mathematics

MDPI

Article

An Efficient Limited Memory Multi-Step Quasi-Newton Method

Issam A. R. Moghrabi [1,2,*] and Basim A. Hassan [3]

[1] Department of Information Systems and Technology, Kuwait Technical College, Abu-Halifa 54753, Kuwait
[2] Department of Computer Science, School of Arts and Science, University of Central Asia, Naryn 722918, Kyrgyzstan
[3] Department of Mathematics, College of Computer Sciences and Mathematics, University of Mosul, Mosul 41002, Iraq; basimah@uomosul.edu.iq
* Correspondence: i.moghrabi@ktech.edu.kw

Abstract: This paper is dedicated to the development of a novel class of quasi-Newton techniques tailored to address computational challenges posed by memory constraints. Such methodologies are commonly referred to as "limited" memory methods. The method proposed herein showcases adaptability by introducing a customizable memory parameter governing the retention of historical data in constructing the Hessian estimate matrix at each iterative stage. The search directions generated through this novel approach are derived from a modified version closely resembling the full memory multi-step BFGS update, incorporating limited memory computation for a singular term to approximate matrix–vector multiplication. Results from numerical experiments, exploring various parameter configurations, substantiate the enhanced efficiency of the proposed algorithm within the realm of limited memory quasi-Newton methodologies category.

Keywords: quasi-Newton methods; multi-step methods; unconstrained optimization; limited memory methods

MSC: 64K10

Citation: Moghrabi, I.A.R.; Hassan, B.A. An Efficient Limited Memory Multi-Step Quasi-Newton Method. *Mathematics* **2024**, *12*, 768. https://doi.org/10.3390/math12050768

Academic Editor: Songting Luo

Received: 22 January 2024
Revised: 15 February 2024
Accepted: 17 February 2024
Published: 4 March 2024

1. Introduction

Unconstrained Optimization is concerned with the minimization of a specific objective function as follows:

minimize $f(x)$, where $f : R^n \to R$.

The aforementioned problem can be solved by employing a class of methods referred to as the quasi-Newton techniques for unconstrained optimization. Only the function and its first derivatives are required for quasi-Newton (QN) methods [1]. The Hessian does not need to be available or even coded. To integrate updates in the function and the corresponding gradient, an approximation matrix to the actual Hessian is retained and updated across the iterations.

Given B_i, the current Hessian approximation, a new Hessian estimation B_{i+1} needs to be constructed, for the new solution estimate x_{i+1}. To find B_{i+1}, we can utilize the Taylor series of order one to the gradient vector about the point x_{i+1} to obtain the following relation (known as the Secant equation) [2–6]:

$$B_{i+1}s_i = y_i, \tag{1}$$

where

$$s_i = x_{i+1} - x_i,$$

and

$$y_i = g_{i+1} - g_i,$$

where $g_i \equiv g(x_i)$, is the gradient at iterate x_i. The computed Hessian matrix approximations must satisfy (1). To estimate the next Hessian approximation matrix B_{i+1} using the current matrix B_i and the vectors s_i and y_i, an update of the form $B_{i+1} = B_i + C_i$ is applied, where C_i is a correctional update matrix. It may be preferable to use $H_{i+1} = H_i + D_i$, where D_i is an update term and $H_{i+1} = B_{i+1}^{-1}$. The update can be of rank one or two. A widely used rank two correction formula is referred to as the BFGS formula. The BFGS is given by

$$B_{i+1}^{BFGS} = B_i + \frac{y_i y_i^T}{y_i^T s_i} - \frac{B_i s_i s_i^T B_i}{s_i^T B_i s_i},$$

and the corresponding inverse matrix as

$$H_{i+1}^{BFGS} = H_i + \left[1 + \frac{y_i^T H_i y_i}{y_i^T s_i}\right] \frac{s_i s_i^T}{y_i^T s_i} - \frac{s_i y_i^T H_i + H_i y_i s_i^T}{y_i^T s_i}. \quad (2)$$

Reported numerical outcomes show that this formula outperforms other updating formulas, particularly when the line search is inexact. BFGS is regarded as a standard update formula [3,5,7,8].

The search direction for the standard quasi-Newton methods is computed using

$$B_i p_i = -g_i.$$

2. Literature Review

Limited memory QN methods (L-BFGS) have gained considerable prominence in optimization due to their ability to handle large-scale problems efficiently, especially with memory constraints. These iterative methods are mostly classified as quasi-Newton techniques [9–12]. The versatility of L-BFGS methods is evident in their application across various domains. In machine learning, they are integral to training support vector machines (SVMs), optimizing logistic regression models, and fine-tuning neural networks. Their memory-efficient design makes them invaluable when working with large datasets. Additionally, L-BFGS plays a pivotal role in structural optimization, a field where finite element analysis involves high-dimensional design spaces. In computational chemistry, L-BFGS aids in optimizing molecular structures, facilitating breakthroughs in drug discovery and materials science.

Nevertheless, L-BFGS distinguishes itself from the classical quasi-Newton methods by storing only a limited number of recent iterations data, often referred to as memory pairs (s_i, y_i), where s_i, represents the alteration in the optimization variable, and y_i denotes the change in the gradient (see (1)). These memory pairs are instrumental in updating the approximate Hessian matrix, achieving a balance between computational efficiency and optimization accuracy.

In the past decade, L-BFGS has seen significant methodological advancements and the emergence of new variants tailored to address specific optimization challenges. For example, L-BFGS-B extends the method to handle bound-constrained optimization problems effectively [10]. Innovations like L-BFGS-TF and L-BFGS++ have been introduced to enhance scalability and convergence properties. These methodological innovations have substantially widened the scope of L-BFGS applications, encompassing areas such as machine learning, deep learning, and numerical simulations.

Limited memory BFGS algorithms often derive using the updated identity for the inverse Hessian given as [10]

$$H_{i+1} = \left(I - \rho_i s_i y_i^T\right) H_i \left(I - \rho_i y_i s_i^T\right) + \rho_i s_i s_i^T,$$

where s_i and y_i are as in (1) and $\rho_i = 1/s_i^T y_i$.

