A modified nearly exact method for solving low-rank trust region subproblem[*]

Zhaosong Lu[†] Renato D. C. Monteiro[‡]

June 1, 2004
1st revision: December 8, 2004
2nd revision: September 16, 2005

Abstract

In this paper, we first discuss how the nearly exact (NE) method proposed by Moré and Sorensen [15] for solving trust region subproblems can be modified to solve large-scale "low-rank" trust region TR subproblems efficiently. Our modified algorithm completely avoids computation of Cholesky factorizations by instead relying primarily on the Sherman-Morrison-Woodbury formula for computing inverses of "diagonal plus low-rank" type matrices. We also implement a specific version of the modified log-barrier (MLB) algorithm proposed by Polyak [18] where the generated log-barrier subproblems are solved by a trust region method. The corresponding direction finding TR subproblems are of the low-rank type and are then solved by our modified NE method. We finally discuss the computational results of our implementation of the MLB method and its comparison with a version of LANCELOT [5] based on a collection extracted from CUTEr [13] of nonlinear programming problems with simple bound constraints.

Key words: nearly exact method, trust region method, large-scale optimization, limited-memory BFGS method, Sherman-Morrison-Woodbury formula

AMS 2000 subject classification: 90C06, 90C30, 65K05

1 Introduction

Trust region algorithms are classical methods for solving both convex and nonconvex nonlinear optimization problems. They are known to possess strong convergence properties (see Fletcher [8]). At each iteration of a trust region method, the following is specified: i) a "simple" approximation $\phi(\tilde{x})$ to the objective function, called the model, and; ii) a region T around the current iterate x, where $\phi(\tilde{x})$ is believed to provide a good approximation to the objective function. An approximate solution p of the subproblem $\min_p\{\phi(x+p) : x+p \in T\}$ is then computed, and the next iterate $\hat{x}$

[*]This work was partially supported by NSF Grants CCR-0203113, INT-9910084 and CCF-0430644, and ONR Grants N00014-03-1-0401 and N00014-05-1-0183.

[†]Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890, USA (email: `zhaolu@andrew.cmu.edu`).

[‡]School of ISyE, Georgia Institute of Technology, Atlanta, Georgia 30332, USA (email: `monteiro@isye.gatech.edu`).

is set to be $\hat{x} := x + p$ provided there is a "significant" objective function progress; otherwise, we define the next iterate $\hat{x}$ as $\hat{x} := x$. In both cases, the region T might be updated and the process is then repeated until a desirable iterate is obtained.

In most trust region methods, the above subproblem is either of or reduces to the following form:

$$\text{minimize}\,\{\, q(p) : \|p\|_M \leq \Delta \,\} \tag{1}$$

where Δ is a positive parameter, M is a symmetric positive-definite matrix referred to as the scaling matrix, $\|\cdot\|_M$ is the *M-norm* defined as

$$\|x\|_M = \sqrt{x^T M x}, \quad \forall x \in \Re^n,$$

and $q : \Re^n \to \Re$ is the quadratic function defined as

$$q(p) = g^T p + \frac{1}{2} p^T H p, \quad \forall p \in \Re^n, \tag{2}$$

for some $g \in \Re^n$ and symmetric matrix $H \in \Re^{n \times n}$. The matrix H can be either the Hessian of the objective function or some approximation of it.

There are at least three well-known methods available in the literature for finding an "approximate" solution of TR subproblem (1), which achieves at least as much reduction in the model q as the reduction achieved by the so called "Cauchy point" (see for example Moré [16] and Chapter 4 of Nocedal and Wright [17]). The first method is the *dogleg* method proposed by Powell [19], and later modified by Dennis and Mei [7], which is appropriate when the model Hessian H is positive definite. Recently, Zhang and Xu [27] proposed a *dogleg* method for the case when H is indefinite, which is based on the estimation of the most negative eigenvalue of H and computation of the stable Bunch-Parlett factorization of H (see [3]). The second method is the *two-dimensional subspace minimization* method proposed by Shultz et al. [21], which can be applied when H is indefinite, though it also requires an estimate of the most negative eigenvalue of H. The third method is the *Steihaug* method proposed independently by Steihaug [23] and Toint [24], which is most appropriate when H is the Hessian of the objective function and when this matrix is large and sparse. Based on similar ideas as in the *Steihaug* method, Gould et al. [12] proposed a method whose resulting TR subproblems are tridiagonal and can be efficiently solved by the Moré and Sorensen algorithm [15].

Besides the three "approximate" methods mentioned above, there is a method due to Moré and Sorensen [15], which finds an approximate solution of the TR subproblem (1) in a stronger sense (see (9)). Following standard convention, we will refer to such solutions as "nearly exact" (NE) solutions and to methods for computing them as NE methods. Since the NE method of [15] requires repeated computations of Cholesky factorizations of diagonal displacements of H, it is suitable only for small- to medium-sized problems.

The main goal of this paper is to develop a method for computing NE solutions of the TR subproblem (1) when H and M are large-scale matrices having the following special structures:

$$\begin{aligned} H &= D + VEV^T, & (3)\\ M &= \tilde{D} + \tilde{V}\tilde{E}\tilde{V}^T \succ 0, & (4) \end{aligned}$$

where D, $\tilde{D}$ and $\tilde{E}$ are positive diagonal matrices, V and $\tilde{V}$ have few number of columns (say less than 10), and E is a diagonal matrix. We will refer to the resulting subproblem as the "low-rank trust region" (LRTR) subproblem. We will show that every step of the NE method of [15] can be properly modified to handle the LRTR subproblem and also that the resulting modified NE (MNE) method is quite efficient and robust for computing NE solutions of large-scale LRTR subproblems.

LRTR subproblems arise in several contexts. For example, when using trust region methods to solve unconstrained or linear-equality constrained minimization problems, the matrix H is usually obtained by using a low-rank update (memoryless) formula and the resulting H has the structure specified in (3). In such a case, the scaling matrix M is chosen as either the identity matrix or some other positive definite matrix whose structure is as specified in (4) and depends on the specific problem at hand. It is well known that many constrained minimization problems can be solved by minimizing a sequence of unconstrained ones, obtained by using either the penalty, log-barrier, augmented Lagrangian multiplier (see for example [17]), or modified log-barrier methods (see Polyak [18]). Thus, the NE method developed in this paper for solving the LRTR subproblem can potentially be used in solving many optimization problems.

Independently to the present work, Wang, Wen and Yuan [25] (see also [26]) proposed the *subspace trust region method* for large-scale unconstrained nonlinear optimization problems, which is based on a subspace trust region method and limited memory BFGS method. The size of the TR subproblems in their method is very small (usually about 6 to 16). These TR subproblems can be solved very quickly and exactly, and hence their methods can be used to solve large-scale unconstrained nonlinear problems. The method proposed in [25] is generally different from our limited memory quasi-Newton TR method based on solving a sequence of full-dimensional LRTR subproblems since the TR subproblem of one method generally cannot be converted into an equivalent one of the other method.

The following notations are used throughout our paper. We denote the k-th coordinate vector by e_k and the identity matrix by I. Their dimensions are always clear from the context. The symbol $\Re^n$ denote the n-dimensional Euclidean space. The set of all $m \times n$ matrices with real entries is denoted by $\Re^{m\times n}$. For $J \subseteq \{1,\ldots,n\}$ and $w \in \Re^n$, we let w_J denotes the subvector $[w_i]_{i\in J}$; moreover, if E is an $m \times n$ matrix then E_J denotes the $m \times |J|$ submatrix of E corresponding to J. For a vector $w \in \Re^n$, $\mathrm{Diag}(w)$ denote the diagonal matrix whose i-th diagonal element is w_i for $i = 1,\ldots,n$, and for any real number α, w^α denote the vector whose i-th component is w_i^α whenever it is well-defined for $i = 1,\ldots,n$. The Euclidean norm, the 1-norm and the ∞-norm are denoted by $\|\cdot\|$, $\|\cdot\|_1$ and $\|\cdot\|_\infty$, respectively. For a matrix E, $\mathrm{Im}(E)$ denotes the subspace generated by the columns of E and $\mathrm{Ker}(E)$ denotes the subspace orthogonal to the rows of E. The superscript ^{T} denotes transpose. For any real symmetric matrix E, $\lambda_{\min}(E)$ (resp., $\lambda_{\max}(E)$) denotes the minimal (resp., maximal) eigenvalue of the matrix E; $E \succeq 0$ (resp., $E \succ 0$) denotes that E is positive semi-definite (resp., positive definite).

Before ending this section, we provide one example to show how the LRTR subproblem naturally arises in the context of solving linearly constrained minimization problems using a log-barrier approach. Indeed, consider the problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b, \\ & l \le x \le u, \end{array} \tag{5}$$

where $f(x)$ is twice continuously differentiable in $\Re^n$, $A \in \Re^{m\times n}$ has full row rank, l, $u \in \Re^n$ may have some components equal to $-\infty$ or $+\infty$, and m is small. The log-barrier approach applied to

(5) consists of solving the following sequence of log-barrier subproblems parametrized by $\mu > 0$:

$$\begin{array}{ll} \text{minimize} & \phi_\mu(x) \\ \text{subject to} & Ax = b, \end{array} \tag{6}$$

where

$$\phi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \log(x_i - l_i) - \mu \sum_{i=1}^{n} \log(u_i - x_i).$$

Assume that x denotes the current iterate towards a (local) solution of (6). To find the next iterate, a typical TR method in this context computes the (potential) displacement p by solving the TR subproblem

$$\begin{array}{ll} \text{minimize} & g^T p + \frac{1}{2} p^T H p \\ \text{subject to} & Ap = 0, \\ & \|W^{-1} p\| \leq \Delta, \end{array} \tag{7}$$

where $W = \text{Diag}([(x-l)^{-2} + (u-x)^{-2}]^{-1/2}) \succ 0$, $g = \nabla \phi_\mu(x)$, $\Delta > 0$, and $H = D + VEV^T$ is an approximation to $\nabla^2 \phi_\mu(x)$ obtained by using a low-rank (memoryless) update formula. Thus, H has the structure as in (3), Now, let $B \subset \{1, \ldots, n\}$ be a basic index set and let N denote its complement. By permuting the columns of A, W and D, we may assume that

$$A = [A_B,\, A_N], \quad W = \text{Diag}(W_B,\, W_N), \quad D = \text{Diag}(D_B, D_N).$$

Since $Ap = 0$, we have $p_B = -A_B^{-1} A_N p_N$. Eliminating p_B from (7), we obtain the following equivalent LRTR subproblem

$$\begin{array}{ll} \text{minimize} & \bar{g}^T p_N + \frac{1}{2} p_N^T \bar{H} p_N \\ \text{subject to} & \|p_N\|_M \leq \Delta, \end{array} \tag{8}$$

where

$$\bar{g} = S^T g, \quad \bar{H} = S^T H S, \quad M = S^T W^{-2} S, \quad S = \begin{bmatrix} -A_B^{-1} A_N \\ I \end{bmatrix}.$$

Thus, we easily see that

$$\begin{aligned} M &= W_N^{-2} + (A_B^{-1} A_N)^T W_B^{-2} (A_B^{-1} A_N), \\ \bar{H} &= D_N + (A_B^{-1} A_N)^T D_B (A_B^{-1} A_N) + (S^T V) E (V^T S). \end{aligned}$$

Noting that A has low full rank and H has the structure specified in (3), we immediately see that the matrices M and $\bar{H}$ themselves have the structure as in (4) and (3), respectively. Thus, the subproblem (8) is indeed an LRTR subproblem.

