AMO - Advanced Modeling and Optimization, Volume 12, Number 1, 2010

Global Convergence of Gauss-Newton-MBFGS Method for Solving the Nonlinear Least Squares Problem

Fei Wang
Department of Mathematics, Hunan City University, Yiyang 413000, China
Dong-Hui Li
College of Mathematical Sciences, South China Normal University, Guangzhou 510631, China
Liqun Qi
Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong

Revised: January 28, 2010

Abstract

In this paper, by using a modified BFGS (MBFGS) update, we propose a structured MBFGS update for the nonlinear least squares problem. We then propose a hybrid method that combines the Gauss-Newton method with the structured MBFGS method for solving the nonlinear least squares problem. We show that the hybrid method is globally and quadratically convergent for zero residual problems, and globally and superlinearly convergent for the nonzero residual problems. We also show that the unit step is essentially accepted. We also present some preliminary numerical results which show that the hybrid method is comparable with existing structured BFGS methods.

Keywords: least squares problems, Gauss-Newton method, structured MBFGS method, global convergence

1. Introduction

Let $r_i : R^n \to R$, $i = 1, \dots, m$ be twice continuously differentiable. Consider the following nonlinear least squares problem:

$$\min_{x \in R^n} f(x) \stackrel{\triangle}{=} \frac{1}{2} R(x)^T R(x) = \frac{1}{2} \sum_{i=1}^{m} r_i(x)^2, \tag{1.1}$$

where $R(x) = (r_1(x), \dots, r_m(x))^T$. Problem (1.1) is called zero residual if at a solution x^*, $R(x^*) = 0$. Otherwise, it is called nonzero residual.

[*] Corresponding author. His work was supported by the NSF of China via grant 10771057.
[†] This author's work was supported by the Hong Kong Research Grant Council.
AMO - Advanced Modeling and Optimization. ISSN: 1841-4311

By direct computation, it is easy to obtain the following expression for the gradient and Hessian matrix of f at x:

$$g(x) \triangleq \nabla f(x) = J(x)^T R(x), \quad G(x) \triangleq \nabla^2 f(x) = C(x) + S(x),$$

where $J(x)$ stands for the Jacobian matrix of $R(x)$ at x,

$$C(x) = J(x)^T J(x) \quad \text{and} \quad S(x) = \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x).$$

The structured quasi-Newton methods for solving (1.1) are extensions of quasi-Newton methods for unconstrained optimization problems. They exploit the structure of $G(x)$. Specifically, a structured quasi-Newton method uses the first order information of $C(x)$ exactly while approximates the second order term $S(x)$ by some matrix $A(x)$ such that matrix $B = C(x) + A(x)$ is an approximation of $G(x)$.

Let x_k be the current iterate. Following Dennis [5], we have

$$G(x_{k+1}) s_k \approx \hat{y}_k,$$

where $s_k = x_{k+1} - x_k$ and

$$\hat{y}_k = J(x_{k+1})^T J(x_{k+1}) s_k + (J(x_{k+1}) - J(x_k))^T R(x_{k+1}).$$

Thus, as an approximation of $S(x_k)$, matrix A_k should be updated such that matrix A_{k+1} satisfies the secant condition

$$A_{k+1} s_k = (J(x_{k+1}) - J(x_k))^T R(x_{k+1}). \tag{1.2}$$

The structured quasi-Newton methods have been studied by many authors [1, 2, 6, 7, 8, 9, 12, 19, 20]. Dennis, Martinez, and Tapia [6] described the structure principle and proved the superlinear convergence of the structured BFGS method. Al-Baili and Fletcher [1], Fletcher and Xu [11] and Lukšan and Spedicato [15] proposed the hybrid method which combines the Gauss-Newton method with the BFGS method. This hybrid method is superlinearly convergent for nonzero problems and quadratically convergent for zero residual problems. Huschens [12] derived a factorized self-adjusting structured quasi-Newton method. This factorized method is superlinearly convergent for nonzero problems and quadratically convergent for zero residual problems. Recent progress in factorized quasi-Newton methods can be found in [18, 21, 22, 24].

As far as we know, there is no global convergence result for the structured quasi-Newton methods. A major difficulty for the globalization of the structured quasi-Newton methods lies in that the direction generated by the structured quasi-Newton method may not be a descent direction of f. It is well-known that in a standard quasi-Newton method such as the BFGS or the DFP method, if the Wolfe-Powell type line search is used, the generated iterative matrices are positive definite and quasi-Newton directions are descent directions for the objective function f. For structured quasi-Newton methods, however, Wolfe-Powell types line search is not enough to guarantee the positive definiteness of the iterative matrices. Therefore the structured quasi-Newton directions may not be descent directions of f. Another difficulty in globalizing the structured quasi-Newton methods lies in the non-convexity of f. The standard quasi-Newton methods including the BFGS method are not globally convergent for non-convex minimizations [4].

In this paper, by the use of a modified BFGS method proposed by Li-Fukushima [13], we develop a structured MBFGS method. An advantage of this structured MBFGS method is that the iterative matrices are positive definite whatever line search is used. By combining this method with the Gauss-Newton method, we propose a hybrid method. The proposed method is globally and superlinearly convergent for nonzero residual problems and is globally and quadratically convergent for zero residual problems. Indeed, we will show that when k is sufficiently large, the proposed hybrid method reduces to the structured MBFGS method if the problem is nonzero residual, or to the Gauss-Newton method if the problem is zero residual.

In the next section, we propose the method and establish its global convergence. In Sections 3, we prove the superlinear/quadratic convergence of the proposed method. We will also show the acceptance of the unit step in Section 3. In Section 4, we present some preliminary numerical results.

2. The Algorithm and Its Global Convergence

As described in Section 1, in a structured quasi-Newton method, the approximation $B_k = C(x_k)+A_k$ of $G(x_k)$ is computed via updating A_k such that the secant equation (1.2) is satisfied. Let us recall the structured secant method by Dennis-Martinez-Tapia [6]. At iteration k, we let

$$s_k = x_{k+1} - x_k, \quad \hat{y}_k^{\#} = (J_{k+1} - J_k)^T R_{k+1}, \quad \hat{y}_k = C(x_{k+1})s_k + \hat{y}_k^{\#}.$$

Let

$$B_{k+1} = C(x_{k+1}) + A_{k+1}. \tag{2.1}$$

where A_{k+1} is an approximation of $S(x_{k+1})$ which is updated by

$$A_{k+1} = A_k + \triangle(s_k, \hat{y}_k^{\#}, A_k, v_k),$$

where $\triangle$ is defined by

$$\triangle(s, y, B, v) = \frac{(y - Bs)v^T + v(y - Bs)^T}{v^T s} - \frac{(y - Bs)^T s v v^T}{(v^T s)^2}.$$

The relation between B_k and B_{k+1} is

$$B_{k+1} = B_k^s + \triangle(s_k, \hat{y}_k, B_k^s, v_k),$$

where $B_k^s = C(x_{k+1}) + A_k$. The BFGS update corresponds to the case where $v_k = \hat{y}_k + [\hat{y}_k^T s_k/(s_k^T B_k^s s_k)]^{1/2} B_k^s s_k$, which is stated as follows.

$$B_{k+1} = B_k^s - \frac{B_k^s s_k s_k^T B_k^s}{s_k^T B_k^s s_k} + \frac{\hat{y}_k \hat{y}_k^T}{\hat{y}_k^T s_k}. \tag{2.2}$$

This structured quasi-Newton method is locally superlinearly convergent [6]. Moreover, if x_k is close to a solution x^* where $G(x^*)$ is positive definite, B_k is positive definite under some conditions. However, when x_k is far away from x^*, it is not know under what conditions B_k is positive definite. We note that the update formula (2.2) is different from the standard BFGS formula due to the difference between B_k^s and B_k and the different definition of $\hat{y}_k$. It is well-known that the standard BFGS update can guarantee the positive definiteness of B_{k+1} if B_k is positive definite and $\hat{y}_k^T s_k > 0$. The latter condition is satisfied if the Wolfe-Powell line search is used. It is well-known that when f is twice continuously differentiable and convex, then the standard BFGS method with Wolfe-Powell line search is globally convergent [17]. However, when f is not convex, the standard BFGS method may fail to be globally convergent [4]. On the other hand, the modified BFGS (MBFGS) method proposed by Li-Fukushima [13] is globally and superlinearly convergent even if f is not convex. We now exploit the idea of this MBFGS method to develop a structured MBFGS method.

