Optimization Methods & Software
Vol. 23, No. 5, October 2008, 651–674

Taylor & Francis
Taylor & Francis Group

Solving the quadratic trust-region subproblem in a low-memory BFGS framework

M.S. Apostolopoulou[a]*, D.G. Sotiropoulos[b] and P. Pintelas[a]

[a]*Department of Mathematics, University of Patras, Patras, Greece;* [b]*Department of Informatics, Ionian University, Corfu, Greece*

(*Received 30 September 2007; final version received 28 June 2008*)

We present a new matrix-free method for the large-scale trust-region subproblem, assuming that the approximate Hessian is updated by the L-BFGS formula with $m = 1$ or 2. We determine via simple formulas the eigenvalues of these matrices and, at each iteration, we construct a positive definite matrix whose inverse can be expressed analytically, without using factorization. Consequently, a direction of negative curvature can be computed immediately by applying the inverse power method. The computation of the trial step is obtained by performing a sequence of inner products and vector summations. Furthermore, it immediately follows that the strong convergence properties of trust region methods are preserved. Numerical results are also presented.

Keywords: trust region; nearly exact method; L-BFGS method; eigenvalues; negative curvature direction; large scale optimization

AMS Subject Classification: 90C06; 90C30; 65F15; 65F05

1. Introduction

An important class of methods for solving both convex and nonconvex nonlinear optimization problems is the trust-region algorithms [9,11,19,31]. They are popular due to their strong convergence and robustness [9]. Under some mild conditions, it can be proved that the sequence of points $\{x_k\}$ generated by a trust region algorithm converges to a point which satisfies both the first- and the second-order necessary conditions [19,31]. At each iteration x_k of a trust-region algorithm, a trial step d_k is usually obtained by solving the following quadratic subproblem

$$\min_{d\in\mathbb{R}^n} \phi_k(d) = g_k^{\mathrm{T}} d + \frac{1}{2} d^{\mathrm{T}} B_k d, \quad \text{s.t. } \|d\| \le \Delta_k, \tag{1}$$

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 a positive definite or indefinite approximate Hessian of f at x_k, $\|\cdot\|$ is the Euclidean norm, and $\Delta_k > 0$ is

*Corresponding author. Email: msa@math.upatras.gr

ISSN 1055-6788 print/ISSN 1029-4937 online

DOI: 10.1080/10556780802413579
http://www.informaworld.com

the so-called trust region radius. If the trial step d_k results in a reduction of the objective function, the new iterate is accepted and Δ_k is enlarged. In the different case, the new iterate is rejected, Δ_k is reduced, and the subproblem (1) is resolved. The minimization of the quadratic subproblem (1), called the trust-region subproblem (TRS), is the main computational step in a trust-region algorithm, since it must be solved at least once in each iteration of a trust region algorithm. Problems of the form (1) arise in many applications as regularization methods for ill-posed problems [17], graph partitioning problems [14] and large-scale nonlinear multicommodity flow problems [23,24] which are an important class of network optimization problems with applications in telecommunications and transportation.

A well-known method for the solution of the TRS is the method proposed by Moré and Sorensen [19]. A nearly exact solution d to the TRS must satisfy an equation of the form $(B + \lambda I)d = -g$, where $I \in \mathbb{R}^{n\times n}$ is the identity matrix and $\lambda \geq 0$ is the Lagrange multiplier. Moreover, $B + \lambda I$ must be positive semi-definite, and relation $\lambda(\|d\| - \Delta) = 0$ must hold [11,19,31]. This method requires the Cholesky factorization of $B + \lambda I$ each time that the subproblem is solved. Furthermore, a direction of negative curvature must be produced in the so-called *hard case*. Therefore, a nearly exact solution can be very costly and even prohibitively expensive when B is very large. This drawback has motivated the development of matrix-free methods that rely on matrix–vector products [12,13,28,29,32].

The aim of this work is to avoid both the factorization and the storage of any matrix, as well as matrix–vector products. To this end, the computation of the approximate Hessian B in the quadratic model (1) is obtained using the L-BFGS formula [16,20], for $m = 1$ or 2. The symbol m denotes the number of vector pairs $\{s_i, y_i\}$, that are used for updating B, where $s_i = x_i - x_{i-1}$ and $y_i = g_i - g_{i-1}$. Studying the properties of B, we are able to obtain its characteristic polynomial via simple formulas. Hence, the eigenvalues can be computed analytically, while the inverse of $(B + \lambda I)$ can be expressed in a closed form. Therefore, the Cholesky factorization for the solution of the linear system $(B + \lambda I)d = -g$ is avoided, and a direction of negative curvature can be produced using the method of inverse power iteration [3,15]. Due to the above reasons, the step d can be obtained by performing a sequence of inner products and vector summations. Therefore, our approach can be applied for the solution of large scale problems.

The paper is organized as follows. In Section 2 we discuss the TRS. In Section 3 we study the properties of the updated matrix while in Section 4 we apply our results in the computation of the step. In Section 5 we present an algorithm for the solution of the TRS that incorporates both the standard and the hard case. We present some preliminary numerical results in Section 6 and some conclusions in Section 7.

Notation. Throughout the paper $\|\cdot\|$ denotes the Euclidean norm and n the dimension of the problem. 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 $\lambda_1 \leq \cdots \leq \lambda_n$ are its eigenvalues sorted into nondecreasing order. We indicate that a matrix is positive semidefinite (positive definite) by $A \geq 0$ ($A > 0$, respectively). The L-BFGS matrix is denoted by $B^{(m)}$.

2. The subproblem

A global solution to the TRS (1) is characterized by the following well-known theorem [11,19,31]:

THEOREM 2.1 *A feasible vector d^* is a solution to Equation* (1) *with corresponding Lagrange multiplier λ^* if and only if d^*, λ^* satisfy $(B + \lambda^* I)d^* = -g$, where $B + \lambda^* I$ is positive semidefinite, $\lambda^* \geq 0$, and $\lambda^*(\Delta - \|d^*\|) = 0$.*

The above result provides the theoretical basis for our step computing function d, and it can be analysed in the following cases:

1. If $\lambda_1 > 0$ and $\|B^{-1}g\| \leq \Delta$, then $\lambda^* = 0$ and $d^* = B^{-1}g$.
2. If $\lambda_1 > 0$ and $\|B^{-1}g\| > \Delta$, then for the unique $\lambda^* > 0$ such that $\|(B + \lambda^* I)^{-1}g\| = \Delta$, $d^* = -(B + \lambda^* I)^{-1}g$.
3. (a) If $\lambda_1 \leq 0$ and there is a $\lambda^* > -\lambda_1$ such that $\|(B + \lambda^* I)^{-1}g\| = \Delta$, then $d^* = -(B + \lambda^* I)^{-1}g$.
   (b) Otherwise, $\lambda^* = -\lambda_1$ and $d^* = -(B - \lambda_1 I)^{\dagger}g + \tau u_1$, where $\tau \in \mathbb{R}$ is such that $\| - (B - \lambda_1 I)^{\dagger}g + \tau u_1\| = \Delta$ and u_1 is an eigenvector that corresponds to λ_1, such that $\|u_1\| = 1$.

Moré and Sorensen [19] have shown that the choice of τ that ensures $\|d\| = \Delta$ is

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

where $p = -(B - \lambda_1 I)^{\dagger}g$. The case 3(b) occurs when B is indefinite and g is orthogonal to every eigenvector corresponding to the most negative eigenvalue λ_1 of the matrix. This case is known as the *hard* case. In this case $\lambda^* = -\lambda_1$, $(B - \lambda_1 I)$ is singular, and a direction of negative curvature must be produced in order to ensure that $\|d\| = \Delta$. When $\lambda^* \in (-\lambda_1, \infty)$ (the *standard* case) and $\lambda^* \neq 0$, the TRS (1) has a solution on the boundary of its constraint set, i.e. $\|d^*\| = \Delta$, and 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 $d(\lambda)$ is a solution of $(B + \lambda I)d = -g$. Moré and Sorensen [19] have proved that it is more convenient to solve the equivalent equation (secular equation) $\psi(\lambda) \equiv (1/\Delta) - (1/\|d(\lambda)\|) = 0$ that exploits the rational structure of $\|d(\lambda)\|^2$ when Newton's method is applied. The resulting iteration is

$$\lambda^{\ell+1} = \lambda^{\ell} + \frac{\|d(\lambda)\|}{\|d(\lambda)\|'}\left(\frac{\Delta - \|d(\lambda)\|}{\Delta}\right), \quad \ell = 0, 1, 2, \ldots, \tag{3}$$

where the derivative of $\|d(\lambda)\|$ is of the form

$$\|d(\lambda)\|' = -\frac{d(\lambda)^{\mathrm{T}}(B + \lambda I)^{-1}d(\lambda)}{\|d(\lambda)\|}.$$

An important ingredient of Newton's iteration (3) is the safeguarding required to ensure that a solution is found. The safeguarding depends on the fact that ψ 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 semipositive definite [19]. Initial bounds for λ have also been proposed by Nocedal and Yuan [22]. The following lemma gives the bounds of λ, when the TRS (1) is solved exactly.

LEMMA 2.2 *If vector d^* is a solution of Equation* (1)*, $\|d^*\| = \Delta$, and $\lambda^* \geq 0$ satisfies $(B + \lambda^* I)d^* = -g$, with $(B + \lambda^* I) \geq 0$ then $0 \leq \lambda^* \leq \|g\|/\Delta - \lambda_1$, where λ_1 is the smallest eigenvalue of B.*

For an approximate solution to Equation (1), $\|B\| + (1 + \epsilon)\|g\|/\Delta$ forms another upper bound for λ, where $\epsilon > 0$ is a small number that ensures that $B + \lambda I$ is positive definite [22]. Therefore, λ can be bounded as

$$\max(0, -\lambda_1 + \epsilon) \leq \lambda \leq \|B\| + (1 + \epsilon)\|g\|/\Delta. \tag{4}$$

3. The updated matrix

The computation of the matrix $B \approx \nabla^2 f(x)$ in the quadratic model (1) is accomplished using the L-BFGS philosophy [16,20] with $m = 1$ or 2. More analytically, B_k is updated by means of the BFGS formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^{\mathrm{T}} B_k}{s_k^{\mathrm{T}} B_k s_k} + \frac{y_k y_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k}, \tag{5}$$

using information from the previous ($m = 1$) or the last two previous ($m = 2$) iterations. Also, it is known that the inverse of B_{k+1} is given by the expression

$$(B_{k+1})^{-1} \equiv H_{k+1} = H_k - \frac{H_k y_k s_k^{\mathrm{T}} + s_k y_k^{\mathrm{T}} H_k}{s_k^{\mathrm{T}} y_k} + \frac{s_k s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k}. \tag{6}$$

We consider the diagonal matrix $B_k^{(0)} = (1/\theta_k) I$, $\theta_k \in \mathbb{R}$, as the initial matrix $B_k^{(0)}$. Motivated by the work of Barzilai and Borwein [4] and Birgin and Martínez [5], we use the *spectral* parameter θ_k defined as [6,26,27]

$$\theta_k = \frac{s_{k-1}^{\mathrm{T}} s_{k-1}}{s_{k-1}^{\mathrm{T}} y_{k-1}}.$$

The above parameter is actually the inverse of the Rayleigh quotient $s^{\mathrm{T}} G s / s^{\mathrm{T}} s$, which lies between the largest and the smallest eigenvalues of the average Hessian G.

Observing the updating scheme, we can see that B_k is updated by the addition of two rank-one matrices. The following lemma, due to Wilkinson [33, pp. 94–98], states that the eigenvalues of a matrix which is updated by a rank-one matrix interlace with the eigenvalues of the original matrix.

LEMMA 3.1 *If symmetric matrices A and A^* differ by a matrix of rank-one, then their eigenvalues λ and λ^* interpolate each other in a weak sense. In particular, if $A^* = A + \sigma v v^{\mathrm{T}}$, where σ is scalar, $\lambda_n \geq \lambda_{n-1} \geq \cdots \geq \lambda_1$ and $\lambda_n^* \geq \lambda_{n-1}^* \geq \cdots \geq \lambda_1^*$, then*

(1) *if $\sigma > 0$, $\lambda_n^* \geq \lambda_n \geq \lambda_{n-1}^* \geq \lambda_{n-1} \geq \cdots \geq \lambda_1^* \geq \lambda_1$*
(2) *if $\sigma < 0$, $\lambda_n \geq \lambda_n^* \geq \lambda_{n-1} \geq \lambda_{n-1}^* \geq \cdots \geq \lambda_1 \geq \lambda_1^*$.*

Moreover, it is known that the trace and the determinant of the BFGS matrices are given by the formulas [8,21]

$$\mathrm{tr}(B_{k+1}) = \mathrm{tr}(B_k) - \frac{\|B_k s_k\|^2}{s_k^{\mathrm{T}} B_k s_k} + \frac{\|y_k\|^2}{s_k^{\mathrm{T}} y_k}, \quad \det(B_{k+1}) = \det(B_k) \frac{s_k^{\mathrm{T}} y_k}{s_k^{\mathrm{T}} B_k s_k}, \tag{7}$$

respectively. Based on both Lemma 3.1 and relations (7), we are able to analyse the eigenstructure of the L-BFGS matrices for $m = 1$ and 2. For the remaining of the paper we assume that B is invertible.

