TECHNICAL
REPORT

A PRACTICAL METHOD FOR SOLVING LARGE-SCALE TRS

M.S. Apostolopoulou[a] D.G. Sotiropoulos[b]

C.A. Botsaris[c] P. Pintelas[d]

No. TR09-01

University of Patras
Department of Mathematics
GR-265 04 Patras, Greece.
http://www.math.upatras.gr/

[a]University of Patras, Department of Mathematics, GR-265 04 Patras, Greece (msa@math.upatras.gr)
[b]Ionian University, Department of Informatics, GR-491 00 Corfu, Greece (dgs@ionio.gr)
[c]University of Central Greece, Department of Regional Economic Development, GR-32100 Levadia, Greece(botsaris@otenet.gr)
[d]University of Patras, Department of Mathematics, GR-265 04 Patras, Greece (pintelas@math.upatras.gr)

Technical Report
No. TR09-01

A practical method for solving large-scale TRS

M.S. Apostolopoulou D.G. Sotiropoulos C.A. Botsaris P. Pintelas

March 2009

Abstract. We present a nearly-exact method for the large scale trust region subproblem (TRS) based on the properties of the minimal-memory BFGS method. Our study in concentrated in the case where the initial BFGS matrix can be any scaled identity matrix. The proposed method is a variant of the Moré-Sorensen method that exploits the eigenstructure of the approximate Hessian B, and incorporates both the standard and the hard case. The eigenvalues are expressed analytically, and consequently a direction of negative curvature can be computed immediately by performing a sequence of inner products and vector summations. Thus, the hard case is handled easily while the Cholesky factorization is completely avoided. An extensive numerical study is presented, for covering all the possible cases arising in the TRS with respect to the eigenstructure of B. Our numerical experiments confirm that the method is suitable for very large scale problems.

1 Introduction

We consider the following quadratic minimization problem:

$$\min_{d\in\mathbb{R}^n} \phi(d) = g^T d + \frac{1}{2} d^T B d, \quad \text{s.t.} \quad \|d\|_2 \leq \Delta, \tag{1}$$

where B is a $n \times n$ real symmetric (possibly indefinite) matrix, $g \in \mathbb{R}^n$, Δ is a positive scalar, and d is the real unknown n-vector. Problem (1) arises in many applications as: forming subproblems for constrained programming [3, 5], regularization methods for ill-posed problems [15], graph partitioning problems [11], large-scale nonlinear multicommodity flow problems [20], image restoration [23], etc. In particular, TRS is important in a class of methods for solving both convex and nonconvex nonlinear optimization problems, the trust-region algorithms [5]. At each iteration x_k of a trust-region algorithm, a trial step d_k is usually obtained by solving the quadratic subproblem (1) where $\phi_k(d)$ is an approximation to the objective function f, $g_k = \nabla f(x_k)$, $B_k \in \mathbb{R}^{n\times n}$ is either the Hessian or a (positive definite or indefinite) approximate Hessian of f at x_k, and $\Delta_k > 0$ is the trust region radius.

Various methods for calculating approximate solutions of TRS have been developed such as the dogleg method [21], the two-dimensional subspace minimization methods [4, 25] and the truncated CG methods [9, 28]. Nearly exact methods for solving (1) have been proposed by Gay [7], Sorensen [26], and Moré and Sorensen [16]. The method of nearly exact solutions uses Newton's method to find a root of a scalar function that is almost linear on the interval of interest. It is based on the Cholesky factorization for solving a linear system of the form $(B + \lambda I)\, d = -g$, where $I \in \mathbb{R}^{n\times n}$ is the identity matrix, $\lambda \geq 0$ is the Lagrange multiplier, and $B + \lambda I$ is positive semi-definite. In addition, in the so called *hard case*, a direction of negative curvature is required to be produced [16]. Therefore, this method can be very costly and even prohibitively expensive when it is applied in very large problems. This drawback has motivated the development of matrix-free methods that rely on matrix-vector products [10, 22, 23, 27].

In this work, we are concentrate on the nearly exact method proposed by Moré and Sorensen [16] for solving the TRS and we propose a variant of this method suitable for large scale problems. Motivated by the works of Birgin & Martínez [3] and Apostolopoulou et al [1], we use a minimal-memory BFGS method to approximate the matrix B. However, in [1] the authors assumed that the initial matrix $B^{(0)}$, is scaled with a specific parameter, the spectral parameter of Barzilai & Borwein [2]. Here we extend these results by studying the eigenstructure of minimal-memory BFGS matrices, in the case where the scaling factor is any non-zero real number. Analytical expressions for the eigenvalues are provided and a direction of negative curvature is computed by performing a sequence of inner products and vector summations. By this way, the method avoids the Cholesky factorization and is very practical for large scale problems.

Notation. Throughout the paper $\|\cdot\|$ denotes the Euclidean norm and n the dimension of the problem. The gradient of a function f at at a point x is denoted by $g(x)$. The Moore-Penrose generalized inverse of a matrix A is denoted by $A^\dagger$. For a symmetric $A \in \mathbb{R}^{n\times n}$, assume that $\lambda_1 \leq \ldots \leq \lambda_n$ are its eigenvalues sorted into non-decreasing order. We indicate that a matrix is positive semi-definite (positive definite) by $A \geq 0$ ($A > 0$, respectively).

2 Properties of the minimal-memory BFGS matrices

In this section we study the eigenstructure of the matrix B in the quadratic model (1). The computation of B is based on the minimal memory BFGS method [14, 17]. The Hessian approximation is updated using the BFGS formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \tag{2}$$

where in the vector pair $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$ is stored curvature information from the most previous iteration. We consider as the initial matrix $B^{(0)}$, the diagonal matrix $B^{(0)} = \theta I$, where $\theta \in \mathbb{R} \setminus \{0\}$ is a scaled parameter. Possible choices for the parameter θ could be: $\theta_{k+1} = 1$, which results the conjugate gradient method proposed by Shanno [24]; $\theta_{k+1} = y_k^T y_k / s_k^T y_k$ which constitutes the most successful scaling factor [18, pp. 200]; or $\theta_{k+1} = s_k^T y_k / s_k^T s_k$, the inverse of Barzilai & Borwein's [2] spectral parameter, which lies between the largest and the smallest eigenvalues of the average Hessian.

By setting $B^{(0)} = \theta_{k+1} I$ in (2), we obtain the minimal-memory BFGS update, defined as

$$B_{k+1} = \theta_{k+1} I - \theta_{k+1} \frac{s_k s_k^T}{s_k^T s_k} + \frac{y_k y_k^T}{s_k^T y_k}. \tag{3}$$

Note that in the quadratic model (1), the approximate Hessian matrix can be positive definite or indefinite. Hence, for the remaining of the paper we only assume that $\|B\|$ is bounded, that is, there is a positive constant M, such that $\|B_k\| \leq M < \infty$, for all k.

Theorem 2.1 *Suppose that one update is applied to the symmetric matrix $B^{(0)} = \theta I$, $\theta \in \mathbb{R} \setminus \{0\}$, using the vector pair $\{s_k, y_k\}$ and the BFGS formula. The characteristic polynomial of the symmetric matrix $B_{k+1} \in \mathbb{R}^{n \times n}$, defined in (3), has the general form*

$$p(\lambda) = (\lambda - \theta_{k+1})^{n-2} \left(\lambda^2 - \beta_1 \lambda + \beta_2\right), \tag{4}$$

where $\beta_1 = \theta_{k+1} + \left(y_k^T y_k\right) / \left(s_k^T y_k\right)$, and $\beta_2 = \theta_{k+1} \left(s_k^T y_k\right) / \left(s_k^T s_k\right)$. Moreover, if the vectors s_k and y_k are linearly independent then the smallest eigenvalue of B_{k+1} is distinct.

