Les Cahiers du GERAD ISSN: 0711–2440

Diagonal quasi-Newton updating methods for large-scale nonlinear least squares

A. Al-Siyabi, M. Al-Baali

G–2024–82

December 2024

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Diagonal quasi-Newton updating methods for large-scale nonlinear least squares

Ahmed Al-Siyabi [a]

Mehiddin Al-Baali [b, c]

[a] *University of Technology and Applied Sciences, Nizwa, Oman*

[b] *Department of Mathematics and Statistics, Sultan Qaboos University, Muscat, Oman*

[c] *GERAD, Montréal (Qc), Canada*

ahmed.alsiyabi@utas.edu.om
albaali@squ.edu.om

December 2024
Les Cahiers du GERAD
G–2024–82

Abstract : We propose some diagonal Hessian approximations of the least-squares nonlinear function. We use suitable diagonal matrices that carry some information of the quasi-Newton condition and/or the specific structure of the least-squares function. It is shown that the proposed methods perform well when used to solve a collection of standard test problems.

Keywords : Unconstrained optimization, large-scale problem, nonlinear least-squares problem, diagonal Hessian approximations, the quasi-Newton condition, the BFGS update

1 Introduction

We consider solving the nonlinear least-squares unconstrained optimization problem

$$\begin{aligned}\min_{x\in\Re^n} f(x) &= \frac{1}{2}\sum_{i=1}^{l}[r_i(x)]^2 = \frac{1}{2}r(x)^T r(x)\\ &= \frac{1}{2}\|r(x)\|^2,\end{aligned} \tag{1}$$

where $r_i\colon \Re^n \to \Re$, $i = 1,\ldots,l$, are smooth functions, $l \geqslant n$, $r = (r_1,...,r_l)^T$ and $\|\cdot\|$ denotes the Euclidean norm. It is assumed that n is large so that a matrix cannot be stored explicitly, but the Jacobian matrix $A(x) = \nabla r^T$ can be stored implicitly so that the products Au and A^Tv, for any vectors $u \in \Re^l$ and $v \in \Re^n$, can be computed. These products can be computed efficiently (see for example Al-Baali [4], and Bouaricha and Moré [13]). The gradient of f can be represented as

$$g(x) = \nabla f(x) = \sum_{i=1}^{l} r^{(i)}(x)\nabla r^{(i)}(x) = A(x)r(x), \tag{2}$$

denoted at x_k by

$$g_k = A_k r_k, \tag{3}$$

where $r_k = r(x_k), A_k = A(x_k)$, etc. The Hessian as

$$G(x) = \nabla^2 f(x) = A(x)\ A(x)^T + \sum_{i=1}^{l} r^{(i)}(x)\ \nabla^2 r^{(i)}(x), \tag{4}$$

referred at x_k by

$$G_k = A_k A_k^T + \sum_{i=1}^{l} r_k^{(i)}\ \nabla^2 r_k^{(i)}. \tag{5}$$

Letting x^* denotes a solution to problem (1), a value of the residual $r(x^*) = 0$ reduces the Hessian matrix $G(x^*)$ to the Gauss-Newton (GN) Hessian $A(x^*)A(x^*)^T$ (in this case, the problem is referred to as zero residual) (for further details, see Fletcher [15], for instance).

Thus, the Hessian G_k can be approximated by the GN Hessian $A_kA_k^T$, for zero and small residual r_k. This matrix has been modified to the Levenberg-Marquardt (LM) Hessian, that is

$$B_k = A_k A_k^T + \lambda_k I, \tag{6}$$

where $\lambda_k \geq 0$ is the LM parameter (see e.g. Fan and Yuan [14] and Nocedal and Wright [24]). We note that the second order part of the Hessian (5) is ignored in both approximations. As a result, the corresponding methods are expected to work well when applied to small residual problems. Nevertheless, when solving sufficiently large residual problems, the methods may work badly (for further details, see Al-Baali and Fletcher [5] and Fletcher [15], for instance).

