A Sparse Quasi-Newton Method Based on Automatic Differentiation for Solving Unconstrained Optimization Problems

Huiping Cao 1,* and Xiaomin An 2

1 School of Science, Xi’An Polytechnic University, Xi’an 710048, China
2 School of Science, Xi’An Technological University, Xi’an 710021, China; anxiaomin@xatu.edu.cn
* Correspondence: huiping_cao@hnu.edu.cn

Abstract: In our paper, we introduce a sparse and symmetric matrix completion quasi-Newton model using automatic differentiation, for solving unconstrained optimization problems where the sparse structure of the Hessian is available. The proposed method is a kind of matrix completion quasi-Newton method and has some nice properties. Moreover, the presented method keeps the sparsity of the Hessian exactly and satisfies the quasi-Newton equation approximately. Under the usual assumptions, local and superlinear convergence are established. We tested the performance of the method, showing that the new method is effective and superior to matrix completion quasi-Newton updating with the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method and the limited-memory BFGS method.

Keywords: symmetric quasi-Newton method; unconstrained optimization problems; matrix completion; automatic differentiation; superlinear convergence; Broyden–Fletcher–Goldfarb–Shanno method

MSC: 65K05; 90C06; 90C53

1. Introduction

We concentrated on the unconstrained optimization problem

$$\min f(x), \quad x \in \mathbb{R}^n, \quad (1)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a twice continuously differentiable function; and $\nabla f(x)$ and $\nabla^2 f(x)$ denote the gradient and Hessian of $f$ at $x$, respectively. The first order necessary condition of (1) is

$$\nabla f(x) = 0,$$

which can be written as the symmetric nonlinear equations

$$F(x) = 0, \quad (2)$$

where $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a continuously differentiable mapping and the symmetry implies that the Jacobian $F'(x)$ satisfies $F'(x) = F'(x)^T$. That symmetric nonlinear system has close relationships with many practical problems, such as the gradient mapping of unconstrained optimization problems, the Karush–Kuhn–Tuckrt (KKT) system of equality constrained optimization problem, the discretized two-point boundary value problem, and the saddle point problem (2) [1–5].

For small or medium-scale problems, classical quasi-Newton methods enjoy superlinear convergence without the calculation of the Hessian [6,7]. Let $x_k$ be the current
iterative point and $B_k$ be the symmetric approximation of the Hessian; then the iteration \{x_k\} generated by quasi-Newton methods is

$$x_{k+1} = x_k + \alpha_k d_k \nabla f(x_k),$$

where $\alpha_k > 0$ is a step length obtained by some line search or other strategies. The search direction $d_k$ can be gotten by solving the equations

$$B_k d_k + \nabla f(x_k) = 0,$$

where the quasi-Newton matrix $B_k$ is an approximation of $\nabla^2 f(x_k)$ and satisfies the secant condition:

$$B_{k+1}s_k = y_k,$$

where $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. The matrix $B_k$ can be updated by different update formulae. The Davidon–Fletcher–Powell (DFP) update,

$$B_{k+1} = \left( I - \frac{y_k s_k^T}{y_k s_k} \right) B_k \left( I - \frac{y_k s_k^T}{y_k s_k} \right)^T + \frac{y_k y_k^T}{y_k s_k},$$

was first proposed by Davidon [8] and developed by Fletcher and Powell [9]. The Broyden–Fletcher–Goldfarb–Shanno (BFGS) update,

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

was proposed independently by Broyden [10], Fletcher [11], Goldfarb [12], and Shanno [13]. One can find more on the topic in references [14–17].

If we assume that $H_k = B_k^{-1}$, then using Sherman–Morrison formula, we have the Broyden’s family update:

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{s_k^T y_k} + \phi_k v_k v_k^T,$$

where $\phi_k \in [0, 1]$ is

$$\phi_k = \sqrt{y_k^T H_k y_k \left( \frac{s_k}{s_k^T y_k} - \frac{H_k y_k}{y_k^T H_k y_k} \right)}. $$

When $\phi_k \equiv 1$, we have a BFGS update. When $\phi_k \equiv 0$, we have a DFP update.

However, a quasi-Newton method is not desirable when applied to solve large-scale problems, because we need to store the full matrix $B_k$. To overcome such drawback, the so-called sparse quasi-Newton methods [14] have received much attention. Early in 1970, Schubert [18] has proposed a sparse Broyden’s rank one method. Then Powell and Toint [19], Toint [20] studied the sparse quasi-Newton method.

Existing sparse quasi-Newton methods usually use a sparse symmetric matrix as an approximation of the Hessian so that both matrices take the same form or have similar structures. If the limited memory technique [21,22] is adopted, which only stores several pairs $(s_k, y_k)$ to construct a matrix $H_k$ by updating the initial matrix $H_0$ $m$ times, the method
can be widely used in practical optimization problems. On the other hand, there are many large-scale problems in scientific fields take the partially separable form

\[ f(x) = \sum_{i=1}^{m} f_i(x), \]

where function \( f_i, i = 1, \ldots, m \) is related to a few variables. For the partially separable unconstrained optimization problems, the partitioned BFGS method [23,24] was proposed and has better performance in practice. The partitioned BFGS method updates each matrix \( B^i_k \) of each element function \( f_i(x) \) separately via BFGS updating and sums these matrices to construct the next quasi-Newton matrix \( B_{k+1} \). Since the size of \( x \) in \( f_i(x) \) is smaller than that of \( n \), the matrix \( B^i_{k+1} \) will be a small matrix, and then the matrix \( B_{k+1} \) will be sparse. The quasi-Newton direction is the solution of the linear equations:

\[ \left( \sum_{i=1}^{m} B^i_k \right) d_k = -\nabla f(x_k). \]

However, the partitioned BFGS method cannot always preserve the positive definite-ness of the matrix \( B_k \) only if that each element function \( f_i(x) \) is convex, so the partitioned BFGS method is implemented with the trust region strategy [25]. Recently, for the partially separable nonlinear equations, Cao and Li [26] have introduced two kinds of partitioned quasi-Newton methods and given their global and superlinear convergence.

Another efficient sparse quasi-Newton method is designed to exploit the sparsity structures of the Hessian. We assume that for all \( x \in \mathbb{R}^n \),

\[ (\nabla^2 f(x))_{ij} = 0, \quad (i, j) \in F, \]

where \( F \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\} \). References [27,28] have proposed sparse quasi-Newton methods, where \( H_{k+1} \) satisfies the secant equation

\[ H_{k+1} y_k = s_k \]

and sparse condition

\[ (H_{k+1})_{ij} = 0, \quad (i, j) \in F \]

simultaneously, where \( H_{k+1} \) is an approximate inverse Hessian. Recently, Yamashita [29] proposed another type of matrix completion quasi-Newton (MCQN) update for solving problem (1) with a sparse Hessian and proved the local and superlinear convergence for MCQN updates with the DFP method. Reference [30] established the convergence of MCQN updates with all of Broyden’s convex family. However, global convergence analysis [31] was presented for two-dimensional functions with uniformly positive definite Hessians.

Another kind of quasi-Newton method for solving large scale unconstrained optimization problems is the diagonal quasi-Newton method, where the Hessian of an objective function is approximated by a diagonal matrix with positive elements. The first version was developed by Nazareth [32], where the quasi-Newton matrix satisfies the least change and weak secant condition [33]:

\[ \min \| H_{k+1} - H_k \| \]

s.t.\[ y_k^T H_{k+1} y_k = y_k^T s_k, \]

where \( \| \cdot \|_F \) is the standard Frobenius norm. Recently, Andrei N. [34] developed a diagonal quasi-Newton method, where the diagonal elements satisfy the least change weak secant condition (3) and minimize the trace of the update. Besides, lots of other techniques, such as forward and central finite differences, the variational principle with a weighted norm, and the generalized Frobenius norm, can be used to derive different kinds of diagonal quasi-
Newton method [35–37]. Under usual assumptions, the diagonal quasi-Newton method is linearly convergent. The authors of [38] adopted a similar technique to derivation with the DFP method and got a low memory diagonal quasi-Newton method. Using the Armijo line search, they established the global convergence and gave the sufficient conditions for the method to be superlinearly convergent.