The outline of the paper is as follows. In Section 2, we review the NE method proposed by Moré and Sorensen [15]. In Section 3, we discuss how this method can be modified in order to solve large-scale LRTR subproblems efficiently. In Section 4, we first review the modified log-barrier (MLB) algorithm proposed by Polyak [18] and implement a specific version of this algorithm where the generated log-barrier subproblems are solved by a trust region method whose direction finding subproblems are of the LRTR type. The LRTR subproblems are then solved by our modified NE method. Section 4 also gives computational results of our implementation of the MLB method and its comparison with a version of LANCELOT [5] based on a collection extracted from CUTEr [13] of nonlinear programming problems with simple bound constraints.

2 Review the NE method for solving TR subproblem

It is well-known that the TR subproblems which arise in a TR method does not need to be solved exactly to guarantee the global convergence of the algorithm. For example, it has been shown by Moré and Sorensen [15] (see also [16]) that, under some mild conditions, good theoretical and numerical convergence results for a standard TR method can be obtained if p is chosen so that

$$q(p) \leq \tau_1 q^* \quad \text{and} \quad \|p\|_M \leq \tau_2 \Delta \tag{9}$$

for some positive constants τ_1 and τ_2, where q^* is the optimal value of TR subproblem (1). (Note that $q^* \leq 0$ and that $q^* = 0$ if and only if $g = 0$ and $H \succeq 0$.) We will refer to such vectors p as NE solutions of (1).

The NE method proposed by Moré and Sorensen [15] is a method for computing a NE solution p of (1). In this section, we review the technical results of the NE method of [15] for solving TR subproblem (1) (see [6, 9, 15, 16, 17] for more details). The computational difficulties of using the NE method to TR subproblems corresponding to large-scale optimization problems are also presented. But, in Section 3, we will show that the NE method can be suitably modified to overcome these difficulties if all the TR subproblems are constructed as LRTR ones.

This section is divided into five subsections. In Subsection 2.1, we discuss the necessary and sufficient optimality conditions for a global solution of the TR subproblem (1). In Subsection 2.2, we discuss some classical and easily verifiable sufficient conditions for p to be a NE solution of (1) (see for example [6] and [15]). In Subsection 2.3, we discuss how Newton method applied to a classical one dimensional nonlinear equation provides an estimate of the optimal Lagrange multiplier associated with the constraint of (1). Since the search for the optimal Lagrange multiplier requires the estimation of ever-improving lower bounds for it, we discuss in Subsection 2.4 how these bounds are normally generated. The complete NE method of [15] and its computational difficulties in the context of large-scale problems are also discussed in this subsection.

2.1 Characterization of the solution of TR subproblem

In this subsection, we provide optimality conditions which characterize the global solutions of subproblem (1).

The proof of the next lemma, which provides the above mentioned optimality conditions, is given in Theorem 7.4.1 on pp. 201 of Conn et al. [6]. (This result was obtained independently by Gay [10] and Sorensen [22].)

Lemma 2.1 *p is a global solution of TR subproblem (1) if only if $\|p\|_M \leq \Delta$ and there exists $\lambda \geq 0$ such that*

$$\begin{aligned} H(\lambda)p &= -g, && (10)\\ \lambda(\Delta - \|p\|_M) &= 0, && (11)\\ H(\lambda) &\succeq 0, && (12) \end{aligned}$$

where $H(\lambda) \equiv H + \lambda M$ for any $\lambda \in \Re$. Moreover, there exists a unique $\lambda^* \geq 0$ such that:

i) $\lambda = \lambda^*$ for every pair (p, λ) as above;

ii) if $H(\lambda^*) \succ 0$ then (1) has a unique global optimal solution.

We now introduce some notation. Define

$$\lambda_1 \equiv \lambda_{\min}(M^{-1/2}HM^{-1/2}), \qquad \hat{\lambda} \equiv \max(-\lambda_1, 0). \tag{13}$$

Moreover, we define

$$p(\lambda) \equiv -H(\lambda)^{-1}g, \tag{14}$$

for every $\lambda \in \Re$ for which the above inverse exists. It is well-known that, when $g \neq 0$, the function $\|p(\lambda)\|_M$ is strictly decreasing and convex on $(\hat{\lambda}, \infty)$.

We now describe how an exact solution for (1) can be computed, depending on which of the following three cases occur:

1) If there exists $\tilde{\lambda} \in (\hat{\lambda}, +\infty)$ such that $\tilde{\lambda}$ solves the equation

$$\|p(\lambda)\|_M = \Delta, \tag{15}$$

then $\lambda^* = \tilde{\lambda} > 0$ and $p(\tilde{\lambda})$ is the unique solution of (1). This case is usually referred to as the "easy" one. (Note that this case occurs if and only if $\lim_{\lambda \downarrow \hat{\lambda}} \|p(\lambda)\|_M > \Delta$.)

2) If $\lambda_1 > 0$ and (15) has no solution in $(\hat{\lambda}, +\infty)$, then $\lambda^* = \hat{\lambda} = 0$ and $p(0)$ is a solution of (1). (This case can easily be detected and usually referred to as the "interior convergence" one.)

3) If $\lambda_1 \leq 0$ and (15) has no solution in $(\hat{\lambda}, +\infty)$, then $\lambda^* = \hat{\lambda} = -\lambda_1$. Hence, there exist $0 \neq u \in \operatorname{Ker}(H + \hat{\lambda}M)$ and $\alpha^M \in \Re$ such that

$$\|p_{\mathrm{crt}} + \alpha^M u\|_M = \Delta,$$

where $p_{\mathrm{crt}} \equiv -H(\hat{\lambda})^{\dagger}g$, and the superscript † denotes the Moore-Penrose generalized inverse. Using Lemma 2.1, one easily sees that $p_{\mathrm{crt}} + \alpha^M u$ is a solution of (1). This case is usually referred to as the "hard" one.

Some steps in the NE algorithm for solving (1) require to test whether $\lambda > \hat{\lambda}$ or $\lambda > \lambda^*$. These two inequalities can be checked by using the following easily verifiable characterizations: i) $\lambda > \hat{\lambda}$ if and only if $\lambda > 0$ and $H(\lambda) \succ 0$; ii) $\lambda > \lambda^*$ if and only if $\lambda > \hat{\lambda}$ and $\|p(\lambda)\|_M < \Delta$.

2.2 Termination conditions

Among the three cases mentioned in Subsection 2.1, only the interior convergence case can be implemented exactly. When the other two cases occur, we can only expect to obtain an approximate solution of (1). In this subsection, we review some sufficient conditions for a vector $p \in \Re^n$ to be a NE solution of (1) when either the easy or hard case occurs.

While looking for a scalar $\lambda > \hat{\lambda}$ satisfying (15), we might simply stop when $|\,\|p(\lambda)\|_M - \Delta\,| \leq k_e\Delta$, where $k_e \in (0,1)$ is a fixed tolerance. In this case, the following result establishes that $p(\lambda)$ is a NE solution of (1). Its proof is similar to the one given in Lemma 7.3.5 on pp. 195 of [6] (see also [15]).

Lemma 2.2 *If $\lambda > \hat{\lambda}$ satisfies $|\,\|p(\lambda)\|_M - \Delta\,| \leq k_e\Delta$ for some $k_e \in (0, 1)$, then $q(p(\lambda)) \leq (1 - k_e)^2 q^*$.*

The following result describes how an approximate version of the hard case yields NE solutions for (1). Its proof is similar to the one given in Lemma 7.3.6 on pp. 196 of [6] (see also [15]).

Lemma 2.3 *Suppose that $\lambda > \lambda^*$, $\alpha \in \Re$ and $u \in \Re^n$ such that $\|u\|_M = 1$ satisfy*

$$\alpha^2 u^T H(\lambda) u \le k_h \left(p(\lambda)^T H(\lambda) p(\lambda) + \lambda \Delta^2 \right) \text{ and } \|p(\lambda) + \alpha u\|_M = \Delta, \tag{16}$$

for some $k_h \in (0, 1)$. Then, $q(p(\lambda) + \alpha u) \le (1 - k_h) q^*$.

The main use of Lemma 2.3 is related with the hard case, but we emphasize that the result can also be applied in other cases. We now explain how to satisfy the conditions of Lemma 2.3 in the hard case. From the discussion in Subsection 2.1, we know that $\lambda^* = -\lambda_1$. Hence, if $\lambda - \lambda^*$ is sufficiently small, then $H(\lambda)$ is nearly singular and thus there exists a vector u such that $\|u\|_M = 1$ and $u^T H(\lambda) u$ is nearly zero. Once the pair (λ, u) is determined, a scalar α satisfying the equality in (16) can be easily obtained by solving the following problem

$$\begin{aligned} \text{minimize}_\alpha \quad & \tfrac{1}{2}(p + \alpha u)^T H (p + \alpha u) + g^T (p + \alpha u) \\ \text{subject to} \quad & \|p + \alpha u\|_M = \Delta, \end{aligned} \tag{17}$$

where $p = p(\lambda)$. Using the well-known formula of α for the case when $M = I$ (see page 558 of [15]), it is easy to see that the optimal solution of (17) is given by

$$\alpha = \frac{\Delta^2 - \|p\|_M^2}{p^T M u + \text{sgn}(p^T M u)[(p^T M u)^2 + \Delta^2 - \|p\|_M^2]^{1/2}}, \tag{18}$$

where the function sgn : $\Re \to \Re$ is defined as $\text{sgn}(t) = 1$ if $t \ge 0$, and $\text{sgn}(t) = -1$ if $t < 0$. It can be easily verified that the right hand side of the inequality in (16) evaluated at a triple (λ, u, α) obtained as above stays bounded away from zero as $\lambda - \lambda^*$ approaches zero. Thus, as $\lambda - \lambda^*$ approaches zero, a triple (λ, u, α) obtained as above will eventually satisfy (16) when the hard case occurs.