Proof. First we show that B_{k+1} has at most two distinct eigenvalues. To this end, we consider the matrix $\bar{B} = \theta_{k+1} I - \theta_{k+1} s_k s_k^T / s_k^T s_k$ with rank $(n-1)$. Applying the interlacing theorem [30, pp. 94–98] on $\bar{B}$, it is easy to see that $\bar{B}$ besides the zero eigenvalue, has one more eigenvalue equals to θ_{k+1} of multiplicity $(n-1)$. If B_{k+1} is positive definite, then the addition of the term $y_k y_k^T / s_k^T y_k$ on $\bar{B}$ yields

$$\lambda_n \geq \theta_{k+1} \geq \lambda_{n-1} \geq \theta_{k+1} \geq \ldots \geq \lambda_2 \geq \theta_{k+1} \geq \lambda_1 \geq 0,$$

where λ_i, $i = 1, \dots, n$ denote the eigenvalues of B_{k+1}. The above relation implies that

$$\lambda_2 = \ldots = \lambda_{n-1} = \theta_{k+1} \quad \text{and} \quad \lambda_n \geq \theta_{k+1} \geq \lambda_1. \tag{5}$$

Suppose now that B_{k+1} is indefinite. Then, if $\theta_{k+1} > 0$, from the interlacing theorem we have that

$$\theta_{k+1} \geq \lambda_n \geq \theta_{k+1} \geq \lambda_{n-1} \geq \theta_{k+1} \geq \ldots \geq \lambda_3 \geq \theta_{k+1} \geq \lambda_2 \geq 0 \geq \lambda_1,$$

which imply that

$$\lambda_3 = \ldots = \lambda_n = \theta_{k+1} \quad \text{and} \quad \theta_{k+1} \geq \lambda_2 \geq \lambda_1. \tag{6}$$

In different case ($\theta_{k+1} < 0$) we yield

$$0 \geq \lambda_n \geq \theta_{k+1} \geq \lambda_{n-1} \geq \theta_{k+1} \geq \ldots \geq \lambda_2 \geq \theta_{k+1} \geq \lambda_1.$$

Consequently,

$$\lambda_2 = \ldots = \lambda_{n-1} = \theta_{k+1} \quad \text{and} \quad \lambda_n \geq \theta_{k+1} \geq \lambda_1. \tag{7}$$

In all the above cases, it is obvious that B_{k+1} has at most two distinct eigenvalues and one eigenvalue equals to θ_{k+1} of multiplicity at least $(n-2)$. Denoting by λ_x and λ_y the two unknown distinct eigenvalues, the characteristic polynomial of B_{k+1} can be written as follows:

$$p(\lambda) = (\lambda - \theta_{k+1})^{n-2}\big[\lambda^2 - (\lambda_x + \lambda_y)\lambda + \lambda_x\lambda_y\big].$$

Taking into account that

$$\operatorname{tr}(B_{k+1}) = \sum_{i=1}^{n} \lambda_i = (n-2)\theta_{k+1} + \lambda_x + \lambda_y \tag{8}$$

and

$$\det(B_{k+1}) = \prod_{i=1}^{n} \lambda_i = \theta_{k+1}^{n-2}\lambda_x\lambda_y \tag{9}$$

we have that

$$p(\lambda) = (\lambda - \theta_{k+1})^{n-2}\big\{\lambda^2 - [\operatorname{tr}(B_{k+1}) - (n-2)\theta_{k+1}]\,\lambda + \det(B_{k+1})/\theta_{k+1}^{n-2}\big\}.$$

Using the well-known properties of the trace and determinant of matrices, we yield:

$$\operatorname{tr}(B_{k+1}) = \operatorname{tr}\left(\theta_{k+1}I - \theta_{k+1}\frac{s_k s_k^T}{s_k^T s_k} + \frac{y_k y_k^T}{s_k^T y_k}\right) = (n-1)\theta_{k+1} + \frac{y_k^T y_k}{s_k^T y_k}, \tag{10}$$

and

$$\begin{aligned}
\det(B_{k+1}) &= \det\left(\theta_{k+1}I - \theta_{k+1}\frac{s_k s_k^T}{s_k^T s_k} + \frac{y_k y_k^T}{s_k^T y_k}\right) \\
&= \det(\theta_{k+1}I)\det\left(I - \frac{s_k s_k^T}{s_k^T s_k} + \frac{y_k y_k^T}{\theta_{k+1}s_k^T y_k}\right) \\
&= \theta_{k+1}^n\left[\left(1 - \frac{s_k^T s_k}{s_k^T s_k}\right)\left(1 + \frac{y_k^T y_k}{\theta_{k+1}s_k^T y_k}\right) + \frac{s_k^T y_k}{\theta_{k+1}s_k^T s_k}\right] \\
&= \theta_{k+1}^{n-1}\frac{s_k^T y_k}{s_k^T s_k}
\end{aligned} \tag{11}$$

Hence,

$$\begin{aligned}\beta_1 &= \operatorname{tr}(B_{k+1}) - (n-2)\theta_{k+1} = \theta_{k+1} + (y_k^T y_k)/(s_k^T y_k),\\ \beta_2 &= \det(B_{k+1})/\theta_{k+1}^{n-2} = \theta_{k+1}(s_k^T y_k)/(s_k^T s_k),\end{aligned}$$

and relation (4) follows immediately.
It remains to show that when s_k and y_k are linearly independent, the smallest eigenvalue is distinct. Suppose that the vectors s_k and y_k are linearly independent and assume that B_{k+1} has at most one distinct eigenvalue, which implies that either $\lambda_x = \theta_{k+1}$ or $\lambda_y = \theta_{k+1}$. Combining relation (8) with (10), and (9) with (11), we have that $(s_k^T y_k)^2 = s_k^T s_k y_k^T y_k$. Hence, the vectors s_k and y_k are collinear, which contradicts the hypothesis. Thus, if the vectors are linearly independent, then B_{k+1} has exactly two distinct eigenvalues. Combining relations (5), (6) and (7), easily we can conclude that λ_1 is always distinct. □

Let us examine the case where the vectors s_k and y_k are collinear. Assuming that $y_k = \kappa s_k$, $\kappa \in \mathbb{R}$, B_{k+1} becomes

$$B_{k+1} = \theta_{k+1} I + (\kappa - \theta_{k+1})\frac{s_k s_k^T}{s_k^T s_k}. \tag{12}$$

Based on Theorem 2.1, the eigenvalues of B_{k+1} sorted into non-decreasing order are related as follows

$$\kappa = \lambda_1 \leq \lambda_2 = \lambda_3 = \ldots = \lambda_n = \theta_{k+1},$$

and $p(\lambda) = (\lambda - \theta_{k+1})^{n-1}(\lambda - \kappa)$. Easily can be verified that in this case the eigenvector corresponding to $\lambda_1 = \kappa$ equals s_k, while the generalized eigenvectors corresponding to θ_{k+1} are $u_i = (-s_k^i/s_k^1, 0, \ldots, 1, 0, \ldots, 0)^T$, where $i = 2, \ldots, n$, denotes the i-th component of s_k, and 1 is located in the i-th row of u_i. Therefore, in this case, no computational effort is required to obtain the eigenvector that corresponds to the smallest eigenvalue.

When s_k and y_k are linearly independent, Theorem 2.1 provides the information that the smallest eigenvalue is distinct. Based on this fact, we utilize the inverse power method [13] to compute the corresponding eigenvector since we know the exact eigenvalue. Given a non-zero starting vector u_0, inverse iteration generates a sequence of vectors u_i, generated recursively by the formula

$$u_i = (B - \sigma I)^{-1}\frac{u_{i-1}}{\|u_{i-1}\|}, \quad i = 1, 2, \ldots \tag{13}$$

where $\sigma = \lambda_1 + \epsilon$, λ_1 is a distinct eigenvalue of B and $\epsilon \to 0^+$. The sequence of iterates u_i converges to an eigenvector associated with an eigenvalue closest to σ. Usually, the starting vector u_0 is chosen to be the normalized vector $(1, 1, \ldots, 1)^T$. Moreover, if this particular eigenvalue λ_1 is known exactly, this method converges in a single iteration [13]. Consequently, when s_k and y_k are linearly independent, for being able to calculate the desirable eigenvector using the inverse iteration, we have to compute the inverse of $B_{k+1} + \lambda I$, for any $\lambda \in \mathbb{R}$.

Proposition 2.2 *Let Λ be the set of eigenvalues of the minimal-memory BFGS matrix B_{k+1} with opposite signs. Then, for any $\lambda \in \mathbb{R} \setminus \Lambda$, the inverse of $(B_{k+1} + \lambda I)$ has the general form*

$$(B_{k+1} + \lambda I)^{-1} = \frac{B_{k+1}^2 - [\beta'(\lambda) - \lambda]\, B_{k+1} + \beta(\lambda) I}{(\lambda + \theta_{k+1})(\lambda^2 + \beta_1\lambda + \beta_2)} \tag{14}$$

where $\beta(\lambda) = (\lambda+\beta_1)(\lambda+\theta_{k+1})+\beta_2$ while the constants β_1 and β_2 are defined in Theorem 2.1.

