A nearly exact method for solving large-scale TRS

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

***Abstract*—We present a matrix-free method for the large scale trust region subproblem (TRS), assuming that the approximate Hessian is updated using a minimal-memory BFGS method, where the initial matrix is a scaled identity matrix. We propose a variant of the Moré-Sorensen method that exploits the eigenstructure of the approximate Hessian, and incorporates both the standard and the hard case. The eigenvalues and the corresponding eigenvectors are expressed analytically, and hence a direction of negative curvature can be computed immediately. The most important merit of the proposed method is that it completely avoids the factorization, and the trust region subproblem can be solved by performing a sequence of inner products and vector summations. Numerical results are also presented.**

***Index Terms*—Trust region subproblem, nearly exact method, L-BFGS method, eigenvalues, negative curvature direction, large scale optimization**

I. Introduction

We consider the following quadratic minimization problem:

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

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 [2], [3], regularization methods for ill-posed problems [11], graph partitioning problems [7], large-scale nonlinear multicommodity flow problems [15], image restoration [17], etc. In particular, problem (1) is important in a class of methods for solving both convex and nonconvex nonlinear optimization problems, namely, the trust-region algorithms [3]. 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 [16], the two-dimensional subspace minimization methods [18] and the truncated CG methods [6], [20]. Nearly exact methods for solving (1) have been proposed by Gay [5], Sorensen [19], and Moré and Sorensen [12]. 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 $B+\lambda I$ positive semi-definite. Moreover, in the so called *hard case*, a direction of negative curvature is required to be produced [12]. Therefore, this method can be very costly and even prohibitively expensive when it is applied in very large problems.

In this work we are concentrated in the method of nearly exact solutions.We study the eigenstructure of minimal-memory BFGS matrices, and apply our results for the solution of large scale subproblems. The proposed nearly exact method avoids the Cholesky factorization for the solution of the linear system $(B+\lambda I)d = -g$, while a direction of negative curvature is produced using the method of inverse iteration [9].

II. Properties of the minimal-memory BFGS matrices

The minimal memory BFGS matrices [10], [13] are updated using the BFGS formula

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

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$, storing curvature information from the most previous iteration. We consider as the initial matrix $B_k^{(0)}$, the diagonal matrix $B_k^{(0)} = \theta_k I$, where $\theta_k \in \mathbb{R}\setminus\{0\}$, and the resulting minimal memory BFGS scheme takes the form

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

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 II.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

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

where $\beta_1 = \theta_{k+1} + y_k^T y_k / s_k^T y_k$, and $\beta_2 = \theta_{k+1} s_k^T y_k / s_k^T s_k$. 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 [22, pp. 94–98] on $\bar{B}$, it is easy to

M. S. Apostolopoulou is with the Department of Mathematics, University of Patras, Patras, Greece. E-mail: msa@math.upatras.gr

D. G. Sotiropoulos is with the Department of Informatics, Ionian University, Corfu, Greece. E-mail: dgs@ionio.gr

C. A. Botsaris is with the Department of Regional Economic Development,University of Central Greece, Levadia, Greece. E-mail: botsaris@otenet.gr

P. Pintelas is with the Department of Mathematics, University of Patras, Patras, Greece. E-mail: pintelas@math.upatras.gr

978-1-4244-3760-3/09/$25.00 © 2009 IEEE

see that $\bar{B}$ besides the zero eigenvalue, has one more eigenvalue equals to θ_{k+1}, of multiplicity $(n-1)$. If $B_{k+1}^{(0)}$ is positive definite, then the addition of the term $y_k y_k^T / s_k^T y_k$ on $\bar{B}$ yields

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

where λ_i, $i = 1, \ldots, n$ 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

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

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

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

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:

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

and

$$\det(B_{k+1}) = \theta_{k+1}^{n-1} \frac{s_k^T y_k}{s_k^T s_k} \tag{11}$$

Hence, $\operatorname{tr}(B_{k+1}) - (n-2)\theta_{k+1} = \theta_{k+1} + \dfrac{y_k^T y_k}{s_k^T y_k}$, $\dfrac{\det(B_{k+1})}{\theta_{k+1}^{n-2}} = \theta_{k+1} \dfrac{s_k^T y_k}{s_k^T s_k}$, 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 relations (8), (10), and (9), (11), we have that $(s_k^T y_k)^2 = s_k^T s_k y_k^T y_k \Rightarrow \cos\phi = \pm 1$, where ϕ denotes the angle of s_k and y_k. This implies that the vectors s_k and y_k are collinear, which contradicts the hypothesis. Hence, 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. ■

If the vectors s_k and y_k are collinear, i.e., $y_k = \kappa s_k$, $\kappa \in \mathbb{R}$, then 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 II.1, the eigenvalues of B_{k+1} sorted into increasing order are

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

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 ith component of s_k, and 1 is in the ith row of u_i.