The main contribution of our paper is to propose a sparse quasi-Newton algorithm based on automatic differentiation for solving (1). Firstly, similarly to the derivation of BFGS update, we can perform a symmetric rank-two quasi-Newton update:

\[ B_{k+1} = B_k - \frac{B_k \sigma_k \sigma_k^T B_k}{\sigma_k^T B_k \sigma_k} + \frac{\nabla^2 f(x_{k+1}) \sigma_k \sigma_k^T \nabla^2 f(x_{k+1})}{\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k}, \]

where \( \sigma_k \in \mathbb{R}^n \) and \( B_{k+1} \) satisfying the adjoint tangent condition [39]

\[ \sigma_k^T B_{k+1} = \sigma_k^T \nabla^2 f(x_{k+1}). \]

For an \( n \times n \) matrix, we denote \( A \succeq 0 \), as \( A \) is positive definite. Then, when \( B_k \succeq 0 \), \( B_{k+1} \succeq 0 \) if and only if \( \sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k > 0 \), which means that the proposed update (4) keeps the positive definiteness, as in BFGS updating. Moreover, when \( B_0 \) is positive definite, the matrices \( \{B_k\} \) updated by the proposed update (4) are positive definite for solving (1) with uniformly positive definite Hessians. In our work, we pay attention to \( \sigma_k = s_k \); then the proposed rank-two quasi-Newton update (4) method satisfies

\[ B_{k+1}s_k = \nabla^2 f(x_{k+1})s_k, \]

which means that \( B_{k+1} \) equals \( \nabla^2 f(x_{k+1}) \) in the direction \( s_k \) exactly. Several lemmas have been given to present the properties of the proposed rank-two quasi-Newton update formula. Secondly, combined with the idea of MCQN method [29], we propose a sparse and symmetric quasi-Newton algorithm for solving (1). Under appropriate conditions, local and superlinear convergence are established. Finally, our numerical results illustrate that the proposed algorithm has satisfying performance.

The paper is organized as follows. In Section 2, we introduce a symmetric rank-two quasi-Newton update based on automatic differentiation and prove several nice properties. In Section 3, by using the idea of matrix completion, we present a sparse quasi-Newton algorithm and show some nice properties. In Section 4, we prove the local and superlinear convergence of the algorithm proposed in Section 3. Numerical results are listed in Section 5, which verify that the proposed algorithm is very encouraging. Finally, we give the conclusion.

2. A New Symmetric Rank-Two Quasi–Newton Update

Similarly to the derivation of BFGS update, we will derive a new symmetric rank-two quasi-Newton update and show several lemmas. Let

\[ B_{k+1} = B_k + \Delta_k, \]

where \( \Delta_k \) is a rank-two matrix and \( B_{k+1} \) satisfies the condition

\[ \sigma_k^T B_{k+1} = \sigma_k^T \nabla^2 f(x_{k+1}), \]

where \( \sigma_k \in \mathbb{R}^n \) and \( \sigma_k \neq 0 \). Similarly to the derivation of BFGS, we have the following symmetric rank-two update:

\[ B_{k+1} = B_k - \frac{B_k \sigma_k \sigma_k^T B_k}{\sigma_k^T B_k \sigma_k} + \frac{\nabla^2 f(x_{k+1}) \sigma_k \sigma_k^T \nabla^2 f(x_{k+1})}{\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k}. \]
If we denote $H_k = B_k^{-1}$ and $H_{k+1} = B_{k+1}^{-1}$, then (6) can be expressed as

$$H_{k+1} = H_k - \frac{H_k \nabla^2 f(x_{k+1}) \sigma_k \sigma_k^T + \sigma_k \sigma_k^T \nabla^2 f(x_{k+1}) H_k}{\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k} + \left(1 + \frac{\sigma_k^T \nabla^2 f(x_{k+1}) H_k \nabla^2 f(x_{k+1}) \sigma_k}{\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k}\right) \frac{\sigma_k \sigma_k^T}{\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k}. \quad (7)$$

It can be seen that the update (6) involves the Hessian $\nabla^2 f(x)$, but we do not need to compute them in practice. For given vectors $x, s$, and $\sigma$, we can get $\nabla^2 f(x) s$ and $\sigma^T \nabla^2 f(x)$ exactly by the forward and reverse mode of automatic differentiation.

Next, several lemmas are presented.

**Lemma 1.** Suppose that $B_k \succeq 0$ and $B_{k+1}$ is updated by (6); then $B_{k+1} \succeq 0$ if and only if $\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k > 0$.

**Proof.** According to the condition (5), one has

$$\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k = \sigma_k^T B_{k+1} \sigma_k.$$ 

If $B_{k+1}$ is positive definite, one has $\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k > 0$.

Let $\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k > 0$ and $B_k \succeq 0$. Then for $\forall d_k \in R^n$, $d_k \neq 0$, it can be derived from (6) that

$$d_k^T B_{k+1} d_k = d_k^T B_k d_k - \frac{(d_k^T B_k \sigma_k)^2}{\sigma_k^T B_k \sigma_k} + \frac{(d_k^T \nabla^2 f(x_{k+1}) \sigma_k)^2}{\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k}.$$ 

According to that $B_k \succeq 0$, there is a symmetric matrix $B^{1/2}_k \succeq 0$, such that $B_k = B^{1/2}_k B^{1/2}_k$. Then we have from Cauchy–Schwarz inequality that

$$d_k^T B_k d_k \leq \|B^{1/2}_k d_k\|^2 \cdot \|B^{-1/2}_k \sigma_k\|^2 = (B_k^{1/2} d_k)^T (B_k^{1/2} \sigma_k), \quad (8)$$

where the equality holds if and only if $d_k = \lambda_k \sigma_k$, $\lambda_k \neq 0$.

If the inequality (8) holds strictly, one has

$$d_k^T B_{k+1} d_k > d_k^T B_k d_k - d_k^T B_k d_k + \frac{(d_k^T \nabla^2 f(x_{k+1}) \sigma_k)^2}{\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k} \geq 0.$$ 

If the equality (8) holds; i.e., there exists a $\lambda_k \neq 0$ such that $d_k = \lambda_k \sigma_k$, then it can be deduced from (8) that

$$d_k^T B_{k+1} d_k \geq \frac{(d_k^T \nabla^2 f(x_{k+1}) \sigma_k)^2}{\sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k} = \lambda_k^2 \sigma_k^T \nabla^2 f(x_{k+1}) \sigma_k > 0.$$ 

In conclusion, $d_k^T B_{k+1} d_k > 0$ for $\forall d_k \in R^n$ and $d_k \neq 0$. \square

**Lemma 2.** If we rewrite update Formula (7) as $H_{k+1} = H_k + E$, where $H_k$ is symmetric and satisfies $\sigma_k^T = \sigma_k^T \nabla^2 f(x_{k+1}) H_k$, then $E$ is the solution of the following minimization problem:

$$\min_E \quad \|E\|_W$$

s.t. \quad $E^T = E,$ \quad $\sigma_k^T \nabla^2 f(x_{k+1}) E = \eta^T$, \quad
where \( \eta = \sigma_k^T - \sigma_k^T \nabla^2 f(x_{k+1})H_k \) and \( W \) satisfies \( \sigma_k^T W = \sigma_k^T \nabla^2 f(x_{k+1}) \).

**Proof.** A suitable Lagrangian function of the convex programming problem is

\[
\varphi = \frac{1}{4} \text{trace}(WE^TWE) + \text{trace}(\Lambda^T(E^T - E)) - \lambda^T W(\nabla^2 f(x_{k+1})\sigma_k - \eta),
\]

where \( \Lambda \) and \( \lambda \) are Lagrange multipliers. Moreover,

\[
\frac{\partial \varphi}{\partial E_{ij}} = \frac{1}{4} (\text{trace}(W e_i e_j^T W E) + \text{trace}(W E^T W e_i e_j^T)) + \text{trace}(\Lambda(e_i e_j^T - e_j e_i^T)) - \lambda^T W e_i e_j^T F'(x_{k+1})\sigma_k = 0,
\]

or according to the symmetry and cyclic permutations, one has

\[
\frac{1}{2} [WEW]_{ij} + \Lambda_{ij} - \Lambda_{ji} = [W\lambda \sigma_k^T \nabla^2 f(x_{k+1})]_{ij}.
\]

Taking the transpose and accumulating eliminates \( \Lambda \) to yield

\[
WEW = W\lambda \sigma_k^T \nabla^2 f(x_{k+1}) + \nabla^2 f(x_{k+1})\sigma_k \lambda^T W,
\]

and by \( \sigma_k^T W = \sigma_k^T \nabla^2 f(x_{k+1}) \) and the nonsingularity of \( W \) we have that

\[
E = \lambda \sigma_k^T + \sigma_k \lambda^T.
\]

Substituting (9) into \( \sigma_k^T \nabla^2 f(x_{k+1})E = \eta^T \) and rewriting gives

\[
\lambda = \frac{\eta - \sigma_k \lambda^T \nabla^2 f(x_{k+1})\sigma_k}{\sigma_k^T \nabla^2 f(x_{k+1})\sigma_k}.
\]

Postmultiplying by \( \sigma_k^T \nabla^2 f(x_{k+1}) \) gives

\[
\lambda^T \nabla^2 f(x_{k+1})\sigma_k = \frac{1}{2} \frac{\sigma_k^T \nabla^2 f(x_{k+1})\eta}{\sigma_k^T \nabla^2 f(x_{k+1})\sigma_k},
\]

so we have

\[
\lambda = \frac{\eta - \frac{1}{2} \frac{\sigma_k^T \nabla^2 f(x_{k+1})\eta}{\sigma_k^T \nabla^2 f(x_{k+1})\sigma_k}}{\sigma_k^T \nabla^2 f(x_{k+1})\sigma_k} = \frac{H_k \nabla^2 f(x_{k+1})\sigma_k - \frac{1}{2} \frac{\sigma_k^T \nabla^2 f(x_{k+1})H_k \nabla^2 f(x_{k+1})\sigma_k}{\sigma_k^T \nabla^2 f(x_{k+1})\sigma_k}}{\sigma_k^T \nabla^2 f(x_{k+1})\sigma_k}.
\]

Substituting this into (9) gives the result (7). \( \square \)

**Lemma 3.** If \( H_k = B_k^{-1} > 0 \) and \( \sigma_k^T \nabla^2 f(x_{k+1})\sigma_k > 0 \). Then \( B_{k+1} \) given by (6) solves the variational problem

\[
\min_{B > 0} \psi(H_k^{1/2} BH_k^{1/2})
\]

s.t. \( B^T = B \), \( \sigma_k^T B = \sigma_k^T \nabla^2 f(x_{k+1}) \).