For solving nonlinear least-squares problems, most of the classical and modified algorithms require the computation of the gradient and the Hessian matrix or its approximation (see Fletcher [15] and Nocedal and Wright [24], for instance). In practice, the Hessian matrix is too expensive to compute and store, especially when considering large-scale problems. Several studies have been made to improve the algorithms for solving the nonlinear least-squares problems (see, for example, Al-Baali [4], Al-Baali and Fletcher [5], Al-Baali and Fletcher [6], Dehghani and Mahdavi-Amiri [11], Fletcher and Xu [16], Lukšan and Vlček [20] and the reference therein). For solving large-scale nonlinear least-squares problems, Mohammad and Santos [21] define a diagonal Hessian approximation by using the structure of least-squares function (1). The authors demonstrate that the proposed algorithm is more

efficient and robust than some published methods. As an extension to this technique, we will consider several proposed methods for maintaining the Hessian approximation B_k diagonal. We first consider some quasi-Newton methods which are defined iteratively in the following way. For given x_1, the search direction is represented by

$$B_k s_k = -g_k, \tag{7}$$

where B_k is some symmetric and positive definite matrix that approximates the Hessian G_k which ensures that s_k is a descent direction (i.e., $s_k^T g_k < 0$) so that a positive steplength α_k, which minimizes $f(x_k+\alpha s_k)$, (exactly or approximately) along s_k from a point x_k, must exists (see for example Al-Baali and Fletcher [6]). In practice, α_k is usually chosen to satisfy the Wolfe-Powell conditions

$$f_{k+1} \leqslant f_k + \sigma_0 \alpha_k g_k^T s_k, \quad g_{k+1}^T s_k \geqslant \sigma_1 g_k^T s_k, \tag{8}$$

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

$$x_{k+1} = x_k + \alpha_k s_k. \tag{9}$$

Initially B_1 is chosen positive definite (usually, $B_1 = I$, the identity matrix). To define the other Hessian approximations B_{k+1}, for $k \geq 1$, many quasi-Newton formulae have been presented for updating B_k. It is considered that Broyden-Fletcher-Goldfarb-Shanno (BFGS) is the most popular quasi-Newton update. It is provided by

$$B_{k+1} = \text{bfgs}(B_k, \delta_k, \gamma_k), \tag{10}$$

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

$$\text{bfgs}(B, \delta, \gamma) = B - \frac{B\delta\delta^T B}{\delta^T B \delta} + \frac{\gamma\gamma^T}{\delta^T \gamma}, \tag{11}$$

$$\delta_k = x_{k+1} - x_k \tag{12}$$

and

$$\gamma_k = g_{k+1} - g_k. \tag{13}$$

Often, in practical implementation, instead of solving the system of equations (7), it is preferable to compute the search direction for the next iteration by

$$s_{k+1} = -H_{k+1} g_{k+1} \tag{14}$$

where $H_k = B_k^{-1}$ and the inverse BFGS update is

$$H_{k+1} = \text{bfgs}^{-1}(H_k, \delta_k, \gamma_k), \tag{15}$$

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

$$\text{bfgs}^{-1}(H, \delta, \gamma) = H + \left(1 + \frac{\gamma^T H \gamma}{\delta^T \gamma}\right) \frac{\delta\delta^T}{\delta^T \gamma} - \frac{\delta\gamma^T H + H\gamma\delta^T}{\delta^T \gamma}. \tag{16}$$

We note that the product $\text{bfgs}^{-1}(H, \delta, \gamma)u$, for any vector u, can be calculated in terms of only the vector pair $\{\delta, \gamma\}$ without storing matrices explicitly, assuming H is stored implicitly.

The following is a breakdown of the structure of this paper. In the next section, we consider several proposals for maintaining B_k diagonal, based on quasi-Newton feature and/or using the nonlinear least-squares features. In Section 3, we discuss the convergence property, while Section 4 discusses some numerical results. It is demonstrated that the proposed methods improve the performance of diagonal matrices significantly. Finally, Section 5 concludes the paper.

2 Diagonal Hessian approximations

It is worth considering the particular low storage method of Gilbert and Lemaréchal [17], who suggest using the BFGS formula (10) with B_k replaced by a diagonal matrix $\hat{B}_k$ and use the diagonal of the updated matrix to obtain a new approximation of G_k as follows:

$$\hat{B}_{k+1} = \text{diag}\left[\text{bfgs}(\hat{B}_k, \delta_k, \gamma_k)\right]. \tag{17}$$

On substituting (10)–(11), this diagonal update can be written as follows:

$$\hat{B}_{k+1} = \hat{B}_k - \frac{\text{diag}[\hat{B}_k\delta_k(\hat{B}_k\delta_k)^T]}{\delta_k^T\hat{B}_k\delta_k} + \frac{\text{diag}[\gamma_k\gamma_k^T]}{\delta_k^T\gamma_k}. \tag{18}$$