When s_k and y_k are linearly independent, for being able to determine the eigenvectors corresponding to distinct eigenvalues of B_{k+1}, we can make use of the inverse power method [9]. Given a non-zero starting vector $u^{(0)}$, inverse iteration generates a sequence of vectors $u^{(i)}$, generated recursively by the formula

$$u^{(i)} = \left(B - \hat{\lambda} I\right)^{-1} \frac{u^{(i-1)}}{\|u^{(i-1)}\|},$$

$i \geq 1$, where $\hat{\lambda} = \lambda + \epsilon$, λ 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 $\hat{\lambda}$. Usually, the starting vector $u^{(0)}$ is chosen to be the normalized vector $(1, 1, \ldots, 1)^T$. Moreover, if this particular eigenvalue λ is known exactly, this method converges in a single iteration [9].

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

$$(B_{k+1} + \lambda I)^{-1} = \frac{B_{k+1}^2 - [\beta'(\lambda) - \lambda]\, B_{k+1} + \beta(\lambda) I}{(\lambda + \theta_{k+1})(\lambda^2 + \beta_1 \lambda + \beta_2)}, \tag{13}$$

where $\beta(\lambda) = (\lambda + \beta_1)(\lambda + \theta_{k+1}) + \beta_2$.

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

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

Hence, the minimal characteristic polynomial is

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

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,

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

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

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

By setting $\beta(\lambda) = (\lambda + \beta_1)(\lambda + \theta_{k+1}) + \beta_2$, we obtain relation (13). ■

Using the above Proposition, along with inverse iteration, we have that the eigenvectors corresponding to distinct eigenvalues of B_{k+1}, are the normalized vectors

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

where $\hat{\lambda} = -\lambda + \epsilon$ is a perturbed distinct eigenvalue of B_{k+1} with opposite sign, and u is the unit vector $\frac{1}{\sqrt{n}}(1, 1, \ldots, 1)^T$. The vectors $B_{k+1}u$ and $B_{k+1}^2 u$ can be obtained by the iterative form

$$B_{k+1} v_i = \theta_{k+1} v_i - \theta_{k+1} \frac{s_k^T v_i}{s_k^T s_k} + \frac{y_k^T v_i}{s_k^T y_k} \quad i = 0, 1, \tag{15}$$

with $v_0 = u$. Using relation (15), after some algebraic computations, the normalized vectors (14) can be expressed by the relation

$$u(\hat{\lambda}) = \frac{c_u(\hat{\lambda}) u + c_s(\hat{\lambda}) s_k + c_y(\hat{\lambda}) y_k}{(\hat{\lambda}^2 + \beta_1\hat{\lambda} + \beta_2)(\hat{\lambda} + \theta_{k+1})}, \tag{16}$$

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

$$c_u(\hat{\lambda}) = \hat{\lambda}^2 + \beta_1\hat{\lambda} + \beta_2 \tag{17}$$

$$c_s(\hat{\lambda}) = \left[\left(\hat{\lambda} + \beta_1\right) s_k^T u - y_k^T u\right] \frac{\theta_{k+1}}{s_k^T s_k} \quad \text{and} \tag{18}$$

$$c_y(\hat{\lambda}) = -\frac{y_k^T u}{s_k^T y_k}\hat{\lambda} - \frac{s_k^T u}{s_k^T s_k}\theta_{k+1} \tag{19}$$

III. Solving the TRS using the minimal-memory BFGS method

In this section we apply the results of Section II for solving the large scale trust-region subproblem (1). A global solution to the TRS (1) is characterized by the following well known theorem (Gay [5], Sorensen [19], Moré & Sorensen [12]):

Theorem III.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 optimal non-negative Lagrange multiplier λ^* belongs to the open interval $(-\lambda_1, \infty)$. When $\lambda^* \neq 0$, the TRS (1) has a solution on the boundary of its constraint set, 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, $\phi(\lambda) \equiv 1/\Delta - 1/\|d(\lambda)\| = 0$ (secular equation), that exploits the rational structure of $\|d(\lambda)\|^2$. The solution λ of the secular equation is based on Newton's method,

$$\lambda^{\ell+1} = \lambda^\ell + \frac{\|d(\lambda)\|}{\|d(\lambda)\|'} \left(\frac{\Delta - \|d(\lambda)\|}{\Delta}\right), \quad \ell = 0, 1, 2, \ldots. \tag{20}$$

In Newton's iteration (20) 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 [12].