For a given identity matrix I, a sequence of vectors $q_{i-m}, \ldots.., q_m$ is defined such that $q_i \equiv \left(I - \rho_i s_i y_i^T\right) q_{i+1}$. Then, the quantities q_i and q_{i+1} are recursively computed. For more details see [10].

Next, we present some successful limited-memory BFGS formulas. These formulas are all variations of the standard L-BFGS formula, and they are designed to enhance the numerical behavior of the L-BFGS algorithm for certain types of problems. One of the well-known limited memory methods is the Liu–Nocedal formula [9]. The updated Hessian matrix is the previous Hessian matrix updated with a correction term. The correction term is a weighted sum of the two recent steps and gradient differences, where the weights are chosen to make certain that the updated Hessian matrix is positive-definite. The formula is given as

$$H_i = H_{i-1} + \frac{s_{i-1}s_{i-1}^T}{s_{i-1}^T y_{i-1}} - \frac{H_{i-1}y_{i-1}y_{i-1}^T H_{i-1}}{y_{i-1}^T H_{i-1} y_{i-1}} + \frac{\rho_i s_{i-2}s_{i-2}^T}{s_{i-2}^T y_{i-2}} - \frac{\rho_i H_{i-1}y_{i-2}y_{i-2}^T H_{i-1}}{y_{i-2}^T H_{i-1} y_{i-2}}$$

Another successful formula is the Byrd–Nocedal–Schnabel formula [13]. The correction term is a weighted sum of the latest m steps and gradient differences, where the weights are chosen to ascertain that the newly computed Hessian matrix is positive-definite. The formula is defined as

$$H_i = H_{i-1} + \frac{s_{i-1}s_{i-1}^T}{s_{i-1}^T y_{i-1}} - \frac{H_{i-1}y_{i-1}y_{i-1}^T H_{i-1}}{y_{i-1}^T H_{i-1} y_{i-1}} + \sum_{j=i-m}^{i-1} \frac{\rho_i s_j s_j^T}{s_j^T y_j} - \sum_{j=i-m}^{i-1} \frac{\rho_i H_{i-1}y_j y_j^T H_{i-1}}{y_j^T H_{i-1} y_j}. \quad (3)$$

A variation of (3) is the Zhu–Byrd–Nocedal formula [14]. The correction component is a weighted sum of the latest m steps and gradient differences, where the weights are also chosen to make certain that the newly computed Hessian matrix is positive-definite. Additionally, the correction term includes a term that is designed to improve the behavior of the L-BFGS algorithm for problems with non-smooth objective functions [15]. The formula is defined by

$$H_i = H_{i-1} + \frac{s_{i-1}s_{i-1}^T}{s_{i-1}^T y_{i-1}} - \frac{H_{i-1}y_{i-1}y_{i-1}^T H_{i-1}}{y_{i-1}^T H_{i-1} y_{i-1}} + \sum_{j=i-m}^{i-1} \frac{\rho_i s_j s_j^T}{s_j^T y_j} - \sum_{j=i-m}^{i-1} \frac{\rho_i H_{i-1}y_j y_j^T H_{i-1}}{y_j^T H_{i-1} y_j} + \frac{\rho_i s_{i-m}s_{i-m}^T}{s_{i-m}^T y_{i-m}} - \frac{\rho_i H_{i-1}y_{i-m}y_{i-m}^T H_{i-1}}{y_{i-m}^T H_{i-1} y_{i-m}}. \quad (4)$$

In (3) and (4), the parameter m controls the number of past update terms that are used to update the Hessian matrix. A larger value of m will result in a more precise approximation of the Hessian matrix, closer to that of the standard QN methods.

Another idea, presented in [16], focuses on applying regularized Newton methods in a versatile category of unconstrained optimization algorithms. The method they propose has the potential to combine favorable characteristics from both approaches. The primary emphasis is on the integration of regularization with limited memory quasi-Newton methods, leveraging the unique structure inherent in limited memory algorithms. The paper explores an alternative globalization technique, akin to the less familiar siblings of line search and trust-region methods, known as regularized Newton methods. The derived methods are referred to as regularized quasi-Newton methods. The methods utilize the relationship:

$$(B_i + \mu_i I)p_i = -g_i,$$

which is used in the computation of the search direction p_i, instead of the one used by the standard quasi-Newton methods. The parameter μ_i serves as the regularization parameter. The derived methods integrate features from both line search and trust region techniques and update a limited memory version of the Hessian matrix estimate. The method displays good behavior overall though it involves the solution of a system of linear equations at each iteration, which increases the time complexity of the algorithm.

A recent paper also introduces a limited memory quasi-Newton type method for unconstrained multi-objective optimization problems [17], which is suitable for large-scale

scenarios. The proposed algorithm approximates the convex combination of the Hessian matrices of the objective function using a positive definite matrix. The update formula for the approximation matrix is an extension of the one used in the L-BFGS method for scalar optimization [18]. The method is well defined even in the nonconvex case and exhibits global and R-linear convergence to Pareto optimality for twice continuously differentiable strongly convex problems. In the method derived in [17], the matrix used to update the inverse Hessian approximation is given by

$$H_{i+1} = \left(I - \rho_i s_i u_i^T\right) H_i \left(I - \rho_i u_i s_i^T\right) + \rho_i s_i s_i^T,$$

where $\rho_i = 1/s_i^T u_i$ and u_i is the summation of the previous y_i dictated by the number of vectors retained in memory. Those y values are multiplied by optimal Lagrange multipliers needed to achieve global convergence.

The performance of the algorithm is evaluated through computational comparisons with state-of-the-art Newton and quasi-Newton approaches in multi-objective optimization. The results show that the proposed approach is generally efficient. While testing the method, it failed to converge on some of our test functions, but it did well on others and managed to find global minimizers.