We note that this formula requires the storage of only three vectors (δ_k, γ_k and that of the diagonal matrix $\hat{B}_k$). Similarly, we can apply the diagonal technique to the inverse BFGS update (15) to obtain

$$\hat{H}_{k+1} = \text{diag}\left[\text{bfgs}^{-1}(\hat{H}_k, \delta_k, \gamma_k)\right] \tag{19}$$

which can be expressed as:

$$\hat{H}_{k+1} = \hat{H}_k + \left(1 + \frac{\gamma_k^T\hat{H}_k\gamma_k}{\delta_k^T\gamma_k}\right)\frac{\text{diag}\left[\delta_k\delta_k^T\right]}{\delta_k^T\gamma_k} - 2\frac{\text{diag}\left[(\hat{H}_k\gamma_k)\delta_k^T\right]}{\delta_k^T\gamma_k}. \tag{20}$$

It is worth noting that this formula requires only the storage of three vectors. This diagonal of inverse BFGS update has been suggested by Gilbert and Lemaréchal [17], with a certain positive definite diagonal matrix $\hat{H}_1$. Thus, the updated diagonal matrices are maintained positive definite if the curvature condition $\delta_k^T\gamma_k > 0$ is satisfied. Although the inverse BFGS updated matrix

$$\text{bfgs}(\hat{B}_k, \delta_k, \gamma_k)^{-1} = \text{bfgs}^{-1}(\hat{H}_k, \delta_k, \gamma_k),$$

since $\hat{H}_k = \hat{B}_k^{-1}$, we note that

$$\text{diag}[\text{bfgs}(\hat{B}_k, \delta_k, \gamma_k)]^{-1} \neq \text{diag}[\text{bfgs}^{-1}(\hat{H}_k, \delta_k, \gamma_k)].$$

In practice, the diagonal choice (17) is preferable to (19) (see Al-Siyabi and Al-Baali [9], and Gilbert and Lemaréchal [17], for details).

In the L-BFGS method of Nocedal [23], Al-Siyabi and Al-Baali [8] construct some schemes for defining the basic matrix B_k^0 and reported encouraging numerical results for the nonlinear least-squares problem (1). They recommend extracting some values of the new GN Hessian $M_{k+1} = A_{k+1}A_{k+1}^T$ which is useful for approximating the Hessian (5) for zero and small residual problems. Thus, we consider using the diagonal GN Hessian to define the Hessian approximation as follows

$$\hat{M}_{k+1} = \text{diag}\left[M_{k+1}\right], \tag{21}$$

noting that $\hat{M}_{k+1}^{(i)} = \text{diag}[M_{k+1}]^{(i)} = \|A_{k+1}^T e_i\|^2$, where e_i is $i-$th coordinate vector, for $i = 1, 2, \ldots, n$, which is positive semi-definite (usually positive definite).

Another possible improvement of the diagonal Hessian is to enforce the quasi-Newton property (or nearly so) by adding a suitable diagonal matrix D_{k+1} to the diagonal GN Hessian $\hat{M}_{k+1}$ such that

$$\hat{B}_{k+1} = \hat{M}_{k+1} + D_{k+1} \tag{22}$$

is positive diagonal elements and satisfies the quasi-Newton condition

$$\left(\hat{M}_{k+1} + D_{k+1}\right)\delta_k = \gamma_k. \tag{23}$$

Solving this equation for $D_{k+1} \succ 0$, we suggest the following safeguarded positive definite diagonal matrix

$$\hat{B}_{k+1}^{(i)} = \begin{cases} \dfrac{\gamma_k^{(i)}}{\delta_k^{(i)}}, & \text{if} \quad \epsilon_2 \leq \dfrac{\gamma_k^{(i)}}{\delta_k^{(i)}} \leqslant \dfrac{1}{\epsilon_3}, \ \delta_k^{(i)} \neq 0, \\ \hat{M}_{k+1}^{(i)}, & \text{otherwise}, \end{cases} \tag{24}$$

where ϵ_2 and ϵ_3 are positive parameters. We also consider a modification to this choice by the following:

$$\hat{B}_{k+1}^{(i)} = \begin{cases} \dfrac{\gamma_k^{(i)}}{\delta_k^{(i)}}, & \text{if} \quad \dfrac{\gamma_k^{(i)}}{\delta_k^{(i)}} > \hat{M}_{k+1}^{(i)}, \ \delta_k^{(i)} \neq 0 \\ \hat{M}_{k+1}^{(i)}, & \text{otherwise}, \end{cases} \tag{25}$$

In practice, the first choice performs better than the second one (see Section 4 for details).

