed in each line search iteration is very close to zero. Results in the context of the truncated Newton package TNPACK for 18 standard test functions, as well as molecular potential functions, show that these strategies can lead to modest performance improvements in general, and significant improvements in special cases.

Keywords: Line search; Descent method; Truncated Newton; Molecular potential minimization

1 INTRODUCTION

Descent methods provide a general framework for constructing globally convergent algorithms for minimizing real functions E on a subdomain $\mathcal{D}$ of the n-dimensional Euclidean space R^n. For a given

*Corresponding author. Tel.: (601) 266-5459. Fax: (601) 266-5818.
E-mail: Dexuan.Xie@usm.edu

point $X^k \in \mathcal{D}$, a descent method generates a search direction P^k and a steplength λ_k, and then defines the updated iterate X^{k+1} in the form

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

where X^0 is a starting point, and P^k is a method-specific search vector.

The search direction P^k is often chosen to satisfy the descent condition

$$g(X^k)^T P^k < 0, \tag{2}$$

where $g(X^k)$ is the gradient of E at X^k, and the superscript T denotes a vector or matrix transpose. A vector satisfying (2) is a "descent direction" because it implies that there exists a steplength λ_k such that

$$E(X^k + \lambda_k P^k) \leq E(X^k). \tag{3}$$

Since many λ_k may satisfy (3), line search schemes are employed at each iteration of the minimization method (1) to select a reasonable value of λ_k that will lead to an efficient descent method.

Such line search schemes are iterative, one-dimensional minimization algorithms for solving:

$$\min_{\lambda > 0} f(\lambda), \tag{4}$$

where

$$f(\lambda) = E(X^k + \lambda P^k), \quad \lambda > 0. \tag{5}$$

Safeguarded polynomial interpolation is typically used to compute an approximate solution of (4). For example, a cubic interpolant is constructed at each step using function and derivative values of f at the two endpoints of the feasible λ interval. This interval is modified at each line search iteration to bracket the minimum until certain termination criteria are met.

The prevailing belief up through the mid 1960s was that λ_k should be chosen as an *exact* solution of (4). Further computing experience has shown, however, that choosing λ_k as an *approximate* solution of (4) can lead to a more efficient descent method. Indeed, while an exact

search can reduce E further in one particular step, the additional cost involved (more function, and possibly gradient, evaluations) may not be overall profitable. This tradeoff between the work involved in computing λ_k at each step and the overall efficiency of the descent method must be carefully balanced. When function evaluations are expensive (e.g., potential functions for large biomolecules), cruder line searches may be preferred at each step.

In this article, we propose a more lenient stopping rule for the line search (termed C2) that is suitable for objective univariate functions that are not necessarily convex in the bracketed search interval. Two well-known stopping (or convergence) rules termed C1 and C1′ in this article (i.e., the strong Wolfe condition and the Wolfe condition [12]) consist of two conditions that are simultaneously satisfied at bracketed search intervals on which the objective univariate function is usually convex. Our amendment C2 halts the line search process when these two standard conditions are satisfied *or* when two others are satisfied at bracketed search intervals on which the function is not strictly convex; C2 thus includes two condition *sets*. The convergence of the overall descent method using C2 follows directly by the same procedure used for C1 or C1′. We have also incorporated a minimum value for λ into the acceptable steplength setting as often suggested. We have found this restraint specification to work well in many molecular applications where the cubic interpolant constructed in the line search algorithm has a minimum close to zero. We also show in the appendix that a unit steplength is a good line search starting point in truncated-Newton type methods.

We present numerical results on two nonconvex functions [11], potential energies of two proteins modeled by the molecular mechanics program CHARMM [3] (started from various points), and 18 test functions from Algorithm 566 [8] to demonstrate the performance of these line search modifications. We use the line search algorithm developed by Moré and Thuente [11], modified here as proposed. As a minimization algorithm for the multivariate objective function, we use the truncated-Newton package TNPACK [5,13,17].

Results show that our modifications can slightly reduce the number of line search iterations (or function evaluations) and hence the required CPU times. The reduction is typically small because the line search algorithm tends to generate an acceptable steplength λ_k near a

+ [16.7.2002–12:14pm] [1–18] [Page No. 3] FIRST PROOFS i:/T&F Gordon & Breach/Goms/GOMS-21024.3d Optimization Methods and Software (GOMS) Paper: GOMS-21024 Keyword

minimum point of f such that f is often convex near iterate X^k at most descent steps. Still, we found special cases in molecular potential energy minimization where the reductions in computational times are more significant. It thus appears that using C2 is no worse than using C1 and, in very special cases, C2 works better. Clearly, the resulting minimum function value of f might be higher in any one iteration of the multivariate minimization problem when C2 is used instead of C1, and the consequence of this effect must be examined in the scope of the overall progress to the minimum of the objective function for the given application.

In Section 2, we present and discuss stopping rules for line search algorithms. In Section 3, we discuss the case where the cubic interpolant has a minimum point very close to zero. Sections 4 and 5 present numerical results using C2. In the Appendix, we prove that the unit steplength is a good starting point for the line search in the context of truncated-Newton methods.