The key part to find a triple (λ, u, α) satisfying (16) in the hard case is the computation of a vector $u = u_\lambda$ such that $u^T H(\lambda) u$ approaches zero as $\lambda \downarrow \lambda^*$. The NE method of [15] computes such a vector u by first computing the Cholesky factor of $H(\lambda)$, and then using the LINPACK technique [4] (see also Appendix of [15]) for estimating its smallest singular value. On the other hand, since our approach for solving the low rank version of (1) does not rely on computation of Cholesky factorizations, it uses an entirely different approach to compute a vector u as above (see Subsection 3.3).

2.3 Newton update for λ

We have seen in the Subsection 2.1 that λ^* is a root of (15) in the easy case. Hence, given an approximation $\lambda > -\lambda_1$ of λ^*, it is natural to try to perform a Newton iteration at λ with respect to (15) to obtain a new approximation of λ^*. In this subsection, we describe the details of a Newton iteration applied to a reformulation of the nonlinear equation (15) and discuss its main properties.

Since the function $\|p(\lambda)\|_M$ goes to infinity as λ tends to $-\lambda_1$, it is highly nonlinear near $-\lambda_1$ (see [16] and [17]). As a result, Newton method applied directly to (15) might not work well when λ

is near $-\lambda_1$. Reinsch [20] and Hebden [14] independently observed that Newton method applied to the following alternative reformulation of (15) works better in practice:

$$\phi(\lambda) \equiv \frac{1}{\|p(\lambda)\|_M} - \frac{1}{\Delta} = 0. \tag{19}$$

The following result describes some important properties of the function $\phi(\lambda)$ and provides the formula of a Newton iteration for (19).

Lemma 2.4 *Suppose $g \neq 0$. Then, $\phi(\lambda)$ is strictly increasing and concave on $(-\lambda_1, \infty)$. Moreover, the Newton iterate at λ with respect to (19) is*

$$\lambda^+ = \lambda + \left(\frac{\|p(\lambda)\|_M - \Delta}{\Delta}\right)\left(\frac{\|p(\lambda)\|_M^2}{p(\lambda)^T M H(\lambda)^{-1} M p(\lambda)}\right). \tag{20}$$

Proof. For $\lambda > -\lambda_1$, let $\bar{p}(\lambda) \equiv -\bar{H}(\lambda)^{-1}\bar{g}$, where $\bar{g} \equiv M^{-1/2}g$ and $\bar{H}(\lambda) \equiv M^{-1/2}H(\lambda)M^{-1/2} = M^{-1/2}HM^{-1/2} + \lambda I$. By (14) we have $\bar{p}(\lambda) = M^{1/2}p(\lambda)$, which together with (19) implies that

$$\phi(\lambda) = \frac{1}{\|\bar{p}(\lambda)\|} - \frac{1}{\Delta}.$$

Hence, by Lemma 7.3.1 on pp. 183 of [6], it follows that the function $\phi(\lambda)$ is strictly increasing and concave when $\lambda \in (-\lambda_1, \infty)$, and that its first derivative is given by

$$\phi'(\lambda) = \frac{\bar{p}(\lambda)^T \bar{H}(\lambda)^{-1} \bar{p}(\lambda)}{\|\bar{p}(\lambda)\|^3}. \tag{21}$$

The formula (20) for the Newton iteration $\lambda^+ = \lambda - \phi'(\lambda)/\phi(\lambda)$ can be easily derived using (21), $\bar{p}(\lambda) = M^{1/2}p(\lambda)$ and $\bar{H}(\lambda) = M^{-1/2}H(\lambda)M^{-1/2}$. ■

The next result gives a few useful properties of Newton method applied to (19).

Proposition 2.5 *Suppose $g \neq 0$. Then the following statements hold:*

a) Suppose $\lambda \in (-\lambda_1, \lambda^*)$. Then all Newton iterates starting from λ will stay in $(-\lambda_1, \lambda^*)$ and converge to the solution λ^* of the equation (19) monotonically. The convergence is globally Q-linear with the ratio at least

$$\gamma_\lambda = 1 - \frac{\phi'(\lambda^*)}{\phi'(\lambda)} < 1$$

and is ultimately Q-quadratic.

b) Suppose $\lambda \in (\lambda^*, \infty)$. Then the next Newton iterate $\lambda^+ \in (-\lambda_1, \lambda^*]$ or $\lambda^+ \in (-\infty, -\lambda_1]$.

Proof. The proof is similar to the ones given in Lemmas 7.3.2 and 7.3.3 on pp. 185-186 of [6]. ■

To compute the Newton iterate λ^+ according to (20), the NE method of [15] first computes the lower Cholesky factor L of $H(\lambda)$, and uses it to first compute a vector p such that $LL^Tp = -g$ and then a vector w such that $Lw = Mp$. By (20), we then have

$$\lambda^+ = \lambda + \left(\frac{\|p\|_M - \Delta}{\Delta}\right)\left(\frac{\|p\|_M^2}{\|w\|^2}\right). \tag{22}$$

In our approach for solving the low rank version of (1) we entirely avoid the computation of Cholesky factorizations by instead computing the inverse of $H(\lambda)$ by means of the Sherman-Morrison-Woodbury (SMW) formula (see Subsection 3.2).

2.4 A safeguard Newton method

Since a Newton iteration might result in infeasible iterates $\lambda^+ \leq -\lambda_1$, the NE method of [15] uses some safeguard strategies to handle such iterates in order to obtain a globally convergent method for obtaining a NE solution of (1). The basic idea used is to bracket λ^* by a lower bound λ^L and an upper bound λ^U and reduce the length of the interval $[\lambda^L, \lambda^U]$ by using a clever bisection strategy. In this subsection, we discuss the details of this hybrid method.

At every iteration of the method, we have two scalars λ^L and λ^U such that $0 \leq \lambda^L \leq \lambda^* \leq \lambda^U$ and a current approximation $\lambda \in [\lambda^L, \lambda^U]$ of λ^*. Each iteration of the method then consists of updating the quantities λ^L, λ^U and λ. We first describe the basic idea used to update λ^L. Suppose that $u \in \Re^n$ is a vector such that $\|u\|_M = 1$. Using (13), we see that for any $\lambda \in \Re$,

$$u^T H(\lambda) u = (M^{1/2}u)^T (M^{-1/2} H M^{-1/2} + \lambda I)(M^{1/2}u) \geq (\lambda_1 + \lambda)\|u\|_M^2 = \lambda_1 + \lambda.$$

Defining

$$\lambda^B = \lambda^B(\lambda, u) \equiv \lambda - u^T H(\lambda) u, \tag{23}$$

it follows from the above inequality that $\lambda^B \leq -\lambda_1 \leq \lambda^*$, or in words, λ_B is a lower bound for λ^*. In view of this discussion, we conclude that a natural update for λ^L is simply to let $\lambda^L \leftarrow \max(\lambda^L, \lambda^B)$.

We are now ready to describe how the NE method of [15] updates the three quantities λ^L, λ^U and λ. Fix some constant $\theta \in (0,1)$ (e.g., $\theta = 0.01$). It is convenient to consider the following three cases separately:

i) Assume $\lambda \leq -\lambda_1$. If $\lambda = -\lambda_1$, we perform the update $\lambda^L \leftarrow \max(\lambda^L, \lambda)$. Otherwise, $H(\lambda)$ is indefinite, and hence there exists a vector u such that $\|u\|_M = 1$ and $u^T H(\lambda) u < 0$. One approach to find such a vector u is to perform a partial Cholesky factorization of $H(\lambda)$ to find a scalar $\delta > 0$ and a vector v such that

$$(H(\lambda) + \delta e_k e_k^T)v = 0 \text{ and } e_k^T v = 1. \tag{24}$$

(for more details, see [15] or pp. 191-192 of [6]). Letting $u = v/\|v\|_M$ in (23) and using (24), we easily see that $\lambda^B = \lambda + \delta/\|v\|_M^2$. Then, we perform the update $\lambda^L \leftarrow \max(\lambda^L, \lambda^B)$. Finally, we perform the update $\lambda \leftarrow \max(\sqrt{\lambda^L \lambda^U}, \lambda^L + \theta(\lambda^U - \lambda^L))$.

ii) Assume $\lambda \in (-\lambda_1, \lambda^*)$. In this case, we perform the updates $\lambda^L \leftarrow \lambda$ and $\lambda \leftarrow \lambda^+$, where λ^+ denotes the Newton iterate defined in (20).

iii) Assume $\lambda > \lambda^*$. In this case, we have seen in the paragraph following Lemma 2.3 that a vector $u \in \Re^n$ such that $\|u\|_M = 1$ can be computed using the LINPACK technique which makes $u^T H(\lambda) u$ small as long as $\lambda + \lambda_1$ is small. This vector u, together with λ, is then used to compute λ^B according to (23), and the updates $\lambda^U \leftarrow \lambda$, $\lambda^L \leftarrow \max(\lambda^L, \lambda^B)$ and $\lambda \leftarrow \max(\lambda^+, \lambda^L)$ are then performed.

The above scheme for updating λ^L, λ^U and λ can be shown to generate a sequence of λ's which approaches λ^*. Indeed, if case ii) occurs then the sequence of λ's generated afterwards approaches λ^* monotonically from the left in view of Proposition 2.5(a). Also, Proposition 2.5(b) implies that if case iii) occurs then either case i) or ii) must occur at the next iteration. Hence, if case ii) never occurs then case i) must occur infinitely often. But every time i) occurs, it is easy to see that the ratio of the length of the interval $[\lambda^L, \lambda^U]$ at the end of the next iteration and its length at the current iteration is bounded above by $\max(\theta, 1-\theta)$. Hence, if case ii) never occurs, the length of the generated intervals $[\lambda^L, \lambda^U]$ converges to zero, and thus the generated sequence of λ's approaches λ^*.