The paper in [19] presents a new numerical method for solving large-scale unconstrained optimization problems, derived from a modified BFGS-type update. The update formula is extended to the framework of a limited memory scheme. The update utilized in that paper takes the form

$$H_{i+1} = \left(I - \rho_i s_i u_i^T\right) H_i \left(I - \rho_i u_i s_i^T\right) + \rho_i s_i s_i^T,$$

where $\rho_i = 1/s_i^T u_i$ and $u_i = y_i + \gamma_i s_i$, for some scalar γ_i to ensure convergence. The paper discusses the global convergence and convergence rate of the algorithm with weak Wolfe–Powell line search.

A modified q-BFGS algorithm is proposed for nonlinear unconstrained optimization problems [20]. It uses a simple symmetric positive definite matrix and a new q-quasi-Newton equation to build an approximate q-Hessian. The method preserves global convergence properties without assuming the convexity of the objective function. Numerical results show improvement over the original q-BFGS method. A limited memory quasi-Newton method is introduced for large-scale unconstrained multi-objective optimization problems. It approximates the convex combination of the Hessian matrices of the objectives and exhibits global and R-linear convergence to Pareto optimality. Empirical comparisons demonstrate the efficiency and effectiveness of the proposed approach [21]. A new L-BFGS method with regularization techniques is proposed for large-scale unconstrained optimization. It guarantees global convergence and is robust in terms of solving more problems [4]. The momentum-accelerated quasi-Newton (MoQ) method approximates Nesterov's accelerated gradient as a linear combination of past gradients, and its performance is evaluated on a function approximation problem.

3. A New Multi-Step Limited Memory Method

In this section, we devise a new QN-limited memory method inspired by the methods in (3) and (4). The method is characterized by its utilization of more of already computed past data to better the quality of the generated Hessian (inverse) approximations.

In the basic Secant equation, a straight line L is utilized to find a new iteration x_{i+1} given the iterate x_i, whereas higher-order interpolants are used in the multi-step approaches [22,23]. The main advantage of using polynomials is that they exploit more of the already computed data in the matrix update rather than simply discarding them. This is expected to result in better Hessian approximations.

Multi-step methods generally update the Hessian estimates to satisfy the form

$$B_{i+1}u_i = w_i, \tag{5}$$

for which the classical Secant relation simply refers to the choices $u_k = s_k$ and $w_k = y_k$, respectively. One of the alternatives to u_i and w_i proposed in [24] is

$$u_i = s_i - \mu_{i-1}s_{i-1} \text{ and } w_i = y_i - \mu_{i-1}y_{i-1}, \tag{6}$$

where $\mu_{i-1} = \delta^2/(1+2\delta)$ and $\delta = \|s_i\|/\|s_{i-1}\|$.

Thus, Equation (5) becomes:

$$H_{i+1}(y_i - \mu_{i-1}y_{i-1}) = s_i - \mu_{i-1}s_{i-1}$$

Concentrating on the BFGS updating method (2), the inverse Hessian approximation at iteration i may be expressed after manipulation as

$$H_{i+1} = H_i + \left(\mathrm{u}_i^T\mathrm{w}_i\right)^{-1}\left\{\left[1 + \frac{\mathrm{w}_i^T r_i}{\mathrm{u}_i^T\mathrm{w}_i}\right]\mathrm{u}_i\mathrm{u}_i^T - u_i r_i^T - r_i u_i^T\right\} \tag{7}$$

where $r_i = H_i w_i$.

Contained memory techniques are developed using the identity

$$p_{i+1} = -H_{i+1}g_{i+1}$$

which is equivalent to (using (7))

$$p_{i+1} = p_i - H_i y_i - \left(\mathrm{u}_i^T\mathrm{w}_i\right)^{-1}\left\{\left(\left[1 + \frac{\mathrm{w}_i^T r_i}{\mathrm{u}_i^T\mathrm{w}_i}\right]u_i^T g_{i+1} - r_i^T g_{i+1}\right)u_i + \left(u_i^T g_{i+1}\right)r_i\right\}. \tag{8}$$

However, the problem in using (7) is that of computing r_i and $H_i y_i$ without actually storing H_i. The simplest option would be to make the choice $H_i = I$ at each iteration and, thus, $r_i = y_i$ and $H_i y_i$, in that case, reduces to y_i. However, this is a numerically improper choice since it abandons previous information collected in the updated matrices. Another proposal is considered here and that is to retain some approximation of the vector r_i and correct every cycle. This can be performed using:

Start with $r_0 = H_0 w_0 = w_0$ (since $H_0 = I$)

$$r_1 = H_1 w_1 = w_1 + \left(u_0^T w_0\right)^{-1}\left\{\left(\left[1 + \frac{w_0 w_0^T}{u_0^T w_0}\right]u_0^T w_1 - \left(w_0^T w_1\right)\right)u_0 - \left(u_0^T w_1\right)w_0\right\} \tag{9}$$

and, recurrently, for any i, we use

$$r_i = H_{i-1}w_i + \left(u_{i-1}^T w_{i-1}\right)^{-1}\left\{\left(\left[1 + \frac{w_{i-1}^T r_{i-1}}{u_{i-1}^T y_{i-1}}\right]u_{i-1}^T w_i - \left(r_{i-1}^T w_i\right)\right)u_{i-1} - \left(u_{i-1}^T w_i\right)r_{i-1}\right\}. \tag{10}$$

Relation (10) can then be utilized in (8) to compute the next direction vector. Still, the term $H_{i-1}w_i$ in (10) is not accessible. The obvious option is to set, r_{i-1} to w_{i-1} in (8). Such a choice is expected to affect negatively the updated matrix. Thus, our goal next is to build an expression for both $H_{i-1}w_i$ and $H_i y_i$ to complete the derivation. By doing so, we will then have derived a method that is expected to be computationally superior to that of setting H_i to be just the identity matrix in (8). This is because more of the previously accumulated information is utilized in updating the matrix. Therefore, the preference computationally is to approximate the aforementioned matrix vector products as will be shown next.

To proceed with the derivation, it should be noted that for a diagonal matrix H_0, the next matrix in recurrence can be expressed as

$$H_1 = H_0 + U(H_0, u_0, w_0)$$

and

$$H_2 = H_0 + U(H_0, u_0, w_0) + U(H_1, u_1, w_1)$$
$$H_3 = H_0 + U(H_0, u_0, w_0) + U(H_1, u_1, w_1) + U(H_2, u_2, w_2) \ldots \text{. and so on}$$

for vectors u and w, as in (5).

