SQU Journal for Science, 2021, 26(2), 126-140 DOI:10.53539/squjs.vol26iss2pp126-140
Sultan Qaboos University

Improved Diagonal Hessian Approximations for Large-Scale Unconstrained Optimization

Ahmed Al-Siyabi* and Mehiddin Al-Baali

Department of Mathematics, College of Science, Sultan Qaboos University, P.O. Box 36, PC 123, Al-Khod, Muscat, Sultanate of Oman. *Email: aalsiyabi@squ.edu.om.

ABSTRACT: We consider some diagonal quasi-Newton methods for solving large-scale unconstrained optimization problems. A simple and effective approach for diagonal quasi-Newton algorithms is presented by proposing new updates of diagonal entries of the Hessian. Moreover, we suggest employing an extra BFGS update of the diagonal updating matrix and use its diagonal again. Numerical experiments on a collection of standard test problems show, in particular, that the proposed diagonal quasi-Newton methods perform substantially better than certain available diagonal methods.

Keywords: Unconstrained optimization; Large-scale problems; Quasi-Newton methods; BFGS update.

تحسين تقريبات هس القطرية في الأمثليات غيرالمقيدة ذات أبعاد عالية

أحمد السيابي و محي الدين البعلي

الملخص: نفترض بعض الطرق المشابهة لطريقة نيوتن بمصفوفة قطرية لحل مسائل الأمثليات غير المقيدة وبأبعاد عالية. يتم تقديم نهج بسيط وفعال لخوارزميات تلك الطرق من خلال اقتراح تعديلات جديدة إلى قطر مصفوفة هس التقريبية. إضافة إلى ذلك، نقترح تطبيق دستور التعديل BFGS بشكل إضافي على مصفوفة القطرالمعدلة واستخدام قطر المصفوفة الناتجة مرة أخرى. تُظهر التجارب العددية على مجموعة من المسائل النموذجية، بشكل خاص، أن طرق نيوتن المشابهة المقترحة لتعديل قطرالمصفوفة التقريبي تعمل بشكل أفضل بكثير من طرق قطرية معروفة.

الكلمات المفتاحية: الأمثليات غير المقيدة، مسائل ذات أبعاد عالية، طرق نيوتن المشابهة، تعديل BFGS.

1. Introduction

This paper is concerned with quasi-Newton methods for solving the large-scale unconstrained optimization problem

$$\min_{x\in\Re^n} f(x), \tag{1.1}$$

where $f:\Re^n \longrightarrow \Re$ is a twice continuously differentiable function. It is assumed that n is large so that a matrix cannot be stored explicitly. The methods are defined like the Newton method with line search framework, except that the Hessian $G(x_k) = \nabla^2 f(x_k)$ is replaced by a symmetric and positive definite matrix B_k. This Hessian approximation satisfies the so-called quasi-Newton condition for $k > 1$, assuming B_1 is given (usually, $B_1 = I$, the identity matrix). The quasi-Newton methods are defined iteratively as follows. At the beginning of each iteration, x_k is available (x_1 is given) so that the gradient $g(x_k) = \Delta f(x_k)$ is computed. If this vector (denoted by g_k) is not sufficiently close to zero, a search direction s_k is provided ($s_1 = -B_1^{-1} g_1$) such that the descent property $s_k^T g_k < 0$ holds. Thus, a positive steplength α_k which reduces $f(x_k)$ along s_k exists. In practice, α_k is usually chosen to satisfy the Wolfe-Powell conditions

$$f_{k+1} \leqslant f_k + \sigma_0 \alpha_k g_k^T s_k, \qquad g_{k+1}^T s_k \geqslant \sigma_1 g_k^T s_k, \tag{1.2}$$

where f_k denotes $f(x_k)$, $0 < \sigma_0 < 0.5$ and $\sigma_0 < \sigma_1 < 1$. Then a new point is given by

$$x_{k+1} = x_k + \alpha_k s_k. \tag{1.3}$$

For the next iteration, the Hessian approximation B_k is updated to B_{k+1} in terms of the two vectors

$$\delta_k = x_{k+1} - x_k, \qquad \gamma_k = g_{k+1} - g_k, \tag{1.4}$$

such that the quasi-Newton condition

$$B_{k+1}\delta_k = \gamma_k \tag{1.5}$$

holds. Hence, the next search direction s_{k+1} is computed by solving the system of linear equations

$$B_{k+1} s_{k+1} = -g_{k+1}. \tag{1.6}$$

Although there exist many quasi-Newton updating formulae in the literature, we will focus on the popular BFGS update

$$B_{k+1} = B_k - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k} + \frac{\gamma_k \gamma_k^T}{\delta_k^T \gamma_k}, \tag{1.7}$$

because this formula has the following useful features. It satisfies the quasi-Newton condition (1.5) and maintains the Hessian approximations positive definite if the curvature condition

$$\delta_k^T \gamma_k > 0 \tag{1.8}$$

holds, which is guaranteed if the Wolfe-Powell conditions (1.2) are satisfied. In addition, the corresponding BFGS method converges superlinearly for convex objective functions. (For more details, see, for example, Fletcher [1]).

Since the updated matrix B_{k+1} cannot be stored explicitly, for sufficiently large values of n it is replaced by another matrix that can be stored implicitly, so that the search direction in (1.6) is simply computed for all k (see for example, Andrei [2], and Nocedal and Wright [3]). Here, we consider several proposals for maintaining B_k diagonal, although the quasi-Newton feature (1.5) is not expected to be satisfied. Thus, a number of diagonal updates have been proposed (see the next section), some of which satisfy the weak quasi-Newton condition

$$\delta_k^T B_{k+1} \delta_k = \delta_k^T \gamma_k. \tag{1.9}$$

However, we will consider the possibility of improving the role of these matrices by trying to impose the quasi-Newton feature through an extra BFGS updating strategy.

The rest of our work is organized as follows. In the next section, we discuss a selection strategy for diagonal entries of quasi-Newton BFGS Hessian and inverse Hessian approximations. Section 3 defines other diagonal updates (some satisfy the weak quasi-Newton condition (1.9)). In Section 4 we discuss the convergence property, while in Section 5 some numerical results are presented. We will see that the proposed technique improves upon the performance of the diagonal matrix updates substantially. Finally, we conclude in Section 6.

2. Diagonal quasi-Newton BFGS updates

Although it is possible to define the inverse Hessian approximation by employing a number of BFGS updates in terms of vector pairs $\{\delta_i, \gamma_i\}$, for some $i \leq k$, without storing a matrix explicitly (see, for example, Buckley and LeNir [4], Fletcher [5], Liu and Nocedal [6], and Shanno [7]). We do not consider them here because we focus on the simple diagonal Hessian approximations and diagonal inverse Hessian approximations.