Proof. The addition of the term λI on B_{k+1} results that the eigenvalues of $(B_{k+1} + \lambda I)$ are $\mu_i = \lambda_i + \lambda$, where λ_i, $i = 1, \ldots, n$ are the eigenvalues of B_{k+1}. Using similar arguments as in proof of Theorem 2.1, we have that the characteristic polynomial of $(B_{k+1} + \lambda I)$ is expressed as follows:

$$q(\mu; \lambda) = [\mu - (\lambda + \theta_{k+1})]^{n-2} \left[\mu^2 - (\beta_1 + 2\lambda)\mu + \lambda^2 + \beta_1\lambda + \beta_2\right].$$

Hence, the minimal characteristic polynomial is

$$q_m(\mu; \lambda) = [\mu - (\theta_{k+1} - \lambda)] \left[\mu^2 - (\beta_1 + 2\lambda)\mu + \lambda^2 + \beta_1\lambda + \beta_2\right].$$

Applying the Caley-Hamilton theorem on $(B_{k+1} + \lambda I)$, we have that

$$q_m\,(B_{k+1} + \lambda I; \lambda) = 0,$$

which yields,

$$[B_{k+1} + \theta_{k+1} I] \left[(B_{k+1} + \lambda I)^2 - (\beta_1 + 2\lambda)\,(B_{k+1} + \lambda I) + \lambda^2 + \beta_1\lambda + \beta_2 I\right] = 0.$$

Multiplying both sides of the above equation by $(B_{k+1} + \lambda I)^{-1}$, we yield

$$\begin{aligned}(B_{k+1} + \lambda I)^{-1} &= \frac{1}{(\lambda^2 + \beta_1\lambda + \beta_2)(\theta_{k+1} + \lambda)}\Big\{(B_{k+1} + \lambda I)^2 \\ &\quad - (\beta_1 + \theta_{k+1} + 3\lambda)\,(B_{k+1} + \lambda I) \\ &\quad + \left[3\lambda^2 + 2\lambda(\beta_1 + \theta_{k+1}) + \theta_{k+1}\beta_1 + \beta_2\right] I\Big\} \\ &= \frac{B_{k+1}^2 - (\lambda + \beta_1 + \theta_{k+1})\, B_{k+1} + [(\lambda + \beta_1)(\lambda + \theta_{k+1}) + \beta_2]\, I}{(\lambda + \beta_1)(\lambda + \beta_2)(\theta_{k+1} + \lambda)}\end{aligned}$$

By setting $\beta(\lambda) = (\lambda + \beta_1)(\lambda + \theta_{k+1}) + \beta_2$, we obtain relation (14). $\square$

It is clear that combining inverse iteration (13) along with Proposition 2.2 we can immediately determine the expression of any eigenvector corresponding to a distinct eigenvalue of B.

Corollary 2.3 *Let $\sigma = \lambda + \epsilon$, $\epsilon \to 0^+$, be an ϵ-shifted distinct eigenvalue of B_{k+1}, and $u_0 = (1, 1, \ldots, 1)^T$. Then, the corresponding eigenvector to σ is given by*

$$u(\sigma) = \frac{c_1(\sigma)u_0 + c_2(\sigma)s_k + c_3(\sigma)y_k}{\sqrt{n}(\sigma^2 - \beta_1\sigma + \beta_2)(\theta_{k+1} - \sigma)}, \tag{15}$$

where the coefficients of the vectors u_0, s_k and y_k are defined as follows:

$$\begin{aligned} c_1(\sigma) &= \sigma^2 - \beta_1\sigma + \beta_2, \\ c_2(\sigma) &= \left[(\beta_1 - \sigma)\sum_{i=1}^{n} s_k^i - \sum_{i=1}^{n} y_k^i\right]\frac{\theta_{k+1}}{s_k^T s_k}, \quad \textit{and} \\ c_3(\sigma) &= \frac{\sigma}{s_k^T y_k}\sum_{i=1}^{n} y_k^i - \frac{\theta_{k+1}}{s_k^T s_k}\sum_{i=1}^{n} s_k^i. \end{aligned}$$

Proof. From Proposition 2.2, by setting $\lambda = -\sigma$ in (14) we obtain the expression of $(B - \sigma I)^{-1}$. Therefore, the application of inverse iteration (13) yields

$$u(\sigma) = (B - \sigma I)^{-1}\frac{u_0}{\|u_0\|} = \frac{B_{k+1}^2 u_0 - [\beta'(-\sigma) + \sigma]\, B_{k+1}u_0 + \beta(-\sigma)u_0}{(-\sigma + \theta_{k+1})(\sigma^2 - \beta_1\sigma + \beta_2)\|u_0\|}. \tag{16}$$

The vectors $B_{k+1}u_0$ and $B_{k+1}^2 u_0$ can be obtained by the iterative form

$$B_{k+1}w_i = \theta_{k+1}w_i - \theta_{k+1}\frac{s_k^T w_i}{s_k^T s_k}s_k + \frac{y_k^T w_i}{s_k^T y_k}y_k \quad i = 0, 1, \tag{17}$$

with $w_0 = u_0$. Using relation (17) and the fact that $\|u_0\| = \sqrt{n}$, after some algebraic computations, relation (16) is reduced to (15). □

3 Solving the TRS using the minimal-memory BFGS method

In this section we apply the results of Section 2 for solving the large scale TRS. A global solution to the TRS (1) is characterized by the following well known theorem (Gay [7], Sorensen [26], Moré & Sorensen [16]):

Theorem 3.1 *A feasible vector d^* is a solution to (1) with corresponding Lagrange multiplier λ^* if and only if d^*, λ^* satisfy $(B + \lambda^* I)\, d^* = -g$, where $B + \lambda^* I$ is positive semi-definite, $\lambda^* \geq 0$, and $\lambda^*(\Delta - \|d^*\|) = 0$.*

From the above Theorem we can distinguish two cases, the *standard* and the *hard* case. In the standard case, the solution of the TRS can be summarized as follows:

1. If $\lambda_1 > 0$ (the matrix is positive definite), then
   - a. if $\|B^{-1}g\| \leq \Delta$, then $\lambda^* = 0$ and $d^* = B^{-1}g$.
   - b. if $\|B^{-1}g\| > \Delta$, then for the unique $\lambda^* \in (0, \infty)$ such that $\|\left(B + \lambda^* I\right)^{-1} g\| = \Delta$, $d^* = -\left(B + \lambda^* I\right)^{-1} g$.

2. If $\lambda_1 \leq 0$ (the matrix is indefinite), then if $\|B^{-1}g\| > \Delta$, for the unique $\lambda^* \in (-\lambda_1, \infty)$ such that $\|\left(B + \lambda^* I\right)^{-1} g\| = \Delta$, $d^* = -\left(B + \lambda^* I\right)^{-1} g$.

As we can observe, the optimal non-negative Lagrange multiplier λ^* belongs to the open interval $(-\lambda_1, \infty)$. When $\lambda^* \neq 0$, the TRS (1) has a boundary solution, i.e., $\|d^*\| = \Delta$. In this case, the given n-dimensional constrained optimization problem is reduced into a zero-finding problem in a single scalar variable λ, namely, $\|d(\lambda)\| - \Delta = 0$, where the trial step $d(\lambda)$ is a solution of the linear system $(B + \lambda I)\, d = -g$. Moré and Sorensen [16] have proved that it is more convenient to solve the equivalent equation (secular equation) $\phi(\lambda) \equiv 1/\Delta - 1/\|d(\lambda)\| = 0$, that exploits the rational structure of $\|d(\lambda)\|^2$. The solution λ of the secular equation is based on Newton's method,

$$\lambda^{\ell+1} = \lambda^{\ell} + \frac{\|d(\lambda)\|}{\|d(\lambda)\|'} \left(\frac{\Delta - \|d(\lambda)\|}{\Delta} \right), \quad \ell = 0, 1, 2, \ldots. \tag{18}$$

In Newton's iteration (18) a safeguarding is required to ensure that a solution is found. The safeguarding depends on the fact that ϕ is convex and strictly decreasing in $(-\lambda_1, \infty)$. It ensures that $-\lambda_1 \leq \lambda^{\ell}$, and therefore $B + \lambda^{\ell} I$ is always semi-positive definite [16].