3.1 *1BFGS update*

We consider the case where the BFGS matrix is updated using information from the previous iteration ($m = 1$). Relation (5) yields

$$B_{k+1}^{(1)} = \frac{1}{\theta_{k+1}} I - \frac{s_k s_k^{\mathrm{T}}}{\theta_{k+1} s_k^{\mathrm{T}} s_k} + \frac{y_k y_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k}, \tag{8}$$

where $B_{k+1}^{(0)} = (1/\theta_{k+1}) I$.

LEMMA 3.2 *Suppose that one update is applied to the symmetric matrix $B^{(0)}$ using the vector pair $\{s_k, y_k\}$ and the BFGS formula. The characteristic polynomial of the resulting matrix $B_{k+1}^{(1)}$ has the general form*

$$p_1(\lambda) = \left(\lambda - \frac{1}{\theta_{k+1}}\right)^{n-2} \left(\lambda^2 - \frac{a_k}{\theta_{k+1}}\lambda + \frac{1}{\theta_{k+1}^2}\right), \tag{9}$$

where $a_k = 1 + \theta_{k+1}(y_k^{\mathrm{T}} y_k / s_k^{\mathrm{T}} y_k)$. Moreover, if $a_k > 2$, then $\lambda_1 < 1/\theta_{k+1} < \lambda_n$, where λ_1 and λ_n are the smallest and largest eigenvalues of $B_{k+1}^{(1)}$, respectively.

Proof First we show that $B_{k+1}^{(1)}$ has at most two distinct eigenvalues. To this end, we consider the matrix $\bar{B} = (1/\theta_{k+1})I - s_k s_k^{\mathrm{T}}/(\theta_{k+1} s_k^{\mathrm{T}} s_k)$ with rank $(n-1)$. Using Lemma 3.1, it is easy to see that $\bar{B}$, besides the zero eigenvalue, has one more eigenvalue equals to $1/\theta_{k+1}$ of multiplicity $(n-1)$. If $B_{k+1}^{(0)}$ is positive definite, then the addition of the term $y_k y_k^{\mathrm{T}}/s_k^{\mathrm{T}} y_k$ on $\bar{B}$ yields (cf. Lemma 3.1)

$$\lambda_n^{B^{(1)}} \geq \frac{1}{\theta_{k+1}} \geq \lambda_{n-1}^{B^{(1)}} \geq \frac{1}{\theta_{k+1}} \geq \cdots \geq \lambda_2^{B^{(1)}} \geq \frac{1}{\theta_{k+1}} \geq \lambda_1^{B^{(1)}} \geq 0;$$

where $\lambda_i^{B^{(1)}}$ denotes the eigenvalues of $B_{k+1}^{(1)}$, otherwise,

$$0 \geq \lambda_n^{B^{(1)}} \geq \frac{1}{\theta_{k+1}} \geq \lambda_{n-1}^{B^{(1)}} \geq \frac{1}{\theta_{k+1}} \geq \cdots \geq \lambda_2^{B^{(1)}} \geq \frac{1}{\theta_{k+1}} \geq \lambda_1^{B^{(1)}}.$$

In both cases, it is obvious that $\lambda_2^{B^{(1)}} = \cdots = \lambda_{n-1}^{B^{(1)}} = 1/\theta_{k+1}$, and

$$\lambda_1^{B^{(1)}} \leq \frac{1}{\theta_{k+1}} \leq \lambda_n^{B^{(1)}}. \tag{10}$$

Relation (10) implies that $B_{k+1}^{(1)}$ has at most two distinct eigenvalues and one eigenvalue equal to $1/\theta_{k+1}$ of multiplicity at least $(n-2)$. Denoting by λ_1 and λ_2 the two unknown distinct eigenvalues, the characteristic polynomial of $B_{k+1}^{(1)}$ has the form $p_1(\lambda) = (\lambda - 1/\theta_{k+1})^{n-2}[\lambda^2 - (\lambda_1 + \lambda_2)\lambda + \lambda_1\lambda_2]$. Since $\mathrm{tr}(B_{k+1}^{(1)}) = \lambda_1 + \lambda_2 + (n-2)/\theta_{k+1}$, and $\det(B_{k+1}^{(1)}) = \lambda_1\lambda_2/\theta_{k+1}^{n-2}$, from Equation (7) we obtain $\lambda_1 + \lambda_2 = a_k + 2/\theta_{k+1}$, while $\lambda_1\lambda_2 = 1/\theta_{k+1}^2$. Consequently, relation (9) follows immediately.

Note that the parameter a_k is bounded from below by two, since

$$a_k = 1 + \theta_{k+1}\frac{y_k^{\mathrm{T}} y_k}{s_k^{\mathrm{T}} y_k} = 1 + \frac{\|s_k\|^2 \|y_k\|^2}{(s_k^{\mathrm{T}} y_k)^2} = 1 + \frac{1}{\cos^2\phi} \geq 2, \tag{11}$$

where ϕ is the angle between s_k and y_k. If $a_k = 2$, then the characteristic polynomial is reduced to $p_1(\lambda) = (\lambda - 1/\theta_{k+1})^n$; thus $B_{k+1}^{(1)} = (1/\theta_{k+1})I = B_{k+1}^{(0)}$. In the different case (when $a_k > 2$), the characteristic polynomial becomes $p_1(\lambda) = (\lambda - 1/\theta_{k+1})^{n-2}(\lambda - \lambda_1)(\lambda - \lambda_2)$, where the eigenvalues $\lambda_{1,2} = (a_k \pm \sqrt{a_k^2 - 4})/(2\theta_{k+1})$ are distinct. From inequalities (10) follows that $\min(\lambda_1, \lambda_2) < (1/\theta_{k+1}) < \max(\lambda_1, \lambda_2)$. ■

LEMMA 3.3 *Let Λ be the set of eigenvalues of $B_{k+1}^{(1)}$ with opposite signs. Then, for any $\lambda \in \mathbb{R} \setminus \Lambda$, the matrix $(B_{k+1}^{(1)} + \lambda I)$ is invertible and its inverse can be expressed by the following closed-form*

$$(B_{k+1}^{(1)} + \lambda I)^{-1} = \frac{1}{\gamma(\lambda)} \sum_{i=0}^{2} (-1)^i \gamma_i(\lambda)(B_{k+1}^{(1)})^i, \tag{12}$$

where the quantities $\gamma = (1/\theta_{k+1} + \lambda)(\lambda^2 + a_k\lambda/\theta_{k+1} + 1/\theta_{k+1}^2)$, $\gamma_2 = 1$, $\gamma_1 = \lambda + (a_k + 1)/\theta_{k+1}$, and $\gamma_0 = \lambda^2 + (a_k + 1)\lambda/\theta_{k+1} + (a_k + 1)/\theta_{k+1}^2$ are functions of λ.

Proof Let $\lambda_i, i = 1, \ldots, n$ be the eigenvalues of $B_{k+1}^{(1)}$ and $\lambda \in \mathbb{R}$. If $\lambda \in \Lambda$, obviously the matrix $B_{k+1}^{(1)} + \lambda I = B_{k+1}^{(1)} - \lambda_i I$ is singular. Hence, for $B_{k+1}^{(1)} + \lambda I$ being invertible, relation $\lambda \in \mathbb{R} \setminus \Lambda$ must hold. Without loss of generality, we assume that $B_{k+1}^{(1)}$ has two distinct eigenvalues λ_1 and λ_2. Consequently, $(B_{k+1}^{(1)} + \lambda I)$ has also two distinct eigenvalues, $\lambda_1 + \lambda$ and $\lambda_2 + \lambda$. Using Lemma 3.2, after some algebraic calculations, we obtain the characteristic polynomial of $(B_{k+1}^{(1)} + \lambda I)$, which is

$$q(x) = (x - c)^{n-2}(x^2 - c_1 x + c_0),$$

where $c = 1/\theta_{k+1} + \lambda$, $c_1 = a_k/\theta_{k+1} + 2\lambda$, and $c_0 = \lambda^2 + a_k\lambda/\theta_{k+1} + 1/\theta_{k+1}^2$. Therefore, its minimal polynomial is $q_m(x) = (x - c)(x^2 - c_1 x + c_0)$. From the Caley–Hamilton theorem, we have that

$$q_m(B_{k+1}^{(1)} + \lambda I) = 0.$$

Multiplying both sides of the above equation with $(B_{k+1}^{(1)} + \lambda I)^{-1}$, after some calculations, we obtain relation (12). ■

It is easy to see that in the special case where $a_k = 2$, Equation (12) is reduced to $(B_{k+1}^{(1)} + \lambda I)^{-1} = (1/\theta_{k+1} + \lambda)^{-1} I$.

3.2 *2BFGS update*

When $m = 2$, the matrix $B_{k+1}^{(2)} = B_k^{(1)} - (B_k^{(1)} s_k s_k^{\mathrm{T}} B_k^{(1)} / s_k^{\mathrm{T}} B_k^{(1)} s_k) + (y_k y_k^{\mathrm{T}} / s_k^{\mathrm{T}} y_k)$, contains curvature information from the two previous iterations.

LEMMA 3.4 *Suppose that two updates are applied to the symmetric matrix $B^{(0)}$ using the vector pairs $\{s_i, y_i\}_{i=k-1}^{k}$ and the BFGS formula. The characteristic polynomial of the resulting matrix $B_{k+1}^{(2)}$ has the general form*

$$p_2(\lambda) = \left(\lambda - \frac{1}{\theta_k}\right)^{n-4} (\lambda^4 - \beta_3\lambda^3 + \beta_2\beta_0\lambda^2 - \beta_1\beta_0\lambda + \beta_0), \tag{13}$$

where the coefficients $\beta_i, i = 0, \ldots, 3$ are

$$\begin{aligned}
\beta_0 &= \frac{1}{\theta_k^4} \frac{s_k^{\mathrm{T}} y_k}{s_k^{\mathrm{T}} B_k^{(1)} s_k}, \quad \beta_1 = (a_{k-1} + 2)\theta_k - 2\frac{s_k^{\mathrm{T}} w_k}{s_k^{\mathrm{T}} y_k} + b_k\theta_{k+1}, \\
\beta_2 &= \theta_k(\beta_1 - a_{k-1}\theta_k)(a_{k-1} + 2) - 2\theta_k^2 + \left(\frac{s_k^{\mathrm{T}} w_k}{s_k^{\mathrm{T}} y_k}\right)^2 - \theta_{k+1}\frac{w_k^{\mathrm{T}} w_k}{s_k^{\mathrm{T}} y_k} \\
&\quad + 2\frac{s_k^{\mathrm{T}} H_k w_k}{s_k^{\mathrm{T}} y_k} - b_k\frac{s_k^{\mathrm{T}} H_k s_k}{s_k^{\mathrm{T}} y_k}, \\
\beta_3 &= \frac{a_{k-1} + 2}{\theta_k} - \frac{\|B_k^{(1)} s_k\|^2}{s_k^{\mathrm{T}} B_k^{(1)} s_k} + \frac{\|y_k\|^2}{s_k^{\mathrm{T}} y_k},
\end{aligned} \tag{14}$$

with $H_k = (B_k^{(1)})^{-1}$, $w_k = H_k y_k$, and $b_k = 1 + y_k^{\mathrm{T}} w_k / s_k^{\mathrm{T}} y_k$.