**Proof.** According to the definition of \( \psi \), where \( \psi : \mathbb{R}^{n \times n} \to \mathbb{R} \) [40] is given by

\[
\psi(A) = \text{tr}(A) - \ln \det(A),
\]

(10)
so we have
\[ \psi(H_k^{1/2}B_k^{1/2}) = \text{trace}(H_kB) - \ln(\det H_k\det B) = \psi(H_kB) = \psi(BH_k). \]  
(11)

We have the Lagrangian function
\[ L(B, \Lambda, \lambda) = \frac{1}{2} \psi(H_k^{1/2}B_k^{1/2}) + \text{trace}(\Lambda^T(B^T - B) + (\sigma_k^T B - \sigma_k^T \nabla^2 f(x_{k+1})) \lambda_k \]
\[ + \text{trace}(\Lambda^T(B^T - B) + (\sigma_k^T B - \sigma_k^T \nabla^2 f(x_{k+1})) \lambda_k, \]

where \( \Lambda \) and \( \lambda \) are the Lagrange multipliers. Moreover, one has
\[ \frac{\partial L}{\partial B_{ii}} = \frac{1}{2} \text{trace} \left( H_k e_i e_i^T - (B^{-1})_{ii} \right) + \text{trace} \left( \Lambda^T(e_i^T e_i^T - e_i e_i^T) \right) + \sigma_k^T e_i e_i^T \lambda \]
\[ = \frac{1}{2} (H_k)_{ii} - (B^{-1})_{ii} + \Lambda_{ii} - \Lambda_{ij} + (\sigma_k^T \lambda)_{ii} = 0. \]  
(12)

Transposing and adding in (12) that
\[ H_k - B^{-1} + \sigma_k^T \lambda + \lambda^T \sigma_k = 0, \]
\[ B^{-1} = H_k + \sigma_k^T \lambda + \lambda^T \sigma_k. \]  
(13)

Combined with the tangent condition, we have that
\[ \sigma_k^T = e_k^T \nabla^2 f(x_{k+1}) H_k + e_k^T \nabla^2 f(x_{k+1}) \lambda \sigma_k + e_k^T \nabla^2 f(x_{k+1}) \sigma_k \lambda^T, \]
and hence
\[ e_k^T \nabla^2 f(x_{k+1}) \sigma_k = \frac{1}{2} \left( 1 - \frac{e_k^T \nabla^2 f(x_{k+1}) H_k \nabla^2 f(x_{k+1}) e_k}{e_k^T \nabla^2 f(x_{k+1}) \sigma_k} \right), \]
and so
\[ \lambda = \frac{\sigma_k - H_k \nabla^2 f(x_{k+1}) - \frac{1}{2} \left( 1 - \frac{e_k^T \nabla^2 f(x_{k+1}) H_k \nabla^2 f(x_{k+1}) e_k}{e_k^T \nabla^2 f(x_{k+1}) \sigma_k} \right)}{e_k^T \nabla^2 f(x_{k+1}) \sigma_k}. \]

Combined with (6), one has the Formula (7).

According to the Sherman–Morrison formula, (7) is equivalent to (6). Since the function \( \psi(H_k^{1/2}B_k^{1/2}) \) is strictly convex on \( B \succeq 0 \), the update formula (6) is the unique solution of the variational problem. □

In this paper, we set \( \sigma_k = s_k \), so one has
\[ B_{k+1}s_k = \nabla^2 f(x_{k+1}) s_k, \]
which means that \( B_{k+1} \) is an exact approximation to \( \nabla^2 f(x_{k+1}) \) in direction \( s_k \). Then we have the symmetric rank-two update formula
\[ B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\nabla^2 f(x_{k+1}) s_k s_k^T \nabla^2 f(x_{k+1})}{s_k^T \nabla^2 f(x_{k+1}) s_k}. \]  
(14)

It can be seen that \( B_{k+1} \) can preserve the symmetry when \( B_k \) is symmetric. If we denote \( w_k = \nabla^2 f(x_{k+1}) s_k \), then we can obtain a similar Broyden convex family update formula:
\[ H_{k+1} = H_k - \frac{H_k w_k w_k^T H_k}{w_k^T H_k w_k} + \frac{s_k s_k^T}{s_k^T w_k} + \phi_k v_k v_k^T, \]  
(15)
where the parameter $\phi_k \in [0, 1]$ is defined as

$$v_k = \sqrt{w_k^T H_k w_k} \left( \frac{s_k}{s_k^T w_k} - \frac{H_k w_k}{w_k^T H_k w_k} \right).$$

(16)

The choice $\phi_k \equiv 1$ corresponds to the BFGS update

$$H_{k+1} = H_k + \left( 1 + \frac{w_k^T H_k w_k}{s_k^T w_k} \right) \frac{s_k^T s_k}{s_k^T w_k} - \frac{\bar{s}_k^T w_k H_k + H_k w_k s_k^T}{s_k^T w_k},$$

$$= H_k + \frac{(s_k - H_k w_k)s_k^T + s_k(s_k - H_k w_k)^T}{s_k^T w_k}.$$

(17)

3. Algorithm and Related Properties

For the update formula (15), we adopt the idea of matrix completion. The next quasi-Newton matrix $H_{k+1}$ is the solution of the following minimization problem:

$$\begin{align*}
\min & \quad \psi(H_k^{-1/2} HH_k^{-1/2}) \\
\text{s.t.} & \quad H_{ij} = H_{ij}^{AD}, (i, j) \in F, \\
& \quad (H^{-1})_{ij} = 0, (i, j) \notin F, \\
& \quad H^T = H, H \succeq 0.
\end{align*}$$

(18)

When $G(V, \bar{F})$ is chordal, the minimization problem (18) can be solved by solving the problem

$$\begin{align*}
\max & \quad \det(H) \\
\text{s.t.} & \quad H_{ij} = H_{ij}^{AD}, (i, j) \in F, \\
& \quad H^T = H, H \succeq 0.
\end{align*}$$

(19)

Then $H_{k+1}$ can be expressed as the sparse clique-factorization formula [29]. Then Algorithm 1 is stated as follows.

**Algorithm 1** (Sparse Quasi-Newton Algorithm based on Automatic Differentiation)

**Step 0.** Compute $\bar{F}$ according to $F$ such that $G(V, \bar{F})$ is a chordal graph, where $V = \{1, 2, \cdots, n\}$. Choose $x_0 \in R^n, \epsilon > 0$ and a matrix $H_0 \in R^{n \times n}, H_0 \succeq 0$ with $(H_0^{-1})_{ij} = 0, \forall (i, j) \notin F$. Let $k := 0$.

**Step 1** If $\|\nabla f(x_k)\| \leq \epsilon$, stop.

**Step 2** $x_{k+1} = x_k - H_k \nabla f(x_k)$.

**Step 3** Update $H_k$ to get $H_k^{AD}, \phi_k \in [0, 1], (i, j) \in F$ by update formula (15).

**Step 4** Get $H_{k+1}$ by the minimization problem (18). When $G(V, \bar{F})$ is a chordal graph, the problem (18) can be solved by solving the problem (19).

**Step 5** Let $k := k + 1$, go to Step 1.

When the $H_k$ in step 3 is updated by Broyden’s class method, the method corresponds to the method in [29]. In the present paper, we focus on the MCQN update with $H^{AD} = H_{k+1}$, where $H_{k+1}$ is given by (15).