2 A MORE LENIENT STOPPING RULE FOR LINE SEARCH METHODS

Stopping or convergence rules for line search schemes have been developed on the basis of convergence theory for descent methods [6]. They usually consist of two conditions: the first enforces sufficient decrease in E, and the second avoids excessively small steplength λ_k. Two different formulations of the second condition have led to two well-known stopping rules. One is the following Condition-set 1 (C1) [9,11]:

C1 (STOPPING RULE FOR LINE SEARCH). The iterative line search process is terminated if the steplength $\lambda_k > 0$ satisfies

$$E(X^k + \lambda_k P^k) \leq E(X^k) + \alpha \lambda_k g(X^k)^T P^k \tag{6}$$

and

$$\left| g(X^k + \lambda_k P^k)^T P^k \right| \leq \beta \left| g(X^k)^T P^k \right|, \tag{7}$$

where α and β are given constants satisfying $0 < \alpha < \beta < 1$ (typically $\alpha = 0.1$ and $\beta = 0.9$).

Using condition (6) in combination with

$$g(X^k + \lambda_k P^k)^T P^k \geq \beta g(X^k)^T P^k \tag{8}$$

instead of (7) gives the other well-known convergence rule for line search termination (termed C1′ here). This is a weaker combination than C1 because a steplength satisfying (7) satisfies (8), but the converse is not true. C1′ is based on the work of Armijo [1] and Goldstein [10], and is often referred to as the Wolfe conditions while C1 is referred to as the strong Wolfe conditions [12].

Condition (6) follows from (2) and the definition of the derivative of f, where f is given in (5). It forces a sufficient decrease in the function E but does not rule out an arbitrarily small steplength. Condition (7) or (8) ensures that the steplength λ_k is sufficiently large.

Here we observe that both C1 and C1′ are based on the assumption that a locally convex region of function f in (5) has been bracketed to determine an acceptable steplength λ_k. In special cases, the bracketing strategy may fail and f may not be strictly convex on some feasible λ subinterval explored in a line search iteration (see Figs. 3, 4 and 5 later). For those subintervals, the line search procedure might generate points that satisfy the sufficient decrease condition but not the second condition of sets C1 and C1′, even though the value of λ_k is not necessarily small. Therefore, an alternative condition is needed in these situations to halt the iteration process efficiently.

To see that an acceptable steplength λ_k determined by C1 or C1′ lies on an interval on which f is convex, we write $f'(\lambda_k) = g(X^k + \lambda_k P^k)^T P^k$ and $f'(0) = g(X^k)^T P^k < 0$. We then use (7) or (8) to write

$$f'(\lambda_k) - f'(0) \geq (\beta - 1) f'(0) \geq (1 - \beta)|f'(0)| \tag{9}$$

since $f'(0) < 0$ and $0 < \beta < 1$. This shows that λ_k must be large enough since the right-hand side of (9) is a positive constant. Using the mean value theorem for f' in the interval $(0, \lambda_k)$ we write $f'(\lambda_k) = f'(0) + \lambda_k f''(\xi_k)$ for $\xi_k \in (0, \lambda_k)$. Dividing by λ_k and combining with (9), we have

$$f''(\xi_k) = \frac{f'(\lambda_k) - f'(0)}{\lambda_k} \geq \frac{1 - \beta}{\lambda_k}|f'(0)| > 0. \tag{10}$$

This indicates that f is convex near ξ_k.

To determine an acceptable λ_k when the bracketing strategy has not found a subinterval on which f is convex, we propose the following Condition-set 2 (C2):

C2 (MORE LENIENT STOPPING RULE FOR LINE SEARCH). The iterative line search process is terminated if it generates a steplength $\lambda_k > 0$ satisfies the sufficient decrease condition (6) and one of the following two conditions: (8) or

$$g(X^k + \lambda_k P^k)^T P^k \leq (2 - \beta) g(X^k)^T P^k. \tag{11}$$

Note that C2 includes two condition sets. The combination of (6) and (8) gives C1′, which determines an acceptable λ_k from subintervals on which f is convex. We use C1′ rather than C1 as part of C2 since C1′ is a weaker stopping rule. To see that condition (11) has been introduced to work on a subinterval on which f is not convex, note that (11) implies

$$f'(\lambda_k) - f'(0) \leq -(1 - \beta)|f'(0)|, \tag{12}$$

indicating that there exists a $\xi_k \in (0, \lambda_k)$ such that $f''(\xi_k) < 0$ for $0 < \beta < 1$. Hence, the new combination of conditions (6) and (11) in C2 works on a subinterval on which f is not convex. C2 can thus lead to a larger range of acceptable steplengths than both C1 and C1′ for general f.