Proof We first show that $B_{k+1}^{(2)}$ has at most four distinct eigenvalues. Without loss of generality, we assume that $B_k^{(1)}$ and $B_{k+1}^{(2)}$ are positive definite matrices (in the different case, the proof

follows by similar arguments). Let $\lambda_i^{B^{(1)}}$ and $\lambda_i^{\bar{B}}$ be the eigenvalues of $B_k^{(1)}$ and $\bar{B} = B_k^{(1)} - (B_k^{(1)} s_k s_k^{\mathrm{T}} B_k^{(1)} / s_k^{\mathrm{T}} B_k^{(1)} s_k)$, respectively. Since $\lambda_2^{B^{(1)}} = \cdots = \lambda_{n-1}^{B^{(1)}} = 1/\theta_k$, by Lemma 3.1 we have that

$$\lambda_n^{B^{(1)}} \geq \lambda_n^{\bar{B}} \geq \frac{1}{\theta_k} \geq \lambda_{n-1}^{\bar{B}} \geq \frac{1}{\theta_k} \geq \cdots \geq \frac{1}{\theta_k} \geq \lambda_2^{\bar{B}} \geq \lambda_1^{B^{(1)}} \geq \lambda_1^{\bar{B}}.$$

The addition of the term $y_k y_k^{\mathrm{T}} / s_k^{\mathrm{T}} y_k$ to the matrix $\bar{B}$ yields

$$\lambda_n^{B^{(2)}} \geq \lambda_n^{\bar{B}} \geq \lambda_{n-1}^{B^{(2)}} \geq \frac{1}{\theta_k} \geq \cdots \geq \frac{1}{\theta_k} \geq \lambda_2^{B^{(2)}} \geq \lambda_2^{\bar{B}} \geq \lambda_1^{B^{(2)}} \geq \lambda_1^{\bar{B}},$$

where $\lambda_i^{B^{(2)}}$ are the eigenvalues of $B_{k+1}^{(2)}$. The above inequalities imply that

$$\lambda_n^{B^{(2)}} \geq \lambda_{n-1}^{B^{(2)}} \geq \frac{1}{\theta_k} \geq \lambda_2^{B^{(2)}} \geq \lambda_1^{B^{(2)}}, \tag{15}$$

and thus, the matrix $B_{k+1}^{(2)}$ has at most four distinct eigenvalues. If we denote by λ_1, λ_2, λ_3, and λ_4 the four unknown eigenvalues, the characteristic polynomial of $B_{k+1}^{(2)}$ takes the form

$$p_2(\lambda) = \left(\lambda - \frac{1}{\theta_k}\right)^{n-4} (\lambda^4 - c_3\lambda^3 + c_2\lambda^2 - c_1\lambda + c_0),$$

where

$$c_3 = \sum_{i=1}^{4} \lambda_i, \quad c_2 = \sum_{\substack{i,j \\ i<j}} \lambda_i \lambda_j, \quad c_1 = \sum_{\substack{i,j,\ell \\ i<j<\ell}} \lambda_i \lambda_j \lambda_\ell, \quad \text{and} \quad c_0 = \prod_{i=1}^{4} \lambda_i.$$

Moreover, $\prod_{i=1}^{4} \lambda_i = \det(B_{k+1}^{(2)}) \theta_k^{n-4}$ and $\sum_{i=1}^{4} \lambda_i = \mathrm{tr}(B_{k+1}^{(2)}) - ((n-4)/\theta_k)$. Utilizing Equation (7), after some algebraic manipulations, we yield the expression in Equation (13) with the parameters in Equation (14). ■

In case where $B_{k+1}^{(2)}$ has four distinct eigenvalues, these can be computed analytically by solving the quartic equation $\lambda^4 - \beta_3\lambda^3 + \beta_2\beta_0\lambda^2 - \beta_1\beta_0\lambda + \beta_0 = 0$, using standard methods [7,10]. However, the exact number of the distinct eigenvalues is strongly depends on the way the vector pairs $\{s_i, y_i\}_{i=k-1}^{k}$ are related. Their relation can be specified by the values of a_i. If $a_i = 2$ the vectors s_i, y_i are collinear, else if $a_i > 2$ the vectors s_i, y_i are linearly independent. The following proposition establishes sufficient conditions for the exact number of the distinct eigenvalues of $B_{k+1}^{(2)}$.

PROPOSITION 3.5 *Let the symmetric matrix $B_{k+1}^{(2)} \in \mathbb{R}^{n\times n}$. Then,*

(1) *If $a_{k-1} = 2$, $B_{k+1}^{(2)}$ has at most two distinct eigenvalues. Moreover,*
   (a) *if $a_k > 2$, then two distinct eigenvalues exist,*
   (b) *if $a_k = 2$ and $\theta_k \neq \theta_{k+1}$, then only one distinct eigenvalue exists, and*
   (c) *if $a_k = 2$ and $\theta_k = \theta_{k+1}$, then all the eigenvalues are equal to $1/\theta_k$.*

(2) *If $a_{k-1} > 2$, $B_{k+1}^{(2)}$ has at least three distinct eigenvalues. Moreover,*
   (a) *if $a_k = 2$, then only three distinct eigenvalues exist, and*
   (b) *if $a_k > 2$, then only four distinct eigenvalues exist.*

Proof Assume that $a_{k-1} = 2$. From Lemma 3.2 we have that $B_k^{(1)} = (1/\theta_k)I$ and consequently $B_{k+1}^{(2)} = (1/\theta_k)I - (1/\theta_k)(s_k s_k^{\mathrm{T}}/s_k^{\mathrm{T}} s_k) + (y_k y_k^{\mathrm{T}}/s_k^{\mathrm{T}} y_k)$. In this case the characteristic polynomial of $B_{k+1}^{(2)}$ becomes

$$p_2(\lambda) = \left(\lambda - \frac{1}{\theta_k}\right)^{n-2} \left[\lambda^2 - \left(\frac{a_k - 1}{\theta_{k+1}} + \frac{1}{\theta_k}\right)\lambda + \frac{1}{\theta_k \theta_{k+1}}\right]. \tag{16}$$

In view of Equation (16), it follows immediately that $B_{k+1}^{(2)}$ has at most two distinct eigenvalues. Consider now the quadratic equation

$$\lambda^2 - \left(\frac{a_k - 1}{\theta_{k+1}} + \frac{1}{\theta_k}\right)\lambda + \frac{1}{\theta_k \theta_{k+1}} = 0,$$

which has the following two solutions

$$\lambda_{1,2} = \frac{\theta_{k+1} + \theta_k(a_k - 1) \pm \sqrt{(\theta_k - \theta_{k+1})^2 + \theta_k(a_k - 2)(a_k \theta_k + 2\theta_{k+1})}}{2\theta_k \theta_{k+1}}.$$

Let $a_k > 2$ and suppose that either λ_1 or λ_2 equals to $1/\theta_k$. Solving the equations $\lambda_i - 1/\theta_k = 0$ with respect to a_k, we obtain $a_k = 2$, which is a contradiction. Thus, when $a_k > 2$ the matrix has exactly two distinct eigenvalues. When $a_k = 2$, the characteristic polynomial becomes $p_2(\lambda) = (\lambda - 1/\theta_k)^{n-1}(\lambda - 1/\theta_{k+1})$. Therefore, in the case where $\theta_k \neq \theta_{k+1}$, the only distinct eigenvalue is $1/\theta_{k+1}$; otherwise $B_{k+1}^{(2)}$ does not have distinct eigenvalues.

Assume now that $a_{k-1} > 2$. In Lemma 3.2 we have proved that in this case, $B_k^{(1)}$ has two distinct eigenvalues. Thus, the matrix $\bar{B} = B_k^{(1)} - (B_k^{(1)} s_k s_k^{\mathrm{T}} B_k^{(1)}/s_k^{\mathrm{T}} B_k^{(1)} s_k)$, besides the zero eigenvalue, has two more distinct eigenvalues. Consequently, $B_{k+1}^{(2)}$ has at least three distinct eigenvalues. If $a_k = 2$, then $s_k = \theta_{k+1} y_k$ and $B_{k+1}^{(2)}$ becomes

$$B_{k+1}^{(2)} = B_k^{(1)} - \frac{B_k^{(1)} y_k y_k^{\mathrm{T}} B_k^{(1)}}{y_k^{\mathrm{T}} B_k^{(1)} y_k} + \frac{y_k y_k^{\mathrm{T}}}{\theta_{k+1} y_k^{\mathrm{T}} y_k}.$$

Hence, $\bar{B} = B_k^{(1)} - (B_k^{(1)} y_k y_k^{\mathrm{T}} B_k^{(1)})/(y_k^{\mathrm{T}} B_k^{(1)} y_k)$, and therefore, the addition of $y_k y_k^{\mathrm{T}}/(\theta_{k+1} y_k^{\mathrm{T}} y_k)$ changes the zero eigenvalue of $\bar{B}$ to $1/\theta_{k+1}$ and leaves the others unchanged. Thus, $B_{k+1}^{(2)}$ has as many distinct eigenvalues as $\bar{B}$, i.e. three. Since one of them is $1/\theta_{k+1}$, utilizing Lemma 3.4, the characteristic polynomial of $B_{k+1}^{(2)}$ becomes

$$p_2(\lambda) = \left(\lambda - \frac{1}{\theta_k}\right)^{n-3} \left(\lambda - \frac{1}{\theta_{k+1}}\right)(\lambda^2 - c_1\lambda + c_2),$$

where $c_1 = a_{k-1}/\theta_k - \|B_k^{(1)} y_k\|^2/y_k^{\mathrm{T}} B_k^{(1)} y_k$ and $c_2 = y_k^{\mathrm{T}} y_k/(\theta_k^2 \theta_{k+1} y_k^{\mathrm{T}} B_k^{(1)} y_k)$. By solving the quadratic equation $\lambda^2 - c_1\lambda + c_2 = 0$, we obtain the other two distinct eigenvalues. Finally, if $a_k > 2$, then y_k is not an eigenvector of $\bar{B} = B_k^{(1)} - (B_k^{(1)} s_k s_k^{\mathrm{T}} B_k^{(1)}/s_k^{\mathrm{T}} B_k^{(1)} s_k)$. Hence, the addition of the term $y_k y_k^{\mathrm{T}}/s_k^{\mathrm{T}} y_k$ results one more distinct eigenvalue for $B_{k+1}^{(2)}$. ■

LEMMA 3.6 *If $B_{k+1}^{(2)}$ has at least two distinct eigenvalues, then*

$$\lambda_1 < 1/\theta_k < \lambda_n, \tag{17}$$

where λ_1 and λ_n are the smallest and largest eigenvalues of $B_{k+1}^{(2)}$, respectively.

Proof When $B_{k+1}^{(2)}$ has at least three distinct eigenvalues, from relation (15) it is evident that Equation (17) holds. Now, if $B_{k+1}^{(2)}$ has two distinct eigenvalues, then one pair of the vector set $\{(s_{k-1}, y_{k-1}),\ (s_k, y_k)\}$ must be collinear. Due to Proposition 3.5, these two vectors are s_{k-1} and y_{k-1}, which implies that $B_k^{(1)}$ has no distinct eigenvalues. Under these circumstances, Lemma 3.1 implies relation (17), which completes the proof. ■

LEMMA 3.7 *Let Λ be the set of eigenvalues of $B_{k+1}^{(2)}$ with opposite signs. For any $\lambda \in \mathbb{R} \setminus \Lambda$, the matrix $(B_{k+1}^{(2)} + \lambda I)$ is invertible, and its inverse can be expressed by the following closed-form*

$$(B_{k+1}^{(2)} + \lambda I)^{-1} = \frac{1}{\nu(\lambda)} \sum_{i=0}^{4} (-1)^i \nu_i(\lambda) (B_{k+1}^{(2)})^i, \tag{18}$$

where $\nu_4 = 1$, $\nu_3 = \lambda + \beta_3 + 1/\theta_k$, $\nu_2 = \lambda\nu_3 + \beta_2\beta_0 + \beta_3/\theta_k$, $\nu_1 = \lambda\nu_2 + \beta_1\beta_0 + \beta_2\beta_0/\theta_k$, $\nu_0 = \lambda\nu_1 + \beta_0 + \beta_0\beta_1/\theta_k$, and $\nu = \lambda\nu_0 + \beta_0/\theta_k$ are functions of λ, while the parameters β_0, β_1, β_2 and β_3 are defined as in Lemma 3.4.