The hard case occurs when B is indefinite and g is orthogonal to every eigenvector corresponding to the most negative eigenvalue λ_1 of the matrix. In this case, there is no $\lambda \in (-\lambda_1, \infty)$ such that $\|\left(B + \lambda I\right)^{-1} g\| = \Delta$. The optimal Lagrange multiplier is $\lambda^* = -\lambda_1$, the matrix $(B - \lambda_1 I)$ is singular, and a direction of negative curvature must be calculated in order to ensure that $\|d\| = \Delta$. Consequently, in the hard case, the optimal solution to the TRS is $d^* = -\left(B - \lambda_1 I\right)^{\dagger} g + \tau u$, where $\tau \in \mathbb{R}$ is such that $\| -\left(B - \lambda_1 I\right)^{\dagger} g + \tau u\| = \Delta$ and u is a normalized eigenvector corresponding to λ_1. Moré and Sorensen [16] have showed that the choice of τ that ensures $\|d\| = \Delta$, is

$$\tau = \frac{\Delta^2 - \|p\|^2}{p^T u_1 + \operatorname{sgn}(p^T u)\sqrt{(p^T u)^2 + \Delta^2 - \|p\|^2}}, \tag{19}$$

where $p = -\left(B - \lambda_1 I\right)^{\dagger} g$.

The above analysis indicates that for been able to solve the TRS (1), we should compute the smallest eigenvalue λ_1 of B, the inverse of $B + \lambda I$ for computing the trial step $d(\lambda)$, and, in the hard case, the unit eigenvector u corresponding to λ_1. Moreover, the definition of a simple safeguarding scheme in Newton's iteration (18) requires also the largest eigenvalue λ_n.

In Section 2, we have studied all these aspects in detail. The Algorithm 1, handles both the standard and the hard case and computes a nearly exact solution of the subproblem (1) without using the Cholesky factorization. Moreover, the knowledge of the extreme eigenvalues, results in a straightforward safeguarding procedure for λ.

Algorithm 1 *(Computation of the trial step)*

Step 1: *Compute the eigenvalues λ_i of B; given $\epsilon \to 0^+$, set the bounds $\lambda_L := \max(0, -\lambda_1 + \epsilon)$ and $\lambda_U := \max|\lambda_i| + (1+\epsilon)\|g\|/\Delta$.*

Step 2: *If $\lambda_1 > 0$, then initialize λ by setting $\lambda := 0$ and compute $d(\lambda)$; if $\|d\| \le \Delta$ stop; else go to Step 4.*

Step 3: *Initialize λ by setting $\lambda := -\lambda_1 + \epsilon$ such that $B + \lambda I$ is positive definite and compute $d(\lambda)$;*

a. if $\|d\| > \Delta$ go to Step 4;

b. if $\|d\| = \Delta$ stop;

c. if $\|d\| < \Delta$ compute τ and u_1 such that $\| -(B+\lambda I)^{-1} g + \tau u_1\| = \Delta$; set $d := d + \tau u_1$ and stop;

Step 4: *Use Newton's method to find $\lambda \in [\lambda_L, \lambda_U]$ and compute $d(\lambda)$;*

Step 5: *If $\|d\| = \Delta$ stop; else update λ_L and λ_U such that $\lambda_L \le \lambda \le \lambda_U$ and go to Step 4.*

In Step 1 of Algorithm 1 the eigenvalues are computed by solving the characteristic equation $p(\lambda) = 0$, defined in relation (4). Obviously, the two possibly distinct eigenvalues are $\left(-\beta_1 \pm \sqrt{\beta_1^2 - 4\beta_2}\right)/2$ while the eigenvalue θ_{k+1} is multiple. The safeguarding scheme required for Newton's iteration (18) uses the parameters λ_L and λ_U such that $[\lambda_L, \lambda_U]$ is an interval of uncertainty which contains the optimal λ^*. The initial bounds λ_L and λ_U have been proposed by Nocedal and Yuan [19]. Clearly, the lower bound λ_L is greater than $-\lambda_1$, which ensures that $B + \lambda I$ is always positive definite.

Note that in Steps 2, 3 and 4, the trial step $d(\lambda)$ is computed by solving the linear system $(B + \lambda I)\, d(\lambda) = -g$. Using Proposition 2.2, along with relations (17), the trial step is computed immediately by the expression

$$d(\lambda) = -\frac{c_g(\lambda) g + c_s(\lambda) s_k + c_y(\lambda) y_k}{(\lambda^2 + \beta_1 \lambda + \beta_2)(\lambda + \theta_{k+1})}, \tag{20}$$

where the coefficients are defined as follows

$$\begin{aligned} c_g(\lambda) &= \lambda^2 + \beta_1 \lambda + \beta_2, \\ c_s(\lambda) &= \left[(\lambda + \beta_1)\, s_k^T g_{k+1} - y_k^T g_{k+1}\right] \frac{\theta_{k+1}}{s_k^T s_k}, \quad \text{and} \\ c_y(\lambda) &= -\frac{y_k^T g_{k+1}}{s_k^T y_k}\lambda - \frac{s_k^T g_{k+1}}{s_k^T s_k}\theta_{k+1}. \end{aligned}$$

In Step 3(c), scalar τ is computed by Eq. (19), while for the computation of the eigenvector u_1 corresponding to λ_1, we take into account the relation between the vectors s_k and y_k. If the vectors s_k and y_k are linearly independent, then $u_1 = u/\|u\|$, where u is computed by means of relation (15) in Corolarry 2.3. In different case, u_1 equals to the normalized vector s_k, as explained in Section 2.

In Steps 4 and 5, Newton's iteration (18) is utilized for computing the Lagrange multiplier λ. In (18), we have that

$$\|d(\lambda)\|' = -d(\lambda)^T \left(B + \lambda I\right)^{-1} d(\lambda)/\|d(\lambda)\|,$$

and thus, the iterative scheme becomes

$$\lambda^{\ell+1} = \lambda^{\ell} + \frac{\|d(\lambda)\|^2}{d(\lambda)^T \left(B_{k+1} + \lambda I\right)^{-1} d(\lambda)} \left(\frac{\|d(\lambda)\| - \Delta}{\Delta}\right), \ \ell = 0, 1, \ldots \tag{21}$$

Denoting by $\Pi = d(\lambda)^T \left(B_{k+1} + \lambda I\right)^{-1} d(\lambda)$ the denominator in (21), and taking into account that $B_{k+1}\, d(\lambda) = -\left[\lambda\, d(\lambda) + g_{k+1}\right]$ holds, then by Proposition 2.2 we obtain that

$$\Pi = \frac{\left[\beta(\lambda) + \lambda\beta'(\lambda)\right] \|d(\lambda)\|^2 + \left[\beta'(\lambda) + \lambda\right] d(\lambda)^T g_{k+1} + \|g_{k+1}\|^2}{(\lambda + \theta_{k+1})(\lambda^2 + \beta_1\lambda + \beta_2)}. \tag{22}$$

If s_k and y_k are collinear and relation $y_k = \kappa s_k$ holds, then relations (20) and (22) are reduced to

$$d(\lambda) = -\frac{1}{(\lambda + \theta_{k+1})}\, g_{k+1} + \frac{(\kappa - \theta_{k+1}) s_k^T g_{k+1}}{(\lambda + \kappa)(\lambda + \theta_{k+1}) s_k^T s_k}\, s_k,$$

and

$$\Pi = \frac{(2\lambda + \beta_1)\|d(\lambda)\|^2 + d(\lambda)^T g_{k+1}}{\lambda^2 + \beta_1\lambda + \beta_2},$$

respectively.

4 Numerical experiments and analysis

For illustrating the behavior of the proposed method in both the standard and the hard case, we use randomly generated TRS instances with dimensions from 100 up to 15 000 000 variables. The experiments are consisted by medium-size problems with dimensions $n = 100, 500, 1000$, by larger-size problems ($n = 10^4, 10^5, 10^6$), and by very large problems ($n = 2 \cdot 10^6, 3 \cdot 10^6, \ldots, 15 \cdot 10^6$).