Like the Levenberg-Marquardt technique, we replace D_{k+1} in (22)–(23) by $\lambda_k I$, where λ_k is non-negative scalar. We suggest a new diagonal Hessian approximation that satisfies the quasi-Newton condition as follows:

$$\left(\hat{M}_{k+1} + \lambda_k I\right) \delta_k = \gamma_k. \tag{26}$$

Since this equation may not hold exactly for any value of λ_k, we define

$$\lambda_k = \arg\min_{\lambda} \|\hat{M}_{k+1}\delta_k - \gamma_k + \lambda\delta_k\|.$$

Solving this one variable minimization subproblem, it follows that

$$\lambda_k = \frac{\delta_k^T \gamma_k - \delta_k^T \hat{M}_{k+1}\delta_k}{\|\delta_k\|^2}. \tag{27}$$

To safeguard the positive definiteness of the diagonal Hessian approximation $\hat{B}_{k+1}$, we suggest

$$\hat{B}_{k+1} = \begin{cases} \hat{M}_{k+1} + \lambda_k I, & \text{if } \delta_k^T \gamma_k > \delta_k^T \hat{M}_{k+1}\delta_k \\ \hat{M}_{k+1}, & \text{otherwise.} \end{cases} \tag{28}$$

Another possible improvement of this choice is to replace it by the diagonal BFGS update (17) with $\hat{B}_k$ replaced by (28) to obtain

$$\hat{B}_{k+1} = \text{diag}\left[\text{bfgs}(\hat{B}_{k+1}^*, \delta_k, \gamma_k)\right], \tag{29}$$

where $\hat{B}_{k+1}^*$ denotes the diagonal updated matrix (28). In practice, this choice works better than choices (21), (25) and (28) but it is slightly worse than choice (24) (see Section 4 for details). Thus, we suggest updating the diagonal matrix (24) by the diagonal BFGS update again. We obtained a slight improvement (see Al-Siyabi [7] for details).

Now, the corresponding algorithm for the above diagonal updates can be outlined in Algorithm 1.

Notice that the search direction is computed in Steps 0 and 5 without explicitly forming $\hat{B}_k^{-1}$ (i.e., $s_k^{(i)} = -g_k^{(i)}/\hat{B}_k^{(i)}$, for $i = 1, 2, \ldots, n$). Moreover, if the diagonal GN Hessian (21) is considered, then Step 3 is not used.

Algorithm 1

Step 0: Given an initial point x_1, a symmetric and positive definite diagonal matrix $\hat{B}_1$ and $\epsilon > 0$ (a tolerance on the gradient norm). Set $k = 1$ and compute the initial search direction $s_1 = -\hat{B}_1^{-1} g_1$.

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 (8) hold.

Step 2: Stop if $\|g_{k+1}\| \leq \epsilon$.

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 positive definite diagonal Hessian approximation $\hat{B}_{k+1}$.

Step 5: Compute the new search direction $s_{k+1} = -\hat{B}_{k+1}^{-1} g_{k+1}$.

Step 6: Set $k = k + 1$, and go to Step 1.

3 Convergence analysis

The convergence property of our suggested diagonal Hessian approximations is studied in this section. Since they are maintained positive definite, the descent condition

$$s_k^T g_k < 0 \tag{30}$$

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

Assumption 1.

a) Consider the set $\Omega = \{x \in \Re^n : f(x) \leq f(x_1)\}$ and let $\tilde{\Omega}$ be an open set containing Ω.

b) The function $f(x)$ is bounded and continuously differentiable in $\tilde{\Omega}$.

c) The gradient $g(x)$ is Lipschitz continuous on $\tilde{\Omega}$, that is, there exists a constant $L > 0$ such that

$$\|g(x) - g(\tilde{x})\| \leq L\|x - \tilde{x}\|, \quad \forall\, x, \tilde{x} \in \tilde{\Omega}. \tag{31}$$

We state the following result which is similar to the well-known result of Zoutendijk [25].