Proof Obviously, as in the proof of Lemma 3.3, for $B_{k+1}^{(2)} + \lambda I$ being invertible, relation $\lambda \in \mathbb{R} \setminus \Lambda$ must hold. From Proposition 3.5, we know that $B_{k+1}^{(2)}$ has at most four distinct eigenvalues, λ_i, $i = 1, \ldots, 4$. Thus, $(B_{k+1}^{(2)} + \lambda I)$ has also at most four distinct eigenvalues, $\lambda_i + \lambda$. Utilizing Lemma 3.4, the characteristic polynomial of $(B_{k+1}^{(2)} + \lambda I)$ takes the form

$$q(x) = (x - c)^{n-4}(x^4 - c_3x^3 + c_2x^2 - c_1x + c_0),$$

where $c = \theta_k^{-1} + \lambda$, $c_3 = 4\lambda + \beta_3$, $c_2 = 6\lambda^2 + 3\beta_3\lambda + \beta_2\beta_0$, $c_1 = 4\lambda^3 + 3\beta_3\lambda^2 + 2\beta_2\beta_0\lambda + \beta_1\beta_0$, and $c_0 = \lambda^4 + \beta_3\lambda^3 + \beta_2\beta_0\lambda^2 + \beta_1\beta_0\lambda + \beta_0$, where the constants β_i are defined in Lemma 3.4. In the general case, its minimal polynomial is

$$q_m(x) = (x - c)(x^4 - c_3x^3 + c_2x^2 - c_1x + c_0).$$

Using the Caley–Hamilton theorem, we have that

$$q_m(B_{k+1}^{(2)} + \lambda I) = 0.$$

Multiplying both sides of the above equation with $B_{k+1}^{(2)} + \lambda I$, after some calculations, we obtain Equation (18). ■

4. The step computing function

In Section 2 we have seen that when the Lagrange multiplier $\lambda \in (-\lambda_1, \infty)$ (*standard* case), then $d(\lambda) = -(B + \lambda I)^{-1} g$. In the hard case, a negative curvature direction must be computed in order to ensure that $\|d\| = \Delta$. In the case where λ_1 is a multiple eigenvalue, due to the structure of B, this computation is achieved by simple algebraic computation. In the different case, we are able to find an analytical expression for the desirable eigenvector using the inverse power iteration method [3, pp. 611].

Given a nonzero starting vector $u^{(0)}$, inverse iteration generates a sequence of vectors $u^{(i)}$, generated recursively by the formula

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

$i \geq 1$, where $\hat{\lambda} = \lambda + \epsilon$, λ is a distinct eigenvalue, 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 $(1, 1, \ldots, 1)^{\mathrm{T}}$. Moreover, if λ is an exact eigenvalue of B, this method converges in a single iteration [15], providing a closed form for the corresponding eigenvector.

4.1 *Computation of the step using the 1BFGS update*

4.1.1 *The standard case*

First we consider the standard case, where $\lambda \in (-\lambda_1, \infty)$. When $a_k > 2$, the trial step is computed using Equation (12), which yields $d_{k+1}(\lambda) = -\frac{1}{\gamma(\lambda)} \sum_{i=0}^{2} (-1)^i \gamma_i(\lambda) (B_{k+1}^{(1)})^i g_{k+1}$. The vectors $B_{k+1}^{(1)} g_{k+1}$ and $(B_{k+1}^{(1)})^2 g_{k+1}$ are computed by the iterative scheme

$$v_{i+1} \equiv B_{k+1}^{(1)} v_i = \frac{1}{\theta_{k+1}} v_i - \frac{s_k^{\mathrm{T}} v_i}{\theta_{k+1} s_k^{\mathrm{T}} s_k} s_k + \frac{y_k^{\mathrm{T}} v_i}{s_k^{\mathrm{T}} y_k} y_k, \quad i = 0, 1, \tag{19}$$

with $v_0 = g_{k+1}$. After some calculations, we obtain the general form of the trial step:

$$d_{k+1}(\lambda) = -\gamma_g(\lambda) g_{k+1} + \gamma_s(\lambda) s_k - \gamma_y(\lambda) y_k, \tag{20}$$

where

$$\gamma_g(\lambda) = \frac{1 - \gamma_1(\lambda)\theta_{k+1} + \gamma_0(\lambda)\theta_{k+1}^2}{\gamma(\lambda)\theta_{k+1}^2},$$

$$\gamma_s(\lambda) = \frac{[1 - \gamma_1(\lambda)\theta_{k+1}] s_k^{\mathrm{T}} g_{k+1} + \theta_{k+1} y_k^{\mathrm{T}} g_{k+1}}{\gamma(\lambda)\theta_{k+1}^2 s_k^{\mathrm{T}} s_k}, \quad \text{and}$$

$$\gamma_y(\lambda) = \frac{[1 - \gamma_1(\lambda)\theta_{k+1} + a_k]\theta_{k+1} y_k^{\mathrm{T}} g_{k+1} - s_k^{\mathrm{T}} g_{k+1}}{\gamma(\lambda)\theta_{k+1}^2 s_k^{\mathrm{T}} y_k}.$$

Moreover, for $\lambda = 0$, we yield $\gamma_g(0) = \theta_{k+1}$, $\gamma_s(0) = (\theta_{k+1} y_k^{\mathrm{T}} g_{k+1} - a_k s_k^{\mathrm{T}} g_{k+1})/s_k^{\mathrm{T}} y_k$, and $\gamma_y(0) = -\theta_{k+1} s_k^{\mathrm{T}} g_{k+1}/s_k^{\mathrm{T}} y_k$. Therefore,

$$d_{k+1}(0) \equiv -(B_{k+1}^{(1)})^{-1} g_{k+1} = -\gamma_g(0)\ g_{k+1} + \gamma_s(0) s_k - \gamma_y(0) y_k. \tag{21}$$

When $a_k = 2$, from Lemma 3.2 we have that $B_{k+1}^{(1)} = (1/\theta_{k+1}) I$. Hence

$$d_{k+1}(\lambda) = -\left(\frac{1}{\theta_{k+1}} + \lambda\right)^{-1} g_{k+1}. \tag{22}$$

For the computation of the Lagrange multiplier λ we can follow the ideas described in Moré and Sorensen [19] by applying Newton's method as given in Equation (3). It can be easily verified that the resulting iteration is

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

for $\ell = 0, 1, 2, \ldots$. Denoting by $\Pi = d(\lambda)^{\mathrm{T}} (B_{k+1}^{(2)} + \lambda I)^{-1} d(\lambda)$ the denominator in (23), using Equation (12) we yield

$$\Pi = -\frac{\gamma_1(\lambda)}{\gamma(\lambda)} d_{k+1}(\lambda)^{\mathrm{T}} B_{k+1}^{(1)} d_{k+1}(\lambda) + \frac{\gamma_0(\lambda)}{\gamma(\lambda)} \|d_{k+1}(\lambda)\|^2 + \frac{\|B_{k+1}^{(1)} d_{k+1}(\lambda)\|^2}{\gamma(\lambda)}.$$

Since $B_{k+1}^{(1)} d_{k+1}(\lambda) = -[\lambda d_{k+1}(\lambda) + g_{k+1}]$ holds, the above relation can be written as

$$\Pi = \sum_{i=0}^{2} \frac{\gamma_i(\lambda)\lambda^i}{\gamma(\lambda)} \|d_{k+1}(\lambda)\|^2 + \sum_{i=1}^{2} \frac{i\gamma_i(\lambda)\lambda^{i-1}}{\gamma(\lambda)} d_{k+1}(\lambda)^{\mathrm{T}} g_{k+1} + \frac{\|g_{k+1}\|^2}{\gamma(\lambda)}. \tag{24}$$

In the special case where $a_k = 2$, utilizing Equation (22), relation (24) is reduced to

$$\Pi = \left(\frac{1}{\theta_{k+1}} + \lambda\right)^{-3} \|g_{k+1}\|^2. \tag{25}$$

4.1.2 *The hard case*

We recall that in the hard case, g is perpendicular to all eigenvectors corresponding to λ_1, $\lambda^* = -\lambda_1$, and a direction of negative curvature must be produced. When $a_k > 2$, from Lemma 3.2 we know that λ_1 is a distinct eigenvalue. In this case, using the inverse iteration along with Equation (12) we have that

$$\hat{u}_1 = \sum_{i=0}^{2} (-1)^i \gamma_i(\hat{\lambda})(B_{k+1}^{(1)})^i \frac{u}{\gamma(\hat{\lambda})} = -\gamma_u(\hat{\lambda})u + \gamma_{us}(\hat{\lambda})s_k - \gamma_{uy}(\hat{\lambda})y_k, \tag{26}$$

where $\hat{\lambda} = -\lambda_1 + \epsilon$, $u = u^{(0)}/\|u^{(0)}\|$,

$$\gamma_u(\hat{\lambda}) = \frac{1 - \gamma_1(\hat{\lambda})\theta_{k+1} + \gamma_0(\hat{\lambda})\theta_{k+1}^2}{\gamma(\hat{\lambda})\theta_{k+1}^2},$$

$$\gamma_{us}(\hat{\lambda}) = \frac{[1 - \gamma_1(\hat{\lambda})\ \theta_{k+1}]s_k^{\mathrm{T}}u + \theta_{k+1}y_k^{\mathrm{T}}u}{\gamma(\hat{\lambda})\ \theta_{k+1}^2 s_k^{\mathrm{T}}s_k}, \quad \text{and}$$

$$\gamma_{uy}(\hat{\lambda}) = \frac{[1 - \gamma_1(\hat{\lambda})\ \theta_{k+1} + a_k]\theta_{k+1}y_k^{\mathrm{T}}u - s_k^{\mathrm{T}}u}{\gamma(\hat{\lambda})\ \theta_{k+1}^2 s_k^{\mathrm{T}}y_k}.$$

Therefore, $u_1 = \hat{u}_1/\|\hat{u}_1\|$ and the trial step is computed by the formula

$$d_{k+1} = -\gamma_g(\hat{\lambda})g_{k+1} + \gamma_s(\hat{\lambda})s_k - \gamma_y(\hat{\lambda})y_k + \tau u_1,$$

where τ is obtained by Equation (2). In case where $a_k = 2$, using the eigendecomposition of $B^{(1)}$ we have that $B^{(1)} = U\Lambda U^{\mathrm{T}}$, where $U = I$ and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_1, \ldots, \lambda_1)$. Easily can be verified that an eigenvector corresponding to λ_1 is $u_1 = e_1 = (1, 0, \ldots, 0)^{\mathrm{T}}$.

4.2 *Computation of the step using the 2BFGS update*

4.2.1 *The standard case*

When $\lambda \in (\lambda_1, \infty)$ and the 2BFGS update is performed, the trial step $d(\lambda) = -(B + \lambda I)^{-1}g$ can be computed from Equation (18), using the following procedure:

PROCEDURE 4.1 *Compute* d.

$d^{(4)} = g$;
for $j = 4, \ldots, 1$ **do**

$v \leftarrow B^{(2)} d^{(j)}$;
$d^{(j-1)} \leftarrow v + (-1)^{j-1} v_{j-1}(\lambda) g$;
end for
return $-d^{(0)}/v(\lambda)$;

The vector $v \in \mathbb{R}^n$ is computed by the formula

$$v \equiv B_{k+1}^{(2)} d^{(j)} = B_k^{(1)} d^{(j)} - \frac{s_k^{\mathrm{T}} B_k^{(1)} d^{(j)}}{s_k^{\mathrm{T}} B_k^{(1)} s_k} B_k^{(1)} s_k + \frac{y_k^{\mathrm{T}} d^{(j)}}{s_k^{\mathrm{T}} y_k} y_k,$$

while the vectors $B_k^{(1)} d^{(j)}$ and $B_k^{(1)} s_k$ are computed by means of (19), substituting v_i with $d^{(j)}$ and s_k, respectively. In case where $\lambda = 0$, then

$$d_{k+1}(0) \equiv -H_{k+1} g_{k+1} = -H_k g_{k+1} + \frac{w_k^{\mathrm{T}} g_{k+1} - b_k s_k^{\mathrm{T}} g_{k+1}}{s_k^{\mathrm{T}} y_k} s_k + \frac{s_k^{\mathrm{T}} g_{k+1}}{s_k^{\mathrm{T}} y_k} w_k, \tag{27}$$

where $H_{k+1} = (B_{k+1}^{(2)})^{-1}$ and $H_k = (B_k^{(1)})^{-1}$. If $a_{k-1} = a_k = 2$, then $B_{k+1}^{(2)}$ takes the form

$$B_{k+1}^{(2)} = \frac{1}{\theta_k} I + \left(\frac{1}{\theta_{k+1}} - \frac{1}{\theta_k} \right) \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} y_k}, \tag{28}$$

and the trial step is obtained by the following equation:

$$d_{k+1}(\lambda) = -\left[\left(\lambda + \frac{1}{\theta_k} + \frac{1}{\theta_{k+1}} \right) I - B_{k+1}^{(2)} \right] \frac{g_{k+1}}{(\lambda + 1/\theta_k)(\lambda + 1/\theta_{k+1})}.$$

