Applied Mathematics and Computation 217 (2010) 882–892

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A curvilinear method based on minimal-memory BFGS updates

M.S. Apostolopoulou [a], D.G. Sotiropoulos [a,*], C.A. Botsaris [b]

[a] University of Patras, Department of Mathematics, GR-265 04 Patras, Greece
[b] University of Central Greece, Department of Regional Economic Development, GR-321 00 Levadia, Greece

ARTICLE INFO

Keywords:
Large scale unconstrained optimization
Curvilinear search
Negative curvature direction
Eigenvalues
L-BFGS method

ABSTRACT

We present a new matrix-free method for the computation of negative curvature directions based on the eigenstructure of minimal-memory BFGS matrices. We determine via simple formulas the eigenvalues of these matrices and we compute the desirable eigenvectors by explicit forms. Consequently, a negative curvature direction is computed in such a way that avoids the storage and the factorization of any matrix. We propose a modification of the L-BFGS method in which no information is kept from old iterations, so that memory requirements are minimal. The proposed algorithm incorporates a curvilinear path and a linesearch procedure, which combines two search directions; a memoryless quasi-Newton direction and a direction of negative curvature. Results of numerical experiments for large scale problems are also presented.

1. Introduction

We consider the following large scale unconstrained optimization problem

$$\min\{f(x)|x \in \mathbb{R}^n\}, \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. Many algorithms have been proposed in the literature to solve such problems [16], while some of them attempt to use second-order information available in the Hessian matrix of f. This information is introduced through the computation of a negative curvature direction. At a point $x \in \mathbb{R}^n$, the vector $d \in \mathbb{R}^n$ is a direction of negative curvature of the Hessian matrix at x if $d^T\nabla^2 f(x)d < 0$. This direction is associated with the normalized eigenvector corresponding to the most negative eigenvalue of the Hessian. In the negative curvature directions, the quadratic model of the objective function is unbounded below, offering a potential for a large reduction in the value of the objective function.

One of the first proposals of using negative curvature directions is that of Fiacco and McCormick [6]. In 1977, McCormick [13] showed how a modification of the Armijo's rule [1] could be used to include cases where second derivative information is used, in particular when the Hessian matrix is not positive semi-definite. Later, Moré and Sorensen in 1979 [14], and Goldfarb in 1980 [8], proposed a similar approach along a curve of the form

$$C = \{x(\alpha) : x(\alpha) = x + \phi_1(\alpha)p + \phi_2(\alpha)d,\ \alpha \geqslant 0\}, \tag{1.2}$$

with $\phi_1(0) = \phi_2(0) = 0$, combining two search directions; the direction p, which is a Newton-type direction, and the direction d, the negative curvature direction. Moreover, in [14] an Armijo-type rule was used, to ensure convergence to a second-order critical point, where the Hessian matrix is positive semi-definite. In [9] the alternative use of a negative curvature direction and a Newton-type direction was proposed, within an appropriate linesearch procedure. In the work of Olivares et al. [17]

* Corresponding author.
E-mail addresses: msa@math.upatras.gr (M.S. Apostolopoulou), dgs@math.upatras.gr (D.G. Sotiropoulos), botsaris@otenet.gr (C.A. Botsaris).

0096-3003/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2010.06.032

the authors established criteria such that at each iteration, either a linesearch procedure using one direction (a Newton-type or a negative curvature direction) or a curvilinear search combining both directions, is performed. Curvilinear paths and negative curvature directions are also used in constrained optimization [16,18,19].

The computation of negative curvature direction is equivalent with the computation of the eigenvector corresponding to the most negative eigenvalue of the Hessian. In [14] the Bunch and Parlett factorization was used, while in [5] this approach was embedded in a nonmonotone framework. In both cases, the computation of the negative curvature direction requires the factorization and the storage of a matrix, which is computationally expensive when the number of variables is large. In [11] the computation of the negative curvature direction was based on a Lanczos procedure. However, the storage of a matrix is required and only a few Lanczos vectors are stored. Other works also use directions of negative curvature produced by preconditioned conjugate gradients and Lanczos methods [9,12]. In these methods it is necessary to repeat the recurrence in order to regenerate the Lanczos vectors whenever they are needed. More recently, iterative methods have been proposed for computing negative curvature directions using planar conjugate gradient algorithms [4].

Our interest is to avoid the factorization or the storage of a matrix, and to compute a suitable negative curvature direction without using any iterative procedure, for large scale unconstrained optimization. To this end, we utilize the L-BFGS method [15] for the computation of an approximate Hessian matrix, using information from the most previous iteration. This specific memoryless BFGS update is providing us the ability to determine analytically the eigenvalues of the approximate Hessian matrix B. Moreover, we are able to compute the eigenvector corresponding to the most negative eigenvalue of B explicitly. Thus, the computation of the pair of search directions is obtained by performing a sequence of inner products and vector summations.

The negative curvature direction produced by the minimal-memory BFGS matrices is used within a modified linesearch procedure. We consider an algorithm that exploits the information contained in the eigenstructure of the approximate Hessian. This information is taken into account by performing either a standard linesearch procedure (when B is positive definite), using only the Newton-type direction, or the curvilinear search proposed by Moré and Sorensen [14], combining both directions (when B is indefinite).

The structure of the paper is as follows: in Section 2 we introduce some basic definitions and in Section 3 we study the properties of minimal-memory BFGS matrices. In Section 4 we give in details the method to compute the descent directions, while in Section 5 we describe the proposed algorithm. The results from the computational experiments are illustrated in Section 6. Finally, in Section 7 we give some concluding remarks.

Notation. Throughout the paper $\|\cdot\|$ denotes the Euclidean norm, n the dimension of the problem and I the $n \times n$ identity matrix. The gradient of f is denoted by g and the approximate Hessian by B. For a symmetric $A \in \mathbb{R}^{n\times n}$, assume that $\lambda_1 \leqslant \cdots \leqslant \lambda_n$ are its eigenvalues sorted into non-decreasing order, and u_i, $i = 1,\ldots,n$ the corresponding eigenvectors. A matrix A is indicated that is positive definite by $A > 0$ (semi-definite by $A \geqslant 0$).

2. Preliminaries

For the remainder of the paper we make the following assumptions.