Note that the implementations of cases i) and iii) of the above scheme for updating λ^L, λ^U and λ are based on the computation of the (partial) Cholesky factorization of a matrix. In our approach for finding NE solutions of the low rank version of (1), these cases must be implemented differently so as to avoid the computation of Cholesky factorizations, which are known to be expensive for large scale problems. These alternative implementations of cases i) and iii) are discussed in detail in Subsections 3.2 and 3.3.

We are now ready to state the whole algorithm of [15] for finding a NE solution of (1).

Algorithm 1 (NE method for solving (1)):

```
Let constants $\theta$, $k_e$, $k_h \in (0,1)$ be given (e.g., $\theta = 0.01$, $k_e = 0.1$, $k_h = 0.2$).
1) Find initial scalars $0 \le \lambda^L < \lambda^U$ such that $\lambda^* \in [\lambda^L, \lambda^U]$ and set $\lambda = \lambda^L$.
2) Attempt to do Cholesky factorization $H(\lambda) = LL^T$ to check whether $\lambda > -\lambda_1$.
3) If $\lambda > -\lambda_1$, solve $LL^T p = -g$ for $p$. Check for interior convergence and easy termination.
   If $\lambda > \lambda^*$, compute a pair $(u, \alpha) \in \Re^n \times \Re$ to check for the hard termination (16).
4) If $\lambda \le -\lambda_1$, then update $\lambda^L$ and $\lambda$ according to case i) as above, and go to step 2).
5) If $\lambda \in (-\lambda_1, \lambda^*)$, then update $\lambda^L$ and $\lambda$ according to case ii) as above, and go to step 2).
6) If $\lambda > \lambda^*$, then update $\lambda^L$, $\lambda^U$ and $\lambda$ according to case iii) as above, and go to step 2).
End
```

For large-scale problems, several computational difficulties arise in the NE method of [15] above. In step 1), it generally takes $\mathcal{O}(n^2)$ amount of arithmetic operations to find the initial λ^U (see Section 7.3.8 of [6]), which is somehow expensive for large-scale problems. The Cholesky or partial Cholesky factorization of $H(\lambda)$ used in steps 2), 4), 6) and 7) needs $\mathcal{O}(n^3)$ amount of arithmetic operations. It will be prohibitive for large-scale problems. In the next section, we will modify the NE method of [15] above to overcome those difficulties for solving the LRTR subproblem.

3 A modified NE method for solving LRTR subproblem

In this section a modified NE (MNE) method for solving large-scale LRTR subproblems is presented. We follow the framework of Algorithm 1 as described in Section 2. Our main effort is to overcome the computational difficulties mentioned in Section 2.

This section is divided into four subsections. A more efficient approach for checking whether $H(\lambda)$ is positive definite is given in Subsection 3.1. We also modify the approach of solving the linear equation $H(\lambda)p = -g$ and computing a Newton iterate in this subsection. A more efficient approach for dealing with hard case termination is developed in Subsection 3.2. In Subsection 3.3, a more efficient approach for improvingλ^L when $\lambda < \lambda_1$ is given. In Subsection 3.4, we develop a cheaper approach to initialize λ^U. We emphasize that all modified approaches completely avoid computing Cholesky or partial Cholesky factorization of large-scale matrices.

3.1 Checking positive definiteness of $H(\lambda)$ and solving $H(\lambda)p = -g$

In step 2) of Algorithm 1, given any $\lambda \geq 0$, the NE method of [15] checks whether $H(\lambda)$ is positive definite by computing the Cholesky factorization of it, which is very expensive and even prohibitive for the large-scale problems. In this subsection, we provide a more efficient method instead, which needs $\mathcal{O}(n)$ amount of arithmetic operation for large-scale LRTR problems. Furthermore, in steps 5) and 6) of Algorithm 1, the NE method of [15] uses the Cholesky factor of $H(\lambda)$ to solve the linear equation $H(\lambda)p = -g$ and compute Newton iterate, respectively. We will modify those approaches as well in this subsection.

The following theorem provides the main tool for the analysis in this subsection.

Theorem 3.1 *Let $\hat{E} \in \mathcal{S}^m$, $\hat{V} \in \Re^{n\times m}$ and an invertible matrix $\hat{D} \in \mathcal{S}^n$ be given and define $\hat{H} \equiv \hat{D} + \hat{V}\hat{E}\hat{V}^T$ and $\hat{W} \equiv \hat{V}^T\hat{D}^{-1}\hat{V}$. Then, the following statements hold:*

i) If $\hat{E}$ is invertible, then $\hat{H}$ is invertible if and only if $\hat{E}^{-1} + W$ is invertible.

ii) If $\hat{D} \succ 0$, then $\hat{H} \succeq 0$ if and only if $\hat{W} + \hat{W}\hat{E}\hat{W} \succeq 0$.

iii) If $\hat{D} \succ 0$, then $\hat{H} \succ 0$ and $\hat{V}$ has full column rank if and only if $\hat{W} + \hat{W}\hat{E}\hat{W} \succ 0$.

Proof. It is well-known that if $\hat{E}^{-1} + \hat{W}$ is invertible then the SWM formula applied to $\hat{H}$ implies that $\hat{H}^{-1} = \hat{D}^{-1} - \hat{D}^{-1}\hat{V}(\hat{E}^{-1} + \hat{W})^{-1}\hat{V}^T\hat{D}^{-1}$. Similarly, if $\hat{H} = \hat{D} + \hat{V}\hat{E}\hat{V}^T$ is invertible then the SMW formula applied to $\hat{E}^{-1} + \hat{W}$ reveals that this matrix is invertible. Hence, i) follows.

To prove statement ii), assume that $\hat{D} \succ 0$. We can then write $\hat{H}$ as $\hat{H} = \hat{D}^{1/2}F\hat{D}^{1/2}$, where $F \equiv I + \hat{D}^{-1/2}\hat{V}\hat{E}\hat{V}^T\hat{D}^{-1/2}$. Clearly, $\hat{H} \succeq 0$ if and only if $F \succeq 0$. In view of the decomposition $\Re^n = \mathrm{Ker}(\hat{V}^T\hat{D}^{-1/2}) + \mathrm{Im}(\hat{D}^{-1/2}\hat{V})$, it follows that, for any $p \in \Re^n$, there exist u, $v \in \Re^n$ and $y \in \Re^m$ such that

$$p = u + v, \quad \hat{V}^T\hat{D}^{-1/2}u = 0, \quad v = \hat{D}^{-1/2}\hat{V}y.$$

This relation, the facts that $Fu = u$ and $u^Tv = 0$, and some simple algebraic manipulation imply that

$$\begin{aligned} p^TFp &= (u+v)^TF(u+v) = u^TFu + 2u^TFv + v^TFv, \\ &= \|u\|^2 + v^TFv = \|u\|^2 + y^T(\hat{W} + \hat{W}\hat{E}\hat{W})y. \end{aligned}$$

By the arbitrariness of p, we easily see that $F \succeq 0$ if only if $\hat{W} + \hat{W}\hat{E}\hat{W} \succeq 0$. This, together with the fact that $\hat{H} \succeq 0$ if only if $F \succeq 0$, implies that ii) holds. Under additional assumption that $\hat{V}$ has full column rank, the statement iii) can be shown by using a similar argument as ii). ■

We now describe an efficient approach to check whether $H(\lambda)$ with $\lambda \geq 0$ is positive definite in step 2) of Algorithm 1. Noting that $H(\lambda) = H + \lambda M$, we see from (3) and (4) that

$$H(\lambda) = \hat{D} + \hat{V}\hat{E}\hat{V}^T, \tag{25}$$

where

$$\hat{D} \equiv D + \lambda\tilde{D} \succ 0, \quad \hat{V} \equiv (V, \tilde{V}), \quad \hat{E} \equiv \mathrm{Diag}(E, \lambda\tilde{E}). \tag{26}$$

Hence, by Theorem 3.1 ii), if $\hat{V}$ has full column rank, then we can check the positive definiteness of $H(\lambda)$ by checking whether

$$X \equiv \hat{W} + \hat{W}\hat{E}\hat{W} \tag{27}$$

is positive definite. On the other hand, if $\hat{V}$ does not have full column rank, then we determine a matrix R with full column rank and a matrix T such that $\hat{V} = RT$. It then follows that $H(\lambda) = \hat{D} + \hat{V}\hat{E}\hat{V}^T = \hat{D} + R\check{E}R^T$, where $\check{E} \equiv T\hat{E}T^T$. Hence, Theorem 3.1 ii) can now be used to check the positive definiteness of $H(\lambda)$ by checking whether $\check{W} + \check{W}\check{E}\check{W} \succ 0$, where $\check{W} \equiv R^T\hat{D}^{-1}R$. For convenience of the presentation, we will assume throughout the remaining subsections that $\hat{V}$ has full column rank.

Recall that one of the requirements for (1) to be a LRTR subproblem is that the number of columns of $\hat{V}$ be small. In this case, X is a small-sized matrix whose positive definiteness can be checked by performing a relatively cheap Cholesky factorization. Since the amount of arithmetic operations to compute X for a large-scale LRTR subproblem is $\mathcal{O}(n)$, the above approach for checking whether $H(\lambda)$ is positive definite only requires $\mathcal{O}(n)$ arithmetic operations. If $H(\lambda)$ turns out to be positive definite, we can then solve the linear system $H(\lambda)p = -g$ by means of SMW formula as

$$\begin{aligned} p &= -H(\lambda)^{-1}g = -\left(\hat{D}^{-1} - \hat{D}^{-1}\hat{V}(\hat{E}^{-1} + \hat{V}^T\hat{D}^{-1}\hat{V})^{-1}\hat{V}^T\hat{D}^{-1}\right)g, \\ &= -\hat{D}^{-1}g + \hat{D}^{-1}\hat{V}(\hat{E}^{-1} + \hat{W})^{-1}\hat{V}^T\hat{D}^{-1}g. \end{aligned}$$

Note that the matrix $\hat{E}^{-1} + \hat{W}$ is invertible due to the fact $H(\lambda) \succ 0$ and Theorem 3.1 i). Since the size of $\hat{E}^{-1} + \hat{W}$ is small, this approach for solving the linear system $H(\lambda)p = -g$ also requires $\mathcal{O}(n)$ arithmetic operations. Note that, in the context of a LRTR subproblem, the Newton iterate λ^+ can be efficiently computed by means of (20), which requires solving another linear system with coefficient matrix $H(\lambda)$.