In what follows, we give some notation for the convenience of analysis. For a nonsingular matrix $P$ satisfying

$$(P^{-1})_{ij} = 0, \forall (i, j) \in F,$$

(20)

we let

$$s_k = P^{-1/2} s_k, \quad w_k = P^{1/2} w_k, \quad H_k = P^{-1/2} H_k P^{-1/2}, \quad H_k^{AD} = P^{-1/2} H_k^{AD} P^{-1/2}.$$
where $H^{AD} = H_{k+1}$ is given by (15). Then we can get from (15) that

$$
\begin{align*}
\hat{H}^{AD} &= \hat{H}_k - \frac{\hat{H}_k \omega_k \omega_k^T \hat{H}_k}{\omega_k^T \hat{H}_k \omega_k} + \frac{s_k s_k^T}{s_k^T \omega_k} + \phi_k \sigma_k^T, \\
&\text{where} \\
\sigma_k &= \sqrt{\omega_k^T \hat{H}_k \omega_k} \left( \frac{s_k}{s_k^T \omega_k} - \frac{\hat{H}_k \omega_k}{\omega_k^T \hat{H}_k \omega_k} \right).
\end{align*}
$$

Similarly to that in [30], we can assume that

$$
\begin{align*}
\tau_k &= \frac{\bar{w}_k^T \hat{H}_k \bar{w}_k}{\|\bar{w}_k\| \|\hat{H}_k \bar{w}_k\|}, \\
q_k &= \frac{\bar{w}_k^T \hat{H}_k \bar{w}_k}{\|\bar{w}_k\|^2}, \\
\eta_k &= \frac{\bar{s}_k^T \hat{H}_k \bar{w}_k}{\|\bar{s}_k\|}, \\
m_k &= \frac{\bar{s}_k^T \hat{H}_k \bar{w}_k}{\|\bar{w}_k\|^2}, \\
M_k &= \frac{\|\bar{s}_k\|^2}{\bar{s}_k^T \omega_k}, \\
\beta_k &= \frac{s_k^T \hat{H}_k^{-1} s_k^T}{s_k^T \bar{s}_k}, \\
\gamma_k &= \frac{\bar{w}_k^T \hat{H}_k \bar{w}_k}{\|\bar{w}_k\|^2}.
\end{align*}
$$

According to [41] and (21), we have

$$
\begin{align*}
\text{tr}(\hat{H}^{AD}) &= \text{tr}(\hat{H}_k) - (1 - \phi_k) \frac{q_k}{\tau_k} - 2 \phi_k \eta_k + \left( 1 + \phi_k \frac{q_k}{m_k} \right) M_k \\
&= \text{tr}(\hat{H}_k) \left( 1 + \phi_k (\beta_k \gamma_k - 1) \right) / \gamma_k.
\end{align*}
$$

Next, we establish a relation between $\hat{H}_{k+1}$ and $\hat{H}^{AD}$, which is very important in the establishment of the local and superlinear convergence of Algorithm 1.

**Proposition 1.** For the Algorithm 1, we have the following relation:

$$
\text{tr}(\hat{H}_{k+1}) = \text{tr}(\hat{H}^{AD}), \quad \det(\hat{H}_{k+1}) \geq \det(\hat{H}^{AD}).
$$

**Proof.** We can obtain from (18) that

$$(H_{k+1})_{ij} = (H^{AD})_{ij}, \quad \forall (i, j) \in F.$$  

Combined with (20), one has that for any $(i, j) \in F$, there at least exists one of the $(H_{k+1} - H^{AD})_{ij}$ and $P_{ij}^{-1}$ equals to zero. Then we can get that

$$
\begin{align*}
\text{tr}(\hat{H}_{k+1} - \hat{H}^{AD}) &= \text{tr}(P^{-1}(H_{k+1} - H^{AD})) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} (P^{-1})_{ij} (H_{k+1} - H^{AD})_{ij} = 0.
\end{align*}
$$

Moreover, since $H^{AD}$ satisfies (19), we must have

$$
\det(H_{k+1}) \geq \det(H^{AD}).
$$

Consequently, one has

$$
\det(\hat{H}_{k+1}) = \det(\hat{H}^{AD}).
$$

Next, we establish a relation between $\hat{H}_{k+1}$ and $\hat{H}^{AD}$, which is very important in the establishment of the local and superlinear convergence of Algorithm 1.

**Proposition 1.** For the Algorithm 1, we have the following relation:

$$
\text{tr}(\hat{H}_{k+1}) = \text{tr}(\hat{H}^{AD}), \quad \det(\hat{H}_{k+1}) \geq \det(\hat{H}^{AD}).
$$

**Proof.** We can obtain from (18) that

$$(H_{k+1})_{ij} = (H^{AD})_{ij}, \quad \forall (i, j) \in F.$$  

Combined with (20), one has that for any $(i, j) \in F$, there at least exists one of the $(H_{k+1} - H^{AD})_{ij}$ and $P_{ij}^{-1}$ equals to zero. Then we can get that

$$
\begin{align*}
\text{tr}(\hat{H}_{k+1} - \hat{H}^{AD}) &= \text{tr}(P^{-1}(H_{k+1} - H^{AD})) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} (P^{-1})_{ij} (H_{k+1} - H^{AD})_{ij} = 0.
\end{align*}
$$

Moreover, since $H^{AD}$ satisfies (19), we must have

$$
\det(H_{k+1}) \geq \det(H^{AD}).
$$

Consequently, one has

$$
\det(\hat{H}_{k+1}) = \det(\hat{H}^{AD}).
$$

**Remark 1.** According to the definition of $\psi$ (10) and the relation between $\hat{H}_{k+1}$ and $\hat{H}^{AD}$ (25), one has that

$$
\psi(\hat{H}_{k+1}) \leq \psi(\hat{H}^{AD}).
$$
When we substitute (23) and (24) into (28), the \( \psi(H_{k+1}) \) and \( \psi(H_k) \) has the relation

\[
\psi(H_{k+1}) \leq \psi(H_k) - (1 - \phi_k) \frac{q_k}{r_k} - 2\phi_k \eta_k + \left( 1 + \phi_k \frac{q_k}{m_k} \right) M_k
- \ln \left( 1 + \phi_k (\beta_k \gamma_k - 1) \right) + \ln \gamma_k.
\]

(29)

4. The Local and Superlinear Convergence

Based on the discussion in Section 3, we prove the local and superlinear convergence of Algorithm 1. First, we list the assumptions.

Assumption 1. Assume that \( x^* \) is a solution of (1) and

\[
\|\Omega\| = \{ x \in \mathbb{R}^n \| x - x^* \| \leq b \},
\]

where \( b > 0 \).

1. The function \( f \in \mathbb{R}^n \to \mathbb{R} \) is twice continuously differentiable on \( \Omega \).

2. There exist two constants, \( m > 0 \) and \( M > 0 \), satisfying

\[
m\|u\|^2 \leq u^T (\nabla^2 f(x))^{-1} u \leq M\|u\|^2, \forall u \in \mathbb{R}^n, x \in \Omega.
\]

(30)

According to Assumption 1, we have constants \( L > 0 \) and \( L > 0 \) such that

\[
\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \forall x, y \in \Omega,
\]

(31)

\[
\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|, \forall x, y \in \Omega.
\]

(32)

We define

\[
\epsilon_k = \max\{\|x_k - x^*\|, \|x_{k+1} - x^*\|\},
\]

(33)

and get from (32) that

\[
\|w_k - \nabla^2 f(x^*) s_k\| = \|\nabla^2 f(x_{k+1}) s_k - \nabla^2 f(x^*) s_k\|
\leq \|\nabla^2 f(x_{k+1}) - \nabla^2 f(x^*)\| \cdot \|s_k\|
\leq L\|x_{k+1} - x^*\| \cdot \|s_k\|
= L\epsilon_k \|s_k\|.
\]

(34)

If we take \( P = H^* \), then one has from (34) that

\[
\|w_k - s_k\| = \|P^{1/2} w_k - P^{-1/2} s_k\|
= \|H^{1/2} \| \cdot \|w_k - \nabla^2 f(x^*) s_k\| \leq L \|H^*\|^{1/2} \epsilon_k \|s_k\|,
\]

(35)

Furthermore, it is easy to deduce that

\[
M_k - 1, \mu_k = \frac{2 - M_k - m_k}{m_k}, \hat{\mu}_k = \frac{(\tilde{w}_k - s_k)^T H_k \tilde{w}_k}{\text{tr}(H_k) s_k^2 \tilde{w}_k}, \ln m_k \leq \frac{1}{2} c_1 \epsilon_k,
\]

(36)

where \( c_1 > 0, c_2 \in (0, b) \), and \( \epsilon_k < c_2 \). We define

\[
\rho_k = q_k - 1 - \ln q_k,
\]

\[
\zeta_k = (1 - \phi_k) \eta_k (\tau_k^2 - 1),
\]

\[
\xi_k = \ln (1 + \phi_k (\beta_k \gamma_k - 1)),
\]

(37)
and rewrite (29) as

\[ \psi(H_{k+1}) \leq \psi(H_k) - \rho_k - \zeta_k - \xi_k + (M_k - 1) \]
\[ + \phi_k q_k m_k + \phi_k \text{tr}(H_k) \mu_k + \ln m_k. \]  