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 $\|(B + \lambda I)^{-1} g\| = \Delta$. The optimal Lagrange multiplier is $\lambda^* = -\lambda_1$, and a direction of negative curvature must be produced in order to ensure that $\|d\| = \Delta$. Hence, the optimal solution to the TRS is $d^* = -(B - \lambda_1 I)^\dagger g + \tau u$, where $\tau \in \mathbb{R}$ is such that $\|-(B - \lambda_1 I)^\dagger g + \tau u\| = \Delta$ and u is a normalized eigenvector corresponding to λ_1. Moré and Sorensen [12] have showed that the choice of τ that ensures $\|d\| = \Delta$, is

$$\tau = \frac{\Delta^2 - \|p\|^2}{p^T u_1 + \operatorname{sgn}(p^T u)\sqrt{(p^T u)^2 + \Delta^2 - \|p\|^2}}, \tag{21}$$

where $p = -(B - \lambda_1 I)^\dagger g$, where the symbol † denotes the Moore-Penrose generalized inverse of the matrix.

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. However, all these quantities required of solving the TRS, have been studied and expressed analytically in the previous section. The following algorithm incorporates 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 III.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 $\|d\| \leq \Delta$ 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 $\|-(B + \hat{\lambda} I)g + \tau u_1\| = \Delta$; 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 $\|d\| \leq \Delta$ stop; else update λ_L and λ_U such that $\lambda_L \leq \lambda \leq \lambda_U$ and go to Step 4.

The safeguarding scheme required for Newton's iteration (20) 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 bounds λ_L and λ_U used in Algorithm III.1 have been proposed by Nocedal and Yuan [14]. Clearly, the lower bound λ_L is greater than $-\lambda_1$, which ensures that $B + \lambda I$ is always positive definite.

The eigenvalues in Step 1 of Algorithm III.1 can be computed by means of the characteristic polynomial (4). In Steps 2 and 3, the computation of the trial step $d(\lambda)$ is based on Proposition II.1. Using relations (15), the trial step can be obtained by the formula

$$d(\lambda) = -\frac{c_g(\lambda) g + c_s(\lambda) s_k + c_y(\lambda) y_k}{(\lambda^2 + \beta_1 \lambda + \beta_2)(\lambda + \theta_{k+1})}, \tag{22}$$

where

$$c_u(\lambda) = \lambda^2 + \beta_1 \lambda + \beta_2$$

$$c_s(\lambda) = \left[(\lambda + \beta_1)\, s_k^T g_{k+1} - y_k^T g_{k+1}\right] \frac{\theta_{k+1}}{s_k^T s_k} \quad \text{and}$$

$$c_y(\lambda) = -\frac{y_k^T g_{k+1}}{s_k^T y_k} \lambda - \frac{s_k^T g_{k+1}}{s_k^T s_k} \theta_{k+1}$$

In Step 3(c) the computation of τ is obtained from Eq. (21). If the vectors s_k and y_k are linearly independent, then u is computed by relation (16). In different case, u equals the normalized vector s_k. In Steps 4 and 5, for computing the Lagrange multiplier λ and updating the interval $[\lambda_L, \lambda_U]$ we follow the ideas described in [12]. In Newton's method (20), the quantity $\|d'(\lambda)\|$ equals $\|d(\lambda)\|' = -d(\lambda)^T (B + \lambda I)^{-1} d(\lambda) / \|d(\lambda)\|$. Hence, relation (20) becomes

$$\lambda^{\ell+1} = \lambda^{\ell} + \frac{\|d_{k+1}(\lambda)\|^2}{d(\lambda)^T (B_{k+1} + \lambda I)^{-1} d(\lambda)} \left(\frac{\|d_{k+1}(\lambda)\| - \Delta}{\Delta} \right), \tag{23}$$

for $\ell = 0, 1, 2, \ldots$.

Denoting by $\Pi = d(\lambda)^T (B_{k+1} + \lambda I)^{-1} d(\lambda)$ the denominator in (23), taking into account that $B_{k+1}\, d_{k+1}(\lambda) = -[\lambda\, d_{k+1}(\lambda) + g_{k+1}]$ holds, and using Proposition II.1 we yield

$$\Pi = \frac{[\beta(\lambda) + \lambda \beta'(\lambda)]\, \|d(\lambda)_{k+1}\|^2}{(\lambda + \theta_{k+1})(\lambda^2 + \beta_1 \lambda + \beta_2)} + \\ + \frac{[\beta'(\lambda) + \lambda]\, d(\lambda)_{k+1}^T g_{k+1} + \|g_{k+1}\|^2}{(\lambda + \theta_{k+1})(\lambda^2 + \beta_1 \lambda + \beta_2)}. \tag{24}$$

If s_k and y_k are collinear, i.e., relation $y_k = \kappa s_k$ holds, then relations (22) and (24) are reduced to

$$d_{k+1}(\lambda) = -\frac{1}{(\lambda + \theta_{k+1})} g_{k+1} + \frac{(\kappa - \theta_{k+1}) s_k^T g_{k+1}}{(\lambda + \kappa)(\lambda + \theta_{k+1}) s_k^T s_k} s_k, \tag{25}$$

and

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

respectively.