Assumption 2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on an open set $\mathscr{B}$ and assume that for some $x_0 \in \mathscr{B}$, the level set $\Omega = \{x \in \mathbb{R}^n | f(x) \leqslant f(x_0)\}$ is a compact subset of $\mathscr{B}$.*

Assumption 2.2. *Let B_k be the matrices produced by the memoryless BFGS update. There exists a positive constant M such that $\|B_k\| \leqslant M$, for all k.*

Under the Assumption 2.1, we can construct a local quadratic model for the objective function f from the corresponding Taylor series expansion for all iterates $x_k \in \Omega$ as

$$q_k(p) \simeq f_k + g_k^T p + \frac{1}{2} p^T B_k p,$$

where B_k is a positive definite matrix. Hence, the quasi-Newton direction p_k can be computed by the formula

$$p_k = \begin{cases} -B_k^{-1} g_k, & \text{if } B_k > 0; \\ -g_k, & \text{otherwise}, \end{cases}$$

while the negative curvature direction d_k is related with the unit eigenvector u_k that corresponds to the most negative eigenvalue of B_k and is obtained by the form

$$d_k = \begin{cases} 0, & B_k > 0; \\ -\mathrm{sgn}(u_k^T g_k) u_k, & \text{otherwise}. \end{cases}$$

Both directions p_k and d_k must be *descent* directions.

Definition 2.3. A pair of directions (p,d) is called a *descent pair* at a point x where $f(x)$ is twice differentiable, if

$$p^T\nabla f(x) < 0, \quad d^T\nabla f(x) \leqslant 0, \quad \text{and} \quad d^T\nabla^2 f(x)d = 0,$$

when $\nabla^2 f(x) \geqslant 0$, and

$$p^T\nabla f(x) \leqslant 0, \quad d^T\nabla f(x) \leqslant 0, \quad \text{and} \quad {}^T\nabla^2 f(x)d < 0,$$

otherwise.

From a more practical point of view, in a given iterate x_k, the pair (p_k, d_k) is assumed to be a sufficient descent pair of directions if the directions $\{p_k\}$ and $\{d_k\}$ are bounded and satisfy the following conditions [8]:

Condition 2.4. $g_k^T p_k = 0$ implies $g_k = 0$ and $p_k = 0$.

Condition 2.5. $g_k^T p_k \to 0$ implies $g_k \to 0$ and $p_k \to 0$.

Condition 2.6. $d_k^T B_k d_k \to 0$ implies $\min(\lambda_1, 0) \to 0$ and $d_k \to 0$, where λ_1 is the smallest eigenvalue of B_k.

Conditions 2.4 and 2.5 are the standard ones for Newton-type directions. They ensure that the $p_k \neq 0$ and $g_k \neq 0$ are not orthogonal and do not become nearly so too rapidly. Condition 2.6 ensures that d_k contains information related to the smallest eigenvalue of the approximate Hessian.

3. Properties of the approximate Hessian

As we mentioned before, we use the minimal-memory BFGS update for computing an approximate Hessian of f. The minimal-memory BFGS matrix is computed using the L-BFGS philosophy [15]. Given an initial matrix $B_k^{(0)}$, the L-BFGS method updates B_k by means of the BFGS formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \tag{3.1}$$

using information from the most recent iterations, which is included in a certain number of stored vector pairs $\{s_i, y_i\}$, where $s_i = x_{i+1} - x_i$, and $y_i = g_{i+1} - g_i$. The minimal-memory BFGS update uses information only from the previous iteration, that is, one update is performed to $B_k^{(0)}$, using the current vector pair $\{s_k, y_k\}$. In our study, we assume that the initial matrix $B^{(0)}$ can be any multiple of the identity matrix, that is, $B_{k+1}^{(0)} = \theta_{k+1} I$, where $\theta_{k+1} \in \mathbb{R} \setminus \{0\}$. The resulting minimal-memory BFGS update scheme takes the form

$$B_{k+1} = \theta_{k+1} I - \theta_{k+1} \frac{s_k s_k^T}{s_k^T s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \tag{3.2}$$

while the inverse of B_{k+1} is given by the following expression

$$B_{k+1}^{-1} = \frac{1}{\theta_{k+1}} I - \frac{1}{\theta_{k+1}} \frac{y_k s_k^T + s_k y_k^T}{s_k^T y_k} + \left[1 + \frac{y_k^T y_k}{\theta_{k+1} s_k^T y_k}\right] \frac{s_k s_k^T}{s_k^T y_k}. \tag{3.3}$$

Taking into account that the determinant and the trace of BFGS matrices are given by the formulas (cf. [16])

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

easily can be verified that the above relations become

$$\det(B_{k+1}) = \theta_{k+1}^{n-1} \frac{s_k^T y_k}{s_k^T s_k}, \quad \text{and} \quad \operatorname{tr}(B_{k+1}) = (n-1)\theta_{k+1} + \frac{y_k^T y_k}{s_k^T y_k}, \tag{3.4}$$

respectively, when B_{k+1} is defined by relation (3.2).

Regarding to θ_{k+1}, a choice that has proved effective in practice is to set $\theta_{k+1} = y_k^T y_k / s_k^T y_k$ [16, pp. 200], or alternatively one can set θ_{k+1} to be the Barzilai and Borwein spectral parameter $s_k^T y_k / s_k^T s_k$ [2].

Theorem 3.1. *Let the symmetric minimal-memory BFGS matrix defined in (3.2). Then, the characteristic polynomial of $B_{k+1} \in \mathbb{R}^{n\times n}$ has the general form*

$$p(\lambda) = (\lambda - \theta_{k+1})^{n-2} \left[\lambda^2 - \left(\theta_{k+1} + \frac{y_k^T y_k}{s_k^T y_k}\right)\lambda + \theta_{k+1} \frac{s_k^T y_k}{s_k^T s_k}\right]. \tag{3.5}$$

Moreover, if the vectors s_k and y_k are linearly independent then the smallest eigenvalue of B_{k+1} is distinct.

Proof. The initial matrix $B_{k+1}^{(0)} = \theta_{k+1} I$ is updated by the addition of two rank one matrices. Hence, B_{k+1} has at least $(n-2)$ eigenvalues equal θ_{k+1}, and at most two distinct eigenvalues, denoting them by ξ_i and ξ_j, respectively. Then, the characteristic polynomial of B_{k+1} can be expressed as follows:

$$p(\lambda) = (\lambda - \theta_{k+1})^{n-2}[\lambda^2 - (\xi_i + \xi_j)\lambda + \xi_i \xi_j]. \tag{3.6}$$

Taking into account that

$$\mathrm{tr}(B_{k+1}) = (n-2)\theta_{k+1} + \xi_i + \xi_j \quad \text{and} \quad \det(B_{k+1}) = \theta_{k+1}^{n-2} \xi_i \xi_j, \tag{3.7}$$

combining relations (3.4) and (3.7) we obtain

$$\xi_i + \xi_j = \theta_{k+1} + (y_k^T y_k)/(s_k^T y_k), \quad \text{and} \quad \xi_i \xi_j = \theta_{k+1}(s_k^T y_k)/(s_k^T s_k).$$

By substituting the above relations in (3.6), we obtain the general form of the characteristic polynomial (3.5).