Let η refer to the upper limit of the correction terms U that can be saved. Because H_0 is diagonal, the limit count of n-vectors that can be utilized to build the matrix approximation is $2\eta + 1$. We have hit our storage limit once H_η is produced. In this case, newer vectors start replacing older ones:

$$H_\eta = H_0 + U(H_0, u_0, w_0) + \ldots + U\big(H_{\eta-1}, u_{\eta-1}, w_{\eta-1}\big).$$

This idea was influenced by Nocedal's algorithm [10]. Nocedal's technique is founded on the idea that the latest data should be provided in the update, and replacing the oldest information is one compelling option. However, there is no assurance that the computed matrices will be positive-definite. Loss of positive-definiteness leads to search directions that are not necessarily descent, thus threatening the method's convergence. To avoid this problem, the standard BFGS updating formula is expressed in the form [2]:

$$H_{i+1} = V_i^T H_i V_i + \rho_i u_i u_i^T, \tag{11}$$

where $\rho_i = \left(u_i^T w_i\right)^{-1}$ and $V_i = I + \rho_i\, w_i u_i^T$, where ρ_i is a parameter that is chosen to ascertain that the newly computed Hessian matrix is positive-definite and m is a parameter that controls the number of past updates that are used to update the Hessian matrix.

Thus, given the above-proposed strategy, Nocedal suggests a form of the update as follows:

$$\begin{array}{l}
\text{For } i+1 \le \eta, \\
H_{i+1} = V_i^T V_{i-1}^T \ldots V_0 H_0 \ldots V_{i-1} V_i \\
+V_i^T \ldots V_1^T \rho_0 u_0 u_0^T V_1 \ldots V_i + \ldots \\
+V_i^T \ldots V_{i-1}^T \rho_{i-2} u_{i-2} u_{i-2}^T V_{i-1} \ldots V_i \\
+V_i^T \rho_{i-1} u_{i-1} u_{i-1}^T V_i + \rho_i u_i u_i^T \\
\text{For } i+1 > \eta, \\
H_{i+1} = V_i^T V_{i-1}^T \ldots V_{i-\eta+1} H_0 \ldots V_{i-\eta+1} V_{i-1} V_i \\
+V_i^T .. V_{i-\eta+2}^T \rho_{i-\eta+1} u_{i-\eta+1} u_{i-\eta+1}^T \ldots V_i + \ldots \\
+V_i^T .. V_{i-1}^T \rho_{i-2} u_{i-2} u_{i-2}^T V_{i-1} \ldots V_i \\
+V_i^T \rho_{i-1} u_{i-1} u_{i-1}^T \mathrm{V}_i + + \rho_i u_i u_i^T.
\end{array} \tag{12}$$

Back to our problem, we adopt Nocedal's approach with a suitably chosen small η to build an approximation to the product $H_{i-1} w_i$ in (8). For example, for $\eta = 1$, we have

$$\begin{array}{l}
H_{i-1} w_i = V_{i-2}^T H_0 V_{i-2} w_i + \left(\rho_{i-1} u_{i-2}^T w_i\right) u_{i-2} \\
= H_0 w_i - \left[\rho_{i-2} u_{i-2}^T w_i\right] H_0 w_{i-2} \\
+\rho_{i-2}^2 \left[\left(w_{i-2}^T H_0 w_{i-2}\right)\left(u_{i-2}^T w_i\right) - \left(u_{i-2}^T H_0 w_i\right)\left(u_{i-2}^T w_{i-2}\right) u_{i-2}\right] \\
+\left(\rho_{i-2} s_{i-2}^T w_i\right) u_{i-2}.
\end{array}$$

Similarly, to compute the product $H_i y_i$ the following may be used:

$$\begin{array}{l}
H_i y_i = V_{i-1}^T H_0 V_{i-1} y_i + \left(\rho_{i-1} u_{i-1}^T y_i\right) u_{i-1} \\
= H_0 y_i - \left[\rho_{i-1} u_{i-1}^T y_i\right] H_0 w_{i-1} \\
+\rho_{i-1}^2 \left[\left(w_{i-1}^T H_0 w_{i-1}\right)\left(u_{i-1}^T y_i\right) - \left(u_{i-1}^T H_0 y_i\right)\left(u_{i-1}^T w_{i-1}\right) u_{i-1}\right] \\
+\left(\rho_{i-1} s_{i-1}^T y_i\right) u_{i-1}.
\end{array}$$

In general, the following is employed to build the above products for a given η and an appropriate replacement of the matrix/vectors involved (Algorithm 1):

Algorithm 1 MSCBFGS

0. LET $k = i - 1$;
1. IF $k \leq \eta$ THEN $(incr = 0; bound = k)$
   ELSE $(incr = k - \eta; bound = \eta)$
2. $q_{bound} = w_i$
3. FOR $l = bound - 1 \ldots 0$
   $\{j = l + incr;$
   $\alpha_l = \rho_j u_j^T q_{l+1};$
   $q_l = q_{l+1} - \alpha_l w_j;\}$
4. $z_0 = H_0 q_0$
5. FOR $l = 0, 1, \ldots, bound - 1;$
   $\{j = l + incr;$
   $\beta_j = \rho_j w_j^T z_l;$
   IF $(k \succ \eta)$ THEN $z_{l+1} = z_l + u\Big(\alpha_l - \beta_j\Big)$

$$\text{ELSE } z_{l+1} = z_l + u(\alpha_l - \beta_l)\}. \tag{13}$$

Our multi-step limited memory BFGS (MSCBFGS) algorithm can be stated as:
Start with an approximate point x_0 to the solution
Start with $H_0 = I$ (or a scalar multiple of it)
$i \leftarrow 0$
Find $g_0 = \mathrm{g}(x_0)$
Repeat
Step 1. Let $p_i = -H_i g_i$
Step 2. Minimize $f(\mathrm{x}_i + \alpha p_i)$ where $\alpha \in R$ to find a step length α_i along p_i. To make certain that the obtained search direction is adequately descent, the parameter α_i is required to satisfy the strong Wolfe conditions [25]