In this section, we describe the BFGS update for maintaining the Hessian approximations B_k or its inverse H_k diagonal for all k. For convenience, we denote $\hat{B} = \operatorname{diag}[B] \equiv \operatorname{diag}\left[\hat{B}^{(1)}, \dots, \hat{B}^{(n)}\right]$ and note that for any two vectors u and v we have

$$\operatorname{diag}(uv^T) = \operatorname{diag}\left[u^{(1)}v^{(1)}, \dots, u^{(n)}v^{(n)}\right],$$
$$\operatorname{diag}(\hat{B}u) = \operatorname{diag}\left[\hat{B}^{(1)}u^{(1)}, \dots, \hat{B}^{(n)}u^{(n)}\right],$$
$$\operatorname{diag}(\hat{B}uv^T) = \operatorname{diag}\left[\hat{B}^{(1)}u^{(1)}v^{(1)}, \dots, \hat{B}^{(n)}u^{(n)}v^{(n)}\right].$$

Thus, in particular, it follows from (1.7) that the diagonal BFGS update of a diagonal matrix $\hat{B}_k$ can be written as follows:

$$\hat{B}_{k+1} = \hat{B}_k - \frac{\operatorname{diag}(\hat{B}_k \delta_k \delta_k^T \hat{B}_k)}{\delta_k^T \hat{B}_k \delta_k} + \frac{\operatorname{diag}(\gamma_k \gamma_k^T)}{\delta_k^T \gamma_k} \tag{2.1}$$

or equivalently,

$$\hat{B}_{k+1}^{(i)} = \hat{B}_k^{(i)} - \frac{\left(\hat{B}_k^{(i)} \delta_k^{(i)}\right)^2}{\sum_{j=1}^{n} \hat{B}_k^{(j)} \left(\delta_k^{(j)}\right)^2} + \frac{\left(\gamma_k^{(i)}\right)^2}{\delta_k^T \gamma_k}, \tag{2.2}$$

for $i = 1, \dots, n$. We note that this formula requires the storage of only three vectors. We first consider the most efficient quasi-Newton BFGS method that is defined by (1.3), (1.7) and (1.6). For convenience, we rewrite the BFGS update (1.7) as bfgs matrix function

$$B_{k+1} = \operatorname{bfgs}(B_k, \delta_k, \gamma_k), \tag{2.3}$$

where

$$\operatorname{bfgs}(B, \delta, \gamma) = B - \frac{B\delta\delta^T B}{\delta^T B \delta} + \frac{\gamma\gamma^T}{\delta^T \gamma} \tag{2.4}$$

is the BFGS formula for updating any symmetric matrix B in terms of any two vectors δ and γ. This formula has the useful features that it maintains the positive definiteness of B if $\delta^T \gamma > 0$ and satisfies the quasi-Newton condition $\operatorname{bfgs}(B, \delta, \gamma)\delta = \gamma$. Thus, the updated matrix (2.3) is maintained positive definitive if the curvature condition (1.8) holds.

Since the updated matrix (2.3) cannot be stored explicitly, Gilbert and Lemaréchal [8] suggest using the diagonal BFGS update

$$\hat{B}_{k+1} = \operatorname{diag}\left[\operatorname{bfgs}(\hat{B}_k, \delta_k, \gamma_k)\right], \tag{2.5}$$

which is equivalent to both (2.1) and (2.2), assuming $\hat{B}_1$ is given diagonal. In this case, $\hat{B}_k$ remains diagonal and positive definite so that the search direction (1.6) is simply computed as

$$s_{k+1} = -\hat{B}_{k+1}^{-1} g_{k+1}, \tag{2.6}$$

without computing the inverse matrix $\hat{B}_{k+1}^{-1} (= \hat{H}_{k+1}$, say) explicitly. In the next section, we will consider a modification of this method.

We now show that the inverse matrix of (2.3) which defines the BFGS inverse Hessian approximation yields a choice for defining $\widehat{H}_{k+1}$ explicitly as follows. Let the BFGS inverse Hessian approximation be defined by

$$H_{k+1} = \text{bfgs}^{-1}(H_k, \delta_k, \gamma_k), \tag{2.7}$$

where for any symmetric matrix H and two vectors δ and γ,

$$\text{bfgs}^{-1}(H, \delta, \gamma) = H + \left(1 + \frac{\gamma^T H \gamma}{\delta^T \gamma}\right)\frac{\delta\delta^T}{\delta^T\gamma} - \frac{\delta\gamma^T H + H\gamma\delta^T}{\delta^T\gamma}. \tag{2.8}$$

Thus, if H is stored implicitly (particularly, when it is diagonal) such that the product Hu is available for any vector u, then the product $\text{bfgs}^{-1}(H, \delta, \gamma)v$, for any vector v, can be computed without storing the updated matrix $\text{bfgs}^{-1}(H, \delta, \gamma)$ explicitly. Hence, the search direction (1.6) for the next iteration can be computed directly by

$$s_{k+1} = -H_{k+1} g_{k+1}, \tag{2.9}$$

although H_{k+1} is a dense nondiagonal matrix. This technique maintains the useful quasi-Newton condition $H_{k+1}\gamma_k = \delta_k$ and will be applied to some diagonal matrices to impose the quasi-Newton condition again (see Al-Siyabi, [9]).

As for the BFGS update, we now apply the diagonal technique to the inverse BFGS update (2.7) to obtain

$$\widehat{H}_{k+1} = \text{diag}\left[\text{bfgs}^{-1}\left(\widehat{H}_k, \delta_k, \gamma_k\right)\right]. \tag{2.10}$$

Although the inverse BFGS updated matrix $\text{bfgs}\left(\widehat{B}_k, \delta_k, \gamma_k\right)^{-1} = \text{bfgs}^{-1}\left(\widehat{H}_k, \delta_k, \gamma_k\right)$, since $\widehat{H}_k = \widehat{B}_k^{-1}$, we note that $\text{diag}\left(\text{bfgs}\left(\widehat{B}_k, \delta_k, \gamma_k\right)\right)^{-1} \neq \text{diag}\left(\text{bfgs}^{-1}\left(\widehat{H}_k, \delta_k, \gamma_k\right)\right)$; i.e., the inverse of the diagonal updated matrix (2.5) differs from (2.10). Therefore, for convenience, we rewrite the search direction (2.6) as follows:

$$s_{k+1} = -\widehat{H}_{k+1} g_{k+1}. \tag{2.11}$$

In practice, the diagonal choice (2.5)– (2.6) is preferred to (2.10)– (2.11). In fact, the diagonal inverse BFGS update of (2.10) has been suggested by Gilbert and Lemaréchal [8], as follows:

$$\widehat{H}_{k+1} = \widehat{H}_k + \left(1 + \frac{\gamma_k^T \widehat{H}_k \gamma_k}{\delta_k^T \gamma_k}\right)\frac{\text{diag}\left(\delta_k \delta_k^T\right)}{\delta_k^T \gamma_k} - 2\frac{\text{diag}\left(\widehat{H}_k \gamma_k \delta_k^T\right)}{\delta_k^T \gamma_k}, \tag{2.12}$$

or equivalently,

$$\widehat{H}_{k+1}^{(i)} = \widehat{H}_k^{(i)} + \left(1 + \frac{\sum_{j=1}^n \widehat{H}_k^{(j)}\left(\gamma_k^{(j)}\right)^2}{\delta_k^T \gamma_k}\right)\frac{\left(\delta_k^{(i)}\right)^2}{\delta_k^T \gamma_k} - 2\frac{\delta_k^{(i)} \gamma_k^{(i)} \widehat{H}_k^{(i)}}{\delta_k^T \gamma_k}, \tag{2.13}$$

with a certain positive definite diagonal matrix $\widehat{H}_1$ so that the updated diagonal matrices are maintained positive definite if the curvature condition $\delta_k^T \gamma_k > 0$ is satisfied.

Now, the corresponding algorithm for the above diagonal quasi-Newton updates can be outlined in the following way, where throughout the paper $\|\cdot\|$ denotes the Euclidean vector norm.