3.2 Handling the hard case termination

Recall that one of the key parts in the implementation of steps 3) and 6) of Algorithm 1 is the computation of a vector $u = u_\lambda$ such that $\|u\|_M = 1$ and $u^T H(\lambda)u$ approaches zero as $\lambda \downarrow -\lambda_1$ (see Subsections 2.2 and 2.4). In this subsection, we provide an efficient approach to find such a vector u in the context of the low-rank version of (1), which completely avoids the computation of the Cholesky factorization of $H(\lambda)$.

Recall from (3) that $H = D + VEV^T$, where $D \succ 0$ and E is diagonal and nonsingular. We can partition E (after performing a symmetric permutation of its rows and columns) as $E = \mathrm{Diag}(E_1, -E_2)$, where both E_1 and E_2 are positive diagonal matrices. Accordingly, we partition V as $V = (V_1, V_2)$, and hence

$$VEV^T = V_1E_1V_1^T - V_2E_2V_2^T. \tag{28}$$

Noting that $H(\lambda) = H + \lambda M$, we can write $H(\lambda)$ as

$$H(\lambda) = F(\lambda) - V_2E_2V_2^T, \tag{29}$$

where $F(\lambda) = D + \lambda M + V_1E_1V_1^T \succ 0$ for any $\lambda \geq 0$ due to the fact that D, $M \succ 0$.

The following technical lemma provides the key tool for our analysis in this subsection.

Lemma 3.2 *Assume that λ_1 defined in (13) is nonpositive. Then,*

$$\lim_{\lambda \downarrow -\lambda_1} \lambda_{\max}\left(V_2^T H(\lambda)^{-1} V_2\right) = \infty,$$

where V_2 is defined as above.

Proof. For any $\lambda > -\lambda_1$, using (29) and SMW formula twice, we have

$$H(\lambda)^{-1} = F(\lambda)^{-1} + F(\lambda)^{-1} V_2 \left(E_2^{-1} - V_2{}^T F(\lambda)^{-1} V_2\right)^{-1} V_2{}^T F(\lambda)^{-1}, \tag{30}$$

and

$$\begin{aligned}\left(E_2^{-1} - V_2{}^T F(\lambda)^{-1} V_2\right)^{-1} &= E_2 + E_2 V_2{}^T \left(F(\lambda) - V_2 E_2 V_2{}^T\right)^{-1} V_2 E_2, \\ &= E_2 + E_2 V_2{}^T H(\lambda)^{-1} V_2 E_2. \end{aligned} \tag{31}$$

Using the definition of $F(\lambda)$ and the fact $M \succ 0$, we easily see that $F(\lambda) \succ D \succ 0$ for any $\lambda > -\lambda_1 \geq 0$. This implies that $\|F(\lambda)^{-1}\|$, and hence $\|F(\lambda)^{-1} V_2\|$, is bounded for all $\lambda > -\lambda_1$. From (30), we obtain that

$$\|H(\lambda)^{-1}\| \leq \|F(\lambda)^{-1}\| + \|F(\lambda)^{-1} V_2\| \left\| \left(E_2^{-1} - V_2{}^T F(\lambda)^{-1} V_2\right)^{-1} \right\| \|V_2{}^T F(\lambda)^{-1}\| \tag{32}$$

By the definitions of $H(\lambda)$ and λ_1, we have $\lim_{\lambda \downarrow -\lambda_1} \|H(\lambda)^{-1}\| = \infty$, which, together with (32) and the fact that $\|F(\lambda)^{-1}\|$ and $\|F(\lambda)^{-1} V_2\|$ are bounded for all $\lambda > -\lambda_1 \geq 0$, implies

$$\lim_{\lambda \downarrow -\lambda_1} \left\| \left(E_2^{-1} - V_2{}^T F(\lambda)^{-1} V_2\right)^{-1} \right\| = \infty. \tag{33}$$

Moreover, from (31), we have

$$\left\| \left(E_2^{-1} - V_2{}^T F(\lambda)^{-1} V_2\right)^{-1} \right\| \leq \|E_2\| + \|E_2\| \left\| V_2{}^T H(\lambda)^{-1} V_2 \right\| \|E_2\|,$$

with together with (33) implies that

$$\lambda_{\max}\left(V_2{}^T H(\lambda)^{-1} V_2\right) = \left\| V_2{}^T H(\lambda)^{-1} V_2 \right\| \to \infty \text{ as } \lambda \downarrow -\lambda_1.$$

■

The following theorem provides an efficient approach to compute the vector u for dealing with the hard case termination in step 3) of Algorithm 1 and updating λ^L in step 6) of Algorithm 1.

Theorem 3.3 *Assume that λ_1 defined in (13) is nonpositive. Suppose $u_\lambda = H(\lambda)^{-1} v / \|H(\lambda)^{-1} v\|_M$, where $v = V_2 r$ and r is a unit eigenvector of $V_2^T H(\lambda)^{-1} V_2$ corresponding to its maximum eigenvalue. Then,*

$$\lim_{\lambda \downarrow -\lambda_1} u_\lambda^T H(\lambda) u_\lambda = 0.$$

Proof. It follows from Cauchy-Schwarz inequality that

$$v^T H(\lambda)^{-1} v \le \|v\|_{M^{-1}} \|H(\lambda)^{-1} v\|_M. \tag{34}$$

Using (34) and the definitions of v and r, we have

$$\begin{aligned} u_\lambda^T H(\lambda) u_\lambda &= \frac{v^T H(\lambda)^{-1} v}{\|H(\lambda)^{-1} v\|_M^2} = \frac{(v^T H(\lambda)^{-1} v)^2}{(v^T H(\lambda)^{-1} v)\|H(\lambda)^{-1} v\|_M^2}, \\ &\le \frac{\|v\|_{M^{-1}}^2}{v^T H(\lambda)^{-1} v} = \frac{r^T V_2^T M^{-1} V_2 r}{r^T V_2^T H(\lambda)^{-1} V_2 r}, \\ &\le \frac{\|V_2^T M^{-1} V_2\|}{\lambda_{\max}(V_2^T H(\lambda)^{-1} V_2)}, \end{aligned}$$

which, together with Lemma 3.2, immediately implies that the conclusion holds. ■

Before ending this subsection, we make two observations. First, since $\lambda^* = -\lambda_1$ in the hard case, Theorem 3.3 implies that the vector u_λ defined in its statement satisfies $\lim_{\lambda \downarrow \lambda^*} u_\lambda^T H(\lambda) u_\lambda = 0$, which is exactly the condition required in the discussion of the hard case (see Subsection 2.2). Second, since the number of columns of V is assumed to be small in the low-rank version subproblem (1), it follows that the matrix $V_2^T H(\lambda)^{-1} V_2$ is small-sized, and hence a unit eigenvector r as in Theorem 3.3 can be easily computed. Moreover, the SMW formula can be used to compute $V_2^T H(\lambda)^{-1} V_2$ and u_λ in $\mathcal{O}(n)$ arithmetic operations.

3.3 Improving λ^L when $\lambda < -\lambda_1$

Recall that the key part in the implementation of step 4) of Algorithm 1 consists of finding a vector u satisfying $\|u\|_M = 1$ and $u^T H(\lambda) u < 0$, whenever $\lambda < -\lambda_1$ (see Subsections 2.4). In this subsection, we provide an efficient approach to find such a vector u in the context of the low-rank version of (1), which completely avoids the computation of a partial Cholesky factorization of $H(\lambda)$.

Assume then that $0 \le \lambda < -\lambda_1$. This implies that $H(\lambda)$ is indefinite, and hence that the matrix X defined in (27) is also indefinite, in view of Theorem 3.1(iii). Hence, letting y be an eigenvector of X corresponding to its minimum eigenvalue, we have that $y^T X y < 0$. Using the definition of $\hat{W}$ and relations (25), (26) and (27), we easily see that

$$X = \hat{W} + \hat{W}\hat{E}\hat{W} = \hat{V}^T \hat{D}^{-1} H(\lambda) \hat{D}^{-1} \hat{V}. \tag{35}$$

Hence, letting $u := \hat{D}^{-1}\hat{V}y / \|\hat{D}^{-1}\hat{V}y\|_M$, we have that $\|u\|_M = 1$ and

$$u^T H(\lambda) u = \frac{y^T X y}{\|\hat{D}^{-1}\hat{V}y\|_M^2} < 0.$$

Note that since X is a small-sized matrix (see Subsection 3.1), it is relatively cheap to compute the vector y as described above. Moreover, since the amount of arithmetic operations to compute X for a large-scale LRTR subproblem is $\mathcal{O}(n)$, the computation of the vector u described above can be carried out in $\mathcal{O}(n)$ arithmetic operations.

3.4 Finding the initial λ^U

Recall that step 1) of Algorithm 1 requires initial estimates of the lower bound λ^L and the upper bound λ^U. An approach for estimating these bounds for a general TR subproblem in $\mathcal{O}(n^2)$ arithmetic operations is described in Section 7.3.8 of [6]. For the large-scale lower-rank version subproblem (1), the above approach is expensive, and hence not suitable.

In our implementation of Algorithm 1, we set $\lambda^L = 0$. We now provide an efficient approach to find an initial estimate of λ^U in the context of the low-rank version of (1) in this subsection.

Recall that H and M have low-rank structure (see (3) and (4)). Assume that the row size of matrices E and $\tilde{E}$ is $\bar{k}$ and $\tilde{k}$, respectively. For the convenience of the presentation, we rewrite H and M in (3) and (4) as follows

$$H = D + \sum_{i=1}^{\bar{k}} E_{ii} v_i v_i^T, \tag{36}$$

$$M = \tilde{D} + \sum_{i=1}^{\tilde{k}} \tilde{E}_{ii} \tilde{v}_i \tilde{v}_i^T, \tag{37}$$

where v_i, $\tilde{v}_i$ are the ith column of V and $\tilde{V}$, respectively.