We let

$$y_k^{\#} = \hat{y}_k^{\#} + t_k s_k = (J_{k+1} - J_k)^T R_{k+1} + t_k s_k$$

and

$$y_k = \hat{y}_k + t_k s_k = C(x_{k+1}) s_k + y^{\#} = C(x_{k+1}) s_k + (J_{k+1} - J_k)^T R_{k+1} + t_k s_k,$$

where

$$t_k = C\|g(x_{k+1})\|^{\alpha} + \max\{-\frac{\hat{y}_k^T s_k}{s_k^T s_k}, 0\},$$

and α is a positive constant.

Let B_{k+1} be defined by (2.1) and A_{k+1} be updated by

$$A_{k+1} = A_k^s + \Delta(s_k, y_k^{\#}, A_k^s, \hat{v}_k), \tag{2.3}$$

where $A_k^s = B_k - C(x_{k+1})$. With this update strategy, we can show that B_k and B_{k+1} satisfy the standard BFGS, namely,

$$B_{k+1} = B_k + \triangle(s_k, y_k, B_k, v_k). \tag{2.4}$$

Indeed, we have

$$B_{k+1} = C(x_{k+1}) + A_{k+1} = B_k + \triangle(s_k, y_k^{\#}, A_k^s, v_k). \tag{2.5}$$

We also have

$$y_k^{\#} - A_k^s s_k = y_k - C(x_{k+1})s_k - A_k^s s_k = y_k - B_k s_k.$$

This implies

$$\triangle(s_k, y_k^{\#}, A_k^s, v_k) = \triangle(s_k, y_k, B_k, v_k).$$

Substituting this to (2.5), we get (2.4). In this paper, we only consider the modified BFGS (MBFGS) update in which $v_k = y_k + [y_k^T s_k/(s_k^T B_k s_k)]^{1/2} B_k s_k$, namely,

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}. \tag{2.6}$$

An advantage of this MBFGS update is that the inequality $y_k^T s_k \geq C\|g_k\|^\alpha \|s_k\|^2 > 0$ holds for all k. Consequently, B_{k+1} is positive definite as long as B_k is positive definite. This property is independent with line search used.

The above MBFGS update is different from the MBFGS method proposed by Li-Fukushima [13]. The difference lies in the definition of y_k. In the MBFGS update formula in [13], $y_k = \nabla f(x_{k+1}) - \nabla f(x_k) + t_k s_k$. It needs to be pointed out that the parameter t_k in (5.2) of [13] neglected a factor $\|g_k\|^{-1}$. The above t_k is the correct one.

It can be shown that under appropriate conditions, the above structured MBFGS method is globally and superlinearly convergent whether the problem is zero or nonzero residual. To make the method maintain quadratic convergence property for zero residual problems, we adopt a hybrid strategy by combining the Gauss-Newton method with the structured BFGS method. This strategy was introduced by Fletcher-Xu [11].

Suppose x^* is a solution of (1.1) at which $g(x^*) = 0$ and $G(x^*)$ is positive define. If $\{x_k\}$ converges to x^* and $f(x^*) \neq 0$, then

$$\lim_{k\to\infty} \frac{f(x_k) - f(x_{k+1})}{f(x_k)} = 0.$$

If $\{x_k\}$ converges to x^* superlinearly and $f(x^*) = 0$, then

$$\lim_{k\to\infty} \frac{f(x_k) - f(x_{k+1})}{f(x_k)} = 1.$$

Based on above observation, Fletcher and Xu [11] adopt the quantity $\frac{f(x_k)-f(x_{k+1})}{f(x_k)}$ to distinguish between the zero residual problem and the nonzero residual problem. We use the same strategy to develop a hybrid method. The steps of the method are stated as follows.

Algorithm 1 (A Gauss-Newton-MBFGS Method)

Step 0 Given constants $\sigma \in (0,1)$, $\rho \in (0,1)$ and $\epsilon \in (0,1)$. Let $x_0 \in R^n$. Let $A_0 \in R^{n\times n}$ be a symmetric matrix such that matrix $B_0 \stackrel{\triangle}{=} C(x_0) + A_0$ is positive definite. Let $k := 0$.

Step 1 Stop if $\|g(x_k)\| = 0$.

Step 2 Find quasi-Newton direction p_k by solving the following system of linear equations

$$B_k p + g(x_k) = 0. \tag{2.7}$$

Step 3 Find the minimum nonnegative number j, say j_k, such that

$$f(x_k + \rho^j p_k) \le f(x_k) + \sigma\rho^j g(x_k)^T p_k. \tag{2.8}$$

Let $\lambda_k = \rho^{j_k}$.

Step 4 Let the next iterate be $x_{k+1} = x_k + \lambda_k p_k$.

Step 5 Update B_k to get B_{k+1} by

$$B_{k+1} = \begin{cases} C(x_{k+1}), & \text{if } \dfrac{f(x_k) - f(x_{k+1})}{f(x_k)} \ge \epsilon, \\ C(x_{k+1}) + A_{k+1}, & \text{otherwise,} \end{cases}$$

where A_{k+1} is determined by (2.3) with $v_k = y_k + [y_k^T s_k/(s_k^T B_k s_k)]^{1/2} B_k s_k$.

Step 6 Let $k := k+1$. Go to Step 1.

Remark Note that for some k, matrix $C(x_{k+1})$ may be singular or nearly singular. In this case, we may use $B_{k+1} = C(x_{k+1}) + \zeta_k I$ instead of $C(x_{k+1})$ if $[f(x_k) - f(x_{k+1})]/f(x_k) \ge \epsilon$. The parameter ζ_k can be chosen as

$$\zeta_k = \begin{cases} \beta\|R(x_{k+1})\|^\gamma, & \text{if } C(x_{k+1}) \text{ is nearly singular,} \\ 0, & \text{otherwise} \end{cases}$$

with constants $\beta > 0$ and $\gamma \in [1,\ 2]$. The convergence theory of the related method can be established in a similar way to Theorems 2.1 and 3.2. More study about this kind method can be found in [23] and [10].

We conclude this section by proving the global convergence of Algorithm 1.

Define the index set

$$K = \{k \mid \frac{f(x_k) - f(x_{k+1})}{f(x_k)} \ge \epsilon\}. \tag{2.9}$$

Let Ω be the level set defined by

$$\Omega = \{x \in R^n \mid f(x) \le f(x_0)\}.$$

In the latter part of the paper, we always assume that the level set Ω is bounded and $J(x)$ is Lipschitz continuous on Ω. It is clear that the sequence $\{x_k\}$ generated by Algorithm 1 is contained in Ω and hence bounded. Moreover, $g(x)$ is Lipschitz continuous on Ω, i.e., there exists a constant $L > 0$ such that

$$\|g(x) - g(y)\| \leq L\|x - y\|, \quad \forall x, y \in R^n, \tag{2.10}$$

The following theorem establishes the global convergence of Algorithm 1.

Theorem 2.1 *Let $\{x_k\}$ be generated by Algorithm 1. Then we have*

$$\liminf_{k\to\infty} \|g(x_k)\| = 0. \tag{2.11}$$

Proof Let the index set K be defined by (2.9). We first consider the case where K is infinite. Let $K = \{k_0 < k_1 < \cdots\}$. It is obvious that

$$f(x_{k_i+1}) \leq (1 - \epsilon)f(x_{k_i}), \quad i = 0, 1, 2, \ldots.$$

Since sequence $\{f(x_k)\}$ is nonincreasing, the last inequality implies

$$f(x_{k_i+1}) \leq (1 - \epsilon)f(x_{k_{i-1}+1}), \quad i = 1, 2, \ldots.$$

This means that the subsequence $\{f(x_{k_i+1})\}$ converges to zero and hence the whole sequence $\{f(x_k)\}$ converges to zero. Consequently, every accumulation point of $\{x_k\}$ is a global solution of problem (1.1). In particular, (2.11) holds.