For each dimension n, 1000 random instances of the TRS (1) were generated as follows. The coordinates of the vectors g, s, and y were chosen independently from uniform distributions in the interval $(-10^2, +10^2)$. For the numerical testing we implemented Algorithm 1 (`MLBFGS`), using FORTRAN 90 and all numerical experiments were performed on a Pentium 1.86 GHz personal computer with 2GB of RAM running Linux operating system.

We compared our method with the `GQTPAR`, and the `GLTR` algorithm. The `GQTPAR` algorithm is available as subroutine `dgqt` in the MINPACK-2 package[1], and is based on the

[1]Available from `ftp://info.mcs.anl.gov/pub/MINPACK-2/gqt/`

ideas described in Moré and Sorensen [16] for computing a nearly exact solution of (1). The method uses the Cholesky factorization for calculating the required derivatives during the computation of the Newton step for λ, as well as for the safeguarding and updating of the Lagrange multiplier. The `GLTR` algorithm proposed by Gould et al. [9], is available as the FORTRAN 90 module `HSL_VF05`[2] in the Harwell Subroutine Library [12]. This method is based on a Lanczos tridiagonalization of the matrix B and on the solution of a sequence of problems restricted to Krylov subspaces of $\mathbb{R}^n$. It is an alternative to the Steihaug-Toint algorithm [28, 29] and it computes an approximate solution of (1). The algorithm requires only matrix-vector multiplications while it exploits the sparsity of B.

Note that the same set of random instances was used for each algorithm. We consider that a TRS instance was successfully solved, if a solution satisfying $\|(B + \lambda I)\,d - g\| \leq 10^{-3}$ was computed for both the standard and the hard case. The maximum number of Newton's iterations allowed was 200 and the trust region radius was fixed as $\Delta = 10$. For dimensions larger than 1000, results are reported only for `MLBFGS` and `GLTR` algorithms, due to the storage requirements of `GQTPAR` for factoring B. Moreover, the `GLTR` algorithm is not reported in the comparison results for the hard case, since it computes an approximate solution and not a nearly exact solution of the TRS.

For each dimension, a table summarizing the performance of all algorithms for simulations that reached solution is presented. The reported metrics are Newton's iterations (*it*) and solution accuracy (*acc*). The descriptives statistics over 1000 experiments are: the mean and standard deviation, as well as the minimum and maximum values. To show both the efficacy and accuracy of our method, we have used the performance profile proposed by Dolan and Moré [6]. The performance profile plots the fraction of problems for which any given method is within a factor of the best solver. The left axis of the plot shows the percentage of the problems for which a method is the fastest (efficiency). The right side of the plot gives the percentage of the problems that were successfully solved by each of the methods (robustness). The reported performance profiles have been created using the Libopt environment [8], an excellent set of tools written in Perl.

4.1 Random experiments in the standard case

Firstly, we present the behavior of the proposed method in the standard case. For each dimension between $n = 100$ and $n = 10^6$, we have considered the following cases:

(a) s_k, y_k are linearly independent and $\theta_{k+1} = 1$;

(b) s_k, y_k are linearly independent and $\theta_{k+1} = (y_k^T y_k)/(s_k^T y_k)$;

(c) s_k, y_k are collinear and $\theta_{k+1} = 1$;

(d) s_k, y_k are collinear and $\theta_{k+1} = (y_k^T y_k)/(s_k^T y_k)$.

[2]Available from `http://www.hsl.rl.ac.uk`

If $\theta_{k+1} = 1$, then the initial matrix $B^{(0)}$ is the identity matrix. In different case, the scalar parameter θ is chosen to be the one proposed in the literature [18] for scaling the initial matrix in the BFGS method. The above cases were considered for analyzing the behavior of the proposed method in all cases concerning the vector pair $\{s, y\}$, as well as the scaling factor. As a total, we have 12 000 medium size experiments ($n = 100$, 500, and 1000) and 12 000 large size experiments ($n = 10^4$, 10^5, and 10^6).

		Mean		Std. Deviation		Minimum		Maximum	
Method	*succ*	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	100.0%	1.84	1.19E-13	1.38	3.40E-12	0	1.80E-14	8	2.04E-10
GQTPAR	99.6%	3.75	8.65E-07	2.85	5.46E-05	1	3.12E-14	21	3.44E-03
GLTR	91.4%	3.68	1.27E-12	1.56	7.71E-11	1	2.73E-14	7	4.29E-09

Table 1: Standard Case: Comparative results for $n = 100$ over 4000 experiments

		Mean		Std. Deviation		Minimum		Maximum	
Method	*succ*	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	100.0%	1.55	4.10E-13	1.10	9.91E-12	0	5.51E-14	7	5.64E-10
GQTPAR	99.9%	3.36	7.57E-12	2.60	5.21E-11	1	8.13E-14	24	1.96E-09
GLTR	85.9%	3.85	6.33E-12	1.45	1.23E-10	1	5.94E-14	7	4.55E-09

Table 2: Standard Case: Comparative results for $n = 500$ over 4000 experiments

		Mean		Std. Deviation		Minimum		Maximum	
Method	*succ*	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	100.0%	1.45	2.55E-13	1.03	5.19E-12	0	7.97E-14	7	3.13E-10
GQTPAR	99.7%	3.20	4.17E-11	2.45	5.62E-10	1	1.11E-13	23	2.91E-08
GLTR	86.5%	3.83	4.41E-10	1.46	2.54E-08	1	8.14E-14	7	1.49E-06

Table 3: Standard Case: Comparative results for $n = 1\,000$ over 4000 experiments

Tables 1-6 summarize the results for the experiments with dimensions from $n = 100$ up to $n = 10^6$. In all tables, the first column denotes the method, while the second column

		Mean		Std. Deviation		Minimum		Maximum	
Method	*succ*	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	100.0%	1.31	5.77E-13	1.06	7.53E-12	0	2.78E-13	8	7.41E-10
GLTR	86.3%	3.84	1.37E-08	1.47	6.93E-07	1	2.79E-13	7	4.04E-05

Table 4: Standard Case: Comparative results for $n = 10\,000$ over 4000 experiments

		Mean		Std. Deviation		Minimum		Maximum	
Method	*succ*	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	100.0%	1.14	4.59E-11	1.04	9.48E-10	0	8.19E-13	9	3.88E-08
GLTR	87.5%	3.87	7.06E-07	1.50	1.07E-05	1	8.27E-13	7	2.51E-04

Table 5: Standard Case: Comparative results for $n = 100\,000$ over 4000 experiments

		Mean		Std. Deviation		Minimum		Maximum	
Method	*succ*	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	100.0%	1.00	7.07E-10	1.07	1.22E-08	0	2.74E-12	9	6.38E-07
GLTR	86.8%	3.74	3.74E-05	1.49	1.41E-04	1	2.75E-12	9	8.59E-04

Table 6: Standard Case: Comparative results for $n = 1\,000\,000$ over 4000 experiments

reports the percentage of the successfully (*succ*) solved instances within the required accuracy (10^{-3}). In Tables 4-6 the `GQTPAR` method was omitted from the comparisons due to memory limitations.

The results in all tables show that the `MLBFGS` algorithm has solved all problems successfully, in all dimensions, in contrast with the other two algorithms that failed to solve all experiments within the required accuracy. Moreover, `MLBFGS` outperforms both `GQTPAR` and `GLTR` algorithms in terms of Newton's iterations, since in all statistical metrics it exhibited the lowest value. Actually, on average the `MLBFGS` required about 1/3 of Newton's iterations for solving the TRS compared with the other two algorithms. Similar observations can be made for the solution accuracy. The proposed method presented the highest solution accuracy among the other two methods in all dimensions. In medium-size problems `MLBFGS` algorithm has solved the TRS instances with average solution accuracy of order 10^{-13}. In larger-size problems, the numerical experiments show that the proposed method can solve the TRS with average accuracy of order 10^{-10}. Overall, it is quite remarkable that even in the high dimensional instances, MLBFGS provided solutions with high accuracy, while in many problems only one Newton's iteration was required for solving the TRS.

Referring to the four cases mentioned above, regarding the choice of θ and the relation between the vectors s and y, based on the numerical experience, it is worth to notice that the choice of the scaling factor had no influence in the behavior of all algorithms. On the other hand, the linear independency of the vector pair $\{s, y\}$ have effected the algorithms. We have observed that in cases *(c)* and *(d)*, where the vectors are collinear, MLBFGS presented the highest accuracy and the lowest number of Newton's iterations. In contrast, GLTR presented the most failures, the lowest solution accuracy and the maximum number of Newton's iterations. Based on the above observations, for testing the behavior of our method in dimensions between $2 \cdot 10^6$ and $15 \cdot 10^6$ we have considered only the first case, which is the easiest for the GLTR algorithm.