It remains to show that when s_k and y_k are linearly independent, the smallest eigenvalue is distinct. First we consider the case where $B_{k+1} > 0$, then the matrix $\overline{B} = \theta_{k+1} I - \theta_{k+1} s_k s_k^T / s_k^T s_k$ besides the zero eigenvalue, has one more eigenvalue equals to θ_{k+1} of multiplicity $(n-1)$. By adding the term $y_k y_k^T / s_k^T y_k$ on $\overline{B}$ and applying the interlacing theorem [20, pp. 94–98], we yield that

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

which imply that $\lambda_{n-1} = \cdots = \lambda_2 = \theta_{k+1}$, and consequently,

$$\lambda_n \geqslant \theta_{k+1} \geqslant \lambda_1. \tag{3.8}$$

Suppose now that B_{k+1} is indefinite. It is clear, using the same arguments as before, that if $\theta_{k+1} > 0$, we yield

$$\theta_{k+1} \geqslant \lambda_2 \geqslant \lambda_1, \tag{3.9}$$

otherwise, if $\theta_{k+1} < 0$, we yield relation (3.8). Clearly, from relations (3.9) and (3.8), we can conclude that if the matrix has two distinct eigenvalues, then λ_1 is one of them. Suppose now that the vectors s_k and y_k are linearly independent and let ϕ denotes the angle between them. We assume at the moment that the smallest eigenvalue λ_1 is multiple. Then, from relations (3.8) and (3.9) we have that $\lambda_1 = \theta_{k+1}$, which implies that either only one eigenvalue is distinct or no eigenvalue is distinct. Combining relations (3.4) and (3.7), in the first case (only one distinct), we obtain

$$\mathrm{tr}(B_{k+1}) = (n-1)\theta_{k+1} + \lambda_n = (n-1)\theta_{k+1} + \frac{y_k^T y_k}{s_k^T y_k} \quad \text{and} \quad \det(B_{k+1}) = \theta_{k+1}^{n-1} \lambda_n = \theta_{k+1}^{n-1} \frac{s_k^T y_k}{s_k^T s_k},$$

while in the second case (no distinct eigenvalue) we have that

$$\mathrm{tr}(B_{k+1}) = n\theta_{k+1} = (n-1)\theta_{k+1} + \frac{y_k^T y_k}{s_k^T y_k} \quad \text{and} \quad \det(B_{k+1}) = \theta_{k+1}^{n} = \theta_{k+1}^{n-1} \frac{s_k^T y_k}{s_k^T s_k}.$$

Therefore, we yield that $(s_k^T y_k)^2 = s_k^T s_k y_k^T y_k$ or $\|s_k\|^2 \|y_k\|^2 \cos^2\phi = \|s_k\|^2 \|y_k\|^2$, which implies that the vectors s_k and y_k are collinear. This contradicts to our hypothesis, and hence, if the vectors s_k and y_k are linearly independent then λ_1 is distinct. □

If the vectors s_k and y_k are collinear, i.e., $y_k = \kappa s_k, \kappa \in \mathbb{R}$, then the characteristic polynomial of B_{k+1} takes the form $p(\lambda) = (\lambda - \theta_{k+1})^{n-2}[\lambda^2 - (\theta_{k+1} + \kappa)\lambda + \theta_{k+1}\ \kappa] = (\lambda - \theta_{k+1})^{n-1}(\lambda - \kappa)$. Clearly, the eigenvalue equals θ_{k+1} has multiplicity $n-1$, while the eigenvalue equals κ is distinct. Furthermore, if the choice of θ_{k+1} is the BB spectral parameter $s_k^T y_k / s_k^T s_k$, then B_{k+1} has one multiple eigenvalue equals θ_{k+1} of multiplicity n.

For analyzing the eigenvectors corresponding to distinct eigenvalues, we take into account that the initial matrix $B_{k+1}^{(0)}$ is updated using the vectors s_k and y_k.

Lemma 3.2. *Suppose that one update is performed to the symmetric matrix $B_{k+1}^{(0)} = \theta_{k+1} I$, using the vector pair $\{s_k, y_k\}$ by applying the BFGS formula. Then, the eigenvectors corresponding to distinct eigenvalues of the updated matrix, are of the form*

$$u_{k+1}(\lambda) = \frac{(\lambda s_k - y_k)^T y_k}{(\lambda s_k - y_k)^T s_k} s_k - y_k. \tag{3.10}$$

Proof. Since $B_{k+1}^{(0)}$ is updated using the vector pair $\{s_k, y_k\}$, the corresponding eigenvectors must be of the form $u = c_1 s_k + c_2 y_k$, where $c_1, c_2 \in \mathbb{R}$. By substituting u into the eigenvalue equation $B_{k+1} u = \lambda u$ and taking into account the secant equation $B_{k+1} s_k = y_k$, we obtain the equation: $c_1 y_k + c_2 B_{k+1} y_k = \lambda c_1 s_k + \lambda c_2 y_k$. Multiplying both sides with s_k^T, we obtain $c_1(s_k^T y_k - \lambda s_k^T s_k) = c_2(\lambda s_k^T y_k - y_k^T y_k)$, and under the hypothesis that λ is a distinct eigenvalue, equivalently, we have that $c_1 = -c_2(\lambda s_k^T y_k - y_k^T y_k)/(\lambda s_k^T s_k - s_k^T y_k)$. By setting $c_2 = -1$, we obtain relation (3.10). □

4. Computation of the descent pair of directions

The descent pair of directions consists of the quasi-Newton direction p defined as

$$p_{k+1} = \begin{cases} -B_{k+1}^{-1} g_{k+1}, & B_{k+1} > 0; \\ -g_{k+1}, & \text{otherwise}. \end{cases} \tag{4.1}$$

and the direction of negative curvature d defined as

$$d_{k+1} = \begin{cases} 0, & B_{k+1} > 0; \\ -\text{sgn}(u_{k+1}^T g_{k+1}) u_{k+1}, & \text{otherwise}, \end{cases} \tag{4.2}$$

where u_{k+1} is a normalized vector that corresponds to the most negative eigenvalue of B_{k+1}. In the sequel we compute the search directions when the approximate Hessian is positive definite or indefinite.