To illustrate, we consider function f_1, plotted in Fig. 1:

$$f_1(\lambda) = \begin{cases} -\lambda^2 - \lambda & \text{if } 0 \leq \lambda \leq 1, \\ \dfrac{3}{\lambda} - 5 & \text{if } \lambda \geq 1\ . \end{cases} \tag{13}$$

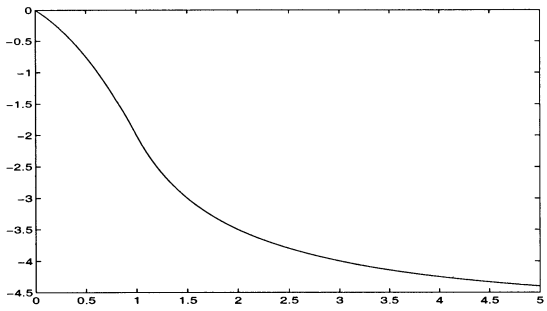

FIGURE 1 Function f_1 (Eq. (13)).

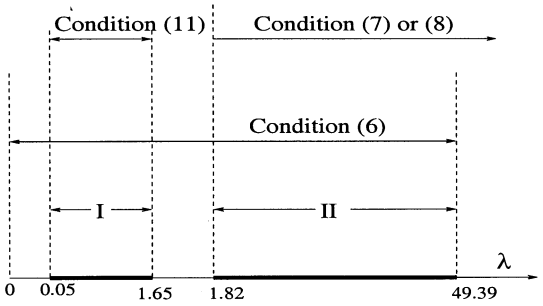

FIGURE 2 The intervals of steplength λ determined by C1 or C1′ (interval *II*) and C2 (*I* and *II*) with $\alpha = 0.1$ and $\beta = 0.9$ for function f_1.

This continuously differentiable function is bounded below. It is easy to show that the λ intervals satisfying the sufficient decrease condition (6) are $[0, 1]$ and $[(5 - \sqrt{25 - 12\alpha})/(2\alpha), (5 + \sqrt{25 - 12\alpha})/(2\alpha)]$; the interval satisfying (7) or (8) is $[\sqrt{3/\beta}, \infty)$. Thus, the range of λ satisfying C1 or C1′ is $II = [\sqrt{3/\beta}, (5 + \sqrt{25 - 12\alpha})/(2\alpha)]$. Since condition (11) produces an additional interval $I = [(1 - \beta)/2, \sqrt{3/(2 - \beta)}]$, the range of λ satisfying C2 is the union of two intervals *I* and *II* (see Fig. 2 for $\alpha = 0.1$ and $\beta = 0.9$).

While C2 can reduce the total number of line search iterations (or function evaluations) due to a larger range of acceptable λ, the accepted λ might not approximate well the minimum point of f, and these effects must be considered in tandem. In fact, condition (8) or (11) only ensures a sufficiently large steplength; it may not approximate a minimum point of f even with $\beta = 0$.

As examples, consider functions f_2 and f_3 [11] (Figs. 3 and 4)

$$f_2(\lambda) = (\lambda + 0.004)^5 - 2(\lambda + 0.004)^4, \tag{14}$$

and

$$f_3(\lambda) = \begin{cases} 1 - \lambda + \dfrac{2(1-\mu)}{l\pi}\sin\left(\dfrac{l\pi}{2}\lambda\right) & \text{if } \lambda \le 1 - \mu, \\ \lambda - 1 + \dfrac{2(1-\mu)}{l\pi}\sin\left(\dfrac{l\pi}{2}\lambda\right) & \text{if } \lambda \ge 1 + \mu, \\ \dfrac{(\lambda-1)^2}{2\mu} + \dfrac{\mu}{2} + \dfrac{2(1-\mu)}{l\pi}\sin\left(\dfrac{l\pi}{2}\lambda\right) & \text{if } 1 - \mu \le \lambda \le 1 + \mu, \end{cases} \tag{15}$$

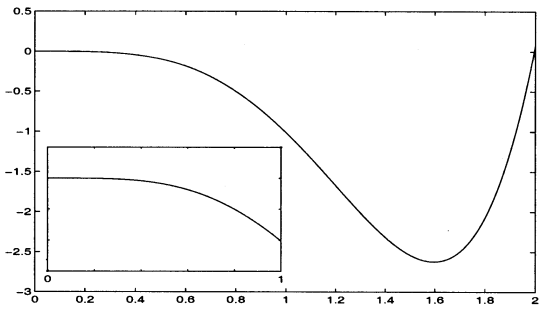

FIGURE 3 Function f_2 (Eq. (14)).