The numerical results of Tables 1-6 are illustrated in Figures 1 and 2. In Figure 3, the performance of MBFGS and GLTR algorithms for $n = 10^6, 2 \cdot 10^6, 3 \cdot 10^6, \ldots, 15 \cdot 10^6$ is presented. Each figure contains three subfigures where the performance profiles based on Newton's iterations, solution accuracy and CPU time (in seconds) are presented.

Figure 1(PAGE 19) presents the performance of all algorithms for the medium-size instances. Clearly, the most robust and efficient method it appears to be the MBFGS, since it presents the best performance regarding all performance metrics. Obviously, the most time consuming method is GQTPAR (Fig. 1(c)), since this method requires the Cholesky factorization. Figure 2 (PAGE 20) illustrates the performance metrics for larger-size problems. According to Figures 2(a) and 2(b), the probability of the MBFGS algorithm to be the best, in terms of Newton's iterations and solution accuracy, is about 100%. When CPU time is used as the performance metric, Figure 2(c) indicates that the inner products and vector summations that were used for obtaining a nearly exact solution to the quadratic subproblem, have costed much less than the Lanczos-based tridiagonalization of B. Therefore, the MBFGS method is much preferable to solve large scale problems than the GLTR, since it appears to be the most efficient. The performance metrics for very large problems are presented in Figure 3 (PAGE 21). Clearly, the MBFGS algorithms outperforms the GLTR algorithm in terms of all performance metrics. Figures 3(a) and 3(c) show that the MBFGS algorithm is the best solver, with probability 100%. In addition, Figure 3(c) clearly shows that MBFGS is the winner, since the method takes advantage of the properties of B. This observation indicates that the proposed method is very practical when the dimension of the problem is very large.

4.2 Random experiments in the hard case

In order to create instances for the hard case, Matlab's routine eigs was used to compute the smallest eigenvalue λ_1 and the corresponding eigenvector u. We initialized the trust-region radius by $\Delta = 10.0\Delta_{hc}$, where $\Delta_{hc} = \| (B - \lambda_1 I)^{\dagger} g \|$. For being g perpendicular to u, the vector g was set $g = (-u(n)/u(1), 0, \ldots, 0, 1)^T$. Since the GLTR algorithm computes an approximate solution to the TRS, we only compared the MBFGS and GQTPAR algorithms. Due to the storage requirements of GQTPAR, we have tested the algorithms in problems with dimensions $n = 100, 500$, and 1000.

For each dimension we have considered only the cases *(a)*, *(b)* and *(c)*, therefore, we have 9 000 medium-size experiments. According to Theorem 2.1, the case *(d)* indicates that B has one eigenvalues equals θ, of multiplicity n. From relation (12) easily can be verified that in this case, B is a scaled diagonal matrix, with eigenvectors equal to $e_i = (0, 0, \ldots, 1, 0, \ldots, 0)^T$, where 1 is in the ith component. We recall that the hard case occurs when all the eigenvectors corresponding to the smallest eigenvalue are perpendicular to g. This implies that in the case *(d)*, the gradient must be zero, and hence we did not examine this case.

Tables 7, 8, and 9, summarize the results for the dimensions 100, 500, and 1000, respectively. All three tables show that the behavior of both `MLBFGS` and `GQTPAR` algorithms is similar in terms of solution accuracy. On the other hand, the `MLBFGS` algorithm significantly outperforms the `GQTPAR` algorithm in terms of Newton iterations. As we can see, the proposed method solved the subproblem in the hard case in a single step, since no Newton's iterations are required. This observation indicates that the hard case can be handled by the

		Mean		Std. Deviation		Minimum		Maximum	
Method	*succ*	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	100.0%	0	9.13E-06	0.0	1.84E-04	0	8.24E-11	0	7.15E-03
GQTPAR	96.8%	27.52	1.41E-06	3.3	1.56E-05	18	6.45E-14	43	6.88E-04

Table 7: Hard Case: Comparative results for $n = 100$ over 3000 experiments

		Mean		Std. Deviation		Minimum		Maximum	
Method	*succ*	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	100.0%	0	9.13E-06	0.0	2.11E-04	0	9.31E-11	0	7.94E-03
GQTPAR	96.0%	29.10	1.25E-06	3.7	1.58E-05	17	1.22E-08	52	7.97E-04

Table 8: Hard Case: Comparative results for $n = 500$ over 3000 experiments

		Mean		Std. Deviation		Minimum		Maximum	
Method	Succ.	*it*	*acc*	*it*	*acc*	*it*	*acc*	*it*	*acc*
MBFGS	99.9%	0	1.23E-05	0.0	2.01E-04	0	1.86E-11	0	6.69E-03
GQTPAR	98.5%	27.44	2.50E-06	3.4	4.06E-05	19	2.58E-08	55	1.67E-03

Table 9: Hard Case: Comparative results for $n = 1\,000$ over 3000 experiments

MLBFGS algorithm easily and fast.

In Figure 4 (Page 22) are presented the results from Tables 7-9. The performance metrics are the solution accuracy and the CPU time (in seconds). The performance based on Newton's iteration is not reported, since the MLBFGS required zero iterations for solving each subproblem. Figure 4(a) indicates that the MLBFGS algorithm outperforms the GQTPAR algorithm, in terms of solution accuracy. The same observation holds and for Figure 4(b), as it was expected, due to storage requirements of GQTPAR.

5 Conclusions

We have studied the eigenstructure of the minimal memory BFGS matrices in the general case where the initial matrix is a scaled identity matrix. Our theoretical results have been applied for the solution of the trust region subproblem. Based on the fact that the algorithm uses only inner products and vector summations, the proposed method is suitable and practical for solving large scale TRS. Our numerical experiments have shown that the proposed method can easily handle both the standard and the hard case. It provides relatively fast solutions with high accuracy and small number of iterations. Moreover, it turns out that the hard case is the “easy” case for the proposed method.

References

[1] M.S. Apostolopoulou, D.G. Sotiropoulos, and P. Pintelas. Solving the quadratic trust-region subproblem in a low-memory BFGS framework. *Optimization Methods and Software*, 23(5):651–674, 2008.

[2] J. Barzilai and J.M. Borwein. Two point step size gradient method. *IMA J. Numer. Anal.*, 8:141–148, 1988.

[3] E.G. Birgin and J.M. Martínez. Structured minimal-memory inexact quasi-newton method and secant preconditioners for augmented lagrangian optimization. *Comput. Optim. Appl.*, 39(1):1–16, 2008.

[4] R.H. Byrd, R.B. Schnabel, and G.A. Shuldz. Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Mathematical Programming*, 40:247–263, 1988.

[5] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, PA, USA, 2000.

[6] E. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[7] D.M. Gay. Computing optimal locally constrained steps. *SIAM J. Sci. Stat. Comput.*, 2(2):186–197, 1981.

[8] J.Ch. Gilbert and X. Jonsson. LIBOPT - an environment for testing solvers on heterogeneous collections of problems. *Submitted to ACM Transactions on Mathematical Software*, January 2009.

[9] N.I.M. Gould, S. Lucidi, M. Roma, and Ph.L. Toint. Solving the trust-region subproblem using the Lanczos method. *SIAM J. Optim.*, 9(2):504–525, 1999.

[10] W.W. Hager. Minimizing a quadratic over a sphere. *SIAM J. Optim.*, 12(1):188–208, 2001.

[11] W.W. Hager and Y. Krylyuk. Graph partitioning and continuous quadratic programming. *SIAM J. Discret. Math.*, 12(4):500–523, 1999.

[12] HSL 2007. A collection of Fortran codes for large scale scientific computation. 2007. Available from: `www.hsl.rl.ac.uk`.

[13] I.C.F. Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 39(2):254–291, June 1997.

[14] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(3):503–528, 1989.

[15] W. Menke. *Geophysical Data Analysis: Discrete Inverse Theory.* Academic Press, San Diego, 1989.

[16] J.J. Moré and D.C. Sorensen. Computing a trust region step. *SIAM J. Sci. Stat. Comput.*, 4(3):553–572, 1983.

[17] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comput.*, 35(151):773–782, 1980.