Next, suppose that K is finite. Without loss of generality, we suppose that K is empty. For the sake of contradiction, we assume that (2.11) does not hold. This means that there is a constant $\eta > 0$ such that $\|g(x_k)\| \geq \eta$ for all k. By the definition of y_k and s_k and the Lipschitz continuity of $J(x)$, it is not difficult to find constants $M_1 \geq m_1 > 0$ such that

$$\frac{y_k^T s_k}{\|s_k\|^2} \geq m_1, \quad \frac{\|y_k\|^2}{y_k^T s_k} \leq M_1.$$

It follows from Theorem 2.1 of [3] that there are positive constants a_1, a_2 and a_3 such that for every $k > 0$, inequalities

$$a_1\|p_i\|^2 \leq p_i^T B_i p_i \leq a_2\|p_i\|^2, \quad \|B_i p_i\| \leq a_3\|p_i\| \tag{2.12}$$

hold for at least a half of the indices $i \leq k$. Let K_1 be the set of indices such that (2.12) holds. Then K_1 is infinite. By the use of Lemma 5.2 of [13], there is a constant $\lambda > 0$ such that $\lambda_i \geq \lambda$ for all $i \in K_1$.

On the other hand, it is easy to see from (2.8) and the monotonicity of $\{f(x_k)\}$ that $\lim_{i\to\infty} \lambda_i g(x_i)^T p_i = 0$, which implies $\lim_{i\in K_1,\, i\to\infty} g(x_i)^T p_i = 0$. This together with (2.7) and (2.12) implies $\lim_{i\in K_1,\, i\to\infty} \|p_i\| = 0$ as well as $\lim_{i\in K_1,\, i\to\infty} \|g(x_i)\| = 0$, which is a contradiction. Consequently, (2.11) holds. □

3. The Unit Step and Convergence Rate

In this section, we show that the unit step is accepted for all k sufficiently large. We then show that Algorithm 1 is superlinearly convergent for nonzero residual problems and is quadratically convergent for zero residual problems. To this end, we need the following assumption.

Assumption A

(1) The sequence $\{x_k\}$ generated by Algorithm 1 converges to x^* at which $g(x^*) = 0$ and $G(x^*)$ is positive definite.

(2) The second derivative G is Hölder continuous at x^*, i.e., there exist constants $\nu > 0$ and $M > 0$ such that

$$\|G(x) - G(x^*)\| \leq M\|x - x^*\|^\nu, \tag{3.1}$$

for all x in a neighborhood of x^*.

It is clear that condition (1) of Assumption A implies that $G(x)$ is uniformly positive definite in a neighborhood $N(x^*)$ of x^*. That is, there exist constants $\Lambda \geq \lambda > 0$ such that

$$\lambda\|p\|^2 \leq p^T G(x)p \leq \Lambda\|p\|^2, \quad \forall x \in N(x^*). \tag{3.2}$$

In particular, (3.2) holds for all x_k when k is sufficiently large. Therefore,

$$\|g(x_k)\| = \|g(x_k) - g(x^*)\| \geq \lambda\|x_k - x^*\|$$

holds for all k sufficiently large. It is also not difficult to see that there are positive constants $m_2 \leq M_2$ and an integer $k_0 \geq 0$ such that inequalities

$$\frac{y_k^T s_k}{\|s_k\|^2} \geq m_2, \quad \frac{\|y_k\|^2}{y_k^T s_k} \leq M_2.$$

hold for all $k \geq k_0$.

Lemma 3.1 *There exists positive constants β_i, $i = 1, 2, 3$, such that for every $t > k_0$, the number of k for which inequalities*

$$\beta_1\|s_k\|^2 \leq s_k^T B_k s_k \leq \beta_2\|s_k\|^2 \quad \text{and} \quad \|B_k s_k\| \leq \beta_3\|s_k\| \tag{3.3}$$

hold is at least $\lceil t/2 \rceil - k_0$.

Proof Let K be defined by (2.9). If $f(x^*) \neq 0$, then K must be finite and the method reduces to the MBFGS method with B_k updated by (2.6). Inequalities in (3.3) follow from Theorem 2.1 of [3]. We turn to the case $f(x^*) = 0$. Without loss of generality, we assume $k_0 = 0$. By condition (1) of Assumption A, matrix $C(x_k)$ is uniformly positive definite. Let $K_t = \{k \in K \mid k \leq t\}$ and $\bar{K}_t = \{1, 2, \ldots, t\}\backslash K_t$. If the number of elements in $\bar{K}_t$ is less than the number of elements in K_t, then the number of $k \le t$ for which $B_k = C(x_k)$ is uniformly positive definite is at least $\lceil t/2 \rceil$. This implies that inequalities (3.3) hold for all $k \in K_t$ with some constants β_i, $i = 1, 2, 3$.

Suppose that the number of elements in K_t is less than the number of elements in $\bar{K}_t$. By Step 5 of Algorithm 1, $\{B_k\}_{k \in K_t}$ is bounded. In a way similar to the proof of Theorem 2.1 in [3], it is not difficult to prove that the number of k for which inequalities (3.3) with some constants β_i, $i = 1, 2, 3$ are satisfied is at least $\lceil (t - k_t)/2 \rceil$, where k_t denotes the number of elements in K_t. On the other hand, for $k \in K_t$, matrix $B_k = C(x_k)$ is uniformly positive definite, which implies that inequalities (3.3) are also satisfied for all $k \in K_t$ (with smaller β_1 and larger β_2 and β_3 if necessary). Therefore, the number of $k \le t$ for which inequalities (3.3) are satisfied is at least $\lceil t/2 \rceil$. The proof is complete. □

The following lemma can be proved in a way similar to the proof of Theorem 3.8 in [14].

Lemma 3.2 *Let Assumption A hold. Then we have*

$$\sum_{k=0}^{\infty} \|x_k - x^*\|^{\nu} < \infty. \tag{3.4}$$

For the sake of convenience, we introduce some notations. Let H_k be the inverse of B_k and $Q = G(x^*)^{-1/2}$. Let $\|\cdot\|_F$ denote the Frobenius norm of matrices. For a matrix A, we let $\|A\|_Q = \|Q^T A Q\|_F$. Denote

$$\tau_k = \max\{\|x_k - x^*\|^{\nu},\ \|x_{k+1} - x^*\|^{\nu}\} \quad \text{and}$$

and

$$\mu_k = \frac{\|Q^{-1}[H_k - G(x^*)^{-1}]y_k\|}{\|H_k - G(x^*)^{-1}\|_{Q^{-1}} \|Q y_k\|}. \tag{3.5}$$

Let $\gamma_k = g_{k+1} - g_k$. We get

$$\begin{aligned} \gamma_k - y_k &= J_{k+1}^T R_{k+1} - J_k^T R_k - [J_{k+1}^T J_{k+1} s_k + (J_{k+1} - J_k)^T R_{k+1} + t_k s_k] \\ &= J_k^T (R_{k+1} - R_k) - J_{k+1}^T J_{k+1} s_k - t_k s_k \\ &= J_{k+1}^T (R_{k+1} - R_k - J_{k+1} s_k) + (J_k - J_{k+1})^T (R_{k+1} - R_k) - t_k s_k. \end{aligned}$$

Therefore, there exist some constants $M_2 > 0$ and $M_3 > 0$ such that for all $k \notin K$ sufficiently large,

$$\begin{aligned} \|\gamma_k - y_k\| &\le \|J_{k+1}\| \|R_{k+1} - R_k - J_{k+1} s_k\| + \|J_k - J_{k+1}\| \|R_{k+1} - R_k\| + t_k \|s_k\| \\ &\le M_2 \|s_k\|^2 + t_k \|s_k\| \le M_2 \|s_k\|^2 + M_3 \|s_k\|^{\alpha+1}. \end{aligned} \tag{3.6}$$

Moreover, we have

$$\begin{aligned}\|y_k - G(x^*)s_k\| &\le \|y_k - \gamma_k\| + \|\gamma_k - G(x^*)s_k\| \\ &\le M_2\|s_k\|^2 + M_3\|s_k\|^{\alpha+1} + M\|s_k\|^{\nu+1} \end{aligned} \tag{3.7}$$

holds for all $k \notin K$ sufficiently large.