Given any $\epsilon > 0$, we can trivially set initial λ^U to be ϵ if $\lambda^* = 0$. Hence, we now assume that $\lambda^* > 0$. It follows from Lemma 2.1 that λ^* together with a global solution p of (1) satisfies

$$(H + \lambda^* M)p = -g, \tag{38}$$

$$\|p\|_M = \Delta, \tag{39}$$

$$H + \lambda^* M \succeq 0 \quad . \tag{40}$$

Multiplying (38) by p^T on the left and using (39) and the fact $M \succ 0$, we obtain

$$p^T H p + \lambda^* \Delta^2 = -p^T g \le \|p\|_M \|g\|_{M^{-1}} = \Delta \|g\|_{M^{-1}}.$$

Hence

$$\lambda^* \le \|g\|_{M^{-1}} \Delta^{-1} - (p^T H p)\Delta^{-2}. \tag{41}$$

Let $\tilde{p} = M^{1/2} p / \Delta$. Noting that $\|\tilde{p}\| = 1$, we have

$$\begin{aligned}
(p^T H p)\Delta^{-2} &= \tilde{p}^T M^{-1/2} H M^{-1/2} \tilde{p}, \\
&\ge \tilde{p}^T M^{-1/2} \left(D + \sum_{\{i | E_{ii} < 0\}} E_{ii} v_i v_i^T \right) M^{-1/2} \tilde{p}, \\
&\ge \lambda_{\min}(D) \|M^{-1/2} \tilde{p}\|^2 + \sum_{\{i | E_{ii} < 0\}} E_{ii} (v_i^T M^{-1/2} \tilde{p})^2, \\
&\ge \zeta \|M^{-1/2} \tilde{p}\|^2,
\end{aligned} \tag{42}$$

where $\zeta = \lambda_{\min}(D) + \sum_{\{i|E_{ii}<0\}} E_{ii} \|v_i\|^2$. Using (37) and the fact that $\tilde{E}_{ii} > 0$ for all i, we have

$$\lambda_{\min}(M) \ge \min_{1 \le i \le n} \tilde{D}_{ii} \text{ and } \lambda_{\max}(M) \le \max_{1 \le i \le n} \tilde{D}_{ii} + \sum_{i=1}^{\tilde{k}} \tilde{E}_{ii} \|\tilde{v}_i\|^2. \tag{43}$$

Note that

$$\lambda_{\max}(M)^{-1} \leq \|M^{-1/2}\tilde{p}\|^2 \leq \lambda_{\min}(M)^{-1}.$$

This together with (43) implies that

$$\left(\max_{1\leq i\leq n} \tilde{D}_{ii} + \sum_{i=1}^{\tilde{k}} \tilde{E}_{ii}\|\tilde{v}_i\|^2\right)^{-1} \leq \|M^{-1/2}\tilde{p}\|^2 \leq \left(\min_{1\leq i\leq n} \tilde{D}_{ii}\right)^{-1}. \quad (44)$$

Using (41), (42), and (44), we see that $\lambda^* \leq \lambda^e$, where λ^e is defined as

$$\lambda^e \equiv \begin{cases} \|g\|_{M^{-1}}\Delta^{-1} - \zeta\left(\max\limits_{1\leq i\leq n} \tilde{D}_{ii} + \sum\limits_{i=1}^{\tilde{k}} \tilde{E}_{ii}\|\tilde{v}_i\|^2\right)^{-1} & \text{if } \zeta \geq 0, \\ \|g\|_{M^{-1}}\Delta^{-1} - \zeta\left(\min\limits_{1\leq i\leq n} \tilde{D}_{ii}\right)^{-1} & \text{if } \zeta < 0. \end{cases} \quad (45)$$

This together with the fact that $\lambda^U = \epsilon$ if $\lambda^* = 0$ implies that $\max(\epsilon, \lambda^e)$ is a proper initial estimate of the upper bound λ^U for any λ^*.

Using the fact that M (4) has low-rank structure, we see that $M^{-1}g$ can be computed by means of SMW formula requiring $\mathcal{O}(n)$ arithmetic operations. This together with (45), and the fact that $\|g\|_{M^{-1}} = \sqrt{g^T M^{-1} g}$ and $\bar{k}$ and $\tilde{k}$ are small, implies that this approach for finding initial estimate of the upper bound λ^U requires $\mathcal{O}(n)$ arithmetic operations in the context of a LRTR subproblem.

4 Some numerical implementation results

In this section, our main goal is to test the numerical performance of "low-rank" trust region methods whose LRTR subproblems are solved by the MNE method proposed in Section 3. For this purpose, we implement a specific version of the modified log-barrier (MLB) algorithm due to Polyak [18] (see Subsection 4.1) and use it to solve a collection of nonlinear programming problems from CUTEr [13] where only simple bound constraints are present. Like the log-barrier method discussed in Section 1, the MLB algorithm also consists of solving a parametrized family of unconstrained nonlinear problems. In our implementation, these subproblems are solved by using a "low-rank" trust region approach similar to the one discussed in Section 1 in the context of the log-barrier method. This section is divided into two subsections. In Subsection 4.1, we discuss the generic MLB algorithm for solving nonlinear programming problems with general inequality constraints and its specialization to problems with simple bound constraints. In Subsection 4.2, we report the computational results of our implementation of the MLB method and its comparison with a version of LANCELOT [5] based on the forementioned collection of problems from CUTEr [13].

4.1 The modified log-barrier algorithm

In this subsection, we discuss the generic MLB algorithm for solving nonlinear programming problems with general inequality constraints and its specialization to problems with simple bound constraints.

Consider the nonlinear programming problem

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad c_i(x) \geq 0, \ i = 1, \cdots, m. \end{aligned} \tag{46}$$

where the functions $f(x)$ and $c_i(x)$, $i = 1, \cdots, m$ are twice continuously differentiable in $\Re^n$. Its associated first-order optimality conditions are:

$$\nabla f(x) - \sum_{i=1}^{m} \lambda_i \nabla c_i(x) = 0, \ \lambda \geq 0,$$

$$\sum_{i=1}^{m} \lambda_i c_i(x) = 0, \ c_i(x) \geq 0, \ i = 1, \cdots, m.$$

The MLB method proposed by Polyak [18] consists of solving a sequence of unconstrained problems with objective functions given by

$$\mathcal{M}(x, \mu^{(k)}, \lambda^{(k)}) = f(x) - \mu^{(k)} \sum_{i=1}^{m} \lambda_i^{(k)} \log\left(\frac{c_i(x)}{\mu^{(k)}} + 1\right), \tag{47}$$

where $\lambda^{(k)} \in \Re^m$ is an estimate of a Lagrange multiplier at a solution of (46) and $\mu^{(k)} > 0$ is a log-barrier parameter. Letting $x^{(k)}$ denote a stationary point of $\mathcal{M}(x, \mu^{(k)}, \lambda^{(k)})$, Polyak [18] has shown under reasonable conditions that there exists a threshold value $\bar{\mu} > 0$ such that for any fixed $\mu \in (0, \bar{\mu})$, the MLB method which updates $\{\lambda^{(k)}\}$ according to

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} / (c_i(x^{(k)})/\mu + 1), \tag{48}$$

generates a sequence of iterates $\{(x^{(k)}, \lambda^{(k)})\}$ which converges to a point satisfying the first-order optimality condition of (46) as $k \to \infty$.

In order to give the detailed description of the MLB method, we introduce the following parameters and definitions (see Breitfeld and Shanno [1] and [2]).

Let $T_1 = 10^{-4}$, $T_2 = 10^{-6}$, $\epsilon_0 = 10^{-5}$, $\sigma = 0.5$ or 0.1. Also, let

$$\begin{aligned} r &= -0.5 \log_{10}(T_1), \quad \epsilon_k = \max(\epsilon_0, 10^{-(k+r-1)}), \\ \nu_1^{(k)} &= \max\left\{ \frac{\|\nabla \mathcal{M}\|}{1 + \|x^{(k)}\|}, - \min_{i=1,\cdots,m} c_i(x^{(k)}) \right\}, \\ \nu_2^{(k)} &= \max\left\{ \nu_1^{(k)}, \frac{\sum_{i=1}^{m} \lambda_i^{(k+1)} |c_i(x^{(k)})|}{1 + \|x^{(k)}\|}, \frac{\|\nabla f(x^{(k)}) - \sum_{i=1}^{m} \lambda_i^{(k+1)} \nabla c_i(x^{(k)})\|_\infty}{1 + \|x^{(k)}\|} \right\}, \\ \nu_3^{(k)} &= \frac{|f(x^{(k)}) - f(x^{(k-1)})|}{1 + |f(x^{(k-1)})|}, \quad \nu_4^{(k)} = \frac{|\nu_2^{(k)} - \nu_2^{(k-1)}|}{\nu_2^{(k-1)}}. \end{aligned} \tag{49}$$

The algorithm below is a complete description of MLB method as implemented in [1] and [2].

Algorithm MLB:

Let $\xi \in (0, 1)$, $\mu^{(1)} > 0$, $\lambda^{(1)} > 0$, $\nu_1^{(0)}$, $\nu_2^{(0)}$, $\nu_3^{(0)}$ be given, and set $k = 1$.

```
For $k = 1, 2, 3, \ldots$, until $\nu_2^{(k-1)} < T_1$ or ($\nu_1^{(k-1)} < T_1$ and $\nu_3^{(k-1)} < T_2$), do
    1. Minimize the function (47) approximately,
       obtaining $x^{(k)}$ such that $\|\nabla \mathcal{M}(x^{(k)}, \mu^{(k)}, \lambda^{(k)})\| \le \epsilon_k$.
    2. Update $\lambda^{(k+1)}$ by (48). If $\nu_4^{(k)} < \xi$, set $\mu^{(k+1)} = \sigma \mu^{(k)}$.
       Increment $k$ by 1, and return to step 1.
End
```

In our implementation, we initialize $\mu^{(0)} = 10^{-2}$ and $\lambda_i^{(0)} = 1$ for $i = 1, \cdots, m$ as suggested in [1] and [18], respectively.

While Breitfeld and Shanno [1] have used a line search method to find $x^{(k)}$, we instead use a trust region method whose associated TR subproblems have quadratic objective functions with low-rank Hessian matrices obtained by means of limited-memory BFGS (L-BFGS) method (see Section 9.1 of [17])). The MNE method is then used to solve the resulting LRTR subproblems.