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k$$

And

$$d_k^T g(x_k + \alpha_k d_k) \geq \sigma\, d_k^T g_k,$$

where $0 < \delta < \sigma < 1$
Step 3. $x_{i+1} = x_i + \alpha_i p_i$
Step 4. Compute $s_i = x_{i+1} - x_i$ $y_i = g_{i+1} - g_i$, $w_i = y_i - \mu_i y_{i-1}$ and
$u_i = s_i - \mu_i s_{i-1}$.
Step 5. If $\mathrm{w}_i^T \mathrm{u}_i > 0$, then compute H_{i+1}^{BFGS} using the recurrence in (8).
Step 6. $i = i + 1$
until $\| g_i \|_2 < \varepsilon$, (where $\varepsilon \in R$ is a convergence parameter).

We now need to establish that (8) produces a descent direction vector. To do so, it suffices to show that the matrix used in the computation of (8) is positive-definite.

Theorem 1. *The matrix H_{i+1} update in (7) is positive-definite if H_i is positive-definite and if $u_i^T w_i > 0$.*

Proof of Theorem 1. We first show that if H_{i+1} is positive-definite, then $u_i^T \mathrm{w}_i > 0$. From (5), it follows that

$$w_i^T u_i = w_i^T H_{i+1} w_i > 0.$$

Now, given that $w_i^T u_i > 0$ and $r_i = H_i w_i$ (from (7)), we proceed to prove that H_{i+1} is positive-definite by showing that $v^T H_{i+1} v > 0$, for any vector $v \neq 0$. Using (7) and (11), we have

$$v^T H_{i+1} \mathrm{v} = v^T V_i^T H_i V_i \mathrm{v} + \rho_i \left(u_i^T v \right)^2, \tag{14}$$

where $\rho = \left(u_i^T w_i \right)^{-1}$, $u_i = s_i - \mu_{i-1} s_{i-1}$, $V_i = I + \rho_i \, w_i u_i^T$, and ρ_i is a parameter that is chosen to make certain that the newly computed Hessian matrix is positive-definite. Since H_i is positive-definite, $V_i^T H_i V_i$ is also positive-definite. The second term is also positive. Thus, the whole expression in (14) is positive-definite. □

It should be noted here that conditions (13) ensure only that $s_i^T y_i > 0$ while there are no guarantees for $w_i^T u_i$. In our implementations, the condition $w_i^T u_i > 0$ is tested, and if it is not satisfied, we revert to using the classical Secant update at that specific iteration with $u_i = s_i$ and $w_i = y_i$.

As the new algorithm is not stripping away the main structure of the BFGS update, the derived method possesses the same convergence properties as those that belong to the same class such as those in [10,26]. The R-linear convergence rate of the MSCBFGS method is proved under the assumption that the objective function is strongly convex. However, the method can also converge R-linearly for non-convex objective functions. The convergence rate of the MSCBFGS method can be affected by the choice of the memory parameter η. A larger value of η can lead to a faster convergence rate, but it can also make the method more sensitive to noise. The proof for the convergence rate is very similar to that conducted in [26].

Estimating the time complexity of the described optimization algorithm involves several factors. The number of iterations (k) in the optimization process, along with the computational costs of algorithmic components ($O(f)$) and convergence conditions ($O(g)$), collectively shape the overall time complexity. A very rough estimate of the algorithm's time complexity could be expressed as $O(k \times (f + g))$, considering the iterative nature of the optimization and the associated computations. However, a more precise analysis would require a detailed breakdown of specific operations and their computational costs. As a result, the per iteration complexity and storage requirement is $O(\eta n)$ where $\eta \leq n$ is the size of the stored frame and n is the problem dimension, thus, reducing the $O(n^2)$ computational complexity and memory requirements of standard quasi-Newton methods.

The main feature of this algorithm is that it computes the search vector at each iteration using a formula that is almost the full memory multi-step BFGS version with only an approximation applied to the term $H_{i-1} w_i$ in (10). This is expected to produce results close to the full multi-step BFGS version, as the results reveal.

4. Numerical Results

The new method (7) is benchmarked against the methods in (3) and (4). Those are experimented on thirty distinct test problems with dimensions varying from 2 to 10,000. The total number of tested problems is 900. The functions tested are classified into four categories and can be found in [7,13,27–29]. The categories are as follows:

a. Low dimension $(2 \leq n \leq 15)$
b. Medium dimension $(16 \leq n \leq 45)$
c. Moderate-High dimension $(46 \leq n \leq 10000)$
d. High dimension $(81 \leq n \leq 10000)$

Each of the test problems possesses one or more of the following properties:

(a) Non-convexity;
(b) Global minimum;
(c) Multi-modality;
(d) Ill-conditioned;
(e) Highly non-linear;

(f) Symmetry around the global minimum;
(g) Periodic;
(h) Discrete optimization domain.

The functions and their properties are listed in Table 1.

Table 1. Set of test problems.

Function Name (Dimension)	Properties
Rosenbrock ($n = 2$)	a, b, c, d, f
Quadratic function ($n = 2$)	b
Watson function ($3 \leq n \leq 31$)	c, e
Extended Rosenbrock ($2 \leq n \leq 10000, n$ even)	a, c, d
Extended Powell ($2 \leq n \leq 10000, n$ divisible by 4)	a, c
Penalty function I ($2 \leq n \leq 10000$)	a, c, e
Variably dimensioned function ($2 \leq n \leq 10000$)	a, c
Trigonometric function ($2 \leq n \leq 10000$)	a, c, g
Modified Trigonometric function ($2 \leq n \leq 10000$)	a, c, g
Broyden Tridiagonal function ($2 \leq n \leq 10000$)	a, c
Discrete Boundary value function ($2 \leq n \leq 10000$)	a, c, h
Oren and Spedicato Power function ($2 \leq n \leq 10000$)	a, c, e
Full Set of Distinct Eigen Values Problem ($2 \leq n \leq 10000$)	a (if all eigenvalues are negative)
Tridiagonal function ($2 \leq n \leq 10000$)	a, c, f
Wolfe function ($2 \leq n \leq 10000$)	a, c, e
Diagonal Rosenbrock's function ($2 \leq n \leq 10000$, n even)	c, e
Generalized Shallow function ($2 \leq n \leq 10000$, n even)	c, e
Freudenstein and Roth ($n = 2$)	a, c, e
Powell Badly Scaled ($n = 2$)	a, d, e
Brown Badly Scaled ($n = 2$)	a, d, e
Beale ($n = 2$)	a, b, c
Bard ($n = 3$)	a, b, c
Freudenstein and Roth ($n = 2$)	a, b, c
Powell Badly Scaled ($n = 2$)	a, b, c
Brown Badly Scaled ($n = 2$)	a, b, c
Beale ($n = 2$)	a, b, c
Bard ($n = 3$)	a, b, c
Gaussian (n = 3)	B