By the use of (3.6) and (3.7), similar to the proof of Lemma 3.7 in [13], it is not difficult to prove the following lemma.

Lemma 3.3 *Let Assumption A hold. Then there are positive constants* $b_i,\ i = 1, 2, \cdots, 7$ *and* $\xi \in (0, 1)$ *such that for all* $k \notin K$ *sufficiently large,*

$$\|B_{k+1} - G(x^*)\|_Q \le (1 + b_1\tau_k)\|B_k - G(x^*)\|_Q + b_2\tau_k + b_3t_k \tag{3.8}$$

and

$$\begin{aligned}\|H_{k+1} - G(x^*)^{-1}\|_{Q^{-1}} &\le (\sqrt{1 - \xi\mu_k^2} + b_4\tau_k + b_5t_k)\|H_k - G(x^*)^{-1}\|_{Q^{-1}} \\ &\quad + b_6\tau_k + b_7t_k. \end{aligned} \tag{3.9}$$

The following theorem shows that the Dennis-Moré condition holds.

Theorem 3.1 *Let Assumption A hold. Then the Dennis-Moré condition*

$$\lim_{k\to\infty} \frac{\|(B_k - G(x^*))s_k\|}{\|s_k\|} = 0 \tag{3.10}$$

holds.

Proof It is clear that for all $k \in K$,

$$\|B_{k+1} - G(x^*)\|_Q = \|C(x_{k+1}) - G(x^*)\|_Q$$

and

$$\|H_{k+1} - G(x^*)^{-1}\|_{Q^{-1}} = \|C(x_{k+1})^{-1} - G(x^*)^{-1}\|_{Q^{-1}}. \tag{3.11}$$

We also have that for all $k \notin K$, inequalities (3.8) and (3.9) hold. By inequality (3.4) and the twice continuous differentiability of the residual function R, it is not difficult to deduce that

$$\sum_{k=0}^{\infty} \tau_k < \infty \quad \text{and} \quad \sum_{k=0}^{\infty} t_k < \infty.$$

This implies that the sequences $\{\|B_k - G(x^*)\|_Q\}$ and $\{\|H_k - G(x^*)^{-1}\|_{Q^{-1}}\}$ are convergent. In particular, sequences $\{\|B_k\|\}$ and $\{\|H_k\|\}$ are bounded. Taking limits in both sides of (3.9) and

(3.11), we get either $\lim_{k\to\infty} \|H_k - G(x^*)^{-1}\|_{Q^{-1}} = 0$ or $\lim_{k\to\infty} \xi_k = 0$. Taking into account that $\{\|B_k\|\}$ and $\{\|H_k\|\}$ are bounded, we get

$$\lim_{k\to\infty} \frac{\|[H_k - G(x^*)^{-1}]y_k\|}{\|Qy_k\|} = 0. \tag{3.12}$$

This together with (2.7) and Assumption A implies (3.10). □

The next theorem shows that Algorithm 1 is superlinearly convergent for nonzero residual problems and is quadratically convergent for zero residual problems. Moreover, the unit step is accepted for all k sufficiently large.

Theorem 3.2 *Let Assumption A hold. If we choose $\sigma \in (0, 1/2)$, then $\{x_k\}$ converges to x^* superlinearly if $f(x^*) \neq 0$ and quadratically if $f(x^*) = 0$. Moreover, when k is sufficiently large, we always have $\lambda_k = 1$.*

Proof It is well-known that (3.10) guarantees that a quasi-Newton method with unit stepsize converges superlinearly. It then suffices to show that when k is sufficiently large, the unit stepsize is accepted. Since $\|p_k\| = \|H_k g_k\| \to 0$, by Taylor's expansion, it is not difficult to get from (2.7) and (3.10) that

$$f(x_k + p_k) - f(x_k) - \sigma g(x_k)^T p_k = -(\frac{1}{2} - \sigma)p_k^T G(x^*)p_k + o(\|p_k\|^2).$$

Therefore, we have for all k sufficiently large,

$$f(x_k + p_k) - f(x_k) - \sigma g(x_k)^T p_k \leq 0,$$

which implies that $\lambda_k = 1$ for all k sufficiently large. By (3.10) again, $\{x_k\}$ is at least superlinearly convergent.

If $f(x^*) = 0$, we have by the superlinear convergence of $\{x_k\}$ that

$$\lim_{k\to\infty} \frac{f(x_k) - f(x_{k+1})}{f(x_k)} = 1. \tag{3.13}$$

This shows that when k is sufficiently large, we always have $k \in K$. In other words, Algorithm 1 reduces to the Gauss-Newton method. Consequently, $\{x_k\}$ is quadratically convergent. □

Theorem 3.2 shows the superlinear/quadratic convergence property of Algorithm 1. Note that the superlinear convergence of $\{x_k\}$ yields (3.13). It is easy to see from Step 5 of Algorithm 1 that when k is sufficiently large, Algorithm 1 reduces to the MBFGS method if $f(x^*) \neq 0$, or the Gauss-Newton method if $f(x^*) = 0$. The MBFGS method and the Gauss-Newton method will not be used alternatively when k is sufficiently large.

4. Numerical Experiments

In this section, we present some preliminary numerical experiments and compare the performance of Algorithm 1 with some existing structured BFGS methods. The parameters in Algorithm 1 are specified as follows. We take $\epsilon = 0.2$, $\sigma = 0.1$ and $\rho = 0.36$. We choose the parameter α by the rule:

$$\alpha = \begin{cases} 0.01, & \text{if } \|g(x_{k+1})\| > 1, \\ 2, & \text{otherwise.} \end{cases}$$

We let the constant C in the update formula be $C = 10^{-6}$ if $\hat{y}_k^T s_k > 0$, and $C = 1$ elsewhere. The initial matrix A_0 is set to be $A_0 = 0.1 f_0^{\frac{1}{2}} I$ if $J_0^T J_0$ is nearly singular. Otherwise, we simple take $A_0 = 0$. The subproblems were solved by factorizing $B_k = L_k^T D_k L^k$, where L_k is an upper triangle matrix and D_k is a diagonal matrix. In the case where Gauss-Newton step is used and $J_k^T J_k$ is nearly singular, we use a modified Gauss-Newton step, namely,

$$B_k = J_k^T J_k + 0.1 f_k^{\frac{1}{2}} I.$$

We stop the iteration process if the inequality $\|g(x_k)\| < 10^{-4}$, or $\|f_k^{\frac{1}{2}}\| < 10^{-6}$ or $f(x_k) - f(x_{k+1}) < 10^{-15} \max\{1, f(x_{k+1})\}$ is satisfied. This stopping criterion was used in [1]. We also stop the algorithm if the number of iteration is greater than 300. The computation was done in a personal computer (Pentium IV, 1.8GHz with relation precision $2^{-52} = 2.2 \times 10^{-16}$) using Matlab 6.5.

We first test the performance of Algorithm 1 on 34 problems that comes from [16]. For each test problem, we test the algorithm starting from different initial points. These initial points are x^0, $x^1 = 100x^0$, $x^2 = -x^0$ and $x^3 = -100x^0$, where x^0 was the given initial point in [16]. We then test the performance of Algorithm 1 on the following three problems that comes from [1].