4.1. The positive definite case

When B_{k+1} is positive definite, the quasi-Newton direction is obtained using the inverse of the minimal-memory BFGS matrix, defined in Eq. (3.3). Therefore, $p_{k+1} = -H_{k+1} g_{k+1}$, or, equivalently

$$p_{k+1} = -\frac{g_{k+1}}{\theta_{k+1}} - \left[\left(1 + \frac{y_k^T y_k}{\theta_{k+1} s_k^T y_k} \right) \frac{s_k^T g_{k+1}}{s_k^T y_k} - \frac{y_k^T g_{k+1}}{\theta_{k+1} s_k^T y_k} \right] s_k + \frac{s_k^T g_{k+1}}{\theta_{k+1} s_k^T y_k} y_k. \tag{4.3}$$

In case where the vectors s_k and y_k are collinear, i.e., $y_k = \kappa s_k$, then B_{k+1} is of the form

$$B_{k+1} = \theta_{k+1} I - (\theta_{k+1} - \kappa) \frac{s_k s_k^T}{s_k^T s_k}, \tag{4.4}$$

while its inverse has the following form:

$$H_{k+1} = \frac{1}{\theta_{k+1}} I - \left(\frac{1}{\theta_{k+1}} - \frac{1}{\kappa} \right) \frac{s_k s_k^T}{s_k^T s_k}. \tag{4.5}$$

Thus, the quasi-Newton direction is computed by the formula

$$p_{k+1} = -\frac{g_{k+1}}{\theta_{k+1}} + \left(\frac{1}{\theta_{k+1}} - \frac{1}{\kappa} \right) \frac{s_k^T g_{k+1}}{s_k^T s_k} s_k. \tag{4.6}$$

4.2. The indefinite case

When B_{k+1} is indefinite, we consider the following cases: (i) If the smallest eigenvalue of B_{k+1} is distinct, then from Lemma 3.2 we know that the corresponding eigenvector is the normalized vector as given in Eq. (3.10). (ii) If the smallest eigenvalue λ_1 of B_{k+1} is multiple, then $y_k = \kappa s_k$ and $\theta_{k+1} \leqslant \kappa$. In this case, from Eq. (4.4) easily can be verified that

$$u_{k+1} = (-s_k^{(n)}/s_k^{(1)}, 0, \ldots, 0, 1)^T \tag{4.7}$$

is the eigenvector corresponding to λ_1, where $s_k^{(i)}$ denotes the ith component of s_k. In addition, if $\theta_{k+1} = \kappa$, then $B_{k+1} = \theta_{k+1} I$, and $u_{k+1} = e_1 = (1, 0, \ldots, 0)^T$.

Obviously, the directions produced by the minimal-memory BFGS update are descent. More analytically, when $B_k > 0$, then p_k is defined as $p_k = -B_k^{-1} g_k$ and $d_k = 0$. Thus relations

$$p_k^T g_k = -g_k B_k^{-1} g_k < 0, \quad d_k^T g_k = 0, \quad \text{and} \quad d_k^T B_k d_k = 0,$$

hold. When B_k is indefinite, then $p_k = -g_k$ and $d_k = -\text{sgn}(u_k^T g_k) u_k$, where u_k is the normalized eigenvector defined in (3.10) or (4.7). Therefore, we have that

$$g_k^T p_k = -\|g_k\|^2 < 0, \quad d_k^T g_k = -\text{sgn}(u_k^T g_k) u_k^T g_k \leqslant 0, \quad \text{and} \quad d_k^T B_k d_k = \lambda_1 \|u_k\|^2 = \lambda_1 < 0.$$

Moreover, p_k and d_k satisfy Conditions 2.4–2.6, since p_k is a quasi-Newton related direction and d_k is associated with the unit eigenvector corresponding to the most negative eigenvalue of B_k [8].

5. The algorithm

We consider an iterative scheme of the form

$$x_{k+1} = \begin{cases} x_k + \alpha_k p_k, & \text{when } B_k > 0; \\ x_k + \alpha_k^2 p_k + \alpha_k d_k, & \text{otherwise}, \end{cases} \tag{5.1}$$

where p_k is the quasi-Newton direction, d_k is the negative curvature direction and α_k is a step size. When $B_k > 0$, the iterative scheme follows a linesearch procedure, otherwise it searches along a curve proposed by Morè and Sorensen [14]. The following algorithm describes a curvilinear search based on an Armijo procedure.

Algorithm 5.1. Curvilinear Memoryless BFGS

Step 1: Given x_0, $m = 1$, $0 < \sigma_1 < \sigma_2 < 1$, $0 < \mu < 1$, $0 < \eta < 1$ and $\epsilon \to 0$; set $k = 0$ and $\ell = 0$; compute g_0;
Step 2: If $\|g_k\| \leqslant \epsilon$ stop; else compute the eigenvalues λ_i of B_k;
Step 3: If $\lambda_1 > 0$ then
(a) compute p_k; set $d_k = 0$ and $\alpha = 1$;
(b) Find $\alpha_k \in [\sigma_1\alpha, \sigma_2\ \alpha]$such that

$$f(x_k + \alpha p_k) - f(x_k) \leqslant \mu\alpha g_k^T p_k; \tag{5.2}$$

and set $\alpha_k = \alpha$.
Else
(c) set $p_k = -g_k$ and compute u_k; set $d_k = -\ \mathrm{sgn}(u_k^T g_k)u_k$ and $\alpha_k = 1$;
(d) Choose the smallest non-negative integer i such that

$$f(x_k + \alpha_k^{2i} p_k + \alpha_k^i d_k) - f(x_k) \leqslant \eta\alpha_k^{2i}\left(g_k^T p_k + \frac{1}{2} d_k^T B_k d_k\right). \tag{5.3}$$

Step 4: Set

$$x_{k+1} = \begin{cases} x_k + \alpha_k p_k, & \text{if } \lambda_1 \geqslant 0; \\ x_k + \alpha_k^{2i} p_k + \alpha_k^i d_k, & \text{otherwise.} \end{cases} \tag{5.4}$$

Step 5: Compute g_{k+1}, $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$; if

$$|s_k^T y_k| > 10^{-6}\|s_k\|\|y_k\|,$$

save the vector pair $\{s_k, y_k\}$ and set $\ell = \ell + 1$; if $\ell > m$ discard the vector pair $\{s_{\ell-m}, y_{\ell-m}\}$ from storage;
Step 6: Set $k = k + 1$ and go to step 2;

Since B_k is allowed to be indefinite, the condition $s_k^T y_k > 0$ is not always hold. For being B_k well defined we skip the update if $|s_k^T y_k| \leqslant 10^{-6}\|s_k\|\|y_k\|$. The eigenvalues in Step 2 of Algorithm 5.1 can be computed by Eq. (3.5). In Step 3(a), the direction p_k is obtained by Eq. (4.3). In Step 3(c), u_k is obtained using Lemma 3.2. If the condition $|s_k^T y_k|/\|s_k\|\|y_k\| \leqslant 1 + \epsilon$, where $\epsilon \to 0^+$, is satisfied, then the vectors are collinear. In this case, if $\theta_{k+1} > \kappa$, then $u_k = s_k/\|s_k\|$, while if $\theta_{k+1} \leqslant \kappa$, then u_k is computed by Eq. (4.7).