Using the 2BFGS update, the expression $\Pi = d(\lambda)^{\mathrm{T}} (B + \lambda I)^{-1} d(\lambda)$ of the denominator in Newton's iteration (23) becomes

$$\begin{aligned} \Pi = \frac{1}{v(\lambda)} \Bigg\{ & \sum_{i=0}^{4} [\lambda^i v_i(\lambda)] \|d_{k+1}(\lambda)\|^2 + \sum_{i=1}^{4} [i \lambda^{i-1} v_i(\lambda)] d_{k+1}(\lambda)^{\mathrm{T}} g_{k+1} \\ & + \sum_{i=2}^{4} [(i-1) \lambda^{i-2} v_i(\lambda)] \|g_{k+1}\|^2 - \sum_{i=3}^{4} [(i-2) \lambda^{i-3} v_i(\lambda)] g_{k+1}^{\mathrm{T}} B_{k+1}^{(2)} g_{k+1} \\ & + \|B_{k+1}^{(2)} g_{k+1}\|^2 \Bigg\}. \end{aligned}$$

When $a_{k-1} = a_k = 2$, the above equation yields

$$\Pi = \frac{[2\lambda + (1/\theta_k) + (1/\theta_{k+1})] \|d_{k+1}(\lambda)\|^2 + d_{k+1}(\lambda)^{\mathrm{T}} g_{k+1}}{[\lambda + (1/\theta_k))(\lambda + (1/\theta_{k+1})]}.$$

4.2.2 *The hard case*

Proposition 3.5 along with Lemma 3.6 reveals that if λ_1 is a distinct eigenvalue of $B^{(2)}$, then either $B^{(2)}$ has at least two distinct eigenvalues or it has exactly one equals to $\lambda_1 = 1/\theta_{k+1}$. In the first

case (at least two distinct eigenvalues), we can find the eigenvector corresponds to λ_1 by applying the inverse iteration. The resulting eigenvector is $u_1 = \hat{u}_1/\|\hat{u}_1\|$, where

$$\hat{u}_1 = \left[\sum_{i=0}^{4}(-1)^i \nu_i(\hat{\lambda})(B_{k+1}^{(2)})^i\right]\frac{u}{\nu(\hat{\lambda})}, \tag{29}$$

$\hat{\lambda} = -\lambda_1 + \epsilon$ and $u = (1, \ldots, 1)^{\mathrm{T}}/\|u\|$. The computation of $\hat{u}_1$ can be obtained using Procedure 4.1. In the latter case (one distinct eigenvalue), from Equation (28) it is straightforward to see that $u_1 = y_k/\|y_k\|$.

When λ_1 is a multiple eigenvalue, then $a_{k-1} = a_k = 2$ and $\lambda_1 = 1/\theta_k$. If $\theta_k \neq \theta_{k+1}$, from Equation (28) easily can be verified that the resulting eigenvector is of the form $u_1 = \hat{u}_1/\|\hat{u}_1\|$, where

$$\hat{u}_1 = \left(-\frac{y_k^{(n)}}{y_k^{(1)}}, 0, \ldots, 0, 1\right)^{\mathrm{T}}, \tag{30}$$

and $y_k^{(i)}$ denotes the ith component of y_k. In contrast, if $\theta_k = \theta_{k+1}$, then $u_1 = e_1 = (1, 0, \ldots, 0)^{\mathrm{T}}$.

5. The trial step algorithm

In the previous section we have discussed how Newton's method can be applied for solving the TRS (1), when the approximate Hessian is computed by the L-BFGS formula for $m = 1$ or 2. The following algorithm is a unified algorithm that incorporates both the standard and the hard case and computes an approximate solution of the subproblem (1). The safeguarding scheme required for Newton's iteration (23) uses the parameters λ_L and λ_{U} such that $[\lambda_{\mathrm{L}}, \lambda_{\mathrm{U}}]$ is an interval of uncertainty which contains the optimal λ^*. From Equation (4) we have that $\lambda_{\mathrm{L}} = \max(0, -\lambda_1 + \epsilon)$, and $\lambda_{\mathrm{U}} = \max_{1\leq i\leq n}|\lambda_i| + (1+\epsilon)\|g\|/\Delta$. Clearly, the lower bound λ_{L} is greater than $-\lambda_1$, which ensures that $B^{(m)} + \lambda I$ is always positive definite.

ALGORITHM 1 *Computation of the trial step.*

Step 1 : Given $m \in \{1, 2\}$, compute the eigenvalues λ_i of $B^{(m)}$; given $\epsilon \to 0^+$, set $\lambda_{\mathrm{L}} := \max(0, -\lambda_1 + \epsilon)$ and $\lambda_{\mathrm{U}} := \max|\lambda_i| + (1+\epsilon)\|g\|/\Delta$.

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 $\| - (B^{(m)} + \hat{\lambda} I)g + \tau u_1\| = \Delta$; set $d := d + \tau u_1$ and stop.

Step 4 : Use Newton's method to find $\lambda \in [\lambda_{\mathrm{L}}, \lambda_{\mathrm{U}}]$ and compute $d(\lambda)$.

Step 5 : If $\|d\| \leq \Delta$ stop; else update λ_{L} and λ_{U} such that $\lambda_{\mathrm{L}} \leq \lambda \leq \lambda_{\mathrm{U}}$ and go to Step 4.

When $m = 1$, the eigenvalues in Step 1 of Algorithm 1 can be computed by Equation (9). In Step 2, the trial step is obtained by Equation (21), while in Steps 3 and 4, by means of Equation (20). In Step 3(c) the computation of τ is obtained from Equation (2), while if $a_k > 2$, u_1 is obtained using Equation (26), otherwise we set $u_1 = e_1$.

When $m = 2$, the eigenvalues can be computed by means of Equation (13). The computation of d in Step 2 is done using Equation (27), while in Steps 3 and 4 by means of Procedure 4.1.

In Step 3(c), τ is obtained from Equation (2). Moreover, if $a_{k-1} > 2$ or $a_k > 2$, then u_1 is computed using Equation (29), along with Procedure 4.1. In the different case, i.e. if $a_{k-1} = a_k = 2$, then:

(1) $u_1 = y_k/\|y_k\|$, if λ_1 is distinct eigenvalue,
(2) $u_1 = \hat{u}_1/\|\hat{u}_1\|$, where $\hat{u}_1$ is defined in Equation (30), if λ_1 has multiplicity $n-1$,
(3) $u_1 = e_1$, if λ_1 has multiplicity n.

As we can see, the computation of the step does not require the Cholesky factorization of $B^{(m)} + \lambda^{(\ell)} I$. Thus, Algorithm 1 can solve inexpensively the subproblem, and therefore it can be iterate until convergence to the optimal λ is obtained with high accuracy. Moreover, the knowledge of the extreme eigenvalues, along with Lemma 2.2, results in a straightforward safeguarding procedure for λ.

The following lemma is established by Powell [25] and gives a lower bound for the maximum reduction $\phi(0) - \phi(d^*)$ of the quadratic model within the trust region $\|d\| \leq \Delta$.

LEMMA 5.1 *If d^* is a solution of the subproblem* (1)*, then*

$$\phi(0) - \phi(d^*) \geq \frac{1}{2}\|g\| \min\{\Delta, \|g\|/\|B\|\}.$$

Global convergence theory of trust region algorithms requires that the trial step d_k must satisfy the inequality

$$\phi_k(0) - \phi(d_k) \geq c\|g_k\| \min\{\Delta_k, \|g_k\|/\|B_k\|\} \tag{31}$$

for all k, where c is a positive constant (see [9,21]). The following theorem shows that relation (31) is satisfied if the trial step is computed by Algorithm 1.

THEOREM 5.2 *Assume that $\|B_k^{(m)}\| \leq M < \infty$, $m = 1, 2$, for all k, where M is a positive constant. If the trial step d_k is computed approximately by Algorithm* 1*, then*

$$\phi_k(0) - \phi_k(d_k(\lambda)) \geq \frac{1}{8}\|g_k\| \min\{\Delta_k, \ \|g_k\|/\|B_k^{(m)}\|\}. \tag{32}$$

Proof We recall that at each iteration k, the quadratic model is of the form

$$\phi_k(d_k) = d_k^{\mathrm{T}} g_k + \frac{1}{2} d_k^{\mathrm{T}} B_k^{(m)} d_k. \tag{33}$$

We consider the following two cases:

(i) If $B_k^{(m)} > 0$, then from Step 2, we have that if $\lambda = 0$ and $\|d_k\| \leq \Delta_k$, the algorithm stops. In this case,

$$g_k^{\mathrm{T}} d_k = -g_k (B_k^{(m)})^{-1} g_k \leq \frac{-\|g_k\|^2}{\|B_k^{(m)}\|}. \tag{34}$$

Then, from (33) and (34) we have that

$$\begin{aligned}
\phi_k(0) - \phi_k(d_k(\lambda)) &= -d_k^{\mathrm{T}}(\lambda) g_k - \frac{1}{2} d_k^{\mathrm{T}}(\lambda) B_k^{(m)} d_k(\lambda) \\
&= -d_k^{\mathrm{T}}(\lambda) g_k - \frac{1}{2} d_k^{\mathrm{T}}(\lambda)(B_k^{(m)} + \lambda I) d_k(\lambda) + \frac{1}{2}\lambda \|d_k(\lambda)\|^2 \\
&= -\frac{1}{2} d_k^{\mathrm{T}}(\lambda) g_k + \frac{1}{2}\lambda \|d_k(\lambda)\|^2 \\
&\geq \frac{1}{2} \frac{\|g_k\|^2}{\|B_k^{(m)}\|}.
\end{aligned} \tag{35}$$

If $\lambda = 0$ and $\|d_k\| > \Delta_k$, Steps 4 and 5 of Algorithm 1 yield λ and d such that

$$0 < \lambda \leq \lambda_{\mathrm{U}} \quad \text{and} \quad \|d_k(\lambda)\| \leq \Delta_k,$$

respectively. Since $B_k^{(m)}$ is positive definite, we have that $\|B_k^{(m)} + \lambda I\| = \lambda_n + \lambda = \|B_k^{(m)}\| + \lambda$, where λ_n is the largest eigenvalue of $B_k^{(m)}$. Therefore, taking into account that $\lambda \leq \|B_k^{(m)}\| + (1+\epsilon)\|g_k\|/\Delta_k$ and

$$\|B_k^{(m)} + \lambda I\| = \|B_k^{(m)}\| + \lambda \leq 2\|B_k^{(m)}\| + \frac{(1+\epsilon)\|g_k\|}{\Delta_k} \leq 2\left(\|B_k^{(m)}\| + \frac{\|g_k\|}{\Delta_k}\right),$$

we have that

$$\begin{aligned} g_k^{\mathrm{T}} d_k(\lambda) &\leq -\frac{\|g_k\|^2}{\|(B_k^{(m)} + \lambda I)^{-1}\|} \\ &\leq -\frac{1}{2}\frac{\|g_k\|^2}{\|B_k^{(m)}\| + \|g_k\|/\Delta_k} \\ &\leq -\frac{1}{4}\|g_k\| \min(\Delta_k, \|g_k\|/\|B_k^{(m)}\|). \end{aligned} \tag{36}$$

Thus, the reduction of the model yields

$$\begin{aligned} \phi_k(0) - \phi_k(d_k(\lambda)) &= -\frac{1}{2} d_k^{\mathrm{T}}(\lambda) g_k + \frac{1}{2}\lambda \|d_k(\lambda)\|^2 \\ &\geq \frac{1}{8}\|g_k\| \min(\Delta_k, \|g_k\|/\|B_k^{(m)}\|). \end{aligned} \tag{37}$$

(ii) If $B_k^{(m)} \leq 0$, then $d_k^{\mathrm{T}}(\lambda) B_k^{(m)} d_k(\lambda) \leq 0$. From Step 3(a) along with Steps 4 and 5 we have that $\hat{\lambda} \leq \lambda \leq \lambda_{\mathrm{U}}$, and $\|d_k(\lambda)\| \leq \Delta_k$, where $\hat{\lambda} = -\lambda_1 + \epsilon$. Therefore, using relation (36), for the reduction of the model we obtain

$$\begin{aligned} \phi_k(0) - \phi_k(d_k(\lambda)) &= -d_k^{\mathrm{T}}(\lambda) g_k - \frac{1}{2} d_k^{\mathrm{T}}(\lambda) B_k^{(m)} d_k(\lambda) \\ &\geq -d_k^{\mathrm{T}}(\lambda) g_k \geq \frac{1}{4}\|g_k\| \min(\Delta_k, \|g_k\|/\|B_k^{(m)}\|). \end{aligned} \tag{38}$$