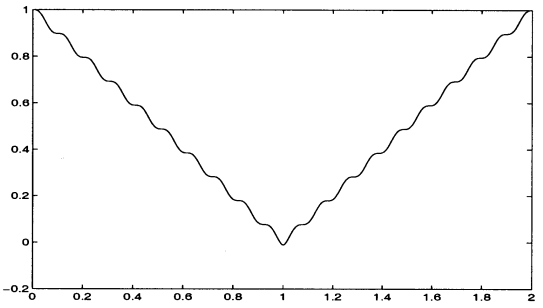

FIGURE 4 Function f_3 (Eq. (15)).

where $\mu = 0.01$ and $l = 39$. These functions were used in [11] to demonstrate the performance of a line search algorithm using C1. To compare the line search using C2 *versus* C1 and C1′, we use the same line search code [11] and parameters (such as the four different starting points for λ $10^{-3}, 10^{-1}, 10$ and 10^3, and $\alpha = \beta = 0.1$) as [11]. Results are reported in Table I. To verify that C2 can generate a sufficiently large steplength, we also tested a very small starting point, $\lambda_0 = 10^{-10}$, with respect to three different values of β (0.1, 0.5 and 0.9). Results are reported in Table II. These two tables show, on one hand, that C2 can reduce the total number of required function evaluations significantly in comparison to C1 and C1′. On the other hand, the values of the accepted λ are very different! C1 leads to the minimum points 1.6 and 1.0 of functions f_2 and f_3, respectively, while the acceptable steplengths generated by C2 are much smaller than these values. Note also that using C1′ requires less function evaluations than C1 but that for function f_2 the minima obtained are also different and depend on the starting point.

TABLE I Line search performance using C2 *vs* C1 and C1′ for functions f_2 (Eq. (14)) and f_3 (Eq. (15))

Starting point	*C1*		*C1′*		*C2*	
	f Eval.	*Accepted λ*	*f Eval.*	*Accepted λ*	*f Eval.*	*Accepted λ*
Function f_2						
10^{-3}	12	1.6	10	1.6	1	0.001
10^{-1}	8	1.6	5	1.6	1	0.1
10	8	1.6	5	1.6	3	0.69
10^{3}	11	1.6	7	1.6	6	0.72
Function f_3						
10^{-3}	12	1.0	8	1.6	2	0.005
10^{-1}	12	1.0	6	1.5	1	0.1
10	10	1.0	3	1.0	2	0.021
10^{3}	13	1.0	7	1.1	3	0.016

TABLE II Condition-sets C1, $C1'$, and C2 for function f_3 (Eq. (15)) with different β and starting point 10^{-10}

Value of β	*C1*		*C1′*		*C2*	
	f Eval.	*Accepted λ*	*f Eval.*	*Accepted λ*	*f Eval.*	*Accepted λ*
0.1	25	1.0	17	1.6	12	0.0056
0.5	25	1.0	17	1.6	12	0.0056
0.9	25	1.0	17	1.6	11	0.0014

Since C2 forces sufficient function decrease according to (6) and guarantees a sufficiently large steplength, it might be tested in practice for overall efficiency of a minimization procedure for a large-scale multivariate function where function evaluations are expensive. In theory, the global convergence for a descent method using C2 can be proven in the same way as for C1 and C1′ [6]. More precisely, the following theorem holds for C2.

THEOREM *Let $E : R^n \to R$ be continuously differentiable in an open convex set $\mathcal{D}$ and bounded below. If the descent iterates X^k defined by* (1) *satisfy the descent condition* (2) *and the line search stopping rule* C2, *and the angles between the search directions P^k and the gradients $g(X^k)$ of E at X^k are bounded away from 90° by a constant for all k, then for any given $X^0 \in \mathcal{D}$,*

$$\lim_{k\to\infty} g(X^k) = 0.$$

Furthermore, if $\{X^k\}$ converges to a point $X^* \in \mathcal{D}$, at which $g(X^*) = 0$ and the second derivative of E is positive definite, and there exists an

index $k_0 > 0$ *such that* $\lambda_k = 1$ *for all* $k \geq k_0$, *then* $\{X^k\}$ *converges to* X^* *q-superlinearly*.

3 A MINIMUM ACCEPTABLE STEPLENGTH

The cubic interpolant constructed in each line search iteration may have a very small minimum point in special cases. Our experience has shown this to occur in practice in computational chemistry problems. The simple modification below incorporates a minimum acceptable value for λ_k.

Let λ_l and λ_t be the two endpoints of an interval I, and f_l, f_t, g_l and g_t the values of $f(\lambda)$ and $f'(\lambda)$ at λ_l and λ_t, respectively. Using them, we can construct a cubic interpolant of f on I, and find its minimum point λ_c as follows:

$$\lambda_c = \lambda_l + \frac{\sqrt{\theta^2 - g_l g_t} - g_l + \theta}{2\sqrt{\theta^2 - g_l g_t} - g_l + g_t}(\lambda_t - \lambda_l), \tag{16}$$

where $\theta = 3(f_l - f_t)/(\lambda_t - \lambda_l) + g_l + g_t$.

When $f_t > f_l$, we have $\theta < 0$. Hence, if $f_t \to +\infty$, we have $\theta \to -\infty$, and

$$\frac{\sqrt{\theta^2 - g_l g_t} - g_l + \theta}{2\sqrt{\theta^2 - g_l g_t} - g_l + g_t} = \frac{|\theta|\sqrt{1 - g_l g_t/\theta^2} - g_l + \theta}{2|\theta|\sqrt{1 - g_l g_t/\theta^2} - g_l + g_t} \to 0. \tag{17}$$

Therefore, if the lower endpoint is set as $\lambda_l = 0$, (16) and (17) give $\lambda_c \approx 0$.