IV. 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 1 000 000 variables. The experiments are consisted of medium-size problems with dimensions $n = 100, 500, 1000$, and by larger-size problems ($n = 10^4, 10^5, 10^6$). For each dimension n, 1000 random instances of TRS 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 III.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` [1], and the `GLTR` [8] algorithm. The `GQTPAR` algorithm is the subroutine `GQTPAR.f` from the MINPACK package, and is based on the ideas described in Moré and Sorensen [12] for computing a nearly exact solution of (1). The method uses the Cholesky factorization for the evaluations of the derivatives and the Newton step for λ, and for the safeguarding and updating of the Lagrange multiplier. The `GLTR` algorithm proposed by Gould et al. [6], is available as the FORTRAN 90 module HSL_VF05 in the Harwell Subroutine Library. 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 [20], [21] 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 throughout for each algorithm. We consider a TRS instance successfully solved, if a solution satisfying $\|(B + \lambda I)\, d - g\| \leq 10^{-5}$ 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 the dimensions 10^4, 10^5 and 10^6, 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.

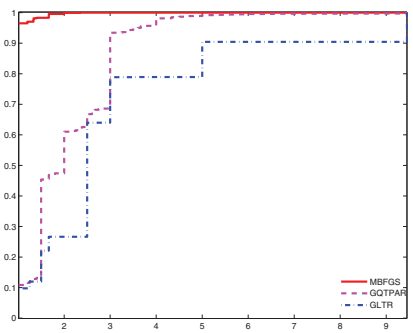
To show both the efficacy and accuracy of our method, we have used the performance profile proposed by Dolan and Moré [4]. The performance profile plots the fraction of problems for which any given method is within a factor of the best time. The left axis of the plot shows the percentage of the problems 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 performance measures that have been used are Newton's iterations, solution accuracy and CPU time in seconds.

A. Random experiments in the standard case

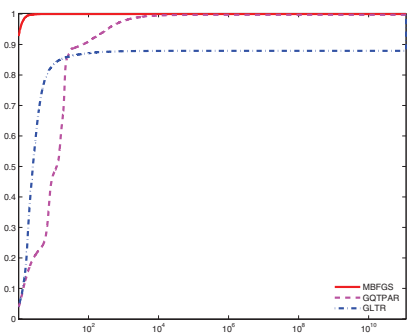
In the sequel, we present the behavior of the proposed method in the standard case. For each dimension we have consider the following cases:

(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 $\theta_{k+1} = (y_k^T y_k)/(s_k^T y_k)$.

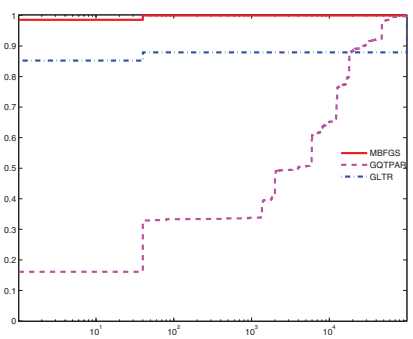
As a total we have 12000 medium size experiments ($n =$ 100, 500, and 1000) and another 12000 large size experiments ($n = 10^4$, 10^5, and 10^6).



(a) Performance based on number of iterations



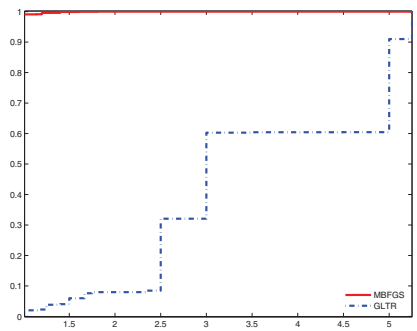
(b) Performance based on accuracy



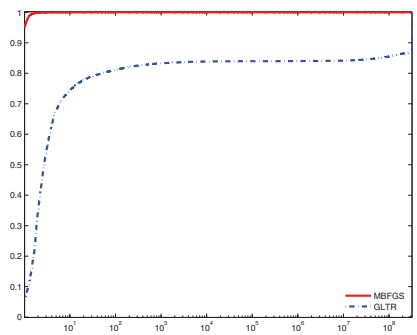
(c) Performance based on CPU time

Fig. 1. Performance profiles for `MBFGS`, `GQTPAR` and `GLTR` on the set of 12000 medium size experiments ($n = 100$, 500, and 1000).