Algorithm 2.1.
Step 0: Given an initial point x_1, a symmetric and positive definite diagonal matrix $\widehat{B}_1$ and $\epsilon > 0$ (an acceptance tolerance on the gradient norm). Set $k = 1$ and compute the initial search direction $s_1 = -\widehat{B}_1^{-1} g_1$.
Step 1: Compute a steplength α_k and a new point $x_{k+1} = x_k + \alpha_k s_k$, such that the Wolfe-Powell conditions (1.2) hold.
Step 2: If $\|g_{k+1}\| \leq \epsilon$, then stop.
Step 3: Compute the vectors $\delta_k = x_{k+1} - x_k$ and $\gamma_k = g_{k+1} - g_k$.
Step 4: Define a new diagonal Hessian approximation $\widehat{B}_{k+1}$ by (2.1) using $\widehat{B}_k$, δ_k and γ_k.
Step 5: Compute the new search direction $s_{k+1} = -\widehat{B}_{k+1}^{-1} g_{k+1}$.
Step 6: Set $k = k + 1$, and go to Step 1.

Note that in steps 0 and 5, the search direction is computed without forming $\widehat{B}_{k+1}^{-1}$ explicitly; i.e., it is calculated as $s_{k+1}^{(i)} = -g_{k+1}^{(i)} / \widehat{B}_{k+1}^{(i)}$, for $i = 1,2,\dots,n$. In Step 4, we define $\widehat{B}_{k+1}$ in particular by the diagonal BFGS Hessian approximation (2.1) or equivalently (2.5), unless otherwise stated. However, if a diagonal inverse Hessian approximation is considered, using for example formula (2.10) for updating $\widehat{H}_k$ to $\widehat{H}_{k+1}$, then steps 0, 4 and 5 are used with $\widehat{B}_j^{-1}$ replaced by $\widehat{H}_j$, for $j = 1, k$ and $k+1$, respectively.

3. Diagonal non quasi-Newton updates

In this section, we describe some diagonal Hessian approximations $\widehat{B}_k$ which satisfy certain useful properties. Although it is possible to define diagonal inverse Hessian approximations $\widehat{H}_k$, we do not consider them here because they are inferior to the direct Hessian approximations (Al-Siyabi [9]).

Nazareth [10] has proposed the following Hessian approximation update:

$$B_{k+1} = B_k + \frac{\delta_k^T\gamma_k - \delta_k^T B_k \delta_k}{\left(\delta_k^T B_k \delta_k\right)^2} B_k \delta_k \delta_k^T B_k. \tag{3.1}$$

This satisfies the weak quasi-Newton condition (1.9) and maintains the Hessian approximations positive definite whenever the curvature condition (1.8) holds. The author suggests replacing B_k by a diagonal matrix $\widehat{B}_k$ to obtain the new diagonal Hessian approximation

$$\widehat{B}_{k+1} = \widehat{B}_k + \frac{\delta_k^T\gamma_k - \delta_k^T \widehat{B}_k \delta_k}{\left(\delta_k^T \widehat{B}_k \delta_k\right)^2} \operatorname{diag}\left(\widehat{B}_k \delta_k \delta_k^T \widehat{B}_k\right), \tag{3.2}$$

or equivalently,

$$\widehat{B}_{k+1}^{(i)} = \widehat{B}_k^{(i)} + \frac{\delta_k^T\gamma_k - \sum_{j=1}^n \widehat{B}_k^{(j)}\left(\delta_k^{(j)}\right)^2}{\left(\sum_{j=1}^n \widehat{B}_k^{(j)}\left(\delta_k^{(j)}\right)^2\right)^2} \left(\widehat{B}_k^{(i)}\right)^2 \left(\delta_k^{(i)}\right)^2, \tag{3.3}$$

for $i = 1, \dots, n$. This also remains positive definite if the curvature condition (1.8) holds. Although a formula for the inverse update of (3.1) can be used to define diagonal inverse Hessian approximations $\widehat{H}_k$, we do not consider it here, because the performance of the corresponding method is worse than that of the above one (for details, see Al-Siyabi [9]).

Moreover, Zhu *et al*. [11] proposed the direct diagonal update

$$\widehat{D}_{k+1} = \widehat{B}_k + \frac{\delta_k^T\gamma_k - \delta_k^T \widehat{B}_k \delta_k}{\operatorname{tr}(\widehat{E}_k)^2} \widehat{E}_k, \tag{3.4}$$

where

$$\widehat{E}_k = \operatorname{diag}\left[\left(\delta_k^{(1)}\right)^2, \left(\delta_k^{(2)}\right)^2, \dots, \left(\delta_k^{(n)}\right)^2\right],$$

or equivalently,

$$\widehat{D}_{k+1}^{(i)} = \widehat{B}_k^{(i)} + \frac{\delta_k^T\gamma_k - \sum_{j=1}^n \widehat{B}_k^{(j)}\left(\delta_k^{(j)}\right)^2}{\sum_{j=1}^n \left(\delta_k^{(j)}\right)^4} \left(\delta_k^{(i)}\right)^2, \tag{3.5}$$

for $i = 1, \dots, n$, which satisfies the weak quasi-Newton condition (1.9).

Since the positive definiteness of this update is not guaranteed, the authors introduced the following safeguarding test. If the inequality $\widehat{D}_{k+1}^{(i)} < \epsilon_1$ holds for any i and some $\epsilon_1 > 0$ (we used $\epsilon_1 = 10^{-6}$), the authors suggest resetting the above updated matrix to the scaled identity matrix $\tau_k^0 I$, where

$$\tau_k^0 = \frac{\gamma_k^T\gamma_k}{\delta_k^T\gamma_k} \tag{3.6}$$

is suggested by Oren and Luenberger [12], which is positive if the curvature condition holds. Thus, the authors propose the positive definite diagonal update as follows:

$$\widehat{B}_{k+1} = \begin{cases} \widehat{D}_{k+1}, & \text{if } \widehat{D}_{k+1}^{(i)} \geq \epsilon_1, \forall i, \\ \tau_k^0 I, & \text{otherwise.} \end{cases} \tag{3.7}$$

Recently, Sim et al. [13] proposed the following diagonal Hessian approximation:

$$\hat{B}_{k+1} = \begin{cases} \hat{C}_{k+1}, & \text{if } \hat{\tau}_k^0 < 1 \\ \hat{\tau}_k^0 I, & \text{otherwise,} \end{cases} \tag{3.8}$$

where

$$\hat{C}_{k+1}^{(i)} = \frac{1}{1+\omega_k\left(\delta_k^{(i)}\right)^2}, \quad \omega_k = \frac{\delta_k^T\delta_k - \delta_k^T\gamma_k}{\sum_{j=1}^n\left(\delta_k^{(j)}\right)^4}, \tag{3.9}$$

and

$$\hat{\tau}_k^0 = \frac{\delta_k^T\gamma_k}{\delta_k^T\delta_k}. \tag{3.10}$$

Since this self-scaling parameter of Oren and Luenberger [12] is positive if the curvature condition holds, the proposed diagonal matrix (3.8) is positive definite. The authors derived the above diagonal matrix $\hat{B}_{k+1}$ as follows. If $\hat{\tau}_k^0 \geq 1$, then they use the other known scaled identity matrix $\hat{B}_{k+1} = \hat{\tau}_k^0 I$. Otherwise, they define positive definite $\hat{B}_{k+1}$ as an approximate solution of the constrained optimization problem

$$\min_{\hat{B}^+} \psi(\hat{B}^+) = \operatorname{tr}(\hat{B}^+) - \ln(\det(\hat{B}^+)) \tag{3.11}$$

$$\text{s.t. } \delta_k^T\hat{B}^+\delta_k = \delta_k^T\gamma_k, \hat{B}^+ \text{diagonal,} \tag{3.12}$$

assuming $\hat{B}^+$ is positive definite and approximating the Lagrange multiplier by a reasonable value of ω_k. Since the ψ function is useful for deriving the BFGS formula (Byrd and Nocedal [14]), it is expected that the first case in (3.8) would work well. In practice, this works a little better than (3.7) and slightly worse than the diagonal BFGS update (2.5) (see Section 5 for details).