In the line search algorithm of Moré and Thuente [11], a sequence of trial values, $\{\lambda^{(j)}\}$, is generated such that $\lambda^{(j)}$ is in the interval I_j satisfying $I_0 \supset I_1 \supset I_2 \supset \cdots$. If $f_t > f_l$, together with another condition (see [11]), the jth iterate is defined by

$$\lambda^{(j)} = \lambda_c. \tag{18}$$

At the start of the line search, we have $I_0 = [0, \lambda^{(0)}]$ with a given starting point $\lambda^{(0)} > 0$ (i.e., $\lambda_l = 0$ and $\lambda_t = \lambda^{(0)}$). Thus, if $f(\lambda^{(0)})$ is very large, we have $\lambda^{(1)} = \lambda_c \approx 0$. It follows that all subsequent trial values, $\lambda^{(j)}$, are very small, leading to failure of the line search algorithm.

The above case occurs in our molecular energy minimization occasionally. We illustrate this problem through an application of the truncated-Newton package TNPACK [5,13,14,17] to the minimization of the protein lysozyme (6090 Cartesian variables) [17]. At the 51th truncated-Newton iteration, the line search fails to converge due to a very small trial value obtained at the first iteration of the line search. We plot here the function $f(\lambda) = E(X_k + \lambda P^k)$ at $k = 51$ on the interval [0, 1] in two figures: the positive values of f are in Fig. 5 and the negative values in Fig. 6.

From the inset of Fig. 6 we see that $f'(0) < 0$, and there exists a minimum of f near 0.01. However, since $f(1) = \mathrm{O}\,(10^m)$ with $m > 10$, as

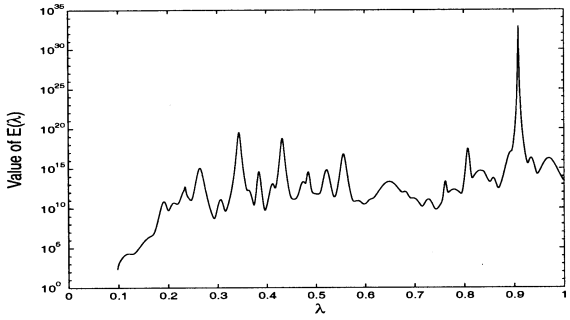

FIGURE 5 The positive part of function $E(X_k + \lambda P^k)$ at step $k = 51$ of the minimization algorithm, where E is the lysozyme potential energy, $X^k$is the 51th truncated-Newton iterate, and P^k is the search direction generated by the preconditioned conjugate gradient method.

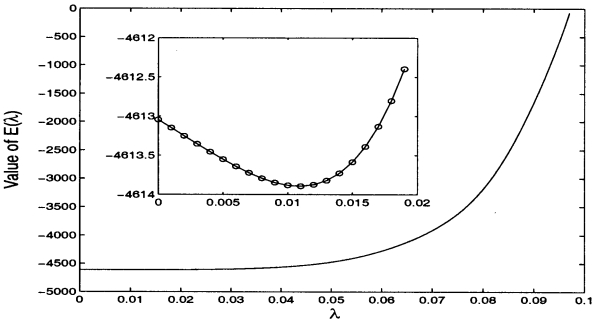

FIGURE 6 The negative part of the function $E(X_k + \lambda P^k)$ as described in Fig. 5. The inset shows the location of the minimum.

shown in Fig. 5, the line search scheme generates a trial value as small as $O(10^{-m})$ and fails to converge.

A simple remedy is the addition of a lower bound after the computation of the trial value λ_c. For example, Moré has suggested (personal communication) to modify (18) as follows:

$$\lambda^{(j)} = \max\{\lambda_l + \sigma(\lambda_t - \lambda_l), \lambda_c\}, \tag{19}$$

where σ is a small number such as 0.001. Another solution is using a backtracking strategy [6]. Too small a λ_c may indicate that function $f(\lambda)$ is poorly modeled by a cubic interpolant on $[\lambda_l, \lambda_t]$. Hence, the interval size should be reduced by setting a new trial value according to

$$\lambda_t := \rho\lambda_t$$

with $\rho \leq 1/2$.

4 NUMERICAL EXPERIMENTS IN BIOMOLECULAR MINIMIZATION

We now test the two line search modifications in the context of TNPACK. We use stopping rule C2 for halting the line search process, and the modification (19) to incorporate a lower bound for the acceptable steplength. The starting point for the line search is $\lambda^{(0)} = 1$. As we show in the appendix, the value unity is the minimum point of the quadric approximation $q(\lambda)$ of f. This is because the minimum point of the quadratic model $q(\lambda)$ is $-g_k^T P^k/(P^k)^T H_k P^k$ (H_k is the Hessian at X^k) and $(P^k)^T H_k P^k = -g_k^T P^k$ when the search direction P^k is a linear preconditioned conjugate gradient (PCG) iterate. TNPACK uses the line search code by Moré and Thuente, which uses C1 to halt line search iterations. Hence, we will only compare C2 *versus* C1 in the numerical experiments made with TNPACK.