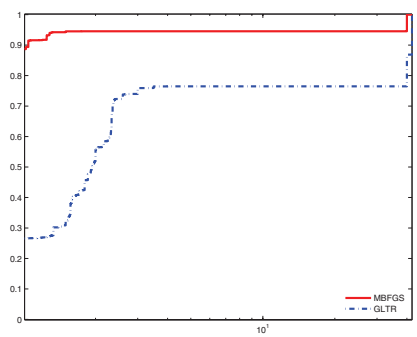
In Figures 1 and 2 are presented the performance profiles based on Newton iterations, solution accuracy and CPU time (in seconds) for the medium and large size experiments, respectively. In Figure 2 the `GQTPAR` method was omitted from the comparisons due to insufficient memory limitations. In both figures it is interesting to observe that the best performance regarding all performance metrics was obtained by `MBFGS` since it is the most robust and efficient method. It is worth noticing that the proposed method significantly outperforms the other two methods in all dimensions.



(a) Performance based on number of iterations



(b) Performance based on accuracy



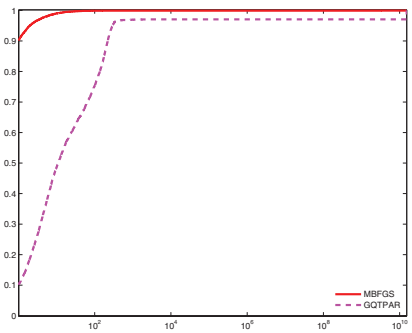
(c) Performance based on CPU time

Fig. 2. Performance profiles for `MBFGS` and `GLTR` on the set of 12000 large size experiments ($n = 10^4$, 10^5, and 10^6).

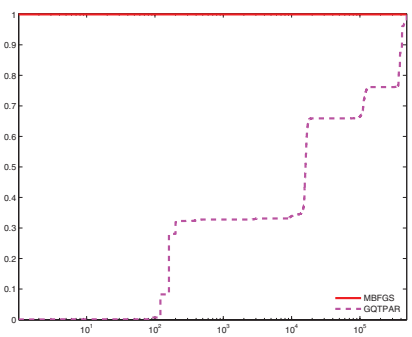
B. Random experiments in the hard case

In order to create instances for the hard case, Matlab's `eigs` routine was used to compute the smallest eigenvalue λ_1 and the corresponding eigenvector u of the reconstructed matrix B from the vector pairs. We initialized the trust-region radius by $\Delta = 10.0\Delta_{hc}$, where

$\Delta_{hc} = \| (B - \lambda_1 I)^{\dagger} g \|$ while the vector of the gradient was computed as $g = (-u(n)/u(1), 0, \ldots, 0, 1)^T$. For each dimension we have consider only the cases *(a)*, *(b)* and *(c)* due to the fact that the case *(d)* occurs only when $g = 0$. Totally, we have run 9000 medium size experiments ($n = 100$, 500, and 1000). In Figure 3 reports the performance profiles regarding `MBFGS` and `GQTPAR` in terms of solution accuracy and CPU time (in seconds) for medium size problems. The iteration performance metric is not reported since the method `MBFGS` does not require any Newton's iterations for solving the problem. Obviously, the proposed methods exhibits top performance relative to all performance metrics.



(a) Performance based on accuracy



(b) Performance based on CPU time

Fig. 3. Performance profiles for `MBFGS` and `GQTPAR` on the set of 9000 medium size experiments ($n = 100$, 500, and 1000).

V. Conclusions

We have studied the eigenstructure of the minimal memory BFGS matrices, in the general case where the initial matrix is any scaled identity matrix. Our theoretical results have been applied for the solution of the trust region subproblem. Based on the fact that the eigenvalues can immediately be computed with high accuracy, the inverse of $B + \lambda I$ can be expressed in a closed form, and consequently the Cholesky factorization can be completely avoided, the proposed method can solve inexpensively large scale subproblems. The numerical experiments have shown that the proposed method can handle easily both the standard and the hard case, while it provides solutions with high accuracy and small running time, since the amount of memory needed is negligible.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, PA, USA, 1994.

[2] 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.

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

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

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

[6] 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.

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

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

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

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

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

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

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

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

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

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

[21] 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.

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