Problem I (Signomial problem)

$$r_i(x) = -e_i + \sum_{k=1}^{l} c_{ik} \prod_{j=1}^{n} x_j^{a_{ijk}} \quad i = 1, 2, \cdots, m.$$

Problem II (Exponential problem)

$$r_i(x) = -e_i + \sum_{k=1}^{l} c_{ik} exp(\sum_{j=1}^{n} a_{ijk} x_j) \quad i = 1, 2, \cdots, m.$$

Problem III (Trigonometric problem)

$$r_i(x) = -e_i + \sum_{j=1}^{n} (a_{ij} sin x_j + b_{ij} cos x_j) \quad i = 1, 2, \cdots, m.$$

The parameters e_i, a_{ijk}, c_{ik}, a_{ij}, b_{ij} in Problems I – III are generated randomly as described in [1]. While testing the performance of Algorithm 1 on these problems, we also choose the initial points x^0, $x^1 = 100x^0$, $x^2 = -x^0$ and $x^3 = -100x^0$, where x^0 was the given initial point in [1].

Table I shows the performance of Algorithm 1 on the above 37 problems where the columns have the following meaning:

Prob: the name of the test problem.
Dim: the dimension $(n-m)$ of the problem.
Init: the initial point.
It: the number of iterations.
GN: the number of iterations where the Gauss-Newton step is used.
Fun: the number of function evaluations.
$f^{1/2}$: the final value of $\sqrt{f(x)}$.
$\|g\|$: the final value of $\|\nabla f(x)\|$.
Un: the number of unit steps.

We see from Table I that for most test problems and most initial points, Algorithm 1 successfully terminates at stationary points of the problems. In general, for zero-residual problems and small residual problems, the Gauss-Newton step was used very often, and for large residual problems, the MBFGS step was used much. The unit step was generally accepted very often. We also observe that Algorithm 1 fails for a few problems with some initial points such as problem JENSAM with x^1, problem OSB1 with x^3 etc.. We use the symbol "-" to denote the case where Algorithm 1 fails. It should be pointed out that for all the test problems and all initial points, the nonconvergence of Algorithm 1 was caused by overflow.

We then compare the performance of Algorithm 1 with the method by Al-Baali, Fletcher and Xu (see [1, 11]), which we simplify as AFX. In the method AFX, the update rule of matrix B_k is given by

$$B_{k+1} = \begin{cases} C(x_{k+1}), & \text{if } \dfrac{f(x_k) - f(x_{k+1})}{f(x_k)} \geq \epsilon, \\ B_k - \dfrac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \dfrac{y_k y_k^T}{y_k^T s_k}, & \text{otherwise}, \end{cases} \tag{4.1}$$

where

$$y_k = \begin{cases} y_{\rm new}, & \text{if } s_k^T y_{\rm new} \geq 0.01 s_k^T y_{\rm old}, \\ y_{\rm old}, & \text{otherwise}, \end{cases} \tag{4.2}$$

$$y_{\rm new} = C(x_{k+1})s_k + (J(x_{k+1} - J_k)^T R(x_{k+1}),$$

and

$$y_{\rm old} = g_{k+1} - g_k.$$

We also compare the algorithm with the method by Huschens (see [12]), which we simplify as Husch. The update rule of matrix B_k in the method Husch is stated as follows.

$$y_k^{\#} = (J_{k+1} - J_k)^T \frac{R_{k+1}}{\|R(x_k)\|},$$

$$y_k = C(x_{k+1})s_k + \|R(x_{k+1})\| y_k^{\#},$$

$$B_k^s = C(x_{k+1}) + \|R(x_{k+1})\| A_k,$$

$$A_{k+1} = A_k + \triangle(s_k, y_k^{\#}, A_k, v(s_k, y_k, B_k^s)),$$

$$B_{k+1} = C(x_{k+1}) + \|R(x_{k+1})\| A_{k+1}.$$

We note that the positive definiteness of B_k in the methods AFX and Husch is not guaranteed. As a result, the generated direction p_k may not be a descent direction of f at x_k and the line search process may fail. In order to ensure the positive definiteness of B_k, we need to modify the update rules. Taking this into account, we only update B_k when the positive definiteness of B_{k+1} is guaranteed. Details are stated as follows. For the method AFX, we let $B_{k+1} = B_k$ if $y_k^T s_k \le 0$ and $(f_k - f_{k+1})/f_k < \epsilon$. Otherwise, we let B_k be determined by (4.1). For the method Husch, we only update B_k when B_k^s is positive definite and $y_k^T s_k > 0$.

Tables II and III list the performance of Algorithm 1, AFX and Husch with Armijo search and Wolfe-Powell search on 37 test problems, respectively. The suffix "-Ar" and '-WP" in the first lines of these two tables mean that the line search used are Armijo search and Wolfe-Powell search, respectively. For each problem, we compare the performance of Algorithm 1, AFX and Husch starting from 10 different initial points. These initial points are $\pm x^0$, $\pm 10x^0$, $\pm 100x^0$, $\pm 1000x^0$ and $\pm 10^4 x^0$. The meaning of the columns in Tables II and III is as follows.

Suc: the number of initial points with which the algorithms terminate at stationary points successfully.

Avit: the average number of iterations for all initial points.

Avgn: the average number of iterations where the Gauss-Newton step is used for all initial points.

Avf: the average number of function evaluations.

Avg: the average number of gradient evaluations.

We see from Tables II and III that Algorithm 1 is comparable with the method AFX either with Armijo search or with Wolfe-Powell search. In particular, for test problem BEALE, problem WOOD etc., the performance of Algorithm 1 is better than that of the method AFX. For other test problems and most initial points, Algorithm 1 in general performs as good as the method AFX does. Both Algorithm 1 and the method AFX perform much better than the method Husch does.

The results in Table II show that for most test problems and most initial points, Algorithm 1 with Armijo search successfully terminates at stationary points in very few iterations. Moreover, among the 40 test problems with 400 initial points, Algorithm 1 possess the highest successful termination ratio. The successful termination ratio of Algorithm 1 is 358/400=89.5%, while the successful termination ratio of AFX-AG method is 322/400=80.5% and the successful termination ratio of Husch-AG method is 267/400=66.75%. The results in Table III show that when Wolfe-Powell search is used, Algorithm 1 also possess the highest successful termination ratio. The successful termination ratio of Algorithm 1 with Wolfe-Powell search 358/400=89.5%, while the successful termination ratio of AFX-WP method is 351/400=87.75% and the successful termination ratio of Husch-WP method is 267/400=66.75%. From Table II and Table III, we can see that the successful termination ratio of Algorithm 1 is independent with the line search used, whereas the successful termination ratio of AFX-AG is less than one of AF-WP.