We consider two protein molecular systems: BPTI and lysozyme. The 58-residue protein BPTI has 568 atoms and thus 1704 Cartesian variables; it is considered "small" by computational chemists. The larger protein lysozyme has 130 residues, 2030 atoms, and 6090 variables.

For each protein, we construct three starting points. Besides the default experimental structure (X^0) from CHARMM, we also use the simple perturbation formula $X^{0,\omega} = X^0 + \omega\Upsilon$ for $\omega = 0.1$ and 1, with the random vector Υ generated by the randomly chosen seed 13434323.

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. CHARMM version 23 was used to compute the potential energy function and its derivatives, with the default parameters of a TNPACK version for CHARMM [17]. All nonbonded interactions of the potential energy function were considered, and a distance-dependent dielectric function was used. The vector norm $\|\cdot\|$ in the table is the standard Euclidean norm divided by $\sqrt{n}$, where n is the number of independent variables of a potential energy function. The latest version of the Moré and Thuente line search code was incorporated at this time.

Table III compares the performance of TNPACK using stopping rules C1 *versus* C2. Different overall pathways are obtained, so comparisons are difficult, but a small improvement using C2 can be noted over all. Interestingly, using C2 tends to produce lower final energy values. C2 also tends to produce less function evaluations, less Newton iterations, and much less CPU time to find a minimum than C1 (a factor of four for BPTI).

TABLE III Minimization performance of TNPACK using the line search stopping rule C1 *vs* C2 for two proteins and three starting points for each (labeled as *a*, *b*, *c*)

Criterion	*Final energy*	*Final* $\|g\|$	*TN (PCG) Itns.*	*E Evals.*	*CPU time*
BPTI (1704 variables)					
$C1_a$	−2705.20	3.3×10^{-6}	67 (1574)	207	5.9 min
$C2_a$	−2772.80	2.8×10^{-6}	62 (1163)	179	4.5 min
$C1_b$	−2796.32	9.5×10^{-6}	820 (2347)	2090	22 min
$C2_b$	−2730.47	1.2×10^{-6}	81 (1215)	202	4.9 min
$C1_c$	−2771.91	7.0×10^{-6}	70 (1686)	200	6.1 min
$C2_c$	−2766.47	9.8×10^{-6}	55 (947)	186	3.9 min
Lysozyme (6090 variables)					
$C1_a$	−4631.20	4.6×10^{-6}	74 (1873)	229	1.5 h
$C2_a$	−4649.90	3.3×10^{-6}	70 (1578)	195	1.3 h
$C1_b$	−4634.53	9.0×10^{-6}	250 (2908)	593	3.0 h
$C2_b$	−4600.88	3.7×10^{-6}	91 (1964)	210	1.6 h
$C1_c$	−4615.49	3.0×10^{-6}	118 (2728)	292	2.2 h
$C2_c$	−4639.50	3.4×10^{-6}	75 (1769)	194	1.4 h

5 NUMERICAL EXPERIMENTS FOR 18 STANDARD TEST FUNCTIONS

We also study performance on the 18 test functions in the Algorithm 566 [7, 8] package by Moré *et al.*, supplemented by the Hessian package, HESFCN, by Averbukh *et al.* [15, 16]. The supplementary package makes possible the testing of minimizers that use second-derivative information.