Tables 2–5 report the results for each category. Table 6 reports the totals. The figures presented in each table indicate the total number of iterations, function/gradient evaluations, the timings, and the points for every algorithm. A point is granted to a method if it obtains the minimal evaluations count in solving a problem. Ties are resolved using the count of iterations. The percentages in each table indicate the improvements (or otherwise) of each method on each of the evaluation categories (iterations, evaluations, and time score) as opposed to the benchmark method of Byrd et al. A lower percentage indicate the savings obtained in comparison to the benchmark method that is assigned the full percentage. Our results for the above method seem to bias method 3 for $\eta = 3$. Bigger η have not introduced worthwhile improvements in performance over the other methods to justify the extra memory expense. Therefore, the test results reported on the new method are conducted with $\eta = 3$.

Table 2. Large problems test results.

Method	Evaluations	Iterations	Time (s)	Scores
Byrd et al. [13]	25,039	22,106	426.42	32
	100%	100%	100%	
Zhu and Byrd [14]	22,699	19,838	392.08	51
	90.66%	89.7%	91.95%	
MSCBFGS	21,609	18,461	319.95	71
	86.30%	83.5%	75.03%	

Table 3. Moderate high dimension problems test results.

Method	Evaluations	Iterations	Time (s)	Scores
Byrd et al. [13]	12,841	9673	3348.5	90
	100%	100%	100%	
Zhu and Byrd [14]	12,343	8864	3075.9	129
	96.12%	91.64%	91.8%	
MSCBFGS	11,654	8717	2980.4	162
	90.76%	90.12%	89.1%	

Table 4. Medium-size dimension problems test results.

Method	Evaluations	Iterations	Time (s)	Scores
Byrd et al. [13]	11,745	11,061	18,645.71	66
	100%	100%	100%	
Zhu and Byrd [14]	11,603	11,132	17,422.51	78
	98.79%	100.64%	93.44%	
MSCBFGS	10,869	9850	13,916.57	110
	92.54%	89.06%	74.64%	

Table 5. Small dimension problems test results.

Method	Evaluations	Iterations	Time (s)	Scores
Byrd et al. [13]	9597	5049	10,744.0	44
	100%	100%	100%	
Zhu and Byrd [14]	9443	4953	10,503.0	41
	98.39%	98.11%	97.76%	
MSCBFGS	9754	5180	10,779.4	26
	101.64%	102.61%	100.3%	

Table 6. Overall scores for the test results for 900 problems.

Method	Evaluations	Iterations	Time (s)	Scores
Byrd et al. [13]	59,079	47,960	31,942	232
	100%	100%	100%	
Zhu and Byrd [14]	56,087	44,787	31,394	299
	94.94%	93.38%	98.28%	
MSCBFGS	53,887	42,209	27,996	369
	91.21%	88.01%	87.65%	

The results reveal clearly that MSCBFGS (corresponding to $\eta = 3$) method is superior to the other two, especially on large problems. Figures 1–3 report the results comparisons on each criterion while Figure 4 presents an overall overview of those.

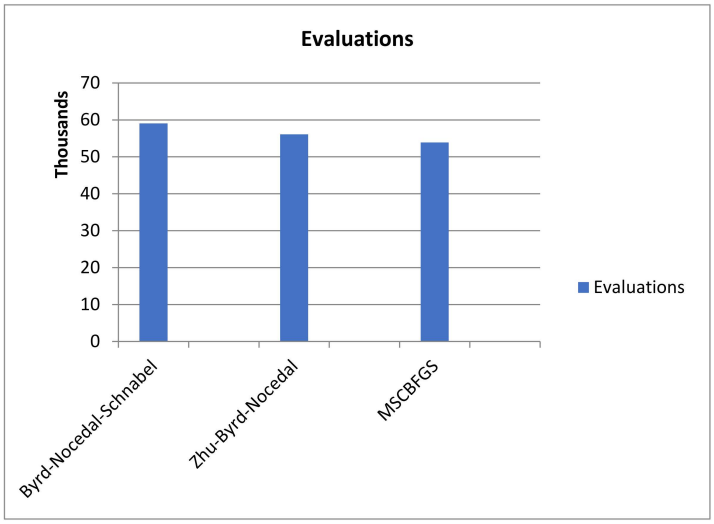

Figure 1. Overall evaluations for 900 problems.

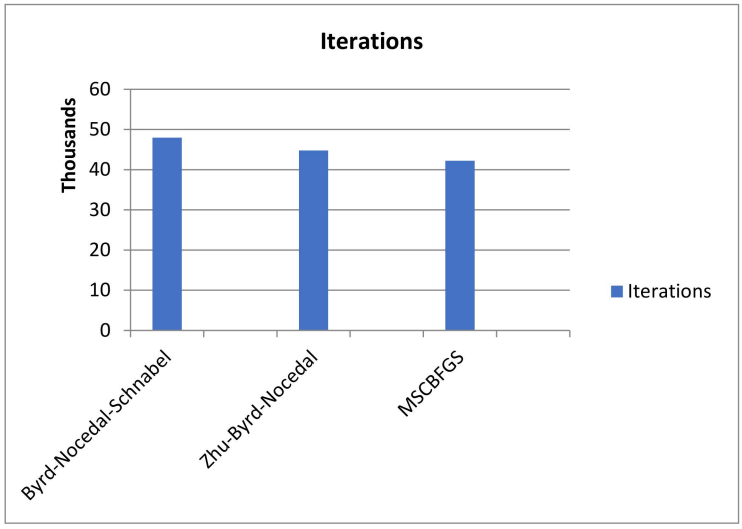

Figure 2. Overall iterations for 900 problems.