Theorem 1. *Suppose Assumption 1 holds. Consider iterations (9) with x_1 is any starting point, the search direction s_k being defined such that the descent condition (30) holds and the steplength α_k satisfies the Wolfe-Powell conditions (8). Then, the so-called Zoutendijk condition*

$$\sum_{k=1}^{\infty} \frac{(s_k^T g_k)^2}{\|s_k\|^2} < \infty, \tag{32}$$

is obtained.

Proof. Similar to many analyses (see, for example, Nocedal and Wright [24]), we state the following proof (for complete illustration). Rearranging the second Wolfe-Powell condition in (8) and using the Lipschitz condition (31), in consequence

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

Substituting $x_{k+1} = x_k + \alpha_k s_k$ and using (30), we obtain:

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

Using this result, the first Wolfe-Powell condition in (8) implies

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

By summing this expression over k and using the assumed bound on f_k, we obtain the Zoutendijk condition (32). □

To obtain the global convergence result for the Algorithm 1, it is assumed that the condition number of the positive definite diagonal Hessian approximate $\hat{B}_k$ is uniformly bounded, that is, there is a constant M such that

$$\kappa(\hat{B}_k) = \frac{\lambda_1}{\lambda_n} \leq M, \ \ \forall\, k, \tag{33}$$

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

Theorem 2. *Suppose that f satisfies Assumption 1. Let x_1 be a starting point and $\hat{B}_1$ be a positive definite diagonal matrix. Consider Algorithm 1 with $\epsilon = 0$ in Step 0, $\hat{B}_{k+1}$ in Step 4 is defined such that condition (33) holds. Then, the algorithm converges globally, that is*

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{34}$$

Proof. Substituting $s_k = -\hat{B}_k^{-1} g_k$ into the Zoutendijk condition (32), we obtain:

$$\frac{1}{M}\sum_{k=1}^{\infty} \|g_k\|^2 = \lambda_n \frac{1}{\lambda_1}\|g_k\|^2 \leq \sum_{k=1}^{\infty} \frac{(s_k^T \hat{B}_k s_k)(g_k^T \hat{B}_k^{-1} g_k)\|g_k\|^2}{\|s_k\|^2\|g_k\|^2} < \infty,$$

where M is given as in (33). Hence, the limit (34) is obtained. □

This result is easily applied to all diagonal methods we consider here, because the diagonal Hessian approximations are defined sufficiently positive and bounded from above.

4 Numerical results

This section considers testing the proposed methods on a set of standard test problems (defined below). All the methods are implemented as in Algorithm 1, which differ only in Step 4 for defining the updated Hessian approximation $\hat{B}_{k+1}$ by choices (17), (21), (24) (with $\epsilon_2 = 10^{-2}$ and $\epsilon_3 = 10^{-14}$), (25), (28) and (29), (referred to as Lq1, Lq2, ..., Lq6, respectively).

In all algorithms, we consider the followings. For Step 0, we choose $\hat{B}_1 = I$ and $\epsilon = 10^{-7}$ is used below in (36). For Step 1, we calculate a value of the steplength α_k such that the strong Wolfe-Powell conditions

$$f_{k+1} \leq f_k + \sigma_0 \alpha_k s_k^T g_k, \quad |s_k^T g_{k+1}| \leq -\sigma_1 s_k^T g_k \tag{35}$$

hold, using the usual values of $\sigma_0 = 10^{-4}$ and $\sigma_1 = 0.9$, which imply the Wolfe-Powell conditions (8). 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 [6] or Fletcher [15]). We stop the run when either

$$\|g_k\| \leq 10^{-7} \max\{\|g_1\|, 1\}, \ \ f_k - f_{k+1} \leq 10^{-14}, \tag{36}$$

or the number of line searches reaches 10^5.

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

The test problems were selected from the collection of Andrei [10], (which belong to the CUTEst collection established by Gould et al. [18], Moré et al. [22] and Himmelblau [19]. We picked 48 test functions (as given in Table 1). We observed that increasing the number of variables in some extended test problems (e.g., Extended Rosenbrock) does not increase the number of line searches, function evaluations, or gradient evaluations required to solve the problems (see for example Al-Baali [3]). To avoid this occurrence, we modify the standard starting point $\overline{x}_1$ (as suggested by the author) to

$$\overline{\overline{x}}_1^{(i)} = \overline{x}_1^{(i)} + \frac{1}{i+1}, \tag{37}$$

for $i = 1, \ldots, n$. We used all 48 test functions with the standard starting points and their modifications (37) for $n = 60, 600$ and 6000 so that the total number of the test problems is 288 tests.