TABLE IV Minimization performance for the 18 test functions of Alg. 566

Prob	*TNPACK*	*Final energy*	*Final* $\|g\|$	*TN (PCG) Itns.*	*E evals.*
1	Modified	2.88×10^{-21}	6.19×10^{-10}	16 (39)	20
	Original	2.87×10^{-32}	6.67×10^{-16}	17 (40)	23
2	Modified	2.43×10^{-1}	3.38×10^{-7}	1156 (6165)	2606
	Original	2.43×10^{-1}	3.05×10^{-7}	1274 (6822)	2963
3	Modified	1.13×10^{-8}	5.60×10^{-11}	2 (3)	4
	Original	1.13×10^{-8}	5.60×10^{-11}	2 (4)	3
4	Modified	7.53×10^{-10}	8.70×10^{-9}	119 (209)	173
	Original	2.98×10^{-7}	1.28×10^{-7}	107 (182)	167
5	Modified	1.05×10^{-18}	3.43×10^{-10}	16 (34)	20
	Original	1.05×10^{-18}	3.43×10^{-10}	16 (34)	20
6	Modified	3.24×10^{-22}	8.04×10^{-11}	9 (14)	10
	Original	1.97×10^{-18}	6.27×10^{-9}	9 (14)	10
7	Modified	4.71×10^{-1}	7.52×10^{-15}	9 (19)	10
	Original	4.71×10^{-1}	7.52×10^{-15}	9 (19)	10
8	Modified	1.52×10^{-5}	3.43×10^{-9}	52 (96)	64
	Original	1.52×10^{-5}	3.43×10^{-9}	52 (96)	64
9	Modified	3.20×10^{-6}	3.95×10^{-11}	34 (92)	42
	Original	3.20×10^{-6}	1.85×10^{-10}	35 (95)	44
10	Modified	5.42×10^{-20}	7.09×10^{-10}	4 (5)	5
	Original	5.42×0^{-20}	7.09×10^{-10}	4 (5)	5
11	Modified	8.58×10^{4}	7.22×10^{-3}	10 (27)	11
	Original	8.58×10^{4}	7.22×10^{-3}	10 (27)	11
12	Modified	1.72×10^{-30}	7.24×10^{-15}	38 (92)	47
	Original	Line search failed at 3rd TN after 30 line search iterations			
13	Modified	2.57×10^{-3}	7.79×10^{-9}	8 (21)	11
	Original	2.57×10^{-3}	7.79×10^{-9}	8 (21)	11
14	Modified	1.18×10^{-23}	2.15×10^{-12}	27 (46)	32
	Original	1.48×10^{-25}	2.97×10^{-13}	33 (57)	43
15	Modified	7.31×10^{-13}	3.13×10^{-9}	21 (75)	22
	Original	7.31×10^{-13}	3.13×10^{-9}	21 (75)	22
16	Modified	3.98×10^{-27}	7.17×10^{-14}	9 (14)	11
	Original	7.94×10^{-24}	2.37×10^{-12}	9 (16)	12
17	Modified	1.39×10^{-30}	2.30×10^{-14}	51 (170)	64
	Original	4.30×10^{-23}	1.43×10^{-10}	52 (175)	64
18	Modified	2.84×10^{-21}	1.82×10^{-10}	7 (11)	11
	Original	2.90×10^{-17}	9.18×10^{-9}	6 (9)	11

+ [16.7.2002–12:14pm] [1–18] [Page No. 14] FIRST PROOFS i:/T&F Gordon & Breach/Goms/GOMS-21024.3d Optimization Methods and Software (GOMS) Paper: GOMS-21024 Keyword

We used the new TNPACK version [18] distributed as Algorithm 702 in *ACM TOMS* [13,14], which was updated as described here, along with other improvements described in [17]. For simplicity, we used all default starting points given in Alg.566 and default values of TNPACK. The preconditioner for PCG was chosen as the diagonal of the Hessian matrix H_k for all problems. The modified TNPACK in Table IV used the line search stopping rule C2 and λ modification (19); other parts remained the same as the original TNPACK described in [13], which used C1.

Table IV displays minimization performance of the modified TNPACK for these 18 functions, together with a comparison with the original TNPACK. The CPU times are all very small (of order of 0.001 s) and are not recorded.

From Table IV we see that the updated version performs overall slightly better for all test functions. For the second function, described by Biggs [2], and the 12th function (Gulf research and development function [15,16]), a significant improvement is observed when C2 is used. The original code failed on the 12th function due to rounding errors in the third truncated-Newton iterate (after 30 line search iterations).

Numerical experiments thus show that using the more lenient line search stopping rule C2 in the context of large-scale minimization problems is no worse than using the common stopping rule C1. However, in special cases, such as highly nonlinear objective functions and very costly functions, the use of C2 can improve overall efficiency.

Acknowledgments

We thank Dr. J. Moré for his valuable suggestions. This research of Tamar Schlick presented in this article was supported in part by the National Science Foundation (ASC-9157582 and BIR 94-23827EQ). T. Schlick is an investigator of the Howard Hughes Medical Institute.

References

[1] L. Armijo (1966). Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific J. Math.*, **16**, 1–3.
[2] M.C. Biggs (1971). A comparison of several current optimization methods. *J. Inst. Maths. Applics.*, **8**, 315–327.

[3] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan and M. Karplus (1983). CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comp. Chem.*, **4**, 187–217.
[4] R.S. Dembo and T. Steihaug (1983). Truncated-Newton algorithms for large-scale unconstrained optimization. *Math. Prog.*, **26**, 190–212.
[5] P. Derreumaux, G. Zhang, B. Brooks and T. Schlick (1994). A truncated-Newton method adapted for CHARMM and biomolecular applications. *J. Comp. Chem.*, **15**, 532–552.
[6] J.E. Dennis, Jr. and R.B. Schnabel (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reprinted with corrections by SIAM in the series SIAM's classics in Applied Mathematics, 1996.
[7] B.S. Garbow, J. Moré and K.E. Hillstrom (1981). Algorithm 566: fortran subroutines for testing unconstrained optimization software. *ACM Trans. on Math. Softw.*, **7**, 136–140.
[8] B.S. Garbow, J. Moré and K.E. Hillstrom (1981). Testing unconstrained optimization software. *ACM Trans. on Math. Softw.*, **7**, 17–41.
[9] P.E. Gill, W. Murray and M.H. Wright (1983). *Practical Optimization*. Academic Press, London, England.
[10] A.A. Goldstein (1967). *Constructive Real Analysis*. Harper & Row, New York.
[11] J.J. Moré and D.J. Thuente (1994). Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, **20**, 286–307.
[12] J. Nocedal (1991). Theory of algorithms for unconstrained optimization. *Acta Numerica*, 199–242.
[13] T. Schlick and A. Fogelson (1992). TNPACK — A truncated Newton minimization package for large-scale problems: I. Algorithm and usage. *ACM Trans. Math. Softw.*, **14**, 46–70.
[14] T. Schlick and A. Fogelson (1992). TNPACK — A truncated Newton minimization package for large-scale problems: II. Implementation examples. *ACM Trans. Math. Softw.*, **14**, 71–111.
[15] S. Figueroa, V.Z. Averbukh and T. Schlick (1992). HESFCN — A FORTRAN package of Hessian subroutines for testing nonlinear optimization software, Technical Report 610. Courant Institute of Math. Sciences, New York University.
[16] V.Z. Averbukh, S. Figueroa and T. Schlick (1994). Remark on algorithm 566. *ACM Trans. Math. Softw.*, **20**, 282–285.
[17] D. Xie and T. Schlick (1999). Efficient implementation of the truncated-Newton algorithm for large-scale chemistry applications. *SIAM J. Optim.*, **10**, 132–154.
[18] D. Xie and T. Schlick (1999). Remark on Algorithm 702 — The updated truncated Newton minimization package. *ACM Trans. Math. Softw.*, **25**, 108–122.