(38)

As \( \gamma_k = q_k / m_k \) and \( 0 \leq q_k \leq \text{tr}(H_k) \), we can obtain from the above inequality and (36) that

\[ \psi(H_{k+1}) \leq \psi(H_k) - \rho_k - \zeta_k - \xi_k + c_1 (1 + \text{tr}(H_k)) \epsilon_k. \]  

(39)

Considering

\[ \lambda - \ln \lambda \geq \max \left( \left( 1 - \frac{1}{e} \right) \lambda, 1 \right), \forall \lambda > 0, \]  

(40)

one has

\[ \psi(A) \geq \max \left( \left( 1 - \frac{1}{e} \right) \text{tr}(A), n \right), \]

where \( A^T = A \) and \( A > 0 \). Moreover, it follows from (40) that

\[ \psi(H_{k+1}) \leq (1 + c_3 \epsilon_k) \psi(H_k) - \rho_k - \zeta_k - \xi_k. \]  

(41)

where \( c_3 = c_1 \left( \frac{1}{n} + \epsilon \right) \). Since \( \tau_k^2 \leq 1 \) and \( \beta_k \gamma_k \geq 1 \), it is obvious that \( \rho_k, \zeta_k, \xi_k > 0 \), and

\[ \psi(H_{k+1}) \leq (1 + c_3 \epsilon_k) \psi(H_k). \]  

(42)

The theorem given below shows that Algorithm 1 converges locally and linearly, where the relation (42) plays an essential role.

**Theorem 1.** Let Assumption 1 hold and sequence \( \{ x_k \} \) be generated by Algorithm 1 with \( a_k \equiv 1 \), where \( H_k \) is updated by (15). Then for any \( \rho \in (0, 1) \), there is a constant \( \tau \| x_0 - x^* \| \leq \tau, \| H_0 - H^* \| \leq \tau \), such that

\[ \| x_{k+1} - x^* \| \leq \rho \| x_k - x^* \|. \]  

(43)

**Proof.** According to the Lemma 4 [29], there are constants \( \tau \in (0, b) \) and \( \delta > 0 \) such that when \( \| x_0 - x^* \| \leq \tau \), one has

\[ \psi(H_0) - n \leq \delta / 2, \]  

(44)

and

\[ \| H - H^* \| \leq \rho / (2L), \]  

(45)

where \( H \succeq 0 \) and \( \bar{H} = H^{* -1/2} H H^{* -1/2} \). Define

\[ \tau = \min \left\{ \bar{\tau}, c_2, \frac{\rho}{L'}, \frac{\rho}{L M} \frac{1 - \rho}{c_3 \ln \left( \frac{2(n + \delta)}{2n + \delta} \right)} \right\}. \]  

(46)

We will prove the inequalities (43) and

\[ \| H_k - H^* \| \leq \frac{\rho}{2L} \]  

(47)

hold for any \( k \geq 0 \) by induction. By the Lipschitz continuity of \( \nabla^2 f(x) \), we have for \( \forall x \in \Omega \),
\[ \|x - x^* - H^* \nabla f(x)\| \leq \|H^*\| \cdot \int_0^1 \|\nabla^2 f(x + t(x - x^*)) - \nabla^2 f(x^*)\| \cdot \|x - x^*\| dt \]
\[ \leq \frac{1}{2} LM \|x - x^*\|^2. \] (48)

Then, when \( k = 0 \), it is easy to deduce (43) by (44) and (45). Moreover, when we take \( \alpha_k \equiv 1 \) and substitute \( x_0 \) into (48), we can obtain
\[ \|x_1 - x^*\| = \|x_0 - H_0 \nabla f(x_0) - x^*\| \leq \|x_0 - x^* - H^* \nabla f(x_0)\| + \|(H_0 - H^*)(\nabla f(x_0)) - \nabla f(x^*)\| \]
\[ \leq \frac{1}{2} LM \|x_0 - x^*\|^2 + \|H_0 - H^*\| \cdot \|\nabla f(x_0) - \nabla f(x^*)\| \]
\[ \leq \left( \frac{1}{2} LM \|x_0 - x^*\| + \frac{\rho}{2} \right) \|x_0 - x^*\| \]
\[ \leq \rho \|x_0 - x^*\|. \] (49)

So we have that (43) and (47) hold for \( k = 1 \). Assume that (43) and (47) hold for \( k = 0, 1, \ldots, l \); then one has
\[ \epsilon_k = \|x_k - x^*\|, \quad \epsilon_k \leq \rho^k \epsilon_0 \leq \rho^k \tau, \quad k = 0, 1, \ldots, l, \]
and
\[ \|x_{l+1} - x^*\| = \|x_l - H_l \nabla f(x_l) - x^*\| \leq \|x_l - x^* - H^* \nabla f(x_l)\| + \|(H_l - H^*)(\nabla f(x_l)) - \nabla f(x^*)\| \]
\[ \leq \frac{1}{2} LM \|x_l - x^*\|^2 + \|H_l - H^*\| \cdot \|\nabla f(x_l) - \nabla f(x^*)\| \]
\[ \leq \left( \frac{1}{2} LM \|x_l - x^*\| + \frac{\rho}{2} \right) \|x_l - x^*\| \]
\[ \leq \rho \|x_l - x^*\|. \] (50)

Then by the definition of \( \tau \) (46), one has
\[ c_3 \sum_{k=0}^l \epsilon_k \leq c_3 \tau \sum_{k=0}^l \rho^k = c_3 \tau \frac{1 - \rho^{l+1}}{1 - \rho} \leq \frac{c_3 \tau}{1 - \rho} \leq \ln \frac{2(n + \delta)}{2n + \delta}. \] (51)

Combine (42) and (44). It can seen that
\[ \psi(H_{l+1}) - n \leq (\psi(H_0) - n) + \left( \prod_{k=0}^l (1 + c_3 \epsilon_k) - 1 \right) \psi(H_0) \]
\[ \leq \frac{\delta}{2} + \left( \frac{n + \delta}{2} \right) \left( \prod_{k=0}^l e^{c_3 \epsilon_k} - 1 \right) \]
\[ \leq \frac{\delta}{2} + \left( \frac{n + \delta}{2} \right) \left( e^{c_3 \sum_{k=0}^l \epsilon_k} - 1 \right) \]
\[ \leq \frac{\delta}{2} + \left( \frac{n + \delta}{2} \right) \left( \frac{2(n + \delta)}{2n + \delta} - 1 \right) = \delta. \] (52)
Thus, we can get that (47) holds for all \( k = l + 1 \). This completes the proof. □

Based on the above discussion and the relation (42), we can show the superlinear convergence of the Algorithm 1.

**Theorem 2.** Let Assumption 1 hold and sequence \( \{ x_k \} \) be generated by Algorithm 1 with \( a_k \equiv 1 \), where \( H_k \) is updated by (15). Then there is a constant \( \tau > 0 \) such that when \( \| x_0 - x^* \| \leq \tau \), \( \| H_0 - H^* \| \leq \tau \), one has

\[
\lim_{k \to \infty} \frac{\| (H_k - H^*)w_k \|}{\| w_k \|} = 0.
\]  

(53)

Then the sequence \( \{ x_k \} \) is superlinearly convergent.

**Proof.** Let \( \tau \) be defined as in (1), and for all \( k \) one has

\[
\psi(\tilde{H}_k) - n \leq \delta.
\]

(54)

It follows from (41) that

\[
\rho_k + \zeta_k + \xi_k \leq \left( \psi(\tilde{H}_{k+1}) - \psi(\tilde{H}_k) \right) + c_3 \epsilon_k \psi(\tilde{H}_k).
\]

Summing the above inequality and combining (51) and (54), we can deduce

\[
\sum_{k \geq 1} (\rho_k + \zeta_k + \xi_k) \leq c_3 \sum_{k \geq 1} \epsilon_k \psi(\tilde{H}_k) \leq c_3 (n + \delta) \ln \frac{2(n + \delta)}{2n + \delta} < \infty,
\]

(55)

which means that the nonnegative constants \( \rho_k, \zeta_k \) and \( \xi_k \) all tend to zero when \( k \to +\infty \). Furthermore, according to the definition of (57), we have that

1. \( q_k \to 1 \); 2. if \( \phi_k \leq \frac{1}{2} \), \( \tau \to 1 \); 3. if \( \phi_k > \frac{1}{2} \), \( \beta_k \gamma_k \to 1 \).