Table 1: List of test functions.

No.	Function's Name	No.	Function's Name
1	Extended Freudenstein & Roth	25	FLETCHR
2	Extended Trigonometric	26	TRIDIA
3	Extended Rosenbrock	27	ARGLINB
4	Generalized Rosenbrock	28	NONDIA
5	Extended White & Holst	29	NONDQUAR
6	Extended Beale	30	DQDRTIC
7	Extended Penalty	31	Broyden Tridiadonal
8	Perturbed Quadratic	32	Almost Perturbed Quadratic
9	Generalized Tridiagonal 1	33	Perturbed Tridiagonal Quadratic
10	Extended Tridiagonal 1	34	Staircase 1
11	Generalized Tridiagonal 2	35	Staircase 2
12	Extended Himmelblau	36	LIARWHD
13	Diagonal 4	37	POWER
14	Generalized White & Holst	38	EDENSCH
15	Generalized PSC1	39	CUBE
16	Extended PSC1	40	NONSCOMP
17	Extended Powell	41	QUARTC
18	Full Hessian FH2	42	SIQUAD
19	Extended BD1 (Block Diagonal)	43	Extended DENSHNB
20	Perturbed quadratic diagonal	44	Extended DENSHNF
21	Extended Hiebert	45	DIXON3DQ
22	Extended quadratic penalty QP1	46	BIGGSB1
23	Extended quadratic penalty QP2	47	Generalized Quartic
24	Extended quadratic exponential EP1	48	SINCOS

To study the behaviour of above algorithms, we compared the numerical results required to solve the tests, using the performance profiles of Dolan and Moré [12] 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 performance of the solvers on a set of test problems in terms of #ls (similarly for #fun, #gra and cpu time). In general, $P_L(\tau)$, the fraction of problems with performance ratio $\tau \geq 0$, is defined by

$$P_L(\tau) = \frac{\text{number of problems where } log_2(\tau_{p,L}) \leq \tau}{\text{total number of problems}}. \tag{38}$$

Here, $\tau_{p,L}$ is the ratio of #ls needed to solve problem p by the L method to the smallest #ls needed to solve problem p. If the L method fails to solve problem p, this ratio $\tau_{p,L}$ is set to ∞ (or some large number). The percentage of test problems for which the L method is best is given by $P_L(\tau)$ for $\tau = 0$, while the percentage of test problems that the L method can solve is given by the value for τ large enough. Thus, a solver with high values of $P_L(\tau)$ 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 set of test problems, we obtained some numerical results. Their comparisons are given in Figures 1–4, with respect to #ls, #fun, #gra and cpu time, respectively.

We observe that Lq3 appear to be the best, better than Lq4, Lq5 and Lq6 while Lq1 and Lq2 are worse than the other methods.

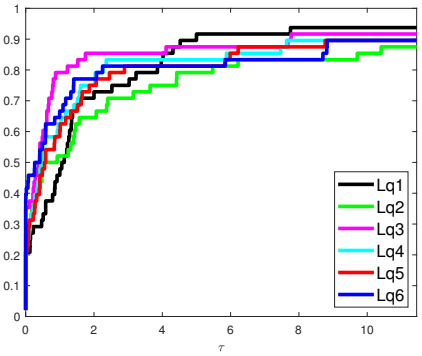

Figure 1: Comparison among Lq$i, i = 1, 2, \ldots, 6$, for #ls.

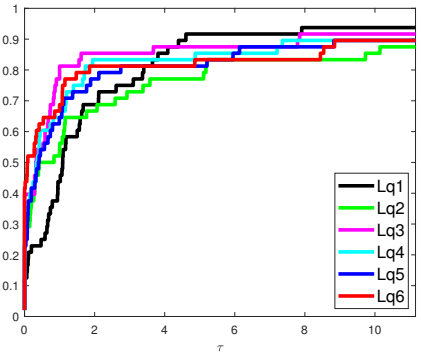

Figure 2: Comparison among Lq$i, i = 1, 2, \ldots, 6$, for #fun.

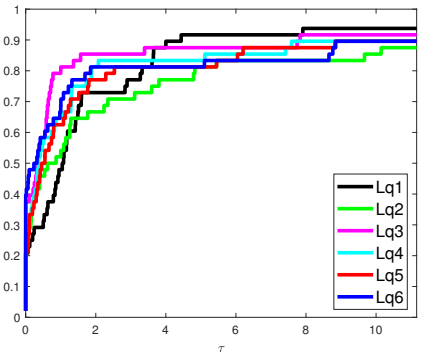

Figure 3: Comparison among Lq$i, i = 1, 2, \ldots, 6$, for #gra.