Andrei [15] considers the possibility of defining a diagonal quasi-Newton Hessian approximation (say, $\hat{B}_{k+1}$) without updating a matrix by enforcing the quasi-Newton condition $\hat{B}_{k+1}\delta_k = \gamma_k$ which he writes as follows:

$$\hat{B}_{k+1}S_k = Y_k, \tag{3.13}$$

where $S_k = \operatorname{diag}[\delta_k^{(1)}, \dots, \delta_k^{(n)}]$ and $Y_k = \operatorname{diag}[\gamma_k^{(1)}, \dots, \gamma_k^{(n)}]$. Because, in general, equation (3.13), subject to $\hat{B}_{k+1}$ positive definite, cannot be solved exactly, for $i = 1,2,\dots,n$, the author suggests the choice

$$\hat{B}_{k+1}^{(i)} = \begin{cases} \frac{\gamma_k^{(i)}}{\delta_k^{(i)}}, & \text{if } \frac{\gamma_k^{(i)}}{\delta_k^{(i)}} \geqslant \epsilon_2 \\ 1, & \text{otherwise,} \end{cases} \tag{3.14}$$

where $\epsilon_2 > 0$. He reports that the corresponding algorithm performs better than certain diagonal quasi-Newton algorithms.

We note that the second case in (3.14) has the drawback of losing the details of $\hat{B}_k$. Therefore, we suggest the following modified diagonal choice

$$\hat{B}_{k+1}^{(i)} = \begin{cases} \frac{\gamma_k^{(i)}}{\delta_k^{(i)}}, & \text{if } \epsilon_2 \leq \frac{\gamma_k^{(i)}}{\delta_k^{(i)}} \leqslant \frac{1}{\epsilon_3}, \\ \hat{B}_k^{(i)}, & \text{otherwise,} \end{cases} \tag{3.15}$$

where $\epsilon_3 > 0$, which maintains the positive definite property. In practice, this choice works better than (3.14) (see Section 5 for details). We let $\epsilon_3 = 10^{-14}$ be small enough so that the second inequality in (3.15) was always satisfied in our experiment.

Although other useful choices for the second case in (3.15) are possible (see Al-Siyabi [9]), we do not consider them here because we can improve the choice (3.15) as follows. Since the above choices for the Hessian approximation $\hat{B}_{k+1}$ do not satisfy the quasi-Newton condition (1.5), we suggest updating this matrix by any quasi-Newton formula (in particular, the BFGS update). Hence, replacing the updated matrix by its diagonal, as in (2.5), we obtain the positive definite improved diagonal BFGS update

$$\hat{B}_{k+1} = \operatorname{diag}[\operatorname{bfgs}(\hat{B}_{k+1}^*, \delta_k, \gamma_k)], \tag{3.16}$$

where $\hat{B}_{k+1}^*$ denotes any of the above $\hat{B}_{k+1}$.

Finally, the corresponding algorithm for the above diagonal quasi-Newton updates can be outlined along the lines of Algorithm 2.1, where the difference among the above updates occurs in Step 4 for defining $\hat{B}_{k+1}$ (given in particular by (3.15)– (3.16), unless otherwise stated). We will show that this improves the performance of choice (3.15) in Section 5.

4. Convergence analysis

In this section, we study the convergence property of our proposed diagonal Hessian approximations. Since they are maintained positive definite, the descent condition

$$s_k^T g_k < 0 \tag{4.1}$$

is satisfied for all k. Before presenting the convergence property, we first state the following standard assumption.

Assumption 4.1.

a) Consider the level set $\Omega = \{x \in \Re^n : f(x) \le f(x_1)\}$ and let $\widetilde{\Omega}$ be an open set containing Ω.
b) The objective function $f(x)$ is bounded and continuously differentiable in $\widetilde{\Omega}$.
c) The gradient $g(x)$ is Lipschitz continuous on $\widetilde{\Omega}$, that is, there exist a constant $\mathrm{L} > 0$ such that

$$\| g(x) - g(\tilde{x}) \| \le L \| x - \tilde{x} \|, \ \forall x, \ \tilde{x} \in \widetilde{\Omega}. \tag{4.2}$$

Similar to the well-known result of Zoutendijk [16], we state the following result.

Theorem 4.1. Suppose Assumption 4.1 holds. Consider the iterations of the form (1.3), with x_1 being any starting point, the search direction s_k being defined such that the descent condition (4.1) holds and the steplength α_k satisfies the
Wolfe-Powell conditions (1.2). Then, the so-called Zoutendijk condition

$$\sum_{k=1}^{\infty} \frac{\left(s_k^T g_k\right)^2}{\|s_k\|^2} < \infty \tag{4.3}$$

is obtained.

Proof. Similar to many analyses (see, for example, Nocedal and Wright [3]), we state the following proof (for complete illustration). Rearranging the second Wolfe-Powell condition in (1.2) and using the Lipschitz condition (4.2), it follows that

$$L\|s_k\|\|x_{k+1} - x_k\| \ge s_k^T\big(g(x_{k+1}) - g(x_k)\big) \ge (\sigma_1 - 1) s_k^T g_k.$$

Substituting $x_{k+1} = x_k + \alpha_k s_k$ and using (4.1), we obtain:

$$\alpha_k \ge \frac{1 - \sigma_1}{L} \frac{|s_k^T g_k|}{\|s_k\|^2}.$$

Using this result, we rewrite the first Wolfe-Powell condition in (1.2) as follows:

$$f_k - f_{k+1} \ge \frac{\sigma_0(1 - \sigma_1)}{L} \frac{(s_k^T g_k)^2}{\|s_k\|^2}.$$

By summing this expression over k and using the assumed bound on the f_k, we obtain the Zoutendijk condition (4.3).

To obtain the global convergence result for Algorithm 2.1, we assume the condition number of the positive definite diagonal Hessian approximate $\hat{B}_k$ to be uniformly bounded, that is, there is a constant M such that

$$\kappa(\hat{B}_k) = \frac{\lambda_1}{\lambda_n} \le M, \ \forall k, \tag{4.4}$$

where λ_1 and λ_n are the largest and smallest eigenvalues of $\hat{B}_k$.

Theorem 4.2. Suppose that f satisfies Assumption 4.1. Let x_1 be a starting point and $\hat{B}_1$ be a positive definite diagonal matrix. Consider Algorithm 2.1 with $\epsilon = 0$ in Step 2, $\hat{B}_{k+1}$ in Step 4 defined such that condition (4.4) holds and that Step 1 defines the steplength α_k such that the Wolfe-Powell conditions (1.2) hold. Then, the algorithm converges globally, that is,

$$\lim_{k \to \infty} \|g_k\| = 0. \tag{4.5}$$

Proof. Substituting $s_k = -\hat{B}_k^{-1} g_k$ into the Zoutendijk condition (4.3), we obtain:

$$\frac{1}{M} \sum_{k=1}^{\infty} \|g_k\|^2 \le \sum_{k=1}^{\infty} \frac{\left(s_k^T \hat{B}_k s_k\right)\left(g_k^T \hat{B}_k^{-1} g_k\right)}{\|s_k\|^2} < \infty,$$

where M is given as in (4.4). Hence, the limit (4.5) is obtained.