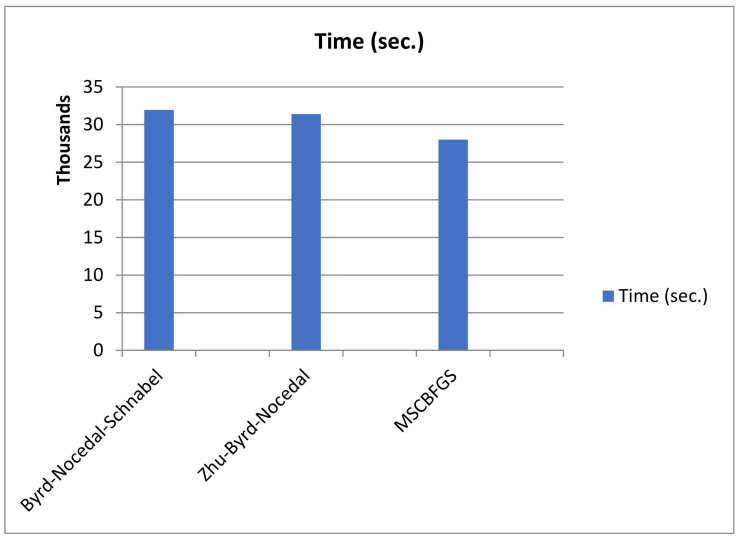

Figure 3. Overall timings for 900 problems.

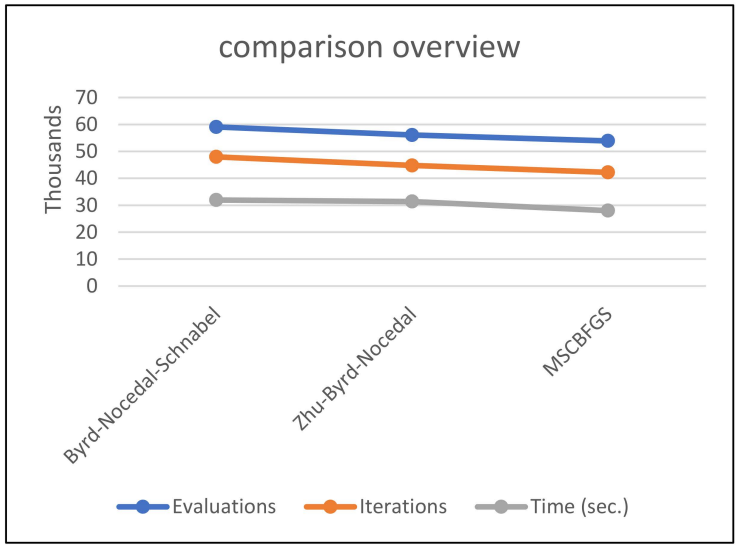

Figure 4. Performance overview of all criteria.

The analysis of the numerical scores for the newly developed limited memory MSCBFGS method in comparison to the Byrd et al. [13] and Zhu and Byrd [14] methods reveals notable improvements in efficiency and effectiveness. In terms of evaluations, the MSCBFGS method requires only 91.21% of the evaluations compared to the Byrd et al. [13] method, indicating enhanced computational efficiency. Additionally, it achieves a further reduction in iterations, requiring only 88.01% of the iterations compared to the original method, demonstrating improved convergence properties. Moreover, the MSCBFGS method significantly reduces computation time, utilizing only 87.65% of the time compared to the Byrd et al. [13] method, while outperforming both the original method and the Zhu and Byrd method in terms of overall effectiveness, as evidenced by its higher score of 369 compared to 232 and 299, respectively. Overall, these results suggest that the newly developed MSCBFGS method offers substantial improvements in efficiency and effectiveness, making it a promising approach for optimization problems.

5. Conclusions

In conclusion, this paper has presented a contribution to the field of optimization through the development of a novel quasi-Newton method known as memory-sensitive or limited BFGS (MSCBFGS). The method has been tailored specifically to address the challenges posed by memory constraints in solving complex large-scale problems. By allowing for flexibility in the choice of a storage variable that governs the retention of past data in the formation of the new Hessian approximation, MSCBFGS exhibits adaptability and versatility.

The outcomes of extensive numerical experiments have underscored the efficacy of the MSCBFGS algorithm. These findings not only affirm the algorithm's potential but also pave the way for its practical application in a range of memory-limited optimization tasks.

One notable advantage of the proposed method lies in its adaptability to varying memory constraints, offering a customizable parameter for controlling the retention of past data. This flexibility enables the algorithm to effectively navigate memory-limited environments, enhancing its applicability to a wide range of optimization problems.

6. Discussion

By utilizing a modified version of the full memory multi-step BFGS update, the method derived in this paper maintains a balance between computational efficiency and accuracy in approximating the Hessian matrix. However, it is crucial to acknowledge certain limitations. One such drawback is the potential sensitivity of the algorithm's performance to the selection of parameters, which may require careful tuning for optimal results. Additionally, while the method demonstrates improved effectiveness within constrained memory contexts, its performance in comparison to alternative approaches warrants further investigation, particularly in scenarios with highly complex or nonlinear objective functions. Overall, the presented method offers promising advancements in addressing memory limitations in quasi-Newton optimization methods, yet ongoing research is essential to fully understand its capabilities and limitations in diverse optimization scenarios.

Future research can deeper investigate the optimization of the memory parameter selection process within MSCBFGS. Exploring adaptive or learning-based techniques to dynamically adjust this parameter during optimization could enhance the algorithm's performance further, especially when taking the nature of the problem being solved into account [30].