We next provide more details of how a low-rank approximation of the Hessian of $\mathcal{M}(\cdot, \mu^{(k)}, \lambda^{(k)})$ is computed in the context of solving nonlinear programming problems with simple bound constraints. For the purpose of this discussion, assume that the constraints of (46) are given by $c_i^l(x) \equiv x_i - l_i \ge 0$ for $i \in I_l$ and $c_i^u(x) \equiv u_i - x_i \ge 0$ for $i \in I_u$ where $I_l, I_u \subseteq \{1, \cdots, n\}$ are index sets corresponding to the lower and upper bound constraints, respectively. Let λ^l and λ^u be the Lagrange multipliers corresponding to the lower and upper bound constraints, respectively. We easily see from (47) that $\nabla^2 \mathcal{M}(x, \mu, \lambda) = \nabla^2 f(x) + \mu Q$, where

$$Q = \sum_{i \in I_l} \frac{\lambda_i^l e_i e_i^T}{(c_i^l(x) + \mu)^2} + \sum_{i \in I_u} \frac{\lambda_i^u e_i e_i^T}{(c_i^u(x) + \mu)^2}. \tag{50}$$

Since Q is a diagonal matrix and can be easily computed, it makes sense to just compute a low-rank approximation F of $\nabla^2 f(x)$ and use $H \equiv F + \mu Q$ as a low-rank approximation of $\nabla^2 \mathcal{M}(x, \mu, \lambda)$. We use the L-BFGS method to obtain a low-rank apprximation of $\nabla^2 f(x)$. This matrix H is used as the Hessian of the objective function of the LRTR subproblem (1) at the point x. Moreover, the other data of this subproblem is determined as $g \equiv \nabla \mathcal{M}(x, \mu, \lambda)$ and $M \equiv \mathrm{Diag}(v)$, where $v \in \Re^n$ is defined as $v_i = 1/\sqrt{Q_{ii}}$ if $i \in I_l \cup I_u$, and $v_i = 1$ if $i \notin I_l \cup I_u$.

4.2 Implementations of some problems from CUTEr

In this subsection, we will report the computational results obtained from the implementation of a specifc version of Algorithm MLB described in Subsection 4.1 and present the comparisons of our method with LANCELOT [5] on a collection of nonlinear programming problems from CUTEr [13] where only simple bound constraints are present.

All computations are performed on a Sun Ultra 10 workstation which has a single UltraSPARC IIi processor running at 440Mhz and 512MB of memory. The sixty test problems are selected from CUTEr. Seventeen of them have simple bound constraints and fixed variables. The remaining problems are unconstrained ones. Each row of Tables 1 and 2 gives the problem name, the number of variables, the number of bound constraints, the number of free variables, and the number of fixed variables on columns one through five, respectively.

Some computational results for our code are also presented in Tables 1 and 2. For each test problem, the total number of iterations performed by the MNE method is given in the sixth column, and the total number of LRTR subproblems (1) solved is given in the seventh column. The average iterations performed by the MNE method for each test problem is given in the eighth column, which is obtained by dividing the entry in the sixth column by the entry in the seventh clolumn.

From the eighth column of Tables 1 and 2, we observe that the average iterations performed by the MNE method are between 1.0 and 2.0 for almost all test problems, except the problems QRTQUAD (2.89) and FLETCBV2 (0). Moreover, the average of the entries in the eighth column over all test problems is 1.25, which is better than the average 1.6 obtained by Moré and Sorensen [15] for the NE method applied to the general TR subproblems (1). Hence, this indicates that the MNE method developed in our paper is efficient and robust for solving the low-rank version TR subproblem (1).

Our code MTR is written in ANSI C and LANCELOT is a FORTAN code. MTR and LANCELOT are both compiled under the default optimization. In MTR, we set the parameter $\sigma = 0.1$ for the Algorithm MLB and store 3 most recent vector pairs that provide curvature information for L-BFGS update. We select the same initial point x^0 as LANCELOT whenever it is strictly feasible; otherwise, we modify the infeasible components of this initial point x^0 in order to make it strictly feasible. We set up an upper bound of one hour computation time (or 3,600 seconds) per problem for both codes. We implement the version of LANCELOT which uses bfgs-approximate-second-derivatives, bandsolver-preconditioned-cg-solver, inexact-cauchy-point, two-norm-trust-region and all its other default settings. LANCELOT terminates when the infinity norm of the projected gradient is less than 10^{-5} or exceeds one hour computation time.

Tables 3 and 4 give the performance of MTR and LANCELOT. The objective function values of both methods are given in the second and third columns. The CPU times (in seconds) are given in the fourth and fifth columns. The iterations given in the sixth and seventh columns represent the total number of conjugate gradient iterations performed by LANCELOT and the total number of TR subproblems generated by MTR, respectively. The total numbers of function and gradient evaluations for both codes are given in the last two columns.

Based on the results of Tables 3 and 4, we now give some conclusions about the performance of the MTR and LANCELOT codes in terms of CPU time and the relative difference (rel. diff.) of the objective values obtained by MTR and LANCELOT. Among those test problems, MTR has:

i) better CPU time and better or close optimal value (i.e., rel. diff. $\leq$ 1.0e-5) for 60% of problems;

ii) better CPU time and worse optimal value (i.e., rel. diff. $>$ 1.0e-5) for 10% of problems;

iii) worse CPU time and better or close optimal value for 23% of problems;

iv) worse CPU time and worse optimal value for 7% of problems.

Based on the above comparison, we see that it is promising to solve large-scale problems with simple bound constraints using our approach.

Acknowledgements

The authors would like to thank Nicholas Gould, Jorge Nocedal, Dominique Orban and Philippe Toint for their kind help in the process of installing the softwares CUTEr and LANCELOT on our machine.

Table 1: The main test problems and some results of MTR

Problem	Var	Free	Bound	Fix	TrsIt	Trs	Ratio
ARGLINA	387	387	0	0	11	6	1.83
ARGLINB	387	387	0	0	25	21	1.19
ARGLINC	388	388	0	0	23	19	1.21
ARWHEAD	10000	10000	0	0	13	12	1.08
BDQRTIC	19999	19999	0	0	180	136	1.32
BRATU1D	13333	13331	0	2	6508	6021	1.08
BROWNAL	547	547	0	0	11	8	1.38
BRYBND	14288	14288	0	0	70	61	1.15
CHEBYQAD	316	0	316	0	383	342	1.12
COSINE	20000	20000	0	0	27	14	1.93
CURLY20	14295	14295	0	0	1597	1471	1.09
CVXBQP1	20000	0	20000	0	1200	1162	1.03
DIXMAANA	19998	19998	0	0	21	15	1.4
DIXMAANB	19998	19998	0	0	21	15	1.4
DIXMAANC	19998	19998	0	0	23	17	1.35
DIXMAAND	19998	19998	0	0	25	19	1.32
DIXMAANJ	19998	19998	0	0	626	617	1.01
DIXMAANK	19998	19998	0	0	419	413	1.01
DIXMAANL	19998	19998	0	0	450	444	1.01
EDENSCH	13333	13333	0	0	55	43	1.28
EIGENBLS	1122	1122	0	0	1695	1608	1.05
EIGENCLS	1122	1122	0	0	1559	1452	1.07
ENGVAL1	19999	19999	0	0	32	26	1.23
ERRINROS	50	50	0	0	819	772	1.06
FLETCBV2	13330	13330	0	0	0	0	0.00
FMINSRF2	19881	19881	0	0	846	841	1.01
FREUROTH	19999	19999	0	0	97	61	1.59
FMINSURF	19881	19881	0	0	2405	2362	1.02
GRIDGENA	6218	0	5560	658	111	98	1.13
HILBERTB	282	282	0	0	16	11	1.45
LIARWHD	19999	19999	0	0	43	32	1.34
LINVERSE	10001	5000	5001	0	110	85	1.29
LMINSURF	19881	19321	0	560	1672	1669	1.00
MANCINO	183	183	0	0	32	21	1.52
NLMSURF	19881	19321	0	560	3618	3579	1.01
NOBNDTOR	12544	6050	6050	444	413	346	1.19
NONDIA	16666	16666	0	0	19	18	1.06

Table 2: The main test problems and some results of MTR(cont'd)

Problem	Var	Free	Bound	Fix	TrsIt	Trs	Ratio
OBSTCLAE	12769	0	12321	448	296	254	1.17
OBSTCLBL	12769	0	12321	448	135	113	1.19
OBSTCLBM	12769	0	12321	448	139	120	1.16
OBSTCLBU	12769	0	12321	448	141	117	1.21
ODC	16900	16384	0	516	682	682	1.00
PENALTY1	19999	19999	0	0	151	116	1.30
PENALTY3	120	120	0	0	94	76	1.24
POWELLSG	20000	20000	0	0	39	31	1.26
POWER	20000	20000	0	0	779	740	1.05
PROBPENL	19999	0	19999	0	10	6	1.67
QRTQUAD	20000	10000	10000	0	104	36	2.89
SENSORS	199	199	0	0	51	42	1.21
SINQUAD	19999	19999	0	0	82	54	1.52
SPARSQUR	16666	16666	0	0	41	37	1.11
TOINTGSS	20000	20000	0	0	29	21	1.38
TORSION2	12544	0	12100	444	357	298	1.20
TORSION4	12544	0	12100	444	203	180	1.13
TORSION6	12544	0	12100	444	149	125	1.19
TORSIONB	12544	0	12100	444	290	269	1.08
TORSIOND	12544	0	12100	444	244	200	1.22
TORSIONF	12544	0	12100	444	143	123	1.16
TQUARTIC	19999	19999	0	0	29	21	1.38
VARDIM	19999	19999	0	0	101	96	1.05