Finally, in Step 3(c), taking into account that $\lambda = \hat{\lambda}$, $u_1^{\mathrm{T}} g_k = 0$ and $\|B_k^{(m)} + \hat{\lambda} I\| \leq \lambda_n + \hat{\lambda} \leq 2\|B_k^{(m)}\|$, where λ_n is the largest eigenvalue of B_k, we obtain

$$g_k^{\mathrm{T}} d_k(\hat{\lambda}) = -g_k (B_k^{(m)} + \hat{\lambda} I)^{-1} g_k \leq -\frac{\|g_k\|^2}{\|B_k^{(m)} + \hat{\lambda}\|} \leq -\frac{\|g_k\|^2}{2\|B_k^{(m)}\|}. \tag{39}$$

Therefore, using relation (39) we have that

$$\begin{aligned} \phi_k(0) - \phi_k(d_k(\hat{\lambda})) &= -d_k^{\mathrm{T}}(\hat{\lambda}) g_k - \frac{1}{2} d_k^{\mathrm{T}}(\hat{\lambda}) B_k^{(m)} d_k(\hat{\lambda}) \\ &\geq -d_k^{\mathrm{T}}(\hat{\lambda}) g_k \geq \frac{1}{2}\frac{\|g_k\|^2}{\|B_k^{(m)}\|}. \end{aligned} \tag{40}$$

Combining relations (35), (37), (38) and (40), relation (32) follows immediately. ■

6. Numerical results

In the first part of this section, we use randomly generated instances of TRS to show both the efficacy and accuracy of our method on problems that have dimensions from 100 up to 100,000 variables. Then, we employ numerical examples to demonstrate the feasibility of the proposed method within a trust-region framework for unconstrained optimization.

For the numerical testing we implemented two versions of Algorithm 1, the first for $m = 1$ (`1BFGS`), and the second for $m = 2$ (`2BFGS`). The algorithms were coded in MATLAB 7.3, and all numerical experiments were performed on a Pentium 1.86 GHz personal computer with 2 GB of RAM running Linux operating system. Double precision IEEE floating point arithmetic with machine precision approximately 2.2×10^{-16} was employed.

We compared our method with the `GQTPAR` [1] and `LSTRS` algorithm [30]. The `GQTPAR` algorithm is a MATLAB version of the subroutine `GQTPAR.f` from the MINPACK package, based on the ideas described in Moré and Sorensen [19], which uses the Cholesky factorization for the solution of the TRS. The `LSTRS` algorithm proposed by Rojas *et al.* [29], remodels the TRS as a parameterized eigenvalue problem, and relies on matrix–vector products, since it computes the smallest eigenvalue and the corresponding eigenvector using a block Lanczos routine.

Note that for large dimensions ($n \geq 1000$) results are reported only for the `1BFGS`, `2BFGS` and `LSTRS` algorithm, since the `GQTPAR` algorithm requires the factorization of $B^{(m)}$.

6.1 *Random instances of TRS*

In this subsection, we present a summary of our random instances. For each dimension n, 100 random instances of TRS were generated as follows. The coordinates of the vectors g, s_m, and $y_m (m = 1, 2)$ were chosen independently from uniform distributions in the interval $(-10^5, +10^5)$. As a total, we have 3200 different instances, divided into two groups of medium-size (with $n = 100, 200, 300, 400, 500$) and larger-size ($n = 10^3, 10^4, 10^5$) problems. In addition, half of the instances illustrate the behaviour of the proposed method in the standard case, and the rest of them in the hard case. 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 $|(\|d\| - \Delta)/\Delta| \leq \text{tol}$ was computed for both the standard and the hard case. In all algorithms, the tolerance tol was set to 10^{-8}. The maximum number of Newton's iterations allowed was 200 and the trust region radius was fixed as $\Delta = 10$.

In order to create instances for the hard case, when $n \leq 500$, MATLAB's `eigs` routine was used to compute the smallest eigenvalue λ_1 and the corresponding eigenvector u of the reconstructed matrix $B^{(m)}$ from the vector pairs. For the `1BFGS`, `2BFGS` and `GQTPAR` algorithms 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 $g = (-u(n)/u(1), 0, \ldots, 0, 1)^{\mathrm{T}}$. For the `LSTRS` algorithm, we set $\Delta = 2.0\Delta_{hc}$ and $g = g - u(g^{\mathrm{T}}u)$. When $n \geq 1000$, we set $\Delta_{hc} = \|(B + \hat{\lambda} I)^{-1} g\|$, where $\hat{\lambda} = -\lambda_1 + \text{tol}$ and $(B + \hat{\lambda} I)^{-1} g$ were computed using the analytical formulas presented in Section 3.

The numerical results are summarized in Tables 1–6. Each row of these tables shows the dimension of the problem (n), the average number of Newton's iterations (it), the average relative accuracy (acc) of the trust-region solution $|(\|d\| - \Delta)/\Delta|$, and the average CPU time (cpu) in seconds. In the case where $\lambda = 0$ and $\|d\| < \Delta$, we used the absolute value of the product $\lambda(\|d\| - \Delta)$ instead of the relative accuracy $|(\|d\| - \Delta)/\Delta|$. Tables 1–4 summarize the results of the medium-size problems, while Tables 5 and 6 summarize the results of the larger-size problems. We recall that in the last two tables no results are reported for the `GQTPAR` algorithm, since it requires too much storage.

Table 1. Comparison results for the standard case ($m = 1$), 100 instances per dimension.

	1BFGS			GQTPAR			LSTRS		
n	it	acc	cpu	it	acc	cpu	it	acc	cpu
100	1.0	4.1e−15	6.8e−04	0.07	5.8e−07	1.1e−02	2.1	4.2e−10	1.4e−02
200	1.0	5.9e−12	7.2e−04	0.05	3.7e−08	5.1e−02	2.0	4.6e−10	1.4e−02
300	1.0	1.7e−16	6.7e−04	0.06	6.2e−13	1.5e−01	2.0	1.2e−10	1.4e−02
400	1.0	2.1e−16	7.8e−04	0.07	1.9e−12	3.4e−01	2.0	7.9e−11	1.4e−02
500	1.0	2.0e−16	6.9e−04	0.03	2.7e−13	6.2e−01	2.0	3.0e−12	1.4e−02

Table 2. Comparison results for the hard case ($m = 1$), 100 instances per dimension.

	1BFGS			GQTPAR			LSTRS		
n	it	acc	cpu	it	acc	cpu	it	acc	cpu
100	0.0	1.1e−16	2.4e−04	10.6	8.4e−03	7.9e−02	15.1	4.9e−11	6.5e−02
200	0.0	1.3e−16	6.2e−04	10.0	2.8e−02	4.2e−01	16.5	2.0e−02	6.9e−02
300	0.0	1.4e−16	4.8e−04	7.7	5.2e−02	9.9e−01	17.1	5.7e−11	8.0e−02
400	0.0	1.4e−16	6.0e−04	5.8	1.7e−03	2.0e+00	17.4	1.0e−10	7.4e−02
500	0.0	1.6e−16	5.9e−04	6.7	4.3e−01	4.0e+00	17.9	1.1e−02	7.8e−02

Table 3. Comparison results for the standard case ($m = 2$), 100 instances per dimension.

	2BFGS			GQTPAR			LSTRS		
n	it	acc	cpu	it	acc	cpu	it	acc	cpu
100	1.0	1.8e−16	1.6e−03	0.14	1.2e−11	1.0e−02	2.0	6.5e−10	1.4e−02
200	1.0	2.9e−16	1.7e−03	0.09	3.3e−12	5.1e−02	2.0	3.6e−10	1.5e−02
300	1.0	3.5e−16	2.0e−03	0.08	1.0e−12	1.5e−01	2.0	2.6e−10	1.5e−02
400	1.0	5.6e−16	2.0e−03	0.09	5.5e−11	3.8e−01	2.0	2.8e−10	1.5e−02
500	1.0	3.1e−12	2.1e−03	0.06	6.1e−09	6.4e−01	2.0	1.3e−10	1.5e−02

Table 4. Comparison results for the hard case ($m = 2$), 100 instances per dimension

	2BFGS			GQTPAR			LSTRS		
n	it	acc	cpu	it	acc	cpu	it	acc	cpu
100	0.0	1.1e−16	1.3e−03	9.0	1.7e−02	6.7e−02	15.9	1.0e−02	7.3e−02
200	0.0	1.2e−16	1.4e−03	6.1	2.3e−02	2.5e−01	15.9	7.6e−02	7.3e−02
300	0.0	1.3e−16	1.6e−03	7.3	2.4e−01	1.0e+00	17.3	4.5e−12	8.3e−02
400	0.0	1.4e−16	1.7e−03	7.4	7.0e−02	1.7e+00	16.7	2.2e−16	8.3e−02
500	0.0	1.6e−16	1.8e−03	6.6	5.3e−01	3.7e+00	18.5	3.0e−02	1.0e−01

Table 5. Comparison results for large scale TRS ($m = 1$), 100 instances per dimension.

		1BFGS			LSTRS		
	n	it	acc	cpu	it	acc	cpu
Standard case	10^3	1.0	1.2e−13	1.8e−03	2.0	1.6e−10	2.4e−02
	10^4	1.0	6.1e−16	2.0e−03	2.0	6.3e−12	3.2e−02
	10^5	1.0	3.1e−10	1.8e−02	2.0	1.1e−12	4.2e−01
Hard case	10^3	0.0	2.0e−16	8.9e−04	18.6	7.2e+00	1.0e−01
	10^4	0.0	5.7e−16	2.4e−03	20.5	3.9e−01	2.8e−01
	10^5	0.0	2.0e−15	1.6e−02	22.8	2.1e+00	5.2e+00

Table 6. Comparison results for large scale TRS ($m = 2$), 100 instances per dimension.

		2BFGS			LSTRS		
	n	it	acc	cpu	it	acc	cpu
Standard case	10^3	1.0	2.3e−16	5.4e−03	2.0	8.6e−11	1.7e−02
	10^4	0.9	1.6e−11	1.4e−02	2.0	7.0e−12	5.4e−02
	10^5	0.9	2.3e−10	2.0e−01	2.0	1.0e−11	9.6e−01
Hard case	10^3	0.0	2.4e−16	2.9e−03	19.9	3.2e+00	1.3e−01
	10^4	0.0	6.2e−16	1.1e−02	22.6	6.9e−01	6.0e−01
	10^5	0.0	1.9e−15	1.5e−01	24.7	4.8e−01	1.2e+01

Table 7. Comparison results for medium-size problems.

$n = 100$						$n = 500$					
$m = 1$			$m = 2$			$m = 1$			$m = 2$		
1BFGS	GQTPAR	LSTRS	2BFGS	GQTPAR	LSTRS	1BFGS	GQTPAR	LSTRS	2BFGS	GQTPAR	LSTRS
74	220	71	73	–	61	75	282	78	63	–	49
81	–	1868	56	–	826	78	–	–	45	–	831
72	–	–	63	–	–	58	–	–	62	–	–
50	–	42	19	–	41	52	–	114	23	1981	42
38	40	42	36	–	28	57	52	57	46	–	46
28	420	23	18	–	15	23	–	10	34	–	–
19	31	19	14	62	12	22	24	23	14	69	14
21	26	22	12	48	16	23	25	23	15	180	14
44	47	38	17	56	16	39	36	35	28	430	19
47	258	53	21	1193	25	26	66	24	19	1921	15
23	41	23	24	39	22	30	36	30	24	40	24
64	62	64	42	77	40	43	436	42	38	160	57
12	12	12	10	35	12	16	24	16	14	190	13
47	87	47	39	262	40	50	110	55	39	214	49
2	2	2	2	2	2	7	7	7	7	17	7
42	1457	89	12	1195	7	42	1435	52	10	1350	14
2	2	2	2	2	2	7	7	7	7	17	7
11	11	10	12	95	11	10	10	10	10	84	10
8	8	8	8	11	8	7	7	7	7	8	7
4	4	4	4	4	4	5	5	5	5	5	5
16	20	16	14	544	12	25	26	25	14	744	14
45	61	44	33	126	35	98	279	103	93	221	77
48	1659	295	29	–	79	52	–	–	30	–	435
280	–	301	98	27	114	199	–	263	100	30	96
8	9	8	9	234	9	11	11	11	10	342	10
9	9	9	10	401	9	11	11	11	12	1964	11
11	11	11	11	813	11	12	12	12	13	1999	12
28	38	28	24	39	27	43	59	42	27	193	28
58	264	49	31	1834	51	66	1243	93	30	–	54
21	14	20	27	120	16	13	17	13	20	66	13
10	12	10	10	15	10	14	16	14	12	29	10
19	25	19	24	270	18	23	26	23	24	951	19

The results in Tables 1 and 3 (standard case) show that GQTPAR algorithm outperforms 1BFGS, 2BFGS and LSTRS algorithms only in terms of average iterations. However, both 1BFGS and 2BFGS exhibit the best performance among the GQTPAR and LSTRS algorithms, with respect to the relative accuracy and the CPU time. On the other hand, when referring to the hard case (Tables 2 and 4), 1BFGS and 2BFGS algorithms significantly outperform the other two algorithms,

especially on the aspect of iterations and solution quality (order of 10^{-16}). Similar observations can be made from Tables 5 and 6, where the dimension of the TRS instances is very large.