Table I Test Results of Algorithm 1 for Problems 1-37

Prob(Dim)	Init	It	Gn	Fun	Un	$f^{1/2}$	$\|g\|$	Init	It	GN	Fun	Un	$f^{1/2}$	$\|g\|$
ROSE (2-2)	x^0	19	11	36	9	0.0	0.0	x^2	2	2	3	2	0.0	0.0
	x^1	3	3	5	2	5.3e-12	1.7e-10	x^3	3	3	5	2	5.1e-12	1.6e-10
FROTH (2-2)	x^0	7	2	10	6	4.9	8.1e-6	x^2	9	4	19	6	4.9	4.3e-6
	x^1	19	13	22	18	4.9	3.4e-7	x^3	14	14	15	14	6.2e-8	3.7e-6
BADSCP (2-2)	x^0	25	25	47	4	6.8e-8	8.8e-3	x^2	54	32	119	21	2.0e-7	2.5e-2
	x^1	1	1	2	1	7.1e-5	7.1e-8	x^3	151	127	211	124	6.2e-7	8.0e-2
BADSCB (2-3)	x^0	21	4	55	13	1.1e-7	1.6e-1	x^2	21	3	56	13	2.7e-10	3.8e-4
	x^1	24	3	56	20	1.7e-7	2.4e-1	x^3	27	3	58	18	1.0e-7	1.5e-1
BEALE (2-3)	x^0	6	6	8	5	9.6e-7	6.5e-6	x^2	9	7	14	6	4.1e-6	2.8e-5
	x^1	172	6	192	162	4.8e-1	3.5e-6	x^3	118	6	132	109	4.8e-1	1.2e-5
JENSAM (2-10)	x^0	13	3	26	8	7.9	3.0e-5	x^2	16	5	64	9	7.9	2.8e-5
	x^1	0	0	1	0	–	–	x^3	0	0	1	0	3.2	3.7e-13
HELIX (3-3)	x^0	9	9	12	7	5.2e-9	1.3e-7	x^2	0	0	1	0	0.0	0.0
	x^1	17	16	30	15	4.8e-10	1.2e-8	x^3	1	1	2	1	0.0	0.0
BARD (3-15)	x^0	4	4	5	4	6.4e-2	1.6e-5	x^2	9	5	16	5	3.0	2.6e-10
	x^1	5	2	14	2	3.0	1.6e-7	x^3	5	3	13	3	3.0	2.2e-10
GAUSS (3-15)	x^0	1	1	2	1	7.5e-5	1.8e-8	x^2	9	9	13	7	7.5e-5	1.9e-7
	x^1	3	2	18	1	4.5e-1	3.5e-12	x^3	0	0	1	0	–	–
MEYER (3-16)	x^0	55	8	148	36	6.6	4.7e+1	x^2	20	7	21	20	2.7e+4	1.7e-7
	x^1	1	1	5	0	4.4e+4	1.0e-7	x^3	4	4	27	0	4.4e+4	1.8e-5
GULF (3-10)	x^0	56	12	89	42	5.2e-3	3.7e-6	x^2	9	1	11	8	2.1	9.2e-5
	x^1	0	0	1	0	1.4e-1	0.0	x^3	0	0	1	0	2.1	2.9e-80
BOX (3-10)	x^0	4	4	5	4	1.0e-5	2.1e-5	x^2	5	4	37	3	4.1e-11	2.7e-6
	x^1	34	22	37	32	1.9e-1	1.1e-6	x^3	0	0	1	0	–	–
SING (4-4)	x^0	8	8	9	8	1.4e-4	1.3e-5	x^2	8	8	9	8	1.4e-4	1.3e-5
	x^1	14	14	15	14	3.3e-4	5.1e-5	x^3	14	14	15	14	3.3e-4	5.1e-5
WOOD (4-6)	x^0	58	12	97	44	1.2e-9	3.6e-8	x^2	5	5	6	5	7.9e-9	2.4e-7
	x^1	63	18	93	51	1.1e-8	3.4e-7	x^3	12	12	13	12	4.5e-11	1.4e-9
KOWOSB (4-11)	x^0	3	3	5	2	1.2e-2	4.3e-5	x^2	15	13	28	6	1.2e-2	1.7e-5
	x^1	8	5	78	2	5.8e-2	5.4e-6	x^3	5	4	33	2	5.6e-2	1.8e-5
BD (4-20)	x^0	20	7	48	7	2.1e+2	2.2e-7	x^2	15	6	33	11	2.1e+2	1.4e-6
	x^1	21	10	55	15	2.1e+2	4.5e-5	x^3	22	13	31	18	2.1e+2	5.1e-5
OSB1 (5-33)	x^0	7	7	11	4	5.2e-3	1.2e-6	x^2	300	11	423	244	1.2e-1	2.4
	x^1	13	7	17	11	7.1e-1	1.6e-5	x^3	0	0	1	0	–	–
BIGGS (6-50)	x^0	44	17	64	29	2.2e-7	7.2e-7	x^2	27	8	28	27	3.9e-1	1.5e-6
	x^1	3	1	7	2	2.1	7.9e-5	x^3	0	0	1	0	–	–
OSB2 (11-65)	x^0	12	7	23	6	1.4e-1	9.3e-5	x^2	0	0	1	0	–	–
	x^1	26	6	32	23	9.5e-1	1.4e-5	x^3	0	0	1	0	–	–
WATSON (20-31)	x^0	4	4	5	4	8.8e-9	5.5e-7	x^2	4	4	5	4	8.8e-9	5.5e-7
	x^1	4	4	5	4	8.8e-9	5.5e-7	x^3	4	4	5	4	8.8e-9	5.5e-7
ROSEX (30-30)	x^0	19	11	36	9	0.0	0.0	x^2	2	2	3	2	0.0	0.0
	x^1	3	3	5	2	2.1e-11	6.6e-10	x^3	3	3	5	2	2.0e-11	6.3e-10
SINGX (40-40)	x^0	8	8	9	8	4.3e-4	4.3e-5	x^2	8	8	9	8	4.3e-4	4.3e-5
	x^1	15	15	16	15	2.6e-4	2.0e-5	x^3	15	15	16	15	2.6e-4	2.0e-5
PEN1 (30-31)	x^0	16	12	42	9	1.1e-2	4.0e-5	x^2	15	11	44	10	1.3e-2	4.7e-5
	x^1	21	18	43	16	1.1e-2	9.3e-6	x^3	26	19	53	19	1.3e-2	7.8e-5
PEN2 (30-60)	x^0	36	9	64	13	1.8e-1	2.7e-5	x^2	42	8	80	17	1.8e-1	4.7e-5
	x^1	78	44	144	21	1.8e-1	1.0e-4	x^3	90	56	163	28	1.8e-1	2.6e-5
VARDIM (30-32)	x^0	12	12	13	12	5.3e-8	7.3e-6	x^2	13	13	14	13	2.8e-8	3.8e-06
	x^1	18	18	19	18	8.9e-13	1.2e-10	x^3	18	18	19	18	1.1e-11	1.5e-9
TRIG (30-30)	x^0	9	6	33	2	4.9e-5	7.0e-5	x^2	4	4	5	4	4.5e-7	6.3e-7
	x^1	54	19	264	9	1.5e-2	3.1e-5	x^3	86	32	466	8	3.2e-3	3.4e-5
ALMOST (30-30)	x^0	2	2	3	2	5.1e-7	4.0e-6	x^2	4	4	5	4	2.4e-6	1.8e-5
	x^1	85	85	90	83	7.1e-1	5.2e-6	x^3	69	69	79	65	7.1e-1	3.0e-11
BV (30-30)	x^0	1	1	2	1	3.6e-5	1.5e-6	x^2	1	1	2	1	9.9e-4	3.6e-5
	x^1	7	7	8	7	7.7e-5	3.0e-6	x^3	7	7	8	7	1.4e-4	5.7e-6
IE (30-30)	x^0	2	2	3	2	4.2e-7	7.7e-7	x^2	3	3	4	3	4.6e-9	8.3e-9
	x^1	8	8	9	8	2.0e-6	3.7e-6	x^3	8	8	9	8	7.0e-6	1.3e-5
TRID (30-30)	x^0	4	4	5	4	7.5e-10	3.3e-9	x^2	81	3	152	70	9.0e-1	4.1e-5
	x^1	10	10	11	10	8.0e-6	3.3e-5	x^3	54	12	147	45	9.0e-1	8.8e-5
BAND (30-30)	x^0	5	5	6	5	1.1e-8	9.2e-8	x^2	13	5	23	9	1.2	3.2e-5
	x^1	16	16	17	16	9.8e-8	7.9e-7	x^3	25	16	35	21	1.2	7.3e-5
LIN (30-50)	x^0	1	1	2	1	3.2	8.6e-15	x^2	0	0	1	0	3.2	1.4e-15
	x^1	1	1	2	1	3.2	4.5e-13	x^3	1	1	2	1	3.2	4.4e-13
LIN1 (30-50)	x^0	2	2	3	2	2.5	2.2e-5	x^2	2	2	3	2	2.5	2.2e-5
	x^1	3	3	4	3	2.5	1.2e-6	x^3	3	3	4	3	2.5	2.2e-6
LIN0 (30-50)	x^0	2	2	3	2	2.6	2.4e-5	x^2	2	2	3	2	2.6	2.4e-5
	x^1	3	3	4	3	2.6	5.2e-6	x^3	3	3	4	3	2.6	3.6e-6
SIG (10-50)	x^0	34	14	35	34	3.2e+2	4.3e-5	x^2	37	14	39	36	3.3e+2	2.0e-5
	x^1	83	66	85	82	3.2e+2	1.0e-4	x^3	84	66	89	80	3.2e+2	2.7e-4
SIG (20-100)	x^0	67	15	88	62	4.4e+2	8.3e-4	x^2	52	16	72	49	4.8e+2	1.3e-3
	x^1	100	62	104	97	4.4e+2	1.3e-3	x^3	106	62	107	106	4.4e+2	6.5e-4
EXP (10-50)	x^0	14	1	17	12	2.7e+1	3.3e-5	x^2	14	2	16	13	2.7e+1	3.3e-5
	x^1	58	46	59	58	2.7e+1	2.1e-5	x^3	69	57	70	69	2.7e+1	9.9e-5
EXP (50-150)	x^0	30	2	33	28	4.0e+1	4.4e-5	x^2	43	3	50	37	4.0e+1	5.0e-5
	x^1	180	159	182	179	4.0e+1	7.9e-5	x^3	164	124	167	162	4.0e+1	5.6e-5
TRG (10-50)	x^0	38	13	40	37	3.0e+1	9.4e-6	x^2	37	13	47	32	2.9e+1	1.8e-5
	x^1	30	12	45	26	2.9e+1	3.0e-4	x^3	26	12	39	24	3.0e+1	1.1e-3
TRG (50-250)	x^0	67	15	114	53	6.0e+1	9.4e-3	x^2	56	12	121	45	6.0e+1	6.5e-3
	x^1	82	17	95	72	5.8e+1	5.7e-4	x^3	105	17	136	100	5.8e+1	5.5e-3