Author Contributions: Conceptualization, I.A.R.M.; methodology, I.A.R.M.; software, B.A.H.; validation, B.A.H. and I.A.R.M.; formal analysis, B.A.H. and I.A.R.M.; investigation, I.A.R.M.; writing—original draft preparation, B.A.H. and I.A.R.M.; writing—review and editing, I.A.R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Kuwait Technical College, Kuwait.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Davidon, W.C. Variable metric methods for minimization. *Argonne Natl. Labs. Rep.* **1971**, *2*, 1–11.
2. Powell, M.J.D. Some Global Convergence Properties of a variable Metric Algorithm for Minimization without Exact Line Searches. In *Nonlinear Programming, SIAM-AMS Proceedings*; Cottle, R.W., Lemke, C.E., Eds.; American Mathematical Society: Providence, RI, USA, 1976; Volume 4, pp. 53–72.
3. Broyden, C.G. The convergence of a class of double-rank minimization algorithms—Part 2: The new algorithm. *J. Inst. Math. Appl.* **1970**, *6*, 222–231. [CrossRef]
4. Dennis, J.E.; Schnabel, R.B. Least change Secant updates for quasi-Newton methods. *SIAM Rev.* **1979**, *21*, 443–459. [CrossRef]
5. Fletcher, R. A new approach to variable metric algorithms. *Comput. J.* **1970**, *13*, 317–322. [CrossRef]
6. Fletcher, R. An Overview of Unconstrained Optimization. In *Mathematical and Physical Sciences*; Springer: Dordrecht, The Netherlands, 1994; Volume 434.
7. Fletcher, R. *Practical Methods of Optimization*; Wiley: Hoboken, NJ, USA, 1987.
8. Oren, S.S.; Luenberger, D. Self-scaling variable metric (SSVM) algorithms V1. *Manag. Sci.* **1974**, *20*, 845–862. [CrossRef]
9. Liu, D.C.; Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **1989**, *45*, 503–528. [CrossRef]
10. Nocedal, J. Updating Quasi-Newton Matrices with Limited Storage. *Math. Program.* **1980**, *35*, 773–782. [CrossRef]
11. Yuan, G.; Wei, Z.; Wu, Y. Modified limited memory BFGS method with non-monotone line search for unconstrained optimization. *J. Korean Math. Soc.* **2010**, *47*, 767–788. [CrossRef]
12. Zhang, J.; Deng, N.; Chen, L. Quasi-Newton equation and related methods for unconstrained optimization. *J. Optim. Theory Appl.* **1999**, *102*, 147–167. [CrossRef]
13. Byrd, R.H.; Nocedal, J.; Schnabel, R.B. Representations of quasi-Newton matrices and their use in large scale optimization. *SIAM J. Optim.* **1994**, *4*, 677–696.
14. Zhu, C.; Byrd, R.H.; Nocedal, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.* **1997**, *23*, 550–560. [CrossRef]
15. Biggs, M.C. Minimization algorithms making use of non-quadratic properties of the objective function. *J. Inst. Math. Its Appl.* **1971**, *8*, 315–327. [CrossRef]
16. Kanzow, C.; Steck, D. Regularization of limited memory quasi-Newton methods for large-scale nonconvex minimization. *Math. Program. Comput.* **2023**, *15*, 417–444. [CrossRef]
17. Lapucci, M.; Mansueto, P. A limited memory Quasi-Newton approach for multi-objective optimization. *Comput. Optim. Appl.* **2023**, *85*, 33–73. [CrossRef]
18. Wei, Z.; Li, G.; Qi, L. The superlinear convergence of a modified BFGS- type method for unconstrained optimization. *Comput. Optim. Appl.* **2004**, *29*, 315–332. [CrossRef]
19. Xiao, Y.H.; Wei, Z.X.; Zhang, L. A modified BFGS method without line searches for nonconvex unconstrained optimization. *Adv. Theor. Appl. Math.* **2006**, *1*, 149–162.
20. Lai, K.K.; Shashi, K.M. Ravina Sharma, Manjari Sharma & Bhagwat Ram. A Modified q-BFGS Algorithm for Unconstrained Optimization. *Mathematics* **2023**, *11*, 1420–1431.
21. Lai, K.K.; Mishra, S.K.; Panda, G.; Chakraborty, S.K.; Samei, M.E.; Ram, B. A limited memory q-BFGS algorithm for unconstrained optimization problems. *J. Appl. Math. Comput.* **2021**, *66*, 183–202. [CrossRef]
22. Ford, J.A.; Moghrabi, I.A.R. Multi-step quasi-Newton methods for optimization. *J. Comput. Appl. Math.* **1994**, *50*, 305–323. [CrossRef]
23. Moghrabi, I.A.R. A non-Secant quasi-Newton Method for Unconstrained Nonlinear Optimization. *Cogent Eng.* **2022**, *9*, 11–25. [CrossRef]
24. Ford, J.A.; Moghrabi, I.A. Alternative Parameter Choices for Multi-Step Quasi-Newton Methods. *Optim. Methods Softw.* **1993**, *2*, 357–370. [CrossRef]
25. Wolfe, P. Convergence conditions for ascent methods. II: Some corrections. *SIAM Rev.* **1971**, *3*, 185–188. [CrossRef]
26. Xiao, Y.; Wei, Z.; Wang, Z. A limited memory BFGS-type method for large-scale unconstrained optimization. *Comput. Math. Appl.* **2008**, *56*, 1001–1009. [CrossRef]
27. Anderi, N. An Unconstrained Optimization Test functions collection. *Adv. Model. Optim.* **2008**, *10*, 147–161.
28. Moré, J.J.; Garbow, B.S.; Hillstrom, K.E. Testing unconstrained optimization software. *ACM Trans. Math. Softw.* **1981**, *7*, 17–41. [CrossRef]
29. Yuan, Y.; Sun, W. *Optimization Theory and Methods*; Science Press: Beijing, China, 1999.
30. Jin, Q.; Mokhtari, A. Non-asymptotic super linear convergence of standard quasi-Newton methods. *J. Strateg. Decis.* **2021**, *121*, 11–28.