First, we have

\[
\frac{\| H^{1/2} x_k \|}{\| w_k \|} = \frac{\| \tilde{H}_k w_k \| - 2 \epsilon_k^T \tilde{H}_k w_k + \| w_k \|}{\| \tilde{H}_k \|} = \frac{\eta_k}{\tau_k} - 2 \eta_k + 1.
\]

(56)

For the case \( \{ k_i : \phi_k \leq \frac{1}{2} \} \), one has \( q_k \to 1 \), and \( \tau_k \to 1 \); and then (53) is true. Moreover, it is easy to deduce that

\[
\frac{\| \tilde{H}_k w_k - s_k \|}{\| w_k \|} \leq \frac{\| \tilde{H}_k^{1/2} \|}{\| \tilde{w}_k \|} \cdot \frac{\| H_k^{1/2} \|}{\| w_k \|} \cdot \frac{\| (H_k - H^*)^{-1/2} s_k \|}{\| \tilde{w}_k \|}
\]

\[
= \frac{\| \tilde{H}_k^{1/2} \|}{\| \tilde{w}_k \|} \left( q_k - 2m_k + \frac{\beta_k \gamma_k}{q_k} \right).
\]

(57)

We also have

\[
\frac{\| \tilde{H}_k w_k - \bar{w}_k \|}{\| w_k \|} - \frac{\| \tilde{H}_k w_k - s_k \|}{\| w_k \|} \leq \frac{\| \bar{w}_k - s_k \|}{\| w_k \|} \to 0.
\]

(58)

For the case \( \{ k_i : \phi_k > \frac{1}{2} \} \), one has \( q_k \to 1 \), \( \beta_k \gamma_k \to 1 \), \( m_k \to 1 \); then (53) is true by (56)–(58). Thus, the relation (53) holds for all \( k \).
Next, we will show that (53) indicates that the sufficient condition [6]
\[
\lim_{k \to \infty} \frac{\| (B_k - \nabla^2 f(x^*) )s_k \|}{\| s_k \|} = 0 \quad (59)
\]
holds. According to (47), one has that there is a constant \( \lambda_{\text{min}} > 0 \) such that \( (\lambda_k)_i \geq \lambda_{\text{min}} \), where \( (\lambda_k)_i \) denotes the eigenvalues of \( H_k \), \( i = 1, 2, \cdots, n \). When we let \( w_k = \nabla^2 f(x_{k+1})s_k \), one has
\[
\| (H_k - H^*)w_k \| = \| (H_k - H^*)\nabla^2 f(x^*)s_k + (H_k - H^*)(\nabla^2 f(x_{k+1}) - \nabla^2 f(x^*))s_k \| \\
\geq \| H_k(\nabla^2 f(x^*) - B_k)s_k \| - \| (H_k - H^*)(\nabla^2 f(x_{k+1}) - \nabla^2 f(x^*))s_k \| \\
\geq \lambda_{\text{min}} \| \| \nabla^2 f(x^*) - B_k \| s_k \| - \| (H_k - H^*)(\nabla^2 f(x_{k+1}) - \nabla^2 f(x^*))s_k \| ,
\]
and
\[
\| (H_k - H^*)w_k \| \\
\leq \frac{\lambda_{\text{min}} \| \| \nabla^2 f(x^*) - B_k \| s_k \| }{\| w_k \|} - \frac{\| (H_k - H^*)(\nabla^2 f(x_{k+1}) - \nabla^2 f(x^*))s_k \| }{\| \nabla^2 f(x_{k+1})s_k \|} \\
\geq \lambda_{\text{min}} \| \| \nabla^2 f(x^*) - B_k \| s_k \| - \lambda_{\text{min}} \| (H_k - H^*)(\nabla^2 f(x_{k+1}) - \nabla^2 f(x^*))s_k \| .
\]
When \( k \to \infty \), since \( x_k \to x^* \), then one has from (53) that
\[
\lim_{k \to \infty} \frac{\| (B_k - \nabla^2 f(x^*) )s_k \|}{\| s_k \|} = 0,
\]
which is the well-known Dennis–Moré condition. Thus, we get the superlinear convergence. \( \square \)

5. Numerical Experiments

The performance in [29] shows that the MCQN update with the BFGS method has better numerical performance than the MCQN update with DFP method. Hence, we compare the numerical performance of Algorithm 1 with the MCQN update with BFGS method and the limited-memory BFGS method.

The 24 test problems with initial points are given in Table 1, which are from [29,42–44]. It can be seen that all the test problems have special Hessian structures such as band matrices, so the chordal extension of the sparsity could be obtained easily. Then \( H_{k+1} \) in Algorithm 1 can be written as the sparse clique-factorization formula.

All the methods were coded in MATLAB R2016a on a Core (TM) i5 PC. The automatic differentiation was computed by ADMAT 2.0, which is available on the cayuga research GitHub page. In Tables 1–4 and Figures 1 and 2, we report the numerical performances of the three methods. For the convenience of statement, we use the following notation in our numerical results.

Pro: the problems;
Dim: the dimensions of the test problem;
Init: the initial points;
Method: the algorithm used to solve the problem;
MCQN-BFGS: MCQN update with the BFGS method;
L-BFGS: limited-memory with the BFGS method;
We adopted the termination criterion as follows:
\[
\frac{\|\nabla f(x)\|}{n} \leq 10^{-5} \text{ or } \text{ite} \geq 5000.
\]

Table 1. The test problems.

| Pro | the Test Functions | Init |
|-----|--------------------|------|
| 1   | TRIDIA [29]        | \(x_0 = (1, 1, \cdots, 1)^T\) |
| 2   | the chained Rosenbrock problem [29] | \(x_0 = (-1.2, -1, \ldots, -1.2, -1)^T\) |
| 3   | the boundary value problem [29] | \(x_0 = (\frac{1}{n+1}, \frac{2}{n+1}, \cdots, \frac{n}{n+1})^T\) |
| 4   | Broyden tridiagonal function [42] | \(x_0 = (-1, -1, \ldots, -1)^T\) |
| 5   | DQRTIC [43]        | \(x_0 = (2, 2, \cdots, 2)^T\) |
| 6   | EDENSCH [43]       | \(x_0 = (0, 0, \ldots, 0)^T\) |
| 7   | ENGVAL1 [43]       | \(x_0 = (2, 2, \cdots, 2)^T\) |
| 8   | COSINE [43]        | \(x_0 = (1, 1, \cdots, 1)^T\) |
| 9   | ERRINROS-modified [43] | \(x_0 = (-1, -1, \ldots, -1)^T\) |
| 10  | FREUROTH [43]      | \(x_0 = (0.5, -2, 0, \ldots, 0)^T\) |
| 11  | MOREBV-different start point [43] | \(x_0 = (0.5, 0.5, \ldots, 0.5)^T\) |
| 12  | TOINTGSS [43]      | \(x_0 = (3, 3, \cdots, 3)^T\) |
| 13  | SCHMVETT [43]      | \(x_0 = (3, 3, \cdots, 3)^T\) |
| 14  | Extended Freudenstein and Roth function [44] | \(x_0 = (0.5, -2, \ldots, 0.5, -2)^T\) |
| 15  | Raydan 1 function [44] | \(x_0 = (1, 1, \cdots, 1)^T\) |
| 16  | Generalized Tridiagonal function [44] | \(x_0 = (2, 2, \cdots, 2)^T\) |
| 17  | Extended Himmelblau function [44] | \(x_0 = (1, 1, \cdots, 1)^T\) |
| 18  | Generalized PSCI function [44] | \(x_0 = (3, 0.1, \cdots, 3, 0.1)^T\) |
| 19  | Extended Tridiagonal 2 function [44] | \(x_0 = (1, 1, \cdots, 1)^T\) |
| 20  | Raydan 2 function [44] | \(x_0 = (1, 1, \cdots, 1)^T\) |
| 21  | Extended Freudenstein and Roth function [44] | \(x_0 = (1, 1, \cdots, 1)^T\) |
| 22  | DQDRTIC function [44] | \(x_0 = (3, 3, \cdots, 3)^T\) |
| 23  | Generalized Quartic function [44] | \(x_0 = (1, 1, \cdots, 1)^T\) |
| 24  | HIMMELBG function [44] | \(x_0 = (1.5, 1.5, \cdots, 1.5)^T\) |

Firstly, we tested all three methods on the above 24 problems, whose dimensions are 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, and 1000. We set \(m = 15\) in the limited-memory BFGS method. Tables 2 and 3 contain the numbers of iterations of the three methods for the test problems. Taking account of the total number of iterations, Algorithm 1 outperformed the MCQN update with BFGS method on 11 problems (2, 4, 5, 7, 9, 10, 12, 14, 18, 23, 24). Additionally, Algorithm 1 outperformed the limited memory BFGS method on 13 problems (1, 2, 3, 7, 9, 12, 15, 16, 18, 19, 20, 21, 23).
### Table 2. Numbers of iterations for problems 1–12.