The results of all tables show that the performance of our approach is promising. It can handle both the standard and the hard case relatively fast, providing solutions with high accuracy and small number of iterations. All of the TRS instances considered in our experiments were successfully solved by the `1BFGS` and `2BFGS` algorithm. In contrast, the other two algorithms in some instances did not achieve the prescribed tolerance, especially in the hard case. In general, one may observe that in the standard case all algorithms behaved similarly, while in the hard case the performance of our approach is very encouraging, showing a great deal of promise for its practical applications.

6.2 *Solving unconstrained optimization problems*

All algorithms were embedded in a trust-region framework for solving large scale unconstrained optimization problems, where the core problem is to solve the TRS (at least once) at each iteration of a trust region algorithm. We used the same main trust region algorithm, where the L-BFGS formula with $m = 1$ and 2 was used to update $B_k^{(m)}$. Since $B_k^{(m)}$ is allowed to be indefinite, the condition $s_k^{\mathrm{T}} y_k > 0$ does not always hold. Hence, for being $B_k^{(m)}$ well defined, we skipped the

Table 8. Comparison results for $n = 1000$.

$m = 1$				$m = 2$			
`1BFGS`		`LSTRS`		`2BFGS`		`LSTRS`	
it	cpu	it	cpu	it	cpu	it	cpu
72	8.7e−02	23	3.6e−01	82	2.9e−01	–	–
80	4.4e−02	–	–	64	1.2e−01	780	1.7e+01
93	9.7e−02	–	–	112	9.5e−01	–	–
48	3.4e−02	136	2.2e+00	27	4.6e−02	57	1.5e+00
58	2.4e−02	64	1.2e+00	48	5.6e−02	47	1.1e+00
37	4.2e−02	–	–	23	3.4e−02	–	–
15	1.2e−02	16	2.4e−01	12	1.7e−02	15	2.9e−01
18	6.5e−03	30	4.4e−01	16	1.4e−01	14	3.1e−01
31	1.4e−02	41	7.3e−01	24	1.6e−01	18	3.7e−01
43	3.2e−02	29	4.3e−01	23	3.5e−02	22	4.0e−01
31	2.5e−02	28	4.1e−01	25	3.7e−02	27	5.4e−01
62	8.4e−02	54	1.1e+00	41	7.6e−02	44	9.2e−01
21	8.4e−03	13	1.7e−01	13	1.3e−02	19	3.9e−01
39	1.5e−02	42	7.3e−01	36	4.1e−02	45	9.5e−01
8	4.3e−03	8	1.1e−01	8	1.4e−02	8	1.7e−01
51	2.9e−02	50	7.3e−01	7	5.7e−03	7	1.1e−01
8	7.9e−03	8	1.1e−01	8	1.4e−02	8	1.6e−01
12	7.3e−03	11	1.3e−01	14	1.3e−01	12	2.1e−01
10	6.9e−03	5	5.9e−02	10	1.4e−02	10	1.7e−01
5	3.4e−03	6	5.7e−02	5	5.0e−03	5	5.9e−02
18	1.0e−02	21	3.1e−01	12	1.7e−02	12	2.0e−01
71	4.7e−02	115	2.1e+00	41	4.5e−02	43	8.9e−01
58	9.4e−02	–	–	130	2.5e+00	823	4.3e+01
265	1.5e−02	253	4.6e+00	144	3.3e−01	136	3.1e+00
9	2.0e−02	11	1.6e−01	9	2.7e−02	9	1.7e−01
9	2.1e−02	11	1.7e−01	9	2.4e−02	9	1.6e−01
10	2.5e−02	13	2.0e−01	10	3.0e−02	10	1.7e−01
28	3.5e−02	37	6.1e−01	29	5.2e−02	27	5.4e−01
77	4.7e−02	67	1.4e+00	43	7.5e−02	54	1.2e+00
53	3.5e−02	16	2.3e−01	53	8.0e−02	29	7.3e−01
15	5.0e−03	15	2.2e−01	12	1.2e−02	14	2.3e−01
29	1.4e−02	27	4.2e−01	25	1.8e−01	23	4.8e−01

storage of the vector pair $\{s_k, y_k\}$ if $|s_k^{\mathrm{T}} y_k| \leq 10^{-12}\|s_k\|\|y_k\|$. The iterations were terminated when $\|g_k\| \leq \text{tol}$, where $\text{tol} = 10^{-5}$. For the solution of the TRS, all the previous mentioned algorithms were used for comparison reasons. For the `1BFGS` and `2BFGS` algorithms the termination criterion for Newton's iterations in the TRS was $|(\|d\| - \Delta)/\Delta| \leq 10^{-5}$ or $|\psi(\lambda)| \leq 10^{-5}$. For the `GQTPAR` algorithm the tolerances were set $\text{rtol} = 10^{-5}$ and $\text{atol} = 10^{-20}$, while for the `LSTRS` algorithm we set epsilon. Delta = epsilon.HC $= 10^{-5}$.

We selected 32 large-scale unconstrained optimization test problems in extended or generalized form [2,18]: Extended Trigonometric, Rosenbrock, White & Holst, Beale, Penalty, Tridiagonal 1, Three Expo Terms, Himmelblau, Block-Diagonal BD1, Block-Diagonal BD2. Generalized Tridiagonal 1, Tridiagonal 2 Quartic GQ1, Quartic GQ2. Raydan 2, Diagonal 4, Diagonal 6, Diagonal 7, Diagonal 8, Full Hessian, Sincos, Broyden tridiagonal, Nondia, Dqdrtic, Dixmaana, Dixmaanb, Dixmaanc, Edensch, Liarwhd, Cosine, Extended Denschnb, Extended Denschnf.

For each test function, five numerical experiments were conducted with number of variables $n = 100, 500$ (medium-size problems), 10^3, 10^4 and 10^5 (larger-size problems), respectively. The initial trust region radius Δ_0 was set $\Delta_0 = 10$, for $n = 100$ and 500, and $\Delta_0 = 100$ for the larger-size problems. For $n = 10^3$, 10^4 and 10^5, we compared our method only with `LSTRS`, due to the

Table 9. Comparison results for $n = 10{,}000$.

$m = 1$				$m = 2$			
`1BFGS`		`LSTRS`		`2BFGS`		`LSTRS`	
it	cpu	it	cpu	it	cpu	it	cpu
87	5.6e−01	83	3.9e+00	61	5.4e−01	79	7.3e+00
86	1.8e−01	–	–	53	6.4e−01	845	9.0e+01
86	5.4e−01	–	–	82	5.7e+00	–	–
62	2.4e−01	142	8.0e+00	19	1.7e−01	72	7.6e+00
65	1.2e−01	78	3.7e+00	69	3.6e−01	–	–
28	2.0e−01	–	–	18	1.6e−01	–	–
21	9.8e−02	18	8.5e−01	17	1.2e−01	14	1.2e+00
23	3.9e−02	27	1.1e+00	13	5.3e−02	18	1.5e+00
48	1.0e−01	35	1.6e+00	20	9.5e−02	20	1.5e+00
49	2.5e−01	33	1.4e+00	20	2.1e−01	47	3.7e+00
41	1.7e−01	33	1.6e+00	27	1.8e−01	22	2.1e+00
92	8.8e−01	104	6.1e+00	85	1.3e+00	44	4.7e+00
12	2.7e−02	18	7.5e−01	11	4.9e−02	12	7.6e−01
45	6.4e−02	43	1.9e+00	34	1.4e−01	36	3.6e+00
2	4.6e−03	9	3.4e−01	2	5.0e−03	2	4.4e−02
54	7.7e−02	67	3.2e+00	12	4.3e−02	7	4.9e−01
2	4.6e−03	9	3.4e−01	2	5.0e−03	2	4.6e−02
11	3.9e−02	11	3.3e−01	9	9.0e−01	11	1.1e+00
8	3.6e−02	7	2.2e−01	8	5.4e−02	8	5.0e−01
4	2.7e−02	7	2.0e−01	4	2.0e−02	4	1.5e−01
18	5.9e−02	19	8.3e−01	14	7.4e−02	13	1.1e+00
146	3.9e−01	55	2.9e+00	84	9.1e−01	39	4.1e+00
46	8.6e−02	–	–	30	2.7e+00	–	–
325	5.9e−01	247	1.2e+01	76	8.0e−01	74	7.9e+00
9	1.4e−01	13	7.5e−01	9	1.6e−01	9	7.3e−01
10	1.6e−01	18	1.1e+00	10	1.8e−01	10	8.4e−01
12	1.9e−01	21	1.4e+00	12	2.1e−01	11	9.4e−01
41	3.4e−01	30	1.9e+00	28	3.0e−01	24	2.1e+00
82	2.0e−01	128	9.8e+00	55	4.7e+00	111	2.1e+01
17	6.5e−02	54	2.4e+00	339	1.2e+01	17	1.3e+00
11	2.1e−02	17	7.4e−01	11	4.6e−02	10	8.0e−01
20	4.9e−02	22	9.7e−01	26	1.2e−01	20	1.6e+00

computational cost of the trust-region algorithm that uses the GQTPAR algorithm for the solution of the TRS.

The numerical results are summarized in Tables 7–10. The reported parameters are: the number of iterations (it) of the trust-region algorithm, which equals the number of function and gradient evaluations per iteration, and the CPU time (cpu) in seconds. In all tables, the symbol '–' is used to indicate that a problem failed to converge with the prescribed accuracy, i.e. $\|g_k\| \leq 10^{-5}$, within a maximum number of iterations (maxiter = 2000). Table 7 summarizes the results for $n = 100{,}500$. In this table we report, for each method, only the number of iterations. Tables 8–10 summarize the results for the rest of the dimensions. Figure 1 illustrates the results of Table 7, and shows the mean iterations of each method. The numerical results of Tables 8–10 can be summarized in Figures 2 and 3, which show the mean iterations and sum CPU time, respectively, for the 1BFGS, 2BFGS and LSTRS algorithms. For calculating both the average iterations and sum CPU time, we considered in all algorithms only the problems that solved with the prescribed accuracy within the iteration limit.

From Table 7 and Figure 1, one can notice that the proposed algorithm solved 100% of the medium-size test problems. Moreover, it has the best performance among the other two

Table 10. Comparison results for $n = 100{,}000$.