Given the descent pair (p_k, d_k) produced by the memoryless BFGS matrix, we want to produce an $\alpha > 0$ such that either $f(x + \alpha p) < f(x)$ when $B > 0$, or $f(x + \alpha^2 p + \alpha d) < f(x)$ otherwise. The following lemma states that there exists an $\bar{\alpha} > 0$ such that $f(x + \alpha^2 p + \alpha d) < f(x)$, $\alpha \in (0, \bar{\alpha}]$.

Lemma 5.2. [14]*Let $\phi : \mathbb{R} \to \mathbb{R}$ be twice continuously differentiable on an open interval* **I** *which contains the origin, with $\phi(\alpha) = f(x + \alpha^2 p + \alpha d)$. Assume that $c \in (0, 1)$. Then there is an $\bar{\alpha} > 0$ in* **I** *such that*

$$\phi(\alpha) \leqslant \phi(0) + c\left[\phi'(0)\alpha + \frac{1}{2}\phi''(0)\alpha^2\right]$$

for all $\alpha \in [0, \bar{\alpha}]$ provided that either $\phi'(0) < 0$, or $\phi'(0) = 0$ and $\phi''(0) < 0$.

Lemma 5.3. *Let f and B_k satisfy Assumptions 2.1 and 2.2 respectively, and assume that (p_k, d_k) is a descent pair of directions produced by the minimal-memory BFGS matrix and Algorithm 5.1. Then there exists an $\alpha_k > 0$ such that (5.2) or (5.3) is satisfied.*

Proof. We consider the following cases:

(i) B_k is positive definite. The pair of directions is $p_k = -B_k^{-1} g_k$ and $d_k = 0$ and the linesearch procedure must satisfy (5.2). If $\alpha_k = 1$ then clearly (5.2) is satisfied. Suppose that inequality (5.2) were never satisfied. Then there exists a sequence α_j converging to 0 as $j \to \infty$ such that $f(x_k + \alpha_j p_k) - f(x_k) \geqslant \mu\alpha_j g_k^T p_k$. By the mean value theorem there exists $\delta \in (0, 1)$ such that

$$\alpha_j g(x_k + \delta\alpha_j p_k)^T p_k > \mu\alpha_j g_k^T p_k. \tag{5.5}$$

Since B_k is positive definite and $\|B_k\| \leqslant M$ for all k, we have that

$$p_k^T B_k p_k \leqslant M\|p\|^2. \tag{5.6}$$

Combining (5.6) and (5.5) we have that

$$M(\alpha_j\|p_k\|)^2 \geqslant \alpha_j[g(x_k+\delta\alpha_j p_k)-g(x_k)]^T p_k > (\mu-1)\alpha_j g_k^T p_k,$$

which implies the following inequality:

$$\frac{(\mu-1)g_k^T p_k}{M\|p_k\|^2} \leqslant \alpha_j. \tag{5.7}$$

For $j\to\infty$, relation (5.7) yields $(\mu-1)g_k^T p_k \leqslant 0$ contradicting the fact that $\mu\in(0,1)$ and $g_k^T p_k<0$.

(ii) B_{k+1} is indefinite. Then $p_k=-g_k, d_k=-\mathrm{sgn}(u_k^T g_k)u_k$, and the curvilinear search must satisfy (5.3). By letting $\phi_k(\alpha_k)=f(x_k+\alpha_k^2 p_k+\alpha_k d_k)$ we have that $\phi_k'(0)=g_k^T d_k\leqslant 0$ and $\phi_k''(0)=g_k^T p_k+d_k^T B_k d_k<0$. From Lemma 5.2 there exists $a_k>0$ such that (5.3) holds. Therefore, there exists $a_k>0$ such that either (5.2) or (5.3) hold. □

Theorem 5.4. *Let f and B_k satisfy Assumptions 2.1 and 2.2 respectively, and suppose that $\{\|p_k\|\}$ and $\{\|d_k\|\}$ are bounded. Let $\{x_k\}$ be the sequence of points produced by the Algorithm 5.1. Then*

$$\lim_{k\to\infty} g_k=0 \quad \textit{and} \quad \lim_{k\to\infty} d_k^T B_k d_k=0. \tag{5.8}$$

Proof. Since Ω is compact, the sequence of iterates $\{x_k\}$ admits at least a limit point which belongs to Ω. Let K_p and K_{pd} be index sets of two subsequences of iterates converging to a limit point such that

(i) for all $k\in K_p$

$$f(x_k+\alpha_k p_k)-f(x_k)\leqslant \mu\alpha_k g_k^T p_k$$

holds, and

(ii) for all $k\in K_{pd}$,

$$f(x_k+\alpha_k^{2i}p_k+\alpha_k^i d_k)-f(x_k)\leqslant \eta\alpha_k^{2i}\left(g_k^T p_k+\frac{1}{2}d_k^T B_k d_k\right)$$

holds. The index sets cannot be both empty.

First suppose that the index set K_p is not empty. This index set is related with the subsequence of iterates $\{x_{K_p}\}$ for which the matrices B_{K_p} are positive definite. Thus, relation (5.6) holds.