Table II Comparison of Algorithm 1, AFX and Husch with Armijo Search

		GN-MBFGS-Ar				AFX-Ar				Husch-Ar		
Prob	Dim	Suc	Avit	Avgn	Avf	Suc	Avit	Avgn	Avf	Suc	Avit	Avf
ROSE	2-2	10	4.4	3.6	7.7	10	4.4	3.6	7.7	3	24.7	31.0
FROTH	2-2	10	17.1	14.0	21.4	10	17.1	14.0	21.4	10	25.6	27.0
BADSCP	2-2	8	37.6	29.0	64.6	6	14.3	10.7	29.5	5	5.8	7.0
BADSCB	2-3	10	22.5	4.6	51.3	4	13.8	5.3	34.8	9	26.4	37.4
BEALE	2-3	9	131.1	6.6	155.0	1	6.0	6.0	8.0	6	22.5	31.7
JENSAM	2-10	7	12.3	7.6	32.1	5	10.8	8.4	14.2	6	33.2	38.0
HELIX	3-3	10	8.9	8.1	19.9	10	8.9	8.1	19.9	9	7.1	9.9
BARD	3-15	10	5.3	3.2	9.8	10	7.5	3.3	11.6	9	24.6	25.9
GAUSS	3-15	7	17.6	4.9	22.0	5	4.2	3.8	6.2	6	8.8	11.5
MEYER	3-16	8	4.8	3.1	20.5	8	3.1	3.0	18.9	0	–	–
GULF	3-10	8	8.1	1.6	13.3	7	1.3	0.1	2.4	7	1.0	2.0
BOX	3-10	6	19.0	9.2	34.8	4	22.0	11.3	30.5	6	24.0	29.8
SING	4-4	10	14.4	14.4	15.4	10	14.4	14.4	15.4	10	29.6	33.0
WOOD	4-6	10	35.8	14.6	49.3	5	11.6	11.6	12.6	6	21.0	22.5
KOWOSB	4-11	10	8.7	6.2	32.3	9	8.4	5.9	31.9	9	13.1	14.8
BD	4-20	10	22.5	12.0	40.7	10	22.5	12.0	41.0	9	22.6	26.6
OSB1	5-33	4	14.8	12.0	17.3	4	14.8	12.0	17.3	3	10.7	12.0
BIGGS	6-50	6	20.0	8.7	29.8	4	16.8	5.8	21.3	4	14.3	16.8
OSB2	11-65	5	11.8	4.8	16.8	4	7.3	4.0	12.0	5	13.2	18.6
WATSON	20-31	10	4.0	4.0	5.0	10	4.0	4.0	5.0	10	19.0	20.0
ROSEX	30-30	10	4.4	3.6	7.7	10	4.4	3.6	7.7	2	17.5	23.0
SINGX	40-40	10	14.6	14.6	15.6	10	14.6	14.6	15.6	10	34.2	36.4
PEN1	30-31	10	22.5	18.5	49.2	10	22.5	18.5	49.2	10	20.6	21.6
PEN2	30-60	9	82.6	47.3	156.3	9	83.3	47.3	157.3	4	132.8	187.8
VARDIM	30-32	10	18.2	18.2	19.2	10	18.2	18.2	19.2	10	30.2	33.0
TRIG	30-30	10	41.5	14.7	193.0	8	27.9	12.0	133.1	10	16.5	21.3
ALMOST	30-30	10	91.7	91.5	98.7	10	91.7	91.5	98.7	2	4.5	5.5
BV	30-30	10	8.2	8.2	9.2	10	8.2	8.2	9.2	10	34.2	35.2
IE	30-30	10	9.5	9.5	10.5	10	9.5	9.5	10.5	8	16.8	18.0
TRID	30-30	10	36.0	10.7	81.0	10	36.4	10.7	81.0	10	17.2	18.2
BAND	30-30	10	20.5	15.9	25.9	10	20.5	15.9	25.9	9	24.2	25.2
LIN	30-50	10	0.9	0.9	1.9	10	0.9	0.9	1.9	10	1.8	2.8
LIN1	30-50	10	3.7	3.4	4.9	10	3.7	3.4	4.9	10	2.3	3.3
LIN0	30-50	10	3.5	3.4	4.5	10	3.5	3.4	4.5	10	2.4	3.5
SIG	10-50	10	83.4	64.2	86.1	10	83.4	64.2	86.2	0	–	–
	10-50	10	98.7	60.0	107.1	10	97.8	60.0	104.6	0	–	–
EXP	10-50	6	31.2	19.3	32.7	6	31.2	19.3	32.7	4	26.5	28.0
	50-150	6	84.8	52.3	87.8	6	84.8	52.3	87.8	3	95.3	97.0
TRG	10-50	10	31.9	11.8	41.9	10	31.8	11.8	43.2	7	36.3	42.0
	50-250	9	70.3	15.2	95.6	7	73.6	15.4	94.4	6	66.3	87.2