| Dim | 10 | 20 | 50 | 100 | 200 | 500 | 1000 | 2000 | 5000 | 10,000 |
|-----|----|----|----|-----|-----|-----|------|------|------|--------|
| (1) Algorithm 1 | 30 | 38 | 51 | 78 | 96 | 146 | 217 | 301 | 424 | 527 |
| (1) MCQN-BFGS | 29 | 38 | 51 | 72 | 95 | 146 | 192 | 298 | 424 | 528 |
| (1) L-BFGS | 26 | 39 | 96 | 158 | 360 | 864 | 1042 | 1759 | 3153 | 3152 |
| (2) Algorithm 1 | 49 | 90 | 166 | 308 | 595 | 1345 | 2699 | 5437 | 3218 | 2725 |
| (2) MCQN-BFGS | 60 | 95 | 200 | 384 | 683 | 1668 | 3249 | 6486 | 4562 | 3207 |
| (2) L-BFGS | 59 | 113 | 260 | 504 | 999 | 2481 | 4947 | 9887 | 24,732 | 49,391 |
| (3) Algorithm 1 | 16 | 26 | 42 | 58 | 59 | 51 | 49 | 60 | 102 | 399 |
| (3) MCQN-BFGS | 15 | 26 | 42 | 50 | 59 | 71 | 54 | 69 | 101 | 402 |
| (3) L-BFGS | 39 | 114 | 279 | 700 | 1503 | 1659 | 2695 | 3370 | 8867 | 27,471 |
| (4) Algorithm 1 | 31 | 25 | 34 | 49 | 43 | 41 | 38 | 36 | 52 | 53 |
| (4) MCQN-BFGS | 30 | 29 | 43 | 45 | 49 | 58 | 61 | 62 | 63 | 56 |
| (4) L-BFGS | 21 | 27 | 40 | 54 | 41 | 38 | 36 | 52 | 50 | 49 |
| (5) Algorithm 1 | 30 | 48 | 60 | 92 | 109 | 99 | 92 | 89 | 81 | 81 |
| (5) MCQN-BFGS | 35 | 49 | 67 | 94 | 111 | 108 | 112 | 98 | 84 | 84 |
| (5) L-BFGS | 28 | 27 | 34 | 31 | 33 | 39 | 41 | 43 | 54 | 81 |
| (6) Algorithm 1 | 23 | 26 | 38 | 44 | 54 | 55 | 47 | 51 | 61 | 51 |
| (6) MCQN-BFGS | 17 | 27 | 36 | 53 | 60 | 55 | 54 | 51 | 50 | 54 |
| (6) L-BFGS | 17 | 19 | 19 | 21 | 21 | 23 | 24 | 26 | 24 | 25 |
| (7) Algorithm 1 | 16 | 21 | 21 | 19 | 17 | 17 | 16 | 15 | 16 | 15 |
| (7) MCQN-BFGS | 20 | 22 | 23 | 22 | 15 | 15 | 15 | 17 | 17 | 16 |
| (7) L-BFGS | 20 | 22 | 26 | 21 | 22 | 21 | 25 | 27 | 28 | 30 |
| (8) Algorithm 1 | 22 | 23 | 24 | 21 | 23 | 27 | 27 | 28 | 29 | 30 |
| (8) MCQN-BFGS | 23 | 25 | 26 | 26 | 26 | 27 | 28 | 29 | 30 | 30 |
| (8) L-BFGS | 9 | 9 | 9 | 9 | 10 | 10 | 10 | 10 | 10 | 10 |
| (9) Algorithm 1 | 74 | 122 | 134 | 149 | 137 | 180 | 148 | 153 | 172 | 170 |
| (9) MCQN-BFGS | 106 | 125 | 145 | 171 | 199 | 181 | 168 | 171 | 174 | 179 |
| (9) L-BFGS | 163 | 245 | 216 | 196 | 189 | 190 | 163 | 169 | 171 | 192 |
| (10) Algorithm 1 | 45 | 45 | 48 | 39 | 45 | 43 | 45 | 45 | 145 | 161 |
| (10) MCQN-BFGS | 47 | 48 | 49 | 43 | 47 | 48 | 41 | 244 | 204 | 279 |
| (10) L-BFGS | 24 | 25 | 24 | 24 | 24 | 22 | 22 | 22 | 20 | 22 |
| (11) Algorithm 1 | 24 | 45 | 97 | 121 | 82 | 67 | 35 | 34 | 21 | 11 |
| (11) MCQN-BFGS | 24 | 45 | 98 | 121 | 82 | 67 | 35 | 34 | 21 | 11 |
| (11) L-BFGS | 33 | 103 | 136 | 127 | 85 | 49 | 32 | 21 | 10 | 9 |
| (12) Algorithm 1 | 13 | 11 | 11 | 12 | 9 | 5 | 6 | 2 | 3 | 2 |
| (12) MCQN-BFGS | 15 | 11 | 13 | 12 | 12 | 7 | 6 | 2 | 3 | 2 |
| (12) L-BFGS | 6 | 8 | 9 | 10 | 13 | 11 | 10 | 10 | 9 | 9 |

### Table 3. Numbers of iterations for problems 13–24.

| Dim | 10 | 20 | 50 | 100 | 200 | 500 | 1000 | 2000 | 5000 | 10,000 |
|-----|----|----|----|-----|-----|-----|------|------|------|--------|
| (13) Algorithm 1 | 19 | 20 | 21 | 22 | 18 | 17 | 15 | 15 | 14 | 13 |
| (13) MCQN-BFGS | 19 | 20 | 21 | 22 | 18 | 17 | 15 | 14 | 13 | 12 |
| (13) L-BFGS | 16 | 18 | 17 | 18 | 18 | 17 | 18 | 18 | 17 | 18 |
| (14) Algorithm 1 | 36 | 52 | 95 | 111 | 169 | 262 | 612 | 567 | 1062 | 1114 |
| (14) MCQN-BFGS | 37 | 54 | 86 | 120 | 190 | 300 | 594 | 679 | 1062 | 1126 |
| (14) L-BFGS | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| (15) Algorithm 1 | 11 | 13 | 23 | 30 | 39 | 45 | 60 | 97 | 196 | 295 |
| (15) MCQN-BFGS | 11 | 12 | 21 | 28 | 37 | 45 | 64 | 95 | 196 | 295 |
| (15) L-BFGS | 13 | 21 | 32 | 50 | 79 | 122 | 207 | 338 | 402 | 770 |
Table 3. Cont.

| Dim | 10  | 20  | 50  | 100 | 200 | 500 | 1000 | 2000 | 5000 | 10,000 |
|-----|-----|-----|-----|-----|-----|-----|------|------|------|--------|
| (16) Algorithm 1 | 29  | 31  | 47  | 69  | 110 | 149 | 165  | 174  | 170  | 163    |
| (16) MCQN-BFGS   | 29  | 35  | 51  | 70  | 100 | 146 | 164  | 173  | 171  | 164    |
| (16) L-BFGS      | 25  | 65  | 80  | 162 | 160 | 156 | 151  | 150  | 144  | 140    |
| (17) Algorithm 1 | 16  | 15  | 11  | 13  | 12  | 12  | 10   | 10   | 9    |        |
| (17) MCQN-BFGS   | 14  | 11  | 18  | 12  | 13  | 12  | 12   | 10   | 9    |        |
| (17) L-BFGS      | 8   | 8   | 8   | 8   | 8   | 8   | 8    | 8    | 8    |        |
| (18) Algorithm 1 | 49  | 21  | 23  | 21  | 19  | 14  | 22   | 21   | 13   | 11     |
| (18) MCQN-BFGS   | 43  | 28  | 22  | 29  | 20  | 15  | 20   | 25   | 23   | 26     |
| (18) L-BFGS      | 36  | 34  | 37  | 36  | 39  | 36  | 40   | 34   | 31   | 37     |
| (19) Algorithm 1 | 13  | 12  | 11  | 13  | 12  | 12  | 10   | 10   | 7    | 5      |
| (19) MCQN-BFGS   | 13  | 12  | 11  | 13  | 12  | 12  | 10   | 10   | 7    | 5      |
| (19) L-BFGS      | 12  | 14  | 15  | 16  | 16  | 17  | 16   | 16   | 16   | 17     |
| (20) Algorithm 1 | 5   | 5   | 4   | 4   | 4   | 4   | 4    | 3    | 3    | 3      |
| (20) MCQN-BFGS   | 5   | 5   | 4   | 4   | 4   | 4   | 4    | 3    | 3    | 3      |
| (20) L-BFGS      | 7   | 7   | 7   | 7   | 7   | 7   | 7    | 7    | 7    | 7      |
| (21) Algorithm 1 | 14  | 11  | 21  | 18  | 18  | 35  | 28   | 26   | 43   |        |
| (21) MCQN-BFGS   | 13  | 10  | 11  | 12  | 18  | 18  | 35   | 28   | 26   | 43     |
| (21) L-BFGS      | 10  | 13  | 15  | 19  | 22  | 30  | 36   | 51   | 56   | 64     |
| (22) Algorithm 1 | 36  | 36  | 26  | 28  | 26  | 27  | 27   | 30   | 28   | 29     |
| (22) MCQN-BFGS   | 36  | 36  | 26  | 28  | 26  | 27  | 27   | 30   | 28   | 29     |
| (22) L-BFGS      | 13  | 13  | 15  | 19  | 22  | 30  | 36   | 51   | 56   | 64     |
| (23) Algorithm 1 | 13  | 17  | 21  | 12  | 12  | 4   | 11   | 10   | 9    |        |
| (23) MCQN-BFGS   | 16  | 18  | 22  | 25  | 15  | 13  | 12   | 12   | 12   | 10     |
| (23) L-BFGS      | 14  | 14  | 16  | 15  | 17  | 24  | 27   | 27   | 27   | 31     |
| (24) Algorithm 1 | 11  | 13  | 18  | 24  | 31  | 37  | 28   | 21   | 13   | 7      |
| (24) MCQN-BFGS   | 12  | 15  | 19  | 25  | 32  | 37  | 28   | 21   | 13   | 7      |
| (24) L-BFGS      | 10  | 10  | 10  | 10  | 10  | 10  | 10   | 10   | 10   | 10     |