5. Numerical results

In this section, we study the performance of our proposed methods on a set of standard unconstrained optimization test problems. All the methods are implemented as in Algorithm 2.1, differing only in Step 4 for defining the Hessian approximate $\hat{B}_{k+1}$ by the choices (2.5), (2.12), (3.2), (3.7) (with $\epsilon_1 = 10^{-6}$), (3.8), (3.14) (with $\epsilon_2 = 10^{-2}$ as in Andrei [15]), and (3.15) (with $\epsilon_2 = 10^{-2}$ and $\epsilon_3 = 10^{-14}$). These choices (referred to as L1, L2, ..., and L7), except for L7, have been proposed by Gilbert and Lemaréchal [8], Nazareth [10], Zhu et al. [11], Sim et al. [13] and Andrei [15], respectively. The choice L7 defines our proposed modification of L6. Because the L2 update defines the inverse Hessian approximate $\widehat{H}_{k+1}$, steps 0, 4 and 5 are used with $\widehat{B}_j^{-1}$ replaced by $\widehat{H}_j$, for $j = 1, k$ and $k + 1$, respectively.

In all algorithms, we consider the followings. For Step 0, we choose $\widehat{B}_1 = I$ and $\epsilon = 10^{-7}$. For Step 1, we calculate a value of the steplength α_k such that the strong Wolfe-Powell conditions

$$f_{k+1} \leq f_k + \sigma_0 \alpha_k s_k^T g_k, \quad |s_k^T g_{k+1}| \leq -\sigma_1 s_k^T g_k \tag{5.1}$$

hold, using the usual values of $\sigma_0 = 10^{-4}$ and $\sigma_1 = 0.9$, which imply the Wolfe-Powell conditions (1.2). We used the MATLAB line search routine 'lswpc' of Al-Baali, which is essentially written with slight differences in Fortran by Fletcher. It is based on using quadratic and cubic interpolations for estimating a value of the steplength α_k. It also guarantees finding a positive value of α_k in a finite number of operations (see Al-Baali and Fletcher [17] and Fletcher [1]). We stopped the run when either

$$\|g_k\| \leq 10^{-7} \max\{\|g_1\|, 1\}, \qquad f_k - f_{k+1} \leq 10^{-14},$$

or the number of line searches reached 10^5.

All codes were written in MATLAB R2017b and the runs were made on a CPU processor with Intel(R) Core (TM) i7-(2.7 GHz) and 16.0 GB RAM memory.

The test problems were selected from the collection of Andrei [18] (which belong to the CUTEst collection established by Gould et al. [19], Himmelblau [20] and Moré et al. [21]). We picked 84 test problems (as given in Table 1). For certain extended test problems (e.g., Extended Rosenbrock), increasing the number of variables does not increase the number of line searches, function and gradient evaluations required to solve the problems (see for example Al-Baali [22]). To avoid this occurrence, the author modifies the standard starting point $\bar{x}_1$ to $\bar{\bar{x}}_1$, where

$$\bar{\bar{x}}_1^{(i)} = \bar{x}_1^{(i)} + \tfrac{1}{i+1}, \tag{5.2}$$

for $i = 1, \dots, n$. We used all 84 test functions with the standard starting points and their modifications (5.2) for n = 900, 9000 and 27000 to define three sets of test problems. Each set (referred to as Set1, Set2 and Set3, respectively) consists of 168 test problems. We also consider all the test problems as a single 504 test problem (referred to as Set4).

To study the behaviour of the above algorithms, we compared the numerical results required to solve the tests, using the performance profiles of Dolan and Moré [23] based on the numbers of line searches (#ls), function evaluations (#fun) and gradient evaluations (#gra) as well as the cpu time in seconds, required to solve the test problems. The Dolan- Moré performance profile can be briefly described as follows. It illustrates the relative solvers performance of the solvers on a set of test problems in terms of #ls (similarly for #fun, #gra and cpu time). In general, $P_M(\tau)$, the fraction of problems with performance ratio $\tau \geq 0$, is defined by

$$P_M(\tau) = \frac{\text{number of problems where } log_2(\tau_{p,M}) \leq \tau}{\text{total number of problems}}. \tag{5.3}$$

Here, $\tau_{p,M}$ is the performance ratio of #ls required to solve problem p by the M method to the lowest #ls required to solve problem p. The ratio $\tau_{p,M}$ is set to ∞ (or some large number) if the M method fails to solve problem p. The values of $P_M(\tau)$ at $\tau = 0$ gives the percentage of test problems for which the method M performs to be best and the value for τ large enough is the percentage of test problems that the M method can solve. Thus, a solver with high values of $P_M(\tau)$ or one with corresponding figure located at the top right performs better than the ones located at lower levels.

Applying the above algorithms to the four sets, Set1, Set2, Set3 and Set4, of test problems, we obtained some numerical results. Their comparisons are given in Figures 1–4, respectively, with respect to #ls, #fun, #gra and cpu time for each figure. We observe that L7 appears to be the best, while showing to be a little better than L1, L2, L4 and L6, with L3 and L5 being a little worse than the other methods.

To give another fair and useful comparison which shows the percentage improvement or worsening of the algorithms, we also considered the comparison rule of Al-Baali (see, e.g., Al-Baali [24] and essentially Al-Baali [25]). To compare two methods (say, M1 and M2) with respect to #ls (similarly for #fun, #gra and cpu time), the author proposes the average ratio measure of the form:

$$r = \tfrac{1}{t} \textstyle\sum_{i=1}^{t} r_i, \tag{5.4}$$

where t is the number of test problems and

$$r_i = \begin{cases} \frac{p_i}{q_i}, & \text{if } p_i \le q_i \\ 2 - \frac{q_i}{p_i}, & \text{if } p_i > q_i \end{cases} \tag{5.5}$$

with p_i and q_i denoting #ls required to solve problem i by the M1 and M2 methods, respectively. If only M1 or only M2 failed to solve the problem, we set $r_i = 2$ and $r_i = 0$, respectively. If both M1 and M2 methods either failed or converged to two different local solutions, for some test problem i then we set $r_i = 1$. The average ratio r in (5.4) always falls in the interval $[0,2]$. A value of $r \le 1$ indicates that M1 is better than M2 by $100(1 - r)$%. Otherwise, when $r > 1$, M1 is worse than M2 (or M2 is better than M1) by $100(1 - r)$% (for more details on this measurement ratio, see Al-Baali [24], for instance).

Using the same numerical results used to obtain the comparison Figures 1- 4 for Set $i, i = 1, \ldots, 4$, we applied the above average ratio measure to compare L1, L2, ..., L6 versus L7 and obtained Tables 2–5, using both the starting points for n = 900, 9000, 27000 and all 3 values of n, respectively. Since we have $r > 1$ in all cases, it is clear that L7 gives the best performance, it being at least 10% better than L1, L2, L4 and L6, and more than 48% better than L3 in terms of #ls, #fun #gra and cpu time. Thus, these observations agree with those in Figures 1–4. We observe that the performance of L6 improves as n increases in comparison with L1, L2, and L4. We also observe that L1, L2 and L4 have nearly identical performances in terms of all measurements, whereas L3 is the worst among all the algorithms.