From (5.7) and (5.2) we obtain

$$f(x_k+\alpha_k p_k)\leqslant f(x_k)-\frac{\mu(1-\mu)}{M}\frac{(g_k^T p_k)^2}{\|p_k\|^2}, \quad k\in K_p.$$

If ϑ_k is the angle between p_k and $-g_k$, the above relation can be written as

$$f(x_k+\alpha_k p_k)\leqslant f(x_k)-\frac{\mu(1-\mu)}{M}\cos^2\vartheta_k\|g_k\|^2. \tag{5.9}$$

Since $s_k=-\alpha_k B_k^{-1}g_k$, and $B_k s_k=\theta_k s_k$, we have that

$$\cos\vartheta_k=-\frac{g_k^T p_k}{\|g\|\|p_k\|}=\frac{s_k^T B_k s_k}{\|s_k\|\|B_k s_k\|}=1.$$

Thus, relation (5.9) becomes

$$f(x_k+\alpha_k p_k)\leqslant f(x_k)-\frac{\mu(1-\mu)}{M}\|g_k\|^2.$$

From the assumption of the theorem the sequence $\{f_k\}$, $k\in K_p$ is bounded below and $\{f_{k+1}-f_k\}$ converges to zero, which implies $\lim_{k\to\infty}\|g_k\|=0$. Moreover, $d_k=0$ for all $k\in K_p$, and obviously $\lim_{k\to\infty} d_k^T B_k d_k=0$.

Suppose now that the index set K_{pd}, which related with the subsequence of iterates $x_{K_{pd}}$ for which the matrices $B_{K_{pd}}$ are indefinite, is not empty. If i_k is the smallest non-negative integer such that $x_{k+1}=x_k+\alpha^{2i_k}p_k+\alpha^{i_k}d_k\in\Omega$ and (5.3) hold, it follows from (5.3) that

$$|f(x_{k+1})-f(x_k)|\geqslant c\alpha^{2i_k}\left|g_k^T p_k+\frac{1}{2}d_k^T B_k d_k\right|$$

for all $k\in K_{pd}$. Since $\{f_{k+1}-f_k\}\to 0$, $k\in K_{pd}$, we have that

$$\alpha^{2i_k}\left|g_k^T p_k+\frac{1}{2}d_k^T B_k d_k\right|\to 0, \quad k\to\infty, \quad k\in K_{pd}.$$

Therefore, taking into account that $g_k^T p_k = -\|g_k\|^2$, either

$$\lim_{k\to\infty} g_k = 0 \quad \text{and} \quad \lim_{k\to\infty} d_k^T B_k d_k = 0, \quad k \in K_{pd},$$

or $\alpha^{i_k} \to 0$ as $k \to \infty$, $k \in K_{pd}$, that is $\lim_{k\to\infty} i_k = \infty$. Suppose that $\alpha_k^{i_k} \to 0$. Then by the definition of i_k we have

$$f(x_k + \alpha^{2i_k} p_k + \alpha^{i_k} d_k) > f(x_k) + c\alpha^{2i_k}\left(g_k^T p_k + \frac{1}{2} d_k^T B_k d_k\right). \tag{5.10}$$

Expanding the left-hand side of (5.10) using Taylor's theorem we have

$$\alpha^{2i_k} g_k^T p_k + \alpha^{i_k} g_k^T d_k + \frac{1}{2}\left(\alpha^{2i_k} p_k + \alpha^{i_k} d_k\right)^T B_k\left(\alpha^{2i_k} p_k + \alpha^{i_k} d_k\right) + o(\alpha^{2i_k}) > c\alpha^{2i_k}\left(g_k^T p_k + \frac{1}{2} d_k^T B_k d_k\right).$$

Using the fact that $g_k^T d_k \leqslant 0$ and accumulating all terms of order higher than $O(\alpha^{2i_k})$ into the $o(\alpha^{2i_k})$ we have

$$o(\alpha^{2i_k}) > -(1-c)\alpha^{2i_k}\left(g_k^T p_k + \frac{1}{2} d_k^T B_k d_k\right). \tag{5.11}$$

Since

$$\lim_{k\to\infty} \frac{o(\alpha^{2i_k})}{(1-c)\alpha^{2i_k}} = 0, \quad -g_k^T p_k \geqslant 0 \quad \text{and} \quad -d_k^T B_k d_k \geqslant 0, \quad k \in K_{pd}, \tag{5.12}$$

with $g_k^T p_k = -\|g_k\|^2$, the conclusion follows from (5.11) and (5.12).

Hence, since $K_p \bigcup K_{pd} \neq \emptyset$, every limit point of the sequence satisfies (5.8). □

6. Numerical results

In order to evaluate the behavior of our algorithm, we have implemented the curvilinear memoryless BFGS Algorithm 5.1, and tested it on problems from the CUTEr collection proposed by Gould et al. [10]. The problems used in the numerical experiments are nonlinear unconstrained problems with a number of variables which ranges between 999 and 100,000, having continuous second derivatives, and bounded below. As a total, the selected test is composed by 58 problems (*ARWHEAD, BROYDN7D, BRYBND, CHAINWOO, COSINE, CRAGGLVY, DIXMAANA, DIXMAANB, DIXMAANC, DIXMAAND, DIXMAANE, DIXMAANF, DIXMAANG, DIXMAANH, DIXMAANI, DIXMAANJ, DIXMAANK, DIXMAANL, DIXON3DQ, DQDRTIC, DQRTIC, EDENSCH, ENGVAL1, EXTROSNB, FLETCBV2, FLETCBV3, FLETCHBV, FLETCHCR, FMINSRF2, FMINSURF, FREUROTH, GENHUMPS, GENROSE, INDEF, LIARWHD, MOREBV, MSQRTALS, MSQRTBLS, NONCVXU2, NONCVXUN, NONDIA, NONDQUAR, PENALTY1, POWELLSG, POWER, QUARTC, SCHMVETT, SPARSINE, SPARSQUR, SPMSRTLS, SROSENBR, TESTQUAD, TOINTGSS, TQUARTIC, TRIDIA, VARDIM, VAREIGVL, WOODS*). For each test function we have considered 3 numerical experiments with number of variables 10^3, 10^4, and 10^5. In all cases, we have used the initial points provided by the CUTEr environment. All the experiments were run on an Pentium 1.86 GHz personal computer, 2 GB of RAM memory and Linux operating system. The proposed algorithm, namely `CMBFGS`,[1] was coded in FORTRAN 90 and the compiler option " -O" was adopted. In our implementation we have used the settings $\mu = \eta = 10^{-4}$.

We have compared our method with the line search L-BFGS method [15], namely `LBFGS` since our algorithm can be viewed as a modification of the memoryless BFGS method. The L-BFGS code was obtained from Jorge Nocedal's web page, and one vector pair was used for the memory. For both methods the scalar parameter θ was defined as $s_k^T y_k / y_k^T y_k$. The termination criterion was $\|g_k\| \leqslant 10^{-5}$. We consider as failure all the runs which the convergence criterion was not fulfilled within a maximum number of iterations (maxiter = 10,000).