[18] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer, New York, 1999.

[19] J. Nocedal and Y. Yuan. Combining trust region and line search techniques. In Y. Yuan, editor, *Advances in Nonlinear Programming*, pages 153–175. Kluwer, Dordrecht, The Netherlands, 1998.

[20] A. Ouorou. Implementing a proximal algorithm for some nonlinear multicommodity flow problems. *Networks*, 49(1):18–27, 2007.

[21] M.J.D. Powell. A new algorithm for unconstrained optimization. In J.B. Rosen, O.L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, New York, NY, 1970.

[22] F. Rendl and H. Wolkowicz. A semidefinite framework for trust region subproblems with applications to large scale minimization. *Mathematical Programming*, 77(2):273–299, 1997.

[23] M. Rojas, S.A. Santos, and D.C. Sorensen. A new matrix-free algorithm for the large-scale trust-region subproblem. *SIAM J. Optim.*, 11(3):611–646, 2000.

[24] D.F. Shanno. Conjugate gradient methods with inexact searches. *Math. Oper. Res.*, 3(3):244–256, 1978.

[25] G.A. Shuldz, R.B. Schnabel, and R.H. Byrd. A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. *SIAM J. Numer. Anal.*, 22(1):47–67, February 1985.

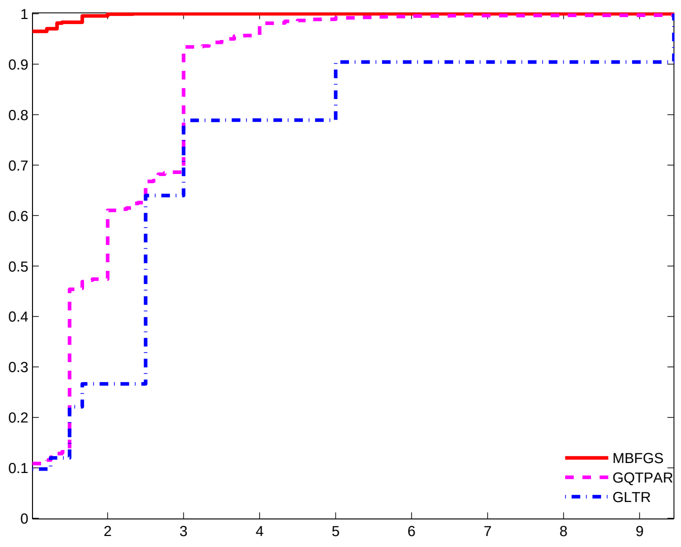
[26] D.C. Sorensen. Newton's method with a model trust region modification. *SIAM J. Numer. Anal.*, 19(2):409–426, 1982.

[27] D.C. Sorensen. Minimization of a large-scale quadratic function subject to a spherical constraint. *SIAM J. Optim.*, 7(1):141–161, 1997.

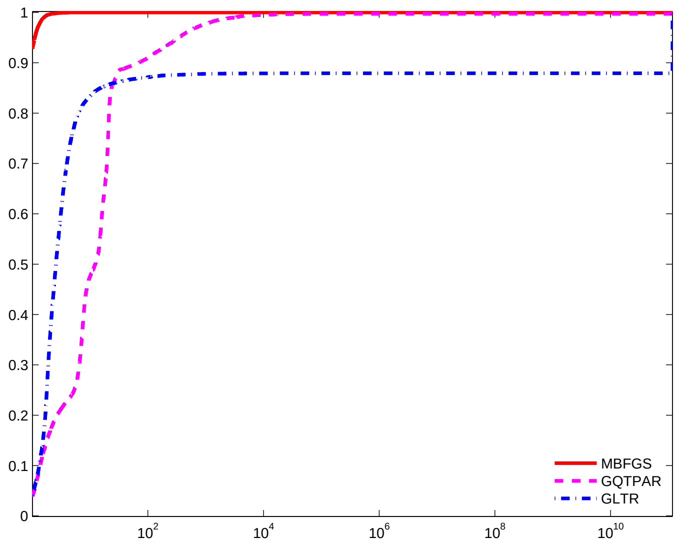
[28] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, 20(3):626–637, 1983.

[29] Ph.L. Toint. Towards an efficient sparsity exploiting newton method for minimization. In I.S. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–87. Academic Press, Inc., New York, NY, 1981.

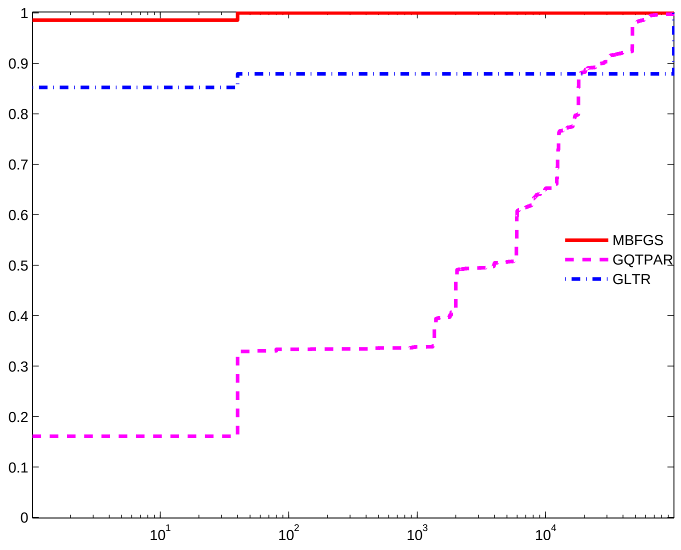
[30] J.H. Wilkinson. *The algebraic eigenvalue problem.* Oxford University Press, London, 1965.