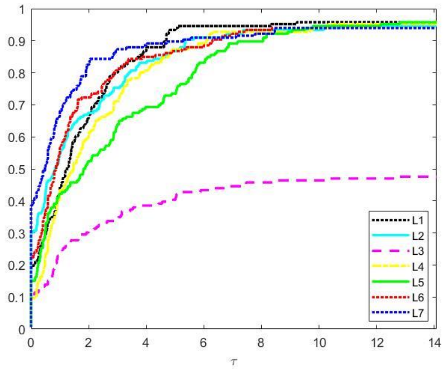
Since the L3, L4, L5, L6 and L7 methods do not consider imposing the quasi-Newton condition (1.5), like that of L1 and L2, we suggest using (3.16) with $\hat{B}^*_{k+1}$ given by (3.2), (3.7), (3.8), (3.14) and (3.15), which define the former five methods, respectively. We compared the corresponding algorithms (referred to as L3a, L4a, L5a, L6a and L7a, respectively) and observed that each Lia, for $i = 3, \ldots, 7$, performs better than Li and that the performance of L7a remains the best (see Al-Siyabi [9]). Thus, we present only the comparison of L7a versus L7 as shown in Figure 5 and Table 6 for Set4 of test problems. We notice that L7a outperforms L7 in terms of all measurements. Moreover, L7a performs better than L7 by at least 6% in terms of #ls, #fun, #gra and cpu time, which provides further illustration of the comparison shown in Figure 5. We also observe from the combination of Figures 1–4 and Figure 5 as well as Tables 2–5 and Table 6 that L7a performs substantially better than L6 with at least 15% improvement in terms of #ls, #fun and #gra and 20% in terms of cpu time.

Table 1. List of test functions.

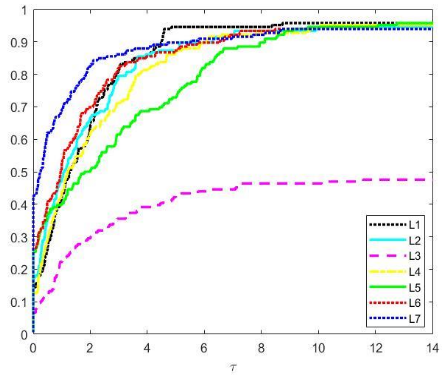
No.	Function’s Name	No.	Function’s Name
1	Extended Freudenstein & Roth	43	ARGLINB
2	Extended Trigonometric	44	ARWHEAD
3	Extended Rosenbrock	45	NONDIA
4	Generalized Rosenbrock	46	NONDQUAR
5	Extended White & Holst	47	BQDRTIC
6	Extended Beale	48	EG2
7	Extended Penalty	49	DIXMAANA
8	Perturbed Quadratic function	50	DIXMAANB
9	Raydan 1	51	DIXMAANC
10	Raydan 2	52	DIXMAAND
11	Diagonal 1	53	DIXMAANE
12	Diagonal 2	54	DIXMAANF
13	Diagonal 3	55	DIXMAANG
14	Hager	56	DIXMAANH
15	Generalized Tridiagonal 1	57	DIXMAANI
16	Extended Tridiagonal 1	58	DIXMAANJ
17	Extended TET (Three exponential terms)	59	DIXMAANK
18	Generalized Tridiagonal 2	60	DIXMAANL
19	Diagonal 4	61	Broyden Tridiagonal
20	Diagonal 5	62	Almost Perturbed Quadratic
21	Extended Himmelblau	63	Staircase 1

Table 1. continued

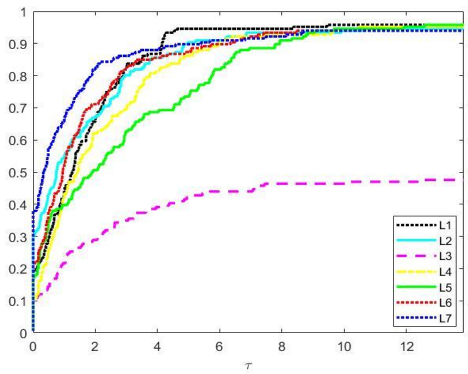
22	Generalized White & Holst	64	Staircase 2
23	Generalized PSC1	65	LIARWHD
24	Extended PSC1	66	ENGVAL1
25	Extended Powell	67	EDENSCH
26	Full Hessian FH1	68	CUBE
27	Full Hessian FH2	69	NONSCOMP
28	Extended BD1 (Block Diagonal)	70	QUARTC
29	Extended Maratos	71	Diagonal 6
30	Extended Cliff	72	SIQUAD
31	Perturbed quadratic diagonal	73	Extended DENSHNB
32	Extended Wood	74	Extended DENSHNF
33	Extended Hiebert	75	COSINE
34	Quadratic QF1	76	Generalized Quartic
35	Extended quadratic penalty QP1	77	Diagonal 7
36	Extended quadratic penalty QP2	78	Diagonal 8
37	Quadratic QF2	79	Full Hessian FH3
38	Extended quadratic exponential EP1	80	SINCOS
39	Extended Tridiagonal 2	81	Diagonal 9
40	FLETCHR	82	HIMMELBG
41	BDQRTIC	83	HIMMELH
42	TRIDIA	84	INDEF



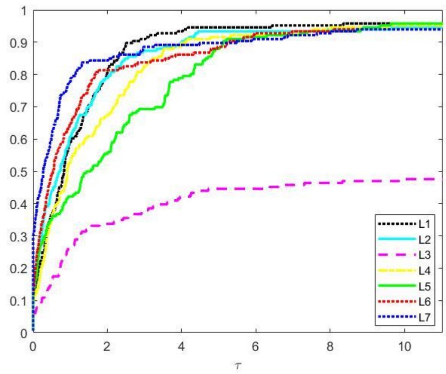
(a) Number of Line Searches.



(b) Number of Function Evaluations.



(c) Number of Gradient Evaluations.

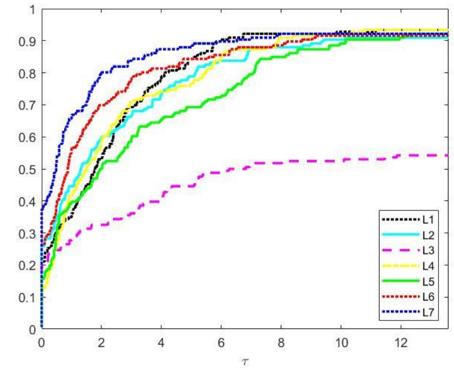


(d) CPU Times.

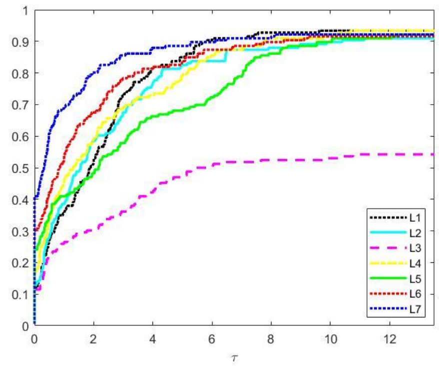
Figure 1. Comparison among L1, L2, L3, L4, L5, L6 and L7, for Set1; $n = 900$.

Table 2. Average ratios r for methods versus L7, for Set1; $n = 900$.