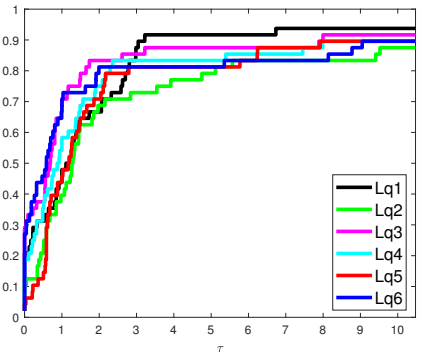

Figure 4: Comparison among Lq$i, i = 1, 2, \ldots, 6$, for cpu time.

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 [2] and essentially Al-Baali [1]). To compare two methods (say, A and B) with respect to #ls (similarly for #fun, #gra and cpu time), the author proposes the average ratio measure of the form:

$$r = \frac{1}{t} \sum_{i=1}^{t} r_i, \tag{39}$$

where t is the number of tests (here, $t = 288$) and

$$r_i = \begin{cases} \dfrac{p_i}{q_i}, & \text{if} \quad p_i \leq q_i \\ 2 - \dfrac{q_i}{p_i}, & \text{if} \quad p_i > q_i, \end{cases} \tag{40}$$

with p_i and q_i denoting #ls required to solve problem i by the A and B methods, respectively. If only A or only B failed to solve the problem, we set $r_i = 2$ and $r_i = 0$, respectively. For some test problem i, we set $r_i = 1$ if both A and B methods fail or converge to two different local solutions. The average ratio r in (39) always belongs to the interval $[0, 2]$. A value of $r \leq 1$ indicates that A is better than B by $100(1 - r)\%$. Otherwise, when $r > 1$, A is worse than B (or B is better than A) by $100(r - 1)\%$ (for additional information on this measurement ratio, see Al-Baali [2], for instance).

Using the same numerical results that we used to obtain the comparison in Figures 1–4, we applied the above average ratio measure to compare the Lq2, Lq3, ..., Lq6 algorithms versus the Lq1 algorithm and obtained Table 2. Since we have $r < 1$ in all cases except for Lq2, it is clear that Lq2 gives the worst performance of algorithms and Lq1 performs worse than Lq3, Lq4, Lq5 and Lq6 in terms of #ls, #fun #gra and cpu time. Thus, these observations agree with those in Figures 1–4. We observe that the best performance algorithm is obtained for the Lq3 by about 20%, 24%, 21% and 15% better than Lq1 algorithm in terms of #ls, #fun, #gra and cpu time, respectively. Table 2 also shows that Lq3 is better than Lq6, Lq4, Lq5, Lq1 and Lq2 in terms of #fun by about 4%, 7%, 11%, 24% and 24%, respectively (similar percentage improvements for the other measurements can be noticed from the table). Thus we obtain the following worsening order: Lq3, Lq6, Lq4, Lq5, Lq1 and Lq2 with Lq3 is slightly better than Lq6 and Lq4. Since, however, the latter three algorithms are defined by (24), (29) and (28), respectively, it follows that the implementation of Lq3 and Lq4 is simpler than that of Lq6 and hence, Lq3 and Lq4 are the winners. Because Table 2 shows that Lq3 performs better than Lq4 with percentage about 12%, 7%, 10% and 12%, in terms of #ls, #fun, #gra and cpu time, respectively, Lq3 is the winner.

Table 2: Comparison of L$qi, i = 2, 3 \ldots, 6$, versus L$q1$.

Method \ Measure	#ls	#fun	#gra	cpu
Lq2	1.067	1	1.057	1.108
Lq3	0.797	0.756	0.789	0.851
Lq4	0.924	0.832	0.899	0.967
Lq5	0.955	0.871	0.934	1.05
Lq6	0.876	0.799	0.862	0.899

5 Conclusion

We studied several proposals for maintaining the Hessian approximations B_k diagonal and positive definite. Based on extensive numerical experiments for 288 test problems, we observed that the Lq3 algorithm, which defines the Hessian approximation by the simple diagonal matrix (24), is the best among all algorithms. All these algorithms require the storage of a few vectors and, somehow, they share the features of the quasi-Newton condition and the structure of the nonlinear least-squares function.