Table 3: Comparison of the Two Methods on the main test problems

Problem	Obj Value		Time		Iter		Nfg	
Name	MTR	LAN	MTR	LAN	MTR	LAN	MTR	LAN
ARGLINA	3.870000e+02	3.870000e+02	0.78	38.48	6	3	14	14
ARGLINB	1.931252e+02	1.931252e+02	2.25	29.00	21	1	44	13
ARGLINC	1.951252e+02	1.951252e+02	2.05	28.89	19	2	40	13
ARWHEAD	0.000000e+00	0.000000e+00	5.52	7.86	12	1	26	12
BDQRTIC	8.008640e+04	8.008640e+04	82.48	21.89	136	17	274	40
BRATU1D	5.840393e+06	1.261025e+08	3600.00	3600.00	6021	4540	12044	16852
BROWNAL	2.572063e-11	2.615568e-11	0.98	81.35	8	5	18	14
BRYBND	1.325909e-12	9.511770e-14	26.00	22.94	60	54	122	78
CHEBYQAD	6.778464e-03	8.047224e-03	224.84	3600.00	342	2394	698	2103
COSINE	-2.000000e+04	-1.999900e+04	8.35	5.16	14	11	30	33
CURLY20	-1.434017e+06	-1.434021e+06	580.10	3600.00	1471	43104	2944	47
CVXBQP1	2.894382e-04	9.000450e+06	577.99	238.05	1162	9760	2338	14
DIXMAANA	1.000000e+00	1.000000e+00	8.13	7.83	15	21	32	34
DIXMAANB	1.000000e+00	1.000000e+00	8.46	9.14	15	17	32	40
DIXMAANC	1.000000e+00	1.000000e+00	9.51	9.79	17	20	36	42
DIXMAAND	1.000000e+00	1.000000e+00	10.57	14.96	19	33	40	66
DIXMAANJ	1.000001e+00	1.000000e+00	325.88	52.51	617	577	1236	88
DIXMAANK	1.000001e+00	1.000000e+00	217.59	46.26	413	389	828	87
DIXMAANL	1.000001e+00	1.000000e+00	233.88	38.11	444	354	890	84
EDENSCH	8.000128e+04	8.000128e+04	15.97	10.18	43	25	88	70
EIGENBLS	1.974164e-03	1.125255e-01	172.40	3600.00	1608	3424	3218	6914
EIGENCLS	2.454111e-02	7.176403e+02	158.73	3600.00	1452	3131	2906	9172
ENGVAL1	2.219933e+04	2.219933e+04	14.35	5.83	26	10	54	28
ERRINROS	3.990416e+01	3.990415e+01	0.83	0.17	772	110	1546	209
FLETCBV2	-5.001006e-01	-5.001006e-01	0.17	0.59	0	0	2	2
FMINSRF2	1.000002e+00	1.000000e+00	413.21	676.48	841	1454	1684	2341
FREUROTH	2.433123e+06	2.433123e+06	39.90	11.97	61	20	124	47
FMINSURF	1.000002e+00	7.466224e+00	1176.94	3600.00	2362	106	4726	211
GRIDGENA	2.352000e+04	2.352000e+04	16.89	10.70	98	126	210	48
HILBERTB	6.262020e-16	8.799383e-12	1.41	12.19	11	31	24	86
LIARWHD	2.130882e-13	5.932646e-14	17.03	13.34	32	43	66	64
LINVERSE	3.409000e+03	3.409000e+03	29.54	3600.00	85	12067	184	16626
LMINSURF	9.000355e+00	9.000000e+00	856.11	3059.60	1669	7650	3340	10168
MANCINO	2.117119e-16	9.787775e-20	17.58	22.76	21	12	44	45
NLMSURF	3.914874e+01	4.290681e+01	1760.28	3600.00	3579	7237	7160	11997
NOBNDTOR	-4.418497e-01	-4.418594e-01	130.18	32.54	346	402	706	72
NONDIA	1.344590e-15	3.812729e-19	7.93	2.63	18	4	38	12

Table 4: Comparison of the Two Methods on the main test problems(cont'd)

Problem	Obj Value		Time		Iter		Nfg	
Name	MTR	LAN	MTR	LAN	MTR	LAN	MTR	LAN
OBSTCLAE	1.894773e+00	1.894763e+00	105.41	389.08	254	6264	522	26
OBSTCLBL	7.285840e+00	7.285834e+00	45.48	165.98	113	2938	240	36
OBSTCLBM	7.285838e+00	7.285834e+00	48.09	291.40	120	4431	254	16
OBSTCLBU	7.285840e+00	7.285834e+00	47.26	58.55	117	1107	248	40
ODC	-1.137898e-02	-1.082613e-02	374.75	3600.00	682	6018	1366	14458
PENALTY1	1.985763e-01	1.985846e-01	61.19	1769.96	116	30	206	124
PENALTY3	9.998808e-04	7.704032e-02	5.63	3600.00	76	14981	154	25899
POWELLSG	2.023535e-07	3.498944e-05	13.52	4.90	31	19	64	40
POWER	2.723993e-07	1.546668e-08	305.59	1222.89	740	68	1482	80
PROBPENL	1.000000e-08	1.007191e-08	4.03	202.40	6	5	22	18
QRTQUAD	-5.375101e+10	-1.526995e+10	34.46	3600.00	36	7354	72	23
SENSORS	-8.336250e+03	-8.064563e+03	18.57	265.23	42	282	86	1008
SINQUAD	-1.040172e+08	1.509966e-05	34.91	208.92	54	781	110	586
SPARSQUR	9.098914e-10	3.009349e-06	17.47	19.87	37	68	76	46
TOINTGSS	1.000050e+01	1.000050e+01	11.79	8.34	21	23	44	48
TORSION2	-4.263058e-01	-4.263350e-01	117.85	340.90	298	5327	610	28
TORSION4	-1.212871e+00	-1.212881e+00	69.53	459.81	180	7990	374	20
TORSION6	-2.859436e+00	-2.859446e+00	49.44	474.08	125	9381	264	18
TORSIONB	-4.183918e-01	-4.184089e-01	106.21	267.73	269	3829	552	20
TORSIOND	-1.204444e+00	-1.204454e+00	82.87	497.59	200	8585	414	22
TORSIONF	-2.850759e+00	-2.850769e+00	50.39	485.76	123	9229	260	16
TQUARTIC	1.098962e-14	2.620727e-09	10.51	15.01	21	53	44	50
VARDIM	1.524712e-15	3.050031e-04	40.28	3600.00	96	2	194	108

References

[1] M. G. Breitfeld and D. F. Shanno: Preliminary computational experience with modified log-barrier functions for large-scale nonlinear programming. *Large Scale Optimization: State of the Art*, Kluwer Academics Publishers, 1994, pp. 45-67.

[2] M. G. Breitfeld and D. F. Shanno: Computational experience with penalty-barrier methods for nonlinear programming. *Annals of Operations Research*, Vol. 62(1996), pp. 439-463.

[3] J. R. Bunch and B. N. Parlett: Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, Vol. 8(1971), pp. 639-655.

[4] A. K. Cline, C. B. Moler, G. W. Stewart and J. H. Wilkinson: An estimate for the condition number of a matrix. *SIAM Journal on Numerical Analysis*, Vol. 16(1979), pp. 368-375.

[5] A. R. Conn, N. I. M. Gould, and P. L. Toint: *LANCELOT: a Fortran package for large-scale nonlinear optimization.* Springer Series in Computational Mathematics, vol. 17, Springer Verlag (Heidelberg, New York), 1992.

[6] A. R. Conn, N. I. M. Gould, and P. L. Toint: *Trust-region methods.* SIAM Publications, Philadelphia, Pennsylvania, USA, 2000.

[7] J. E. Dennis and H. H. W. Mei: Two new unconstrained optimization algorithms which use function and gradient values. *Journal of Optimization Theory and Application*, Vol. 28(1979), pp. 453-482.

[8] R. Fletcher: Practical Methods of Optimization. Unconstrained Optimization, Vol. 1, John Wiley, New York, 1980.

[9] C. Fortin and H. Wolkowicz: The trust region subproblem and semidefinite programming. *Optimization methods and Software*, 19(1), pp. 41-67, 2004.

[10] D. M. Gay: Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing*, Vol. 4(1981), No. 2, pp. 186-197.

[11] G. H. Golub and C. E. Van Loan: *Matrix Computations: Second Edition.* The John Hopkins University Press, Baltimore, MD 21211, 1989.

[12] N. I. M. Gould, S. Lucidi, M. Roma and P. L. Toint: Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, Vol. 9(1999), No. 2, pp. 504-525.

[13] N. I. M. Gould, D. Orban and P. L. Toint: General CUTEr documentation. Technical Report TR/PA/02/13, CERFACS, Toulouse (France), 2003.

[14] M. D. Hebden: An Algorithm for minimization using exact second derivatives. Atomic Energy Research Establishment, Report T. P. 515, Harwell, England.

[15] J. J. Moré and D. C. Sorensen: Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, Vol. 4(1983), No. 3, pp. 553-572.

[16] J. J. Moré: Recent developments in algorithms and software for trust region methods. *Mathematical Programming: The State of the Art*, Springer-Verlag, Berlin, Germany, 1983, pp. 258-287.

[17] J. Nocedal and S. J. Wright: *Numerical optimization.* Springer-Verlag, New York, USA, 1999.

[18] R. Polyak: Modified barrier functions (theory and methods). *Mathematical Programming*, 54(1992), pp. 177-222.

[19] M. J. D. Powell: A hybrid method for nonlinear equations. *in Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, ed, Gordon and Breach, New York, 1970.

[20] C. H. Reinsch: Smoothing by spline functions II. *Numerical Mathematics*, No. 16(1971), pp. 451-454.

[21] G. A. Shultz, R. B. Schnabel and R. H. Byrd: A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. *SIAM Journal on Numerical Analysis*, No. 22(1985), pp. 47-67.

[22] D. C. Sorensen: Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, No. 19(1982), pp. 404-426.

[23] T. Steihaug: The conjugate gradient method and trust regions in large-scale optimization. *SIAM Journal on Numerical Analysis*, No. 20(1983), pp. 626-637.

[24] P. L. Toint: Towards an efficient sparsity exploiting newton method for minimization. In Iain S. Duff, editor, *Sparse matrices and their uses*, Institute of Mathematics and its Applications Conference Series, pages xii+387, London, 1981. Academic Press Inc. [Harcourt Brace Jovanovich Publishers].

[25] Z. H. Wang, Z. W. Wen, and Y. Yuan: A subspace trust region method for large scale unconstrained optimization. In Y. Yuan, editor, *Numerical Linear Algebra and Optimization*, Proceeding of the 2003 International Conference on Numerical Optimization and Linear Algebra, Science Press, Beijing/New York, 2004, pp. 265-274.

[26] Y. Yuan: A Subspace Trust Region Algorithm. Presented at the Conference of Multiscale Optimization Methods and Applications, Center for Applied Optimization, University of Florida, USA, February 26-28, 2004.

[27] J. Zhang and C. Xu: A class of indefinite dogleg path methods for unconstrained minimization. *SIAM Journal on Optimization*, Vol. 9(1999), No. 3, pp. 646-667.