APPENDIX A STARTING POINT JUSTIFICATION

In a line search iterative method, a good starting point, $\lambda^{(0)}$, can be chosen as a minimum point of the following quadratic Newton model function of λ:

$$q(\lambda) = E(X^k) + \lambda g_k^T P^k + \tfrac{1}{2}\lambda^2 (P^k)^T H_k P^k,$$

because $q(\lambda) \approx E(X^k + \lambda P^k)$. That is, we have

$$\lambda^{(0)} = -\frac{g_k^T P^k}{(P^k)^T H_k P^k}. \tag{20}$$

In practice, however, formula (20) is rarely used due to the expensive computation of the inner product $(P^k)^T H_k P^k$. But, when search direction P^k is generated by the preconditioned conjugate gradient (PCG) method, as in truncated-Newton methods [4], we have the following theorem:

THEOREM *Let p_j be the jth PCG iterate with $p_1 = 0$. Then*

$$p_{j+1}^T H_k p_{j+1} = -g_k^T p_{j+1} \quad \textit{for all } j \geq 1. \tag{21}$$

Hence, when $P^k = p_{j+1}$, we get

$$(P^k)^T H_k P^k = -g_k^T P^k,$$

which results in $\lambda^{(0)} = 1$. This prove that $\lambda^{(0)} = 1$ is a good starting point for the line search in the context of TN.

The Proof of Theorem Let d_j, r_j and M_k be the jth PCG search direction, residual vector, and the preconditioner, respectively. From PCG theory we know that

$$d_{j+1}^T H_k d_j = 0, \quad \text{and} \quad p_{j+1} = p_j + \alpha_j d_j \quad \text{for } j \geq 1,$$

where $\alpha_j = r_j^T M_k^{-1} r_j / d_j^T H_k d_j$, and $p_1 = 0$. They imply that

$$\begin{aligned} &d_j^T H_k p_j \\ &\quad = d_j^T H_k (p_{j-1} + \alpha_{j-1} d_{j-1}) = d_j^T H_k p_{j-1} = \cdots = d_j H_k p_1 = 0, \end{aligned} \tag{22}$$

and

$$p_{j+1} = \sum_{\nu=1}^{j} \alpha_\nu d_\nu. \tag{23}$$

We have shown in [17] that

$$-g_k^T d_\nu = r_\nu^T M_k^{-1} r_\nu \quad \text{for all } \nu \geq 1. \tag{24}$$

Hence, combining (23) and (24) gives

$$-g_k^T p_{j+1} = \sum_{\nu=1}^{j} \alpha_\nu (-g_k^T d_\nu) = \sum_{\nu=1}^{j} \alpha_\nu r_\nu^T M_k^{-1} r_\nu. \tag{25}$$

Further, using (22) and $\alpha_j = r_j^T M_k^{-1} r_j / d_j^T H_k d_j$, we have

$$p_{j+1}^T H_k p_{j+1} = p_j^T H_k p_j + \alpha_j^2 d_j^T H_k d_j + 2\alpha_j d_j^T H_k p_j = p_j^T H_k p_j + \alpha_j r_j^T M_k^{-1} r_j.$$

From this it follows by induction that

$$p_{j+1}^T H_k p_{j+1} = \sum_{\nu=1}^{j} \alpha_\nu r_\nu^T M_k^{-1} r_\nu, \quad j \geq 1. \tag{26}$$

The combination of (25) and (26) proves (21).

Note that the values of α_ν and $r_\nu^T M_k^{-1} r_\nu$ in (26) have been calculated before computing the matrix product $p_{j+1}^T H_k p_{j+1}$. Hence, using identity (26), we can reduce the work amount of computing $p_{j+1}^T H_k p_{j+1}$ to only $O(n)$ floating point operations.