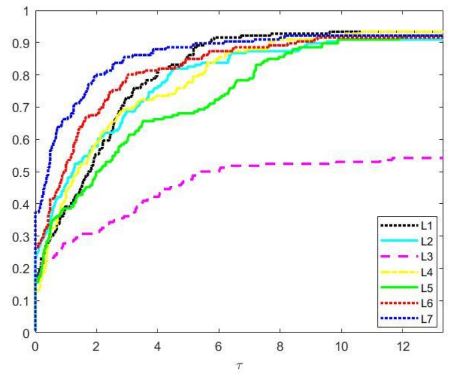
Method /Measure	#ls	#fun	#gra	cpu
L1	1.153	1.203	1.155	1.130
L2	1.074	1.158	1.0714	1.097
L3	1.598	1.622	1.596	1.597
L4	1.207	1.211	1.209	1.173
L5	1.214	1.232	1.211	1.224
L6	1.124	1.144	1.129	1.077



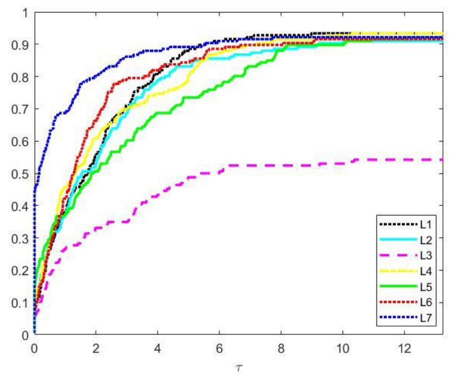
(a) Number of Line Searches.



(b) Number of Function Evaluations.



(c) Number of Gradient Evaluations.

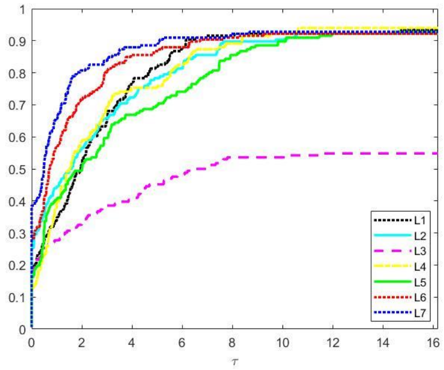


(d) CPU Times.

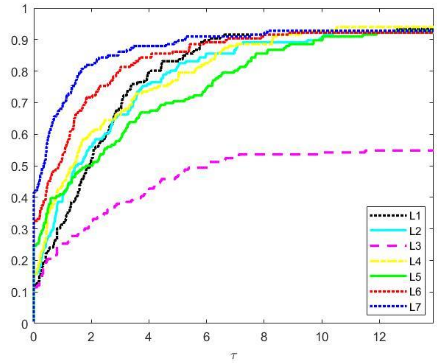
Figure 2. Comparison among L1, L2, L3, L4, L5, L6 and L7, for Set2; $n = 9000$.

Table 3. Average ratios r for methods versus L7, for Set2; $n = 9000$.

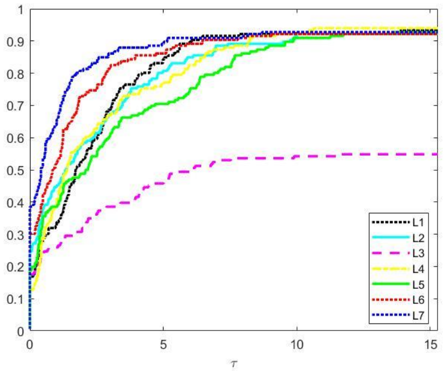
Method/ Measure	#ls	#fun	#gra	cpu
L1	1.190	1.259	1.205	1.281
L2	1.136	1.229	1.150	1.268
L3	1.479	1.542	1.506	1.566
L4	1.197	1.197	1.201	1.240
L5	1.237	1.249	1.250	1.290
L6	1.102	1.134	1.121	1.235



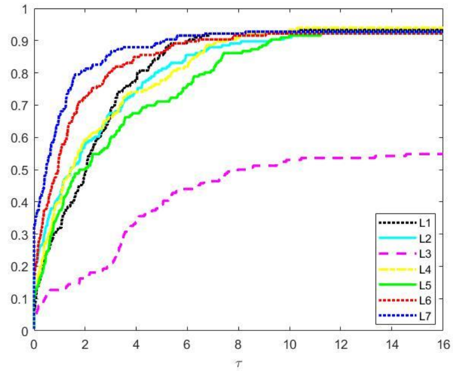
(a) Number of Line Searches.



(b) Number of Function Evaluations.



(c) Number of Gradient Evaluations.

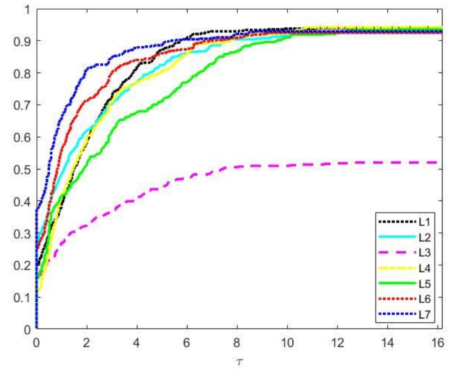


(d) CPU Times.

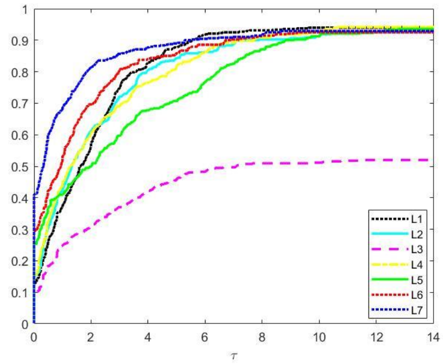
Figure 3. Comparison among L1, L2, L3, L4, L5, L6 and L7, for Set3; $n = 27000$.

Table 4. Average ratios r for methods versus L7, for Set3; $n = 27000$.

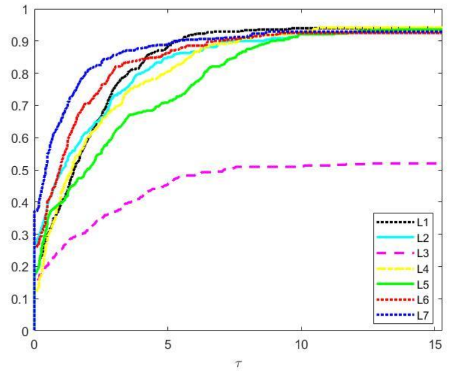
Method / Measure	#ls	#fun	#gra	Cpu
L1	1.239	1.311	1.254	1.290
L2	1.137	1.232	1.152	1.166
L3	1.482	1.556	1.507	1.671
L4	1.228	1.222	1.225	1.223
L5	1.231	1.243	1.238	1.280
L6	1.079	1.114	1.095	1.096



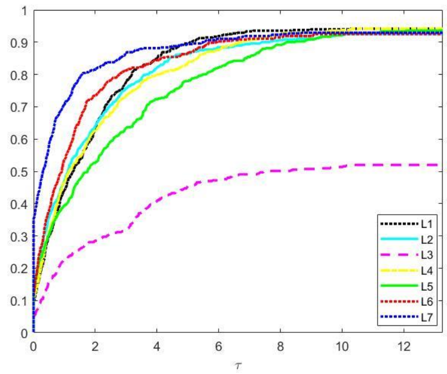
(a) Number of Line Searches.



(b) Number of Function Evaluations.



(c) Number of Gradient Evaluations.

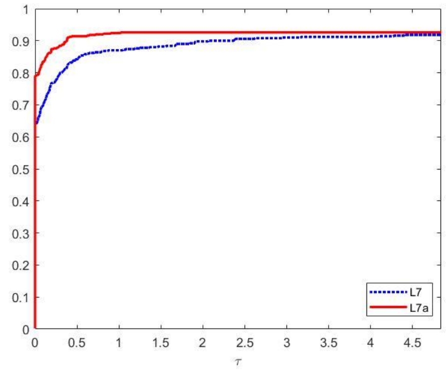


(d) CPU Times.