Table 1 presents a summary of the total CPU time (in seconds) needed for each one of the algorithms, counted only in the successfully solved problems. We can observe that, even if the `CMBFGS` algorithm solved more problems than the `LBFGS` algorithm, the total CPU time is similar for both algorithms. Additionally, in Table 1 is also summarized the number of problems that `CMBFGS` and `LBFGS` algorithms fail to solve, respectively. In the smallest dimension ($n = 1000$), both algorithms failed to solve the problems *FLETCBV3, FLETCHBV, INDEF, NONCVXUN*. In addition, the `LBFGS` algorithm fail to solve the problem *FREUROTH*. For $n = 10{,}000$ the 15 problems that the `CMBFGS` failed to solve are: *ARWHEAD, CHAINWOO, CRAGGLVY, FLETCBV3, FLETCHBV, FLETCHCR, FREUROTH, GENHUMPS, GENROSE, INDEF, NONCVXU2, NONCVXUN, SPARSINE, TESTQUAD, VARDIM*; the 16 failures of the `LBFGS` algorithm are the problems *ARWHEAD, BROYDN7D, CHAINWOO, CRAGGLVY, ENGVAL1, FLETCBV3, FLETCHBV, FLETCHCR, FREUROTH, GENROSE, INDEF, NONCVXU2, NONCVXUN, SPARSINE, TESTQUAD, VARDIM*. Finally in the largest dimension ($n = 100{,}000$) both algorithms failed to solve the 17 problems *BROYDN7D, CHAINWOO, CRAGGLVY, ENGVAL1, FLETCBV3, FLETCHBV, FLETCHCR, FREUROTH, GENHUMPS, GENROSE, INDEF, NONCVXU2, NONCVXUN, SPARSINE, TESTQUAD, TRIDIA, VARDIM*. The `LBFGS` algorithm also failed to solve the problems *EDENSCH, PENALTY1*. Table 2 shows the total number of the computed negative curvature directions in the `CMBFGS` algorithm, excluding those problems where the algorithm failed to solve.

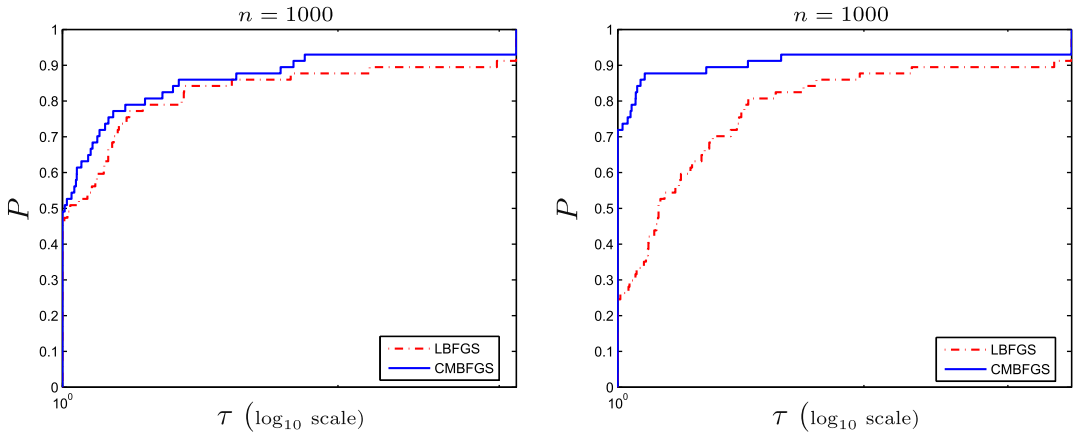
[1] Available from http://www.math.upatras.gr/~msa/.

Table 1
Total CPU time on successfully solved problems and total failures in all dimensions.

	CPU		Failures	
	CMBFGS	LBFGS	CMBFGS	LBFGS
$n = 1000$	29.58	26.77	4	5
$n = 10{,}000$	70.36	85.94	15	16
$n = 100{,}000$	570.71	510.99	17	19
Summary	670.65	623.70	26	30

Table 2
Total number of computed negative curvature directions in successfully solved problems.

Dimension	Eigenvectors
$n = 1{,}000$	110
$n = 10{,}000$	24
$n = 1{,}000{,}000$	24
Total number of computed eigenvectors:	158



(a) Performance based on function evaluations (b) Performance based on gradient evaluations

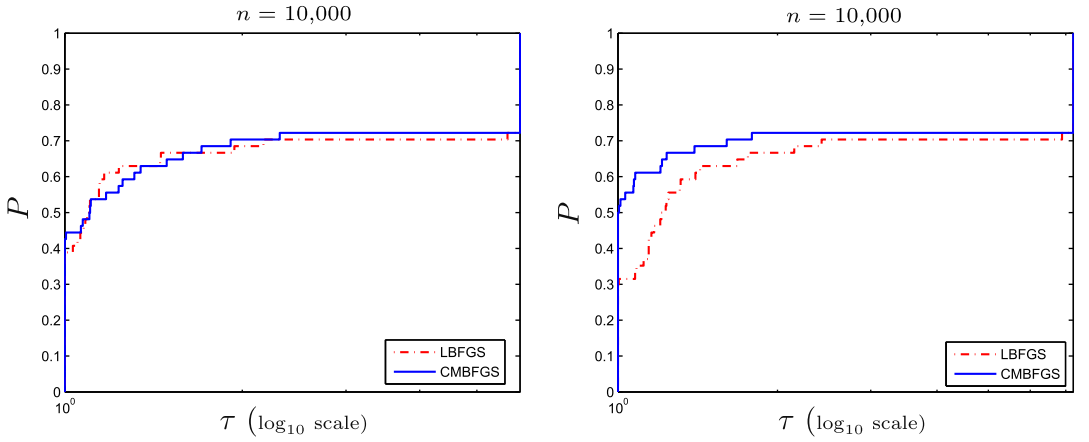
Fig. 1. Performance profiles of CMBFGS and LBFGS, in 1000 dimensions.

We have used the performance profile proposed by Dolan and Moré [3] to display the performance of each implementation on the set of test problems, in terms of function and gradient evaluations. The performance profile plots the fraction P of problems for which any given method is within a factor τ 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). We have used the Libopt[2] environment [7] for measuring the efficiency and the robustness of our algorithm in terms of function and gradient evaluations. The Libopt environment is, among others, an excellent set of tools that can be used for comparing the results obtained by solvers on a specified set of problems, and profiling them. It has been written in Perl and uses Unix/Linux operating system.