(a) Performance based on number of iterations

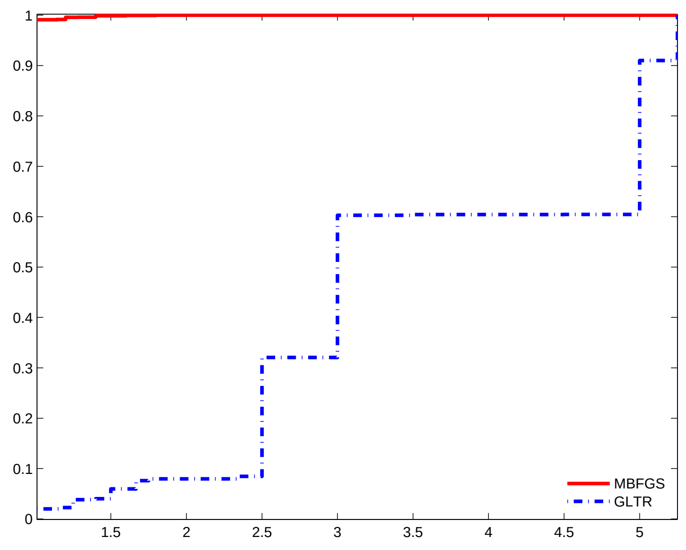


(b) Performance based on accuracy

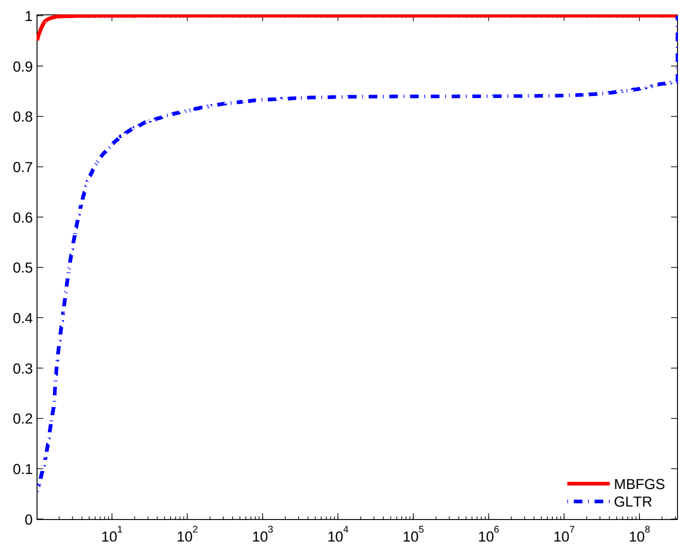


(c) Performance based on CPU time

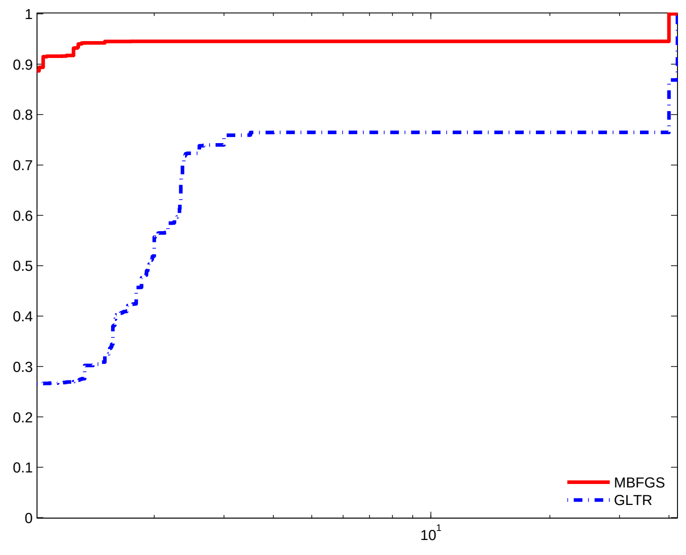
Figure 1: Standard Case: Performance profiles for `MBFGS`, `GQTPAR` and `GLTR` on the set of 12000 medium size problems ($n = 100$, 500, and 1000).



(a) Performance based on number of iterations

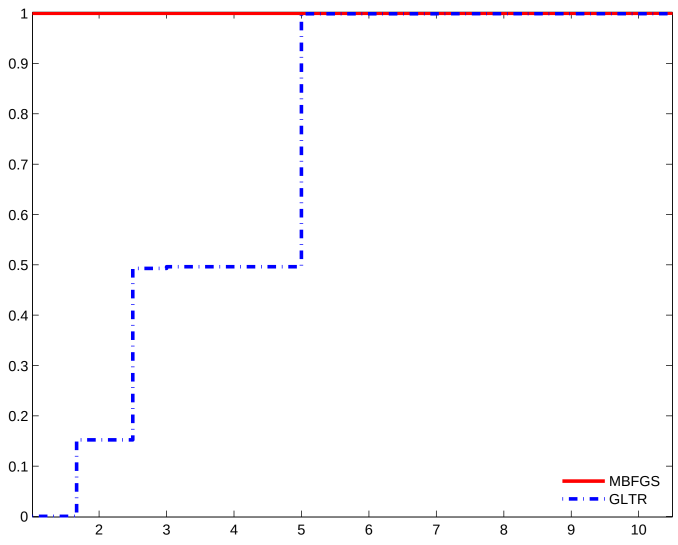


(b) Performance based on accuracy

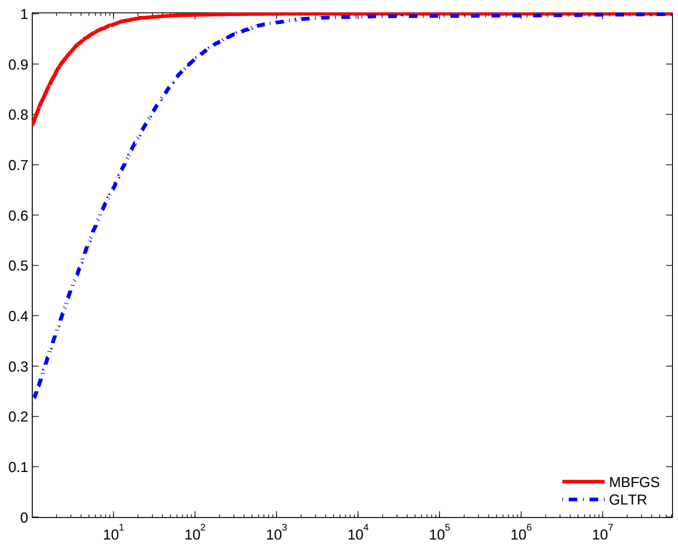


(c) Performance based on CPU time

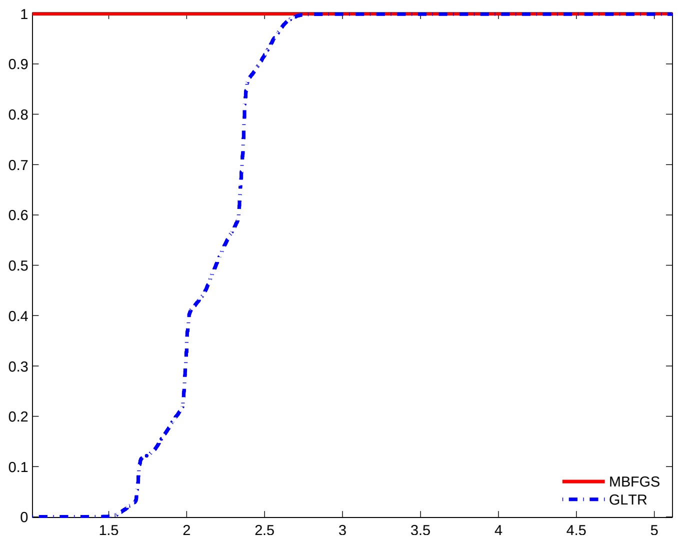
Figure 2: Standard Case: Performance profiles for `MBFGS` and `GLTR` on the set of 12000 large size problems ($n = 10^4$, 10^5, and 10^6).



(a) Performance based on number of iterations

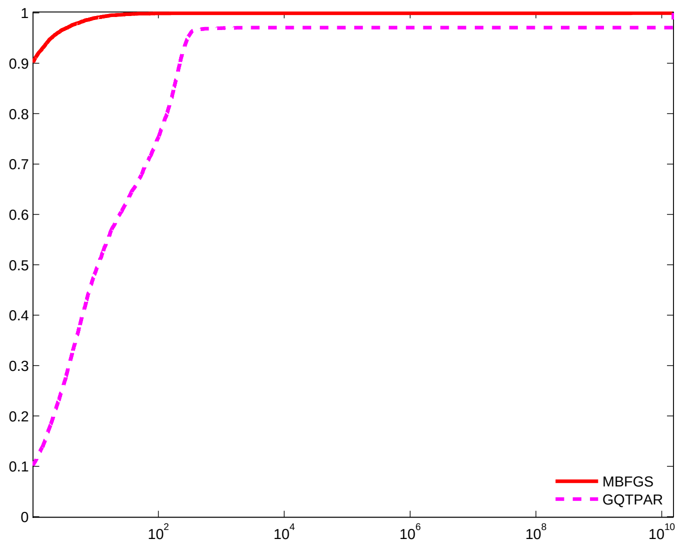


(b) Performance based on accuracy

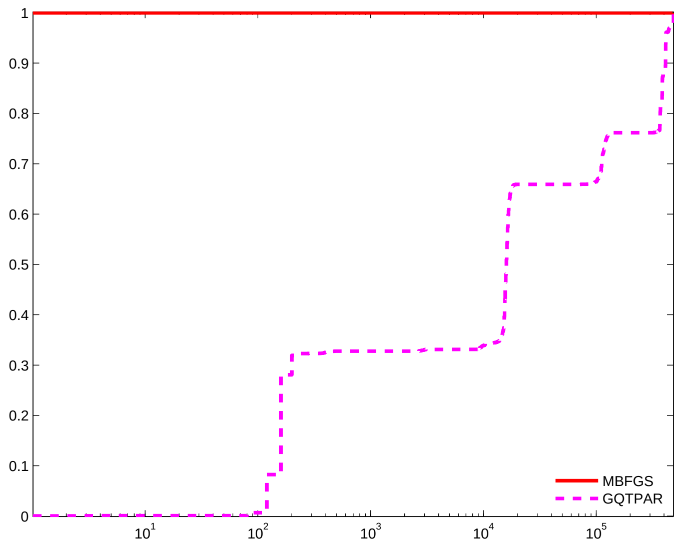


(c) Performance based on CPU time

Figure 3: Standard Case: Performance profiles for `MBFGS` and `GLTR` on the set of 15000 large size problems ($n = 1 \cdot 10^6$, $2 \cdot 10^6$, $3 \cdot 10^6$, …, $15 \cdot 10^6$).



(a) Performance based on accuracy



(b) Performance based on CPU time

Figure 4: Hard Case: Performance profiles for MBFGS and GQTPAR on the set of 9000 medium size problems ($n = 100$, 500, and 1000).