References

[1] M. Al-Baali. A rule for comparing two methods in practical optimization. Dipartimento di Sistemi, Universita della Calabria, Italy, Technica1 Report, 119, 1991.

[2] M. Al-Baali. Numerical experience with a class of self-scaling quasi-Newton algorithms. Journal of Optimization Theory and Applications, 96(3), pp. 533–553, 1998.

[3] M. Al-Baali. Improved Hessian approximations for the limited memory BFGS method. Numerical Algorithms, 22(1), pp. 99–112, 1999.

[4] M. Al-Baali. Quasi-Newton algorithms for large-scale nonlinear least-squares. In High Performance Algorithms and Software for Nonlinear Optimization, Eds. G. Di Pillo and A. Murli, Kluwer Academic, pp. 1–21, 2003.

[5] M. Al-Baali and R. Fletcher. Variational methods for non-linear least-squares. Journal of the Operational Research Society, 36(5), pp. 405–421, 1985.

[6] M. Al-Baali and R. Fletcher. An efficient line search for nonlinear least squares. Journal of Optimization Theory and Applications, 48(3), pp. 359–377, 1986.

[7] A. Al-Siyabi. Efficient Methods for Large-Scale Nonlinear Least-Squares. Ph.D. thesis, Sultan Qaboos University, Muscat, Oman, 2021.

[8] A. Al-Siyabi and M. Al-Baali. New basic Hessian approximations for large-scale nonlinear least-squares optimization. International Conference on Intelligent Computing and Optimization, Koh Sami, Thailand, pp. 608–619. Springer, 2019.

[9] A. Al-Siyabi and M. Al-Baali. Improved diagonal updates for large-scale unconstrained optimization. Sultan Qaboos University Journal for Science, 26(2), pp. 52–66, 2021.

[10] N. Andrei. An unconstrained optimization test functions collection. Advanced Modeling Optimization, 10(1), pp. 147–161, 2008.

[11] R. Dehghani and N. Mahdavi-Amiri. Scaled nonlinear conjugate gradient methods for nonlinear least-squares problems. Numerical Algorithms, 82(1), pp. 1–20, 2019.

[12] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2), pp. 201–213, 2002.

[13] A. Bouaricha and J. J. Moré. Impact of partial separability on large-scale optimization. Computational Optimization and Applications, 7(1), pp. 27–40, 1997.

[14] J. Y. Fan and Y. X. Yuan. On the quadratic convergence of the Levenberg-Marquardt method without nonsingularity assumption. Computing, 74(1), pp. 23–39, 2005.

[15] R. Fletcher. Practical Methods of Optimization, 2nd Edition. Chichester, Wiley, 1987 (reprint 2013).

[16] R. Fletcher and C. Xu. Hybrid methods for nonlinear least squares. IMA Journal of Numerical Analysis, 7(3), pp. 371–389, 1987.

[17] J. C. Gilbert and C. Lemaréchal. Some numerical experiments with variable-storage quasi-Newton algorithms. Mathematical Programming, 45(1-3), pp. 407–435, 1989.

[18] N. I. Gould, D. Orban and P. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. Computational Optimization and Applications, 60(3), pp. 545–557, 2015.

[19] D. M. Himmelblau. Applied Nonlinear Programming. Mcgraw-Hill. New York, 1972.

[20] L. Lukšan and J. Vlček. A hybrid method for nonlinear least squares that uses quasi-Newton updates applied to an approximation of the Jacobian matrix. Programs and Algorithms of Numerical Mathematics, pp. 99–106, 2019.

[21] H. Mohammad and S. A. Santos. A structured diagonal Hessian approximation method with evaluation complexity analysis for nonlinear least squares. Computational and Applied Mathematics, 37(5), pp. 6619–6653, 2018.

[22] J. J. Moré, B. S. Garbow and K. E. Hillstrom. Testing unconstrained optimization software. ACM Transactions on Mathematical Software (TOMS), 7(1), pp. 17–41, 1981.

[23] J. Nocedal. Updating quasi-Newton matrices with limited storage. Mathematics of Computation, 35(151), pp. 773–782, 1980.

[24] J. Nocedal and S. Wright. Numerical Optimization. Springer Science & Business Media, 2006.

[25] G. Zoutendijk. Nonlinear programming, computational methods. Integer and Nonlinear Programming, pp. 37–86, 1970.