For the sake of precise comparison, we adopted the performance profiles from [45], which are distribution functions of a performance metric. We denote $P$ and $S$ as the test set and the set of solvers; and $N_p$ and $N_s$ as the number of problems and number of solvers, respectively. For solver $s \in S$ and problem $p \in P$, we define $t_{p,s}$ as the number of iterations or number of function evaluations required for solve problem $p$ using solver $s$. Then, using the performance ratio $r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,q} : q \in S\}}$, we define

$$\rho_s(t) = \frac{1}{N_p} \text{size}\{p \in P : r_{p,s} \leq t\},$$

where $r_{p,s} \leq r_M$ for some constant for all $p$ and $s$. The equality holds if and only if solver $s$ cannot solve problem $p$. Therefore, $\rho_s : R \to [0,1]$ was the probability for $s \in S$ satisfying $r_{p,s} \leq t$, $t \in R$ among the best possible ratios.

Figure 1 evaluates the number of iterations of and the MCQN update with BFGS method by using performance profiles. It can be seen that the top curve corresponds to Algorithm 1, which shows that Algorithm 1 had better performance than the MCQN update with BFGS method. Additionally, Figure 2 demonstrates that Algorithm 1 had better performance than the limited-memory BFGS method.
Figure 1. Performance profiles based on the numbers of iterations.

Figure 2. Performance profiles based on the numbers of iterations.
Secondly, for a further comparison of Algorithm 1 and the MCQN update with BFGS method, we tested five different initial points, $x_0, 2x_0, 4x_0, 7x_0$, and $10x_0$, where $x_0$ is specified in Table 1. The dimensions of the test problems was 1000. Table 4 reports the number of iterations required of the two methods for 24 test problems, which also demonstrates that Algorithm 1 was effective and superior to the MCQN update with BFGS method.

| Pro | Algorithm 1 | MCQN-BFGS |
|-----|-------------|------------|
| Init | $x_0$ | $2x_0$ | $4x_0$ | $7x_0$ | $10x_0$ | $x_0$ | $2x_0$ | $4x_0$ | $7x_0$ | $10x_0$ |
| (1) | 217 | 189 | 192 | 194 | 196 | 192 | 210 | 213 | 220 | 213 |
| (2) | 2699 | 2684 | 2641 | 1192 | 2694 | 3249 | 4880 | 5056 | 2157 | 4961 |
| (3) | 49 | 47 | 55 | 68 | 65 | 54 | 213 | 210 | 228 | 294 |
| (4) | 43 | 47 | 36 | 81 | 80 | 60 | 213 | 210 | 228 | 294 |
| (5) | 92 | 83 | 86 | 91 | 89 | 112 | 106 | 85 | 94 | 94 |
| (6) | 47 | 47 | 47 | 47 | 47 | 54 | 54 | 54 | 54 | 54 |
| (7) | 16 | 30 | 53 | 19 | 21 | 15 | 19 | 27 | 21 | 22 |
| (8) | 27 | 31 | 16 | 35 | 54 | 28 | 31 | 16 | 33 | 56 |
| (9) | 148 | 159 | 154 | 184 | 157 | 168 | 151 | 147 | 191 | 154 |
| (10) | 45 | 45 | 164 | 170 | 175 | 41 | 263 | 203 | 192 | 194 |
| (11) | 35 | 70 | 96 | 112 | 127 | 35 | 70 | 96 | 112 | 127 |
| (12) | 6 | 6 | 2 | 2 | 2 | 6 | 6 | 2 | 2 | 2 |
| (13) | 15 | 16 | 17 | 14 | 14 | 15 | 16 | 17 | 14 | 14 |
| (14) | 612 | 312 | 504 | 511 | 318 | 594 | 523 | 503 | 481 | 563 |
| (15) | 60 | 57 | 208 | 452 | 1024 | 64 | 58 | 203 | 532 | 941 |
| (16) | 165 | 180 | 173 | 153 | 129 | 164 | 183 | 191 | 197 | 215 |
| (17) | 12 | 11 | 10 | 7 | 21 | 12 | 11 | 8 | 7 | 30 |
| (18) | 22 | 23 | 25 | 60 | 41 | 20 | 23 | 30 | 65 | 75 |
| (19) | 12 | 12 | 14 | 16 | 24 | 12 | 12 | 14 | 22 | 20 |
| (20) | 4 | 6 | 5 | 6 | 4 | 4 | 6 | 5 | 6 | 4 |
| (21) | 35 | 30 | 35 | 112 | 719 | 35 | 30 | 35 | 112 | 719 |
| (22) | 27 | 28 | 28 | 29 | 29 | 27 | 28 | 28 | 29 | 29 |
| (23) | 8 | 22 | 23 | 14 | 35 | 12 | 13 | 24 | 19 | 70 |
| (24) | 28 | 20 | 21 | 21 | 28 | 20 | 21 | 21 | 21 | 21 |

6. Conclusions

In this paper, we presented a symmetric rank-two quasi-Newton update method based on an adjoint tangent condition for solving unconstrained optimization problems. Combined with the idea of matrix completion, we proposed a sparse quasi-Newton algorithm and established its local and superlinear convergence. Extensive numerical results demonstrated that the proposed algorithm outperformed other methods and can be used to solve large-scale unconstrained optimization problems.

**Author Contributions:** Conceptualization, H.C.; methodology, H.C. and X.A.; software, H.C. and X.A.; formal analysis, H.C.; writing—original draft preparation, H.C. and X.A.; writing—review and editing H.C. and X.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** The work is supported by the National Natural Science Foundation of China, grant number 11701577; the Natural Science Foundation of Hunan Province, China, grant number 2020JJ5960; and the Scientific Research Foundation of Hunan Provincial Education Department, China, grant number 18C0253.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The date used to support the research plan and all the code used in this study are available from the corresponding author upon request.

**Conflicts of Interest:** The authors declare no conflict of interest.
36. Andrei, N. Diagonal Approximation of the Hessian by Finite Differences for Unconstrained Optimization. *J. Optim. Theory Appl.* **2020**, *185*, 859–879. [CrossRef]
37. Andrei, N. A new accelerated diagonal quasi-Newton updating method with scaled forward finite differences directional derivative for unconstrained optimization. *Optimization* **2021**, *70*, 345–360. [CrossRef]
38. Leong, W.J.; Enshaei, S.; Kek, S.L. Diagonal quasi-Newton methods via least change updating principle with weighted Frobenius norm. *Numer. Algorithms* **2021**, *86*, 1225–1241. [CrossRef]
39. Schlenkrich, S.; Griewank, A.; Walther, A. On the local convergence of adjoint Broyden methods. *Math. Program.* **2010**, *121*, 221–247. [CrossRef]
40. Byrd, R.H.; Nocedal, J. A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. *SIAM J. Numer. Anal.* **1989**, *26*, 727–739. [CrossRef]
41. Byrd, R.H.; Nocedal, J.; Yuan, Y.X. Global convergence of a class of quasi-Newton methods on convex problems. *SIAM J. Numer. Anal.* **1987**, *24*, 1171–1190. [CrossRef]
42. Moré, J.J.; Garbow, B.S.; Hillstrom, K.E. Testing unconstrained optimization software. *ACM Trans. Math. Softw. (TOMS)* **1981**, *7*, 17–41. [CrossRef]
43. Luksan, L.; Matonoha, C.; Vlcek, J. Modified CUTE Problems for Sparse Unconstrained Optimization; Technical Report 1081; Institute of Computer Science, Academy of Sciences of the Czech Republic: Prague, Czech Republic, 2010.
44. Andrei, N. An unconstrained optimization test functions collection. *Adv. Model. Optim.* **2008**, *10*, 147–161.
45. Dolan, E.D.; Moré, J.J. Benchmarking optimization software with performance profiles. *Math. Program.* **2002**, *91*, 201–213. [CrossRef]