Fig. 1 presents the performance profiles for problems with 1000 variables. Figs. 1(a) and (b) show the results where the function and gradient evaluations are used as the performance metric, respectively. In Fig. 1(a) we can observe that both algorithms perform similarly, with CMBFGS algorithm exhibits a slight improvement, in terms of function evaluations. Fig. 1(b) reports that, the proposed curvilinear search algorithm is much more efficient than the classical line search algorithm, in terms of gradient evaluations.

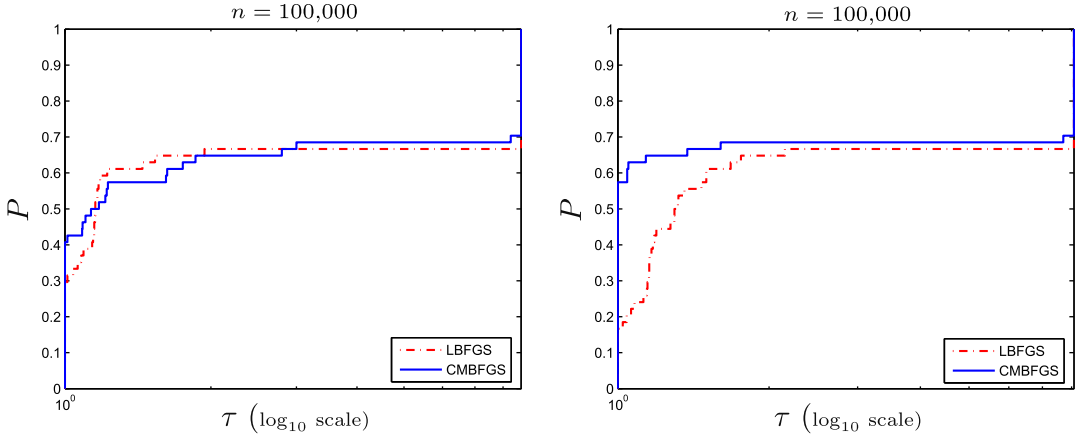
Similar observations can be made by Figs. 2 and 3, that illustrate the performance comparisons for $n = 10{,}000$ and $n = 100{,}000$, respectively. More analytically, both Figs. 2(a) and 3(a) indicate that, in terms of function evaluations, CMBFGS and LBFGS have almost identical performance. Figs. 2(b) and 3(b) presents the numerical results in terms of gradient evaluations. In both Figures, CMBFGS achieved the top performance, in terms of both efficiency and robustness.

[2] Available from http://www-rocq.inria.fr/~gilbert/modulopt/libopt/.



(a) Performance based on function evaluations (b) Performance based on gradient evaluations

Fig. 2. Performance profiles of CMBFGS and LBFGS, in 10,000 dimensions.



(a) Performance based on function evaluations (b) Performance based on gradient evaluations

Fig. 3. Performance profiles of CMBFGS and LBFGS, in 100,000 dimensions.

The best overall performance relative to both function and gradient metric, was obtained by CMBFGS, especially in terms of gradient evaluations. It turns out that the main advantage of the proposed method regarding the use of the information gained by the minimal-memory BFGS matrices, is the efficiency for solving large scale unconstrained optimization problems.

7. Conclusions

In this work, we have proposed an algorithmic model based on a modification of the minimal-memory BFGS method for solving large scale unconstrained optimization problems, that incorporates a curvilinear and a linesearch procedure. This model exploits the curvature information provided by the minimal-memory BFGS matrices using a pair of search directions, a memoryless quasi-Newton direction and a direction of negative curvature. The proposed method for the computation of the negative curvature direction is accomplished avoiding any storage and matrix factorizations. Hence, the amount of memory needed for computing both the smallest eigenvalue and corresponding eigenvector is negligible, providing that very large problems can be solved efficiently.

Acknowledgements

The authors thank the anonymous referees for their valuable comments and suggestions.

References

[1] L. Armijo, Minimization of functions having lipschitz continous first partial derivatives, Pac. J. Math. 16 (1) (1966) 1–3.
[2] J. Barzilai, J.M. Borwein, Two point step size gradient method, IMA J. Numer. Anal. 8 (1988) 141–148.
[3] E. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.
[4] G. Fasano, M. Roma, Iterative computation of negative curvature directions in large scale optimization, Comput. Optim. Appl. 38 (1) (2007) 81–104.
[5] M.C. Ferris, S. Lucidi, M. Roma, Nonmonotone curvilinear line search methods for unconstrained optimization, Comput. Optim. Appl. 6 (1996) 117–136.
[6] A.V. Fiacco, G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley, New York, 1968.
[7] J.C. Gilbert, X. Jonsson, LIBOPT – an environment for testing solvers on heterogeneous collections of problems. The manual, version 2.1. INRIA Technical Report, RT-331, 2009.
[8] D. Goldfarb, Curvilinear path steplength algorithms for minimization which use directions of negative curvature, Math. Program. 18 (1980) 31–40.
[9] N. Gould, S. Lucidi, M. Roma, Ph.L. Toint, Exploiting negative curvature directions in linesearch methods for unconstrained optimization, Optim. Method Softw. 14 (2000) 75–98.
[10] N.I.M. Gould, D. Orban, Ph.L. Toint, CUTEr and SifDec: a constrained and unconstrained testing enviroment, revisited, ACM Trans. Math. Softw. 29 (4) (2003) 373–394.
[11] S. Lucidi, F. Rochetich, M. Roma, Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization, SIAM J. Optim. 8 (4) (1998) 916–939.
[12] S. Lucidi, M. Roma, Numerical experiences with new truncated newton methods in large scale unconstrained optimization, Comput. Optim. Appl. 7 (1997) 71–87.
[13] G.P. McCormick, A modification of Armijo's step-size rule for negative curvature, Math. Program. 13 (1977) 111–115.
[14] J. Moré, D. Sorensen, On the use of directions of negative curvature in a modified Newton method, Math. Program. 16 (1979) 1–20.
[15] J. Nocedal, Updating quasi-Newton matrices with limited storage, Math. Comput. 35 (151) (1980) 773–782.
[16] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, New York, 1999.
[17] A. Olivares, J.M. Moguerza, F.J. Prieto, Nonconvex optimization using negative curvature within a modified linesearch, Eur. J. Oper. Res. 189 (3) (2008) 706–722.
[18] C-j. Wang, Dogleg paths and trust region methods with back tracking technique for unconstrained optimization, Appl. Math. Comput. 177 (1) (2006) 159–169.
[19] Y. Wang, D. Zhu, An affine scaling optimal path method with interior backtracking curvilinear technique for linear constrained optimization, Appl. Math. Comput. 207 (1) (2009) 178–196.
[20] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, London, 1965.