Table III Comparison of Algorithm 1, AFX and Husch with Wolfe-Powell Search

	GN-MBFGS-WP					AFX-WP					Husch-WP			
Prob	Suc	Avit	Avgn	Avf	Avg	Suc	Avit	Avgn	Avf	Avg	Suc	Avit	Avf	Avg
ROSE	10	4.4	3.6	7.7	5.4	10	4.4	3.6	7.7	5.4	3	23.0	29.3	24.3
FROTH	10	17.1	14.0	21.4	18.1	10	17.1	14.0	21.4	18.1	10	25.0	26.9	26.5
BADSCP	8	40.9	27.6	64.9	44.0	8	43.1	29.3	70.1	46.3	5	5.8	7.0	6.8
BADSCB	10	13.8	4.8	35.3	18.4	10	13.8	4.8	35.3	18.4	10	20.5	32.1	28.3
BEALE	9	103.8	6.9	132.0	111.8	3	10.7	7.3	18.0	14.0	6	20.7	29.5	22.5
JENSAM	7	11.1	7.0	21.7	13.9	7	11.1	7.0	21.7	13.9	6	7.3	13.3	10.7
HELIX	10	8.9	8.1	19.9	9.9	10	8.9	8.1	19.9	9.9	10	7.8	12.2	10.2
BARD	10	5.3	3.2	9.8	6.3	10	7.8	3.3	12.5	9.4	10	13.7	19.9	19.3
GAUSS	7	19.6	7.0	27.7	24.9	7	18.7	7.4	27.6	24.3	6	3.8	8.7	7.8
MEYER	7	4.0	2.3	17.7	10.0	7	2.3	2.1	13.9	6.1	0	–	–	–
GULF	8	7.4	1.5	11.4	8.8	8	7.4	1.5	11.8	8.8	7	1.0	2.4	2.4
BOX	6	20.0	12.8	39.7	31.0	6	22.7	13.2	42.5	30.5	5	15.8	25.2	22.8
SING	10	14.4	14.4	15.4	15.4	10	14.4	14.4	15.4	15.4	10	30.2	33.8	31.6
WOOD	10	33.5	14.3	42.8	36.5	10	33.0	14.3	43.1	35.9	6	21.0	22.5	22.0
KOWOSB	10	10.0	5.9	33.6	12.1	10	10.5	6.5	36.5	13.5	8	11.4	15.1	14.6
BD	10	22.5	12.0	40.7	23.5	10	22.5	12.0	41.0	23.5	9	22.6	28.6	23.9
OSB1	6	33.5	15.7	58.7	41.0	5	19.8	16.6	31.6	26.4	5	11.6	19.2	16.4
BIGGS	6	15.3	6.3	21.7	18.2	6	14.8	6.0	20.5	17.8	5	11.6	15.8	15.0
OSB2	5	14.0	10.8	29.0	26.4	5	13.6	10.6	28.6	26.0	5	11.8	20.2	19.0
WATSON	10	4.0	4.0	5.0	5.0	10	4.0	4.0	5.0	5.0	10	19.0	20.0	20.0
ROSEX	10	4.4	3.6	7.7	5.4	10	4.4	3.6	7.7	5.4	1	12.0	15.0	13.0
SINGX	10	14.6	14.6	15.6	15.6	10	14.6	14.6	15.6	15.6	10	33.8	38.0	36.0
PEN1	10	22.5	18.4	52.1	24.5	10	22.4	18.4	50.6	24.4	10	20.6	21.6	21.6
PEN2	9	81.4	47.3	154.9	83.0	9	82.1	47.3	155.4	83.6	3	134.3	175.7	135.3
VARDIM	10	18.2	18.2	19.2	19.2	10	18.2	18.2	19.2	19.2	10	30.2	33.0	31.2
TRIG	10	37.8	14.8	178.4	39.5	10	37.9	14.7	183.8	39.3	8	11.6	15.4	12.9
ALMOST	10	92.2	91.9	101.7	94.1	10	92.2	91.9	101.7	94.1	2	4.5	5.5	5.5
BV	10	8.2	8.2	9.2	9.2	10	8.2	8.2	9.2	9.2	10	34.1	35.3	35.3
IE	10	9.5	9.5	10.5	10.5	10	9.5	9.5	10.5	10.5	8	16.8	18.0	17.8
TRID	10	35.0	10.7	79.4	36.1	10	35.5	10.7	79.8	36.6	10	17.0	18.4	18.2
BAND	10	20.3	15.9	26.1	21.4	10	20.3	15.9	26.1	21.4	9	24.2	25.2	25.2
LIN	10	0.9	0.9	1.9	1.9	10	0.9	0.9	1.9	1.9	10	1.8	6.3	6.3
LIN1	10	3.7	3.4	7.7	6.5	10	3.7	3.4	7.7	6.4	10	2.3	3.5	3.4
LIN0	10	3.5	3.4	4.5	4.5	10	3.5	3.4	4.5	4.5	10	2.4	6.6	5.5
SIG	10	83.4	64.2	86.1	84.4	10	83.4	64.2	86.2	84.4	0	–	–	–
	10	97.4	60.0	104.9	99.0	10	97.5	60.0	105.2	99.4	0	–	–	–
EXP	6	31.2	19.3	32.7	32.2	6	31.2	19.3	32.7	32.2	4	26.5	28.0	27.5
	6	84.8	52.3	87.8	85.8	6	84.8	52.3	87.8	85.8	3	95.3	97.0	96.3
TRG	10	31.6	11.8	53.5	40.3	10	31.6	11.8	53.8	40.2	7	34.6	46.0	38.3
	8	74.3	15.5	115.9	87.4	8	74.5	15.5	120.9	91.8	6	57.0	101.7	67.8

References

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

[2] M. C. Bartholomew-Biggs, The estimation of the Hessian matrix in nonlinear least squares problems with non-zero residuals, *Mathematical Programming*, 12 (1977), 67-80.

[3] R. Byrd and J. Nocedal, A tool for the analysis of quasi-Newton methods with application to unconstrained minimization, *SIAM Journal on Numerical Analysis*, 26 (1989), 727-739.

[4] Y. Dai, Convergence properties of the BFGS algorithm, *SIAM Journal on Optimization*, 13 (2003), 693-701.

[5] J. E. Dennis, JR., A brief survey of convergence results for quasi-Newton methods, *SIAM-AMS Proceedings*, 9 (1976), 185-199.

[6] J. E. Dennis, H.J. Martinez, and R.A. Tapia, Convergence theory for the structured BFGS secant method with an application to nonlinear least squares, *Journal of Optimization Theory and Applications*, 61 (1989), 161-178.

[7] J. E. Dennis Jr., D. M. Gay and R. E. Welsch, An adaptive nonlinear least-squares algorithm, *ACM transactions on mathematical software*, 7 (1981), 348-368.

[8] J. E. Dennis, JR. and H. F. Walker, Convergence theorems for least-change secant update methods, *SIAM Journal on Numerical Analysis*, 18 (1981), 949-987.

[9] J.R. Engels and H.J. Martínez, Local and superlinear convergence for partially known quasi-Newton methods, *SIAM Journal on Optimization*, 1 (1991), 42-56.

[10] J.Y. Fan and Y.X. Yuan, On the convergence of a new Levenberg-Marquardt method, Technical Report, AMSS, Chinese Academy of Sciences, 2001.

[11] R. Fletcher and C. Xu, Hybrid methods for nonlinear least squares, *IMA Journal of Numerical Analysis*, 7 (1987), 371-389.

[12] J. Huschens, On the use of product structure in secant methods for nonlinear least squares problems, *SIAM Journal on Optimization*, 4 (1994), 108-129.

[13] D. H. Li and M. Fukushima, A modified BFGS method and its global convergence in nonconvex minimization, *Journal of Computational and Applied Mathematics*, 129 (2001), 15-35.

[14] D. H. Li and M. Fukushima, On The Global Convergence of the BFGS Method for Non-convex Unconstrained Optimization Problems, *SIAM Jornal on Optimization*, 11 (2001), 1054-1064.

[15] L. Lukšan and E. Spedicato, Variable metric methods for unconstrained optimization and nonlinear least squares, *Journal oc Computational and Applied Mathematics*, 124 (2000), 61-95.

[16] J.J. Moré, B.S. Garbow and K. Hillstrom, Testing unconstrained optimization software, *ACM Transactions on Mathematical Software*, 7 (1981), 17-41.

[17] M. J. D. Powell, Some global convergence properties of a variable metric algorithm for minimization without exact line searches, in *Nonlinear Programming*, SIAM-AMS Proceedings, Vol. IX, R.W. Cottle and C.E. Lemke, eds. AMS Providence, RI, 1976, 55-92.

[18] H. Yabe, Quadratic and superlinear convergence of the Huschens method for nonlinear least squares problems, *Computational Optimization and Applications*, 10 (1998), 79-103.

[19] H. Yabe, and T. Takahashi, Structured quasi-Newton methods for nonlinear least squares problems, *TRU Mathematics*, 24 (1988), 195–209.

[20] H. Yabe, and T. Takahashi, Numerical comparison among structured quasi-Newton methods for nonlinear least squares problems, *Journal of the Operations Research Society of Japan*, 34 (1991), 287–305.

[21] H. Yabe, and T. Takahashi, Factorized quasi-Newton methods for nonlinear least squares problems, *Mathematical Programming*, 51 (1991), 75–100.

[22] H. Yabe and N. Yamaki, Convergence of a factorized Broyden-like family for nonlinear least squares problems, *SIAM Journal on Optimization*, 5 (1995), 770-791.

[23] N. Yamashita and M. Fukushima, On the rate of convergence of the Levenberg-Marquardt method, *Computing*, 15 [Suppl] (2001), pp. 239–249.

[24] J. Z. Zhang, L. H. Chen and N. Y. Deng, A family of scaled factorized Broyden-like methods for nonlinear least squares problems, *SIAM Journal on Optimization*, 10 (2000), 1163–1179.