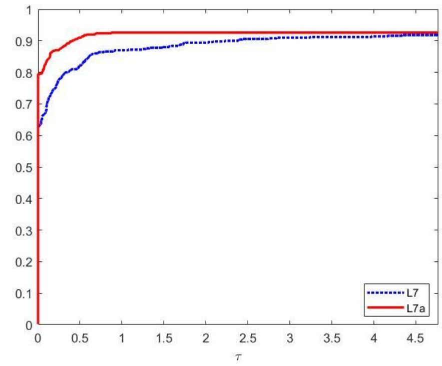
Figure 4. Comparison among L1, L2, L3, L4, L5, L6 and L7, for Set4; $n \in [900, 9000, 27000]$.

Table 5. Average ratios r for methods versus L7, for Set4; $n \in [900, 9000, 27000]$.

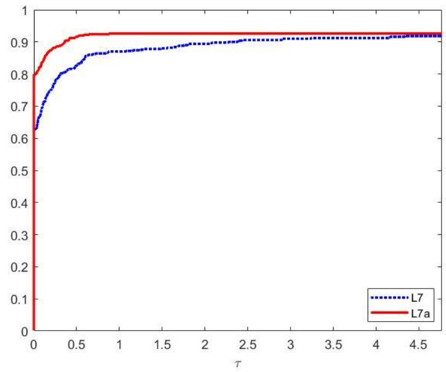
Method /Measure	#ls	#fun	#gra	cpu
L1	1.211	1.285	1.229	1.267
L2	1.126	1.222	1.141	1.191
L3	1.487	1.556	1.512	1.585
L4	1.203	1.200	1.204	1.217
L5	1.223	1.235	1.233	1.262
L6	1.087	1.118	1.101	1.145



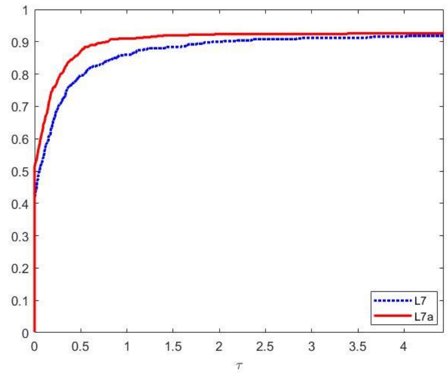
(a) Number of Line Searches.



(b) Number of Function Evaluations.



(c) Number of Gradient Evaluations.



(d) CPU Times.

Figure 5. Comparison among L7 and L7a, for Set4; $n \in [900, 9000, 27000]$.

Table 6. Average ratios r for L7a versus L7, for Set4; $n \in [900, 9000, 27000]$.

Method / Measure	#ls	#fun	#gra	cpu
L7a	0.942	0.933	0.934	0.943

6. Conclusion

We first studied some diagonal Hessian approximation methods for large-scale unconstrained optimization, then presented new diagonal Hessian approximations and established their global convergence. Based on extensive numerical experiments, we observed that a number of our algorithms were more efficient and more robust than several similar available methods. Two of the proposed methods require storing only a few vectors while sharing certain desirable features of the quasi-Newton methods.

Conflict of interest

The authors declare no conflict of interest.

Acknowledgment

We thank the two anonymous referees for careful reading of the paper and for many constructive comments and suggestions. The first author would like to thank the Oman Ministry of Education for the award of a scholarship.

References

1. Fletcher, R. Practical Methods of Optimization. John Wiley & Sons, 2013.
2. Andrei, N. Nonlinear Conjugate Gradient Methods for Unconstrained Optimization. *In Springer Optimization and Its Applications*, 2020, Volume **158**.
3. Nocedal, J., and Wright, S. Numerical Optimization. Springer Science & Business Media, 2006.
4. Buckley, A. and LeNir, A. QN-like variable storage conjugate gradients. *Mathematical Programming*, 1989, **27(2)**, 155-175.
5. Fletcher, R. Low storage methods for unconstrained optimization. *In Computational Solution of Nonlinear Systems of Equations, E.L.Allgower and K.Georg (Eds.), Lectures in Applied Mathematics,* Vol.**26**, AMS Publications, Providence, RI. *Mathematical Programming*, 1990, **91(2)**, 201-213.
6. Liu, D.C. and Nocedal, J. On the limited memory BFGS method for large scale optimization, 1989. *Mathematical Programming*, **45(1-3)**, 503-528.
7. Shanno, D.F. Conjugate gradient methods with inexact searches. *Mathematics of Operations Research,* 1978, **3(3)**, 244-256.
8. Gilbert, J.C. and Lemaréchal, C. Some numerical experiments with variable-storage quasi-Newton algorithms. *Mathematical Programming*, 1989, **45(1-3)**, 407-435.
9. Al-Siyabi, A. Efficient Methods for Large-Scale Nonlinear Least-Squares Unconstrained Optimization. Ph.D. thesis, Sultan Qaboos University, Muscat, Oman, 2021.
10. Nazareth, J. If quasi-Newton then why not quasi-Cauchy. *SIAG/Opt Views-and-news*, 1995, **6**, 11-14.
11. Zhu, M., Nazareth, J.L., and Wolkowicz, H. The quasi-Cauchy relation and diagonal updating. *SIAM Journal on Optimization*, 1999, **9(4)**, 1192-1204.
12. Oren, S.S., and Luenberger, D.G. Self-scaling variable metric (SSVM) algorithms, part I: Criteria and sufficient conditions for scaling a class of algorithms, *Management Science*, 1974, **20**, 845-862.
13. Sim, H.S., Leong, W.J., and Chen, C.Y. Gradient method with multiple damping for large-scale unconstrained optimization. *Optimization Letters*, 2019, **13(3)**, 617-632.
14. Byrd, R.H. and Nocedal, J.A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. *SIAM Journal on Numerical Analysis*, 1989, **26(3)**, 727-739.
15. Andrei, N. A new diagonal quasi-Newton updating method with scaled forward finite differences directional derivative for unconstrained optimization. *Numerical Functional Analysis and Optimization*, 2019, **40(13)**, 1467-1488.
16. Zoutendijk, G. Nonlinear programming computational method. In J. Abadie (Ed.), *Integer and Nonlinear Programming, North-Holland, Amsterdam*, 1970, pp. 37-86.
17. Al-Baali, M. and Fletcher, R. An efficient line search for nonlinear least squares. *Journal of Optimization Theory and Applications*, 1986, **48(3)**, 359-377.
18. Andrei, N. An unconstrained optimization test functions collection. *Adv. Model. Optim*., 2008, **10(1)**, 147-161.
19. Gould N.I., Orban D., Toint Ph. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 2015, **60(3)**, 545-557.
20. Himmelblau, D. Applied Nonlinear Programming McGraw-Hill, New York, 1972.
21. Moré, J.J., Garbow, B.S. and Hillstrom, K.E. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software (TOMS)*, 1981, **7**, 17-41.
22. Al-Baali, M. Improved Hessian approximations for the limited memory BFGS method. *Numerical Algorithms*, 1999, **22(1)**, 99-112.
23. Dolan, E.D. and Moré, J.J. Benchmarking optimization software with performance profiles, 2002.
24. Al-Baali, M. Numerical experience with a class of self-scaling quasi-Newton algorithms. *Journal of Optimization Theory and Applications*, 1998, **96(3)**, 533-553.
25. Al-Baali, M. A rule for comparing two methods in practical optimization. *Technica1 Report 119, Dipartimento di Sistemi, Universita della Calabria, Italy*, 1991.

Received 6 May 2021
Accepted 6 July 2021