$m = 1$				$m = 2$			
1BFGS		LSTRS		2BFGS		LSTRS	
it	cpu	it	cpu	it	cpu	it	cpu
39	2.8e+00	78	5.3e+01	77	7.6e+00	39	5.9e+01
102	3.9e+00	–	–	59	3.6e+01	734	1.1e+03
96	7.7e+00	–	–	68	1.0e+02	–	–
58	2.9e+00	65	5.9e+01	21	1.7e+00	58	8.2e+01
94	1.9e+00	–	–	77	4.7e+00	79	1.3e+02
28	2.2e+00	–	–	40	6.7e+01	–	–
25	1.2e+00	46	5.2e+01	73	3.4e+02	16	2.1e+01
36	6.9e−01	32	2.2e+01	20	1.4e+00	17	2.0e+01
40	1.1e+00	42	2.9e+01	20	1.7e+01	20	2.6e+01
39	2.2e+00	46	3.5e+01	25	2.2e+00	22	2.9e+01
32	1.6e+00	32	2.5e+01	26	2.1e+00	23	2.8e+01
101	1.2e+01	154	1.8e+02	75	1.6e+01	96	1.6e+02
14	3.0e−01	20	1.2e+01	15	7.4e−01	13	1.4e+01
49	9.6e−01	51	4.0e+01	38	2.0e+00	52	6.1e+01
8	2.1e−01	70	2.2e+02	8	3.9e−01	8	7.2e+00
61	1.6e+00	56	4.5e+01	10	5.5e−01	8	7.5e+00
8	2.1e−01	11	5.6e+00	8	3.9e−01	8	7.2e+00
12	4.2e−01	13	6.6e+00	10	1.6e+01	12	9.1e+00
6	2.5e−01	9	4.0e+00	6	3.5e−01	6	4.4e+00
5	2.2e−01	8	3.4e+00	5	2.7e−01	5	3.1e+00
27	9.1e−01	–	–	81	4.2e+02	16	1.9e+01
97	3.5e+00	44	2.9e+01	83	1.5e+01	83	1.2e+02
264	8.5e+01	–	–	33	8.4e+01	–	–
247	8.9e+00	294	2.5e+02	127	3.5e+01	81	1.3e+02
12	1.9e+00	14	1.2e+01	11	2.1e+00	12	1.5e+01
12	2.0e+00	42	3.2e+01	12	2.3e+00	12	1.4e+01
13	2.1e+00	46	3.2e+01	14	2.6e+00	14	1.7e+01
38	3.5e+00	36	2.8e+01	30	3.7e+00	48	4.6e+01
85	3.3e+00	160	2.6e+02	90	1.8e+02	114	4.3e+02
25	1.1e+00	33	2.9e+01	–	–	–	–
16	3.0e−01	22	1.6e+01	12	5.4e−01	12	1.2e+01
33	8.5e−01	22	1.4e+01	19	1.0e+00	22	2.4e+01

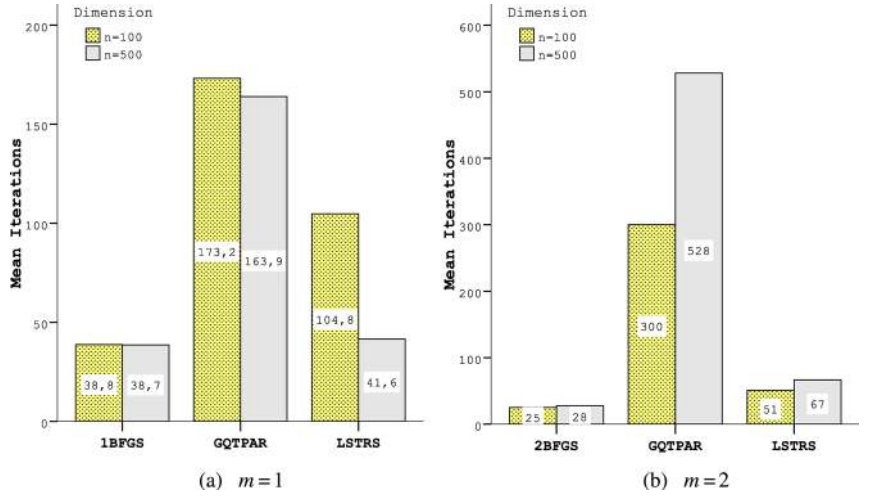

Figure 1. Average iterations for medium-size problems.

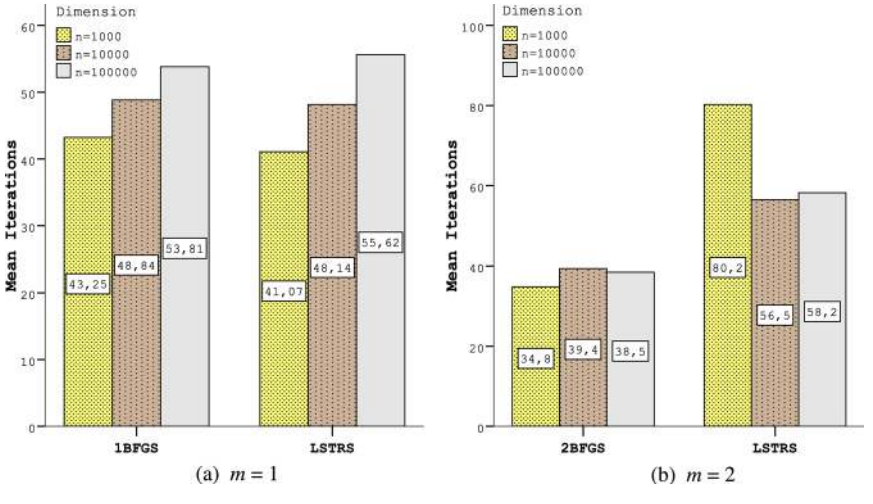

Figure 2. Average iterations for larger-size problems.

approaches, since it exhibits the lowest average by means of iterations, for both $m = 1$ and 2. The results in Tables 8–10 along with Figure 2 show that for $m = 1$, the trust-region algorithms that use `1BFGS` and `LSTRS` algorithms for solving the TRS, behaved similarly in terms of iterations. However, `1BFGS` solved all test problems in all three dimensions, while `LSTRS` presented some failures. For $m = 2$, the `2BFGS` algorithm has the lowest mean iteration, although it presented one failure. Finally, according to Figure 3, we can see that our approach needs the smallest amount of CPU time for solving large-size problems. Based on the above comparisons, we can observe that our method presented less failures, number of iterations and CPU time for solving the test problems, in all dimensions, than the other approaches. As a consequence, we can conclude that the proposed method is promising for solving large-scale unconstrained problems, within a trust-region framework.

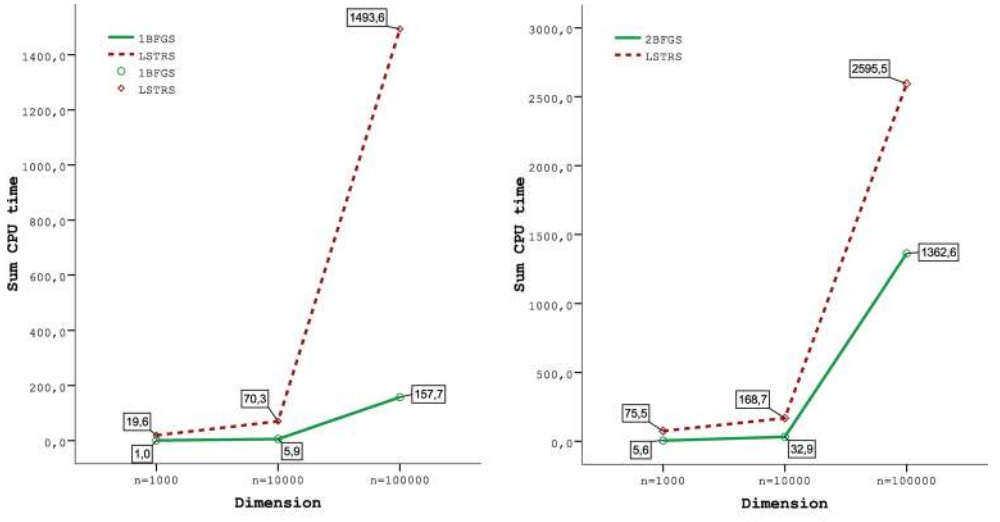

Figure 3. Total CPU time for larger-size problems.

7. Conclusions

We have shown how the `1BFGS` and `2BFGS` methods can be applied for solving the TRS. Our results have been based on the properties of the L-BFGS matrices for $m = 1$ or 2. The eigenvalues can immediately be computed with high accuracy, while the inverse of $B^{(m)} + \lambda I$ can be expressed in a closed form. Hence, the linear system which has to be solved for the computation of the trial step does not require the use of factorization. Finally, the negative curvature direction can be computed either by applying one step of the power inverse method or by making some simple algebraic computations. Thus, the proposed method can completely avoid the Cholesky factorization and handle easily both the standard and the hard case. It turns out that the main advantages of the proposed method are higher solution accuracy compared to other approaches, small running time, and the efficiency for solving large instances of the TRS since the amount of memory needed is negligible.

Acknowledgements

We would like to thank an anonymous referee for helpful suggestions and useful comments that significantly improved the presentation of this work.

References

[1] E. Anderson et al., *LAPACK User's Guide*. SIAM, Philadelphia, PA, USA, 1994.
[2] N. Andrei, *Unconstrained optimization test functions: algebraic expression*. Available at: http://www.ici.ro/camo/neculai/SCALCG/testuo.pdf.
[3] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, 1996.
[4] J. Barzilai and J. Borwein, *Two point step size gradient method*, IMA J. Numer. Anal. 8 (1988), pp. 141–148.
[5] E. Birgin and J. Martínez. *A spectral conjugate gradient method for unconstrained optimization*, Appl. Math. Optim. 43 (2001), pp. 117–128.
[6] E. Birgin, J. Martínez, and M. Raydan. *Nonmonotone spectral projected gradient methods on convex sets*, SIAM J. Optim. 10 (2000), pp. 1196–1211.
[7] P. Borwein and T. Erdélyi, *Polynomials and Polynomial Inequalities, Graduate Texts in Mathematics*, Vol. 161, Springer-Verlag, New York, 1995.

[8] R. Byrd and J. Nocedal, *A tool for the analysis of quasi-Newton methods with application to unconstrained optimization*, SIAM J. Numer. Anal. 26 (1989), pp. 727–739.
[9] A. Conn, N. Gould, and P. Toint. *Trust-Region Methods*, MPS/SIAM Series on Optimization. SIAM, Philadelphia, PA, USA, 2000.
[10] W. Faucette, *A geometric interpretation of the solution of the general quartic polynomial*, Amer. Math. Monthly 103 (1996), pp. 51–57.
[11] D. Gay, *Computing optimal locally constrained steps*, SIAM J. Sci. Stat. Comput. 2 (1981), pp. 186–197.
[12] N. Gould, S. Lucidi, M. Roma, and P. Toint, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim. 9 (1999), pp. 504–525.
[13] W. Hager, *Minimizing a quadratic over a sphere*, SIAM J. Optim. 12 (2001), pp. 188–208.
[14] W. Hager and Y. Krylyuk, *Graph partitioning and continuous quadratic programming*, SIAM J. Discrete Math. 12 (1999), pp. 500–523.
[15] I. Ipsen, *Computing an eigenvector with inverse iteration*, SIAM Rev. 39 (1997), pp. 254–291.
[16] D. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Program. 45 (1989), pp. 503–528.
[17] W. Menke, *Geophysical Data Analysis: Discrete Inverse Theory*, Academic Press, San Diego, 1989.
[18] J. Moré, B. Garbow, and K. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Softw. 7(1981), pp. 17–41.
[19] J. Moré and D. Sorensen, *Computing a trust region step*, SIAM J. Sci. Stat. Comput. 4 (1983), pp. 553–572.
[20] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Math. Comput. 35 (1980), pp. 773–782.
[21] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, New York, 1999.
[22] J. Nocedal and Y. Yuan, *Combining trust region and line search techniques*, in *Advances in Nonlinear Programming*, Y. Yuan, ed., Kluwer, Dordrecht, The Netherlands, 1998, pp. 153–175.
[23] A. Ouorou, *Implementing a proximal algorithm for some nonlinear multicommodity flow problems*, Networks 49 (2007), pp. 18–27.
[24] ———, *A proximal subgradient projection algorithm for linearly constrained strictly convex problems*, Optim. Methods Softw. 22 (2007), pp. 617–636.
[25] M. Powell, *Convergence properties of a class of minimization algorithms*, in *Nonlinear Programming 2*, O. Mangasarian, R. Meyer, and S. Robinson, eds., Academic Press, New York, 1975. pp. 1–27.
[26] M. Raydan, *On the Barzilai and Borwein choice of steplength for the gradient-method*, IMA J. Numer. Anal. 13 (1993), pp. 321–326.
[27] ———, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim. 7 (1997), pp. 26–33.
[28] F. Rendl and H. Wolkowicz, *A semidefinite framework for trust region subproblems with applications to large scale minimization*, Math. Program. 77 (1997), pp. 273–299.
[29] M. Rojas, S. Santos, and D. Sorensen, *A new matrix-free algorithm for the large-scale trust-region subproblem*, SIAM J. Optim. 11 (2000), pp. 611–646.
[30] ———, *Algorithm 873: LSTRS: MATLAB software for large-scale trust-region subproblems and regularization*, ACM Trans. Math. Softw. 34 (2008), pp. 1–28.
[31] D. Sorensen, *Newton's method with a model trust region modification*, SIAM J. Numer. Anal. 19 (1982), pp. 409–426.
[32] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal. 20 (1983), pp. 626–637.
[33] J. Wilkinson. *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.