Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 716019, 9 pages
http://dx.doi.org/10.1155/2014/716019

Research Article

A Limited Memory BFGS Method for Solving Large-Scale Symmetric Nonlinear Equations

Xiangrong Li, Xiaoliang Wang, and Xiabin Duan

College of Mathematics and Information Science, Guangxi University, Nanning, Guangxi 530004, China

Correspondence should be addressed to Xiangrong Li; xrli68@163.com

Received 21 June 2014; Accepted 14 July 2014; Published 5 August 2014

Academic Editor: Gonglin Yuan

A limited memory BFGS (L-BFGS) algorithm is presented for solving large-scale symmetric nonlinear equations, where a line search technique without derivative information is used. The global convergence of the proposed algorithm is established under some suitable conditions. Numerical results show that the given method is competitive to those of the normal BFGS methods.

1. Introduction

Consider

$$h(x) = 0, \quad x \in \mathfrak{R}^n, \tag{1}$$

where $h : \mathfrak{R}^n \to \mathfrak{R}^n$ is continuously differentiable, the Jacobian $\nabla h(x)$ of g is symmetric for all $x \in \mathfrak{R}^n$, and n denotes the large-scale dimensions. It is not difficult to see that if g is the gradient mapping of some function $f : \mathfrak{R}^n \to \mathfrak{R}$, problem (1) is the first order necessary condition for the problem $\min_{x\in\mathfrak{R}^n} f(x)$. Furthermore, considering

$$\min f(u) \quad \text{s.t. } a(u) = 0, \tag{2}$$

where a is a vector-valued function, then the KKT conditions can be represented as the system (1) with $x = (u, v)$ and $h(u, v) = (\nabla f(u) + \nabla a(u)v, a(u))$, where v is the vector of Lagrange multipliers. The above two cases show that problem (1) can come from an unconstrained problem or an equality constrained optimization problem in theory. Moreover, there are other practical problems that can also take the form of (1), such as the discretized two-point boundary value problem, the saddle point problem, and the discretized elliptic boundary value problem (see Chapter 1 of [1] in detail). Let θ be the norm function $\theta(x) = (1/2)\|h(x)\|^2$; then problem (1) is equivalent to the following global optimization problem:

$$\min \theta(x) \quad x \in \mathfrak{R}^n, \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm.

In this paper we will focus on the line search method for (1), where its normal iterative formula is defined by

$$x_{k+1} = x_k + \alpha_k d_k, \tag{4}$$

where d_k is the so-called search direction and α_k is a steplength along d_k. To begin with, we briefly review some methods for α_k.

(i) Normal Line Search (Brown and Saad [2]). The stepsize α_k is determined by

$$\theta(x_k + \alpha_k d_k) - \theta(x_k) \le \sigma\alpha_k \nabla\theta(x_k)^T d_k, \tag{5}$$

where $\sigma \in (0, 1)$ and $\nabla\theta(x_k) = \nabla h(x_k)h(x_k)$. The convergence is proved and some good results are obtained. We all know that the nonmonotone idea is more interesting than the normal technique in many cases. Then a nonmonotone line search technique based on this motivation is presented by Zhu [3].

(ii) Nonmonotone Line Search (Zhu [3]). The stepsize α_k is determined by

$$\theta(x_k + \alpha_k d_k) - \theta(x_{l(k)}) \le \sigma\alpha_k \nabla\theta(x_k)^T d_k, \tag{6}$$

$\theta(x_{l(k)}) = \max_{0\le j\le m(k)}\{\theta(x_{k-j})\}, m(0) = 0, m(k) = \min\{m(k-1) + 1, M\}$ (for $k \ge 1$), and M is a nonnegative integer. The global convergence and the superlinear convergence are established under mild conditions, respectively. It is not difficult to see that, for the above two line search techniques, the Jacobian matrix ∇h_k must be computed at each iteration, which obviously increase the workload and the CPU time consumed. In order to avoid this drawback, Yuan and Lu [4] presented a new backtracking inexact technique.

(iii) A New Line Search (Yuan and Lu [4]). The stepsize α_k is determined by

$$\|h(x_k + \alpha_k d_k)\|^2 \le \|h(x_k)\|^2 + \delta\alpha_k^2 h_k^T d_k, \tag{7}$$

where $\delta \in (0,1)$ and $h_k = h(x_k)$. They established the global convergence and the superlinear convergence. And the numerical tests showed that the new line search technique is more effective than those of the normal line search technique. However, these three line search techniques can not directly ensure the descent property of d_k. Thus more interesting line search techniques are studied.

(iv) Approximate Monotone Line Search (Li and Fukushima [5]). The stepsize α_k is determined by

$$\begin{aligned}\theta(x_k + \alpha_k d_k) - \theta(x_k) \le & -\delta_1\|\alpha_k d_k\|^2 \\ & -\delta_2\|\alpha_k h_k\|^2 + \epsilon_k\|h_k\|^2,\end{aligned} \tag{8}$$

where $\alpha_k = r^{i_k}$, $r \in (0,1)$, i_k is the smallest nonnegative integer i satisfying (8), $\delta_1 > 0$ and $\delta_2 > 0$ are constants, and ϵ_k is such that

$$\sum_{k=0}^{\infty}\epsilon_k < \infty. \tag{9}$$

The line search (8) can be rewritten as

$$\begin{aligned}\|h(x_k + \alpha d_k)\|^2 \le & (1+\epsilon_k)\|h_k\|^2 - \delta_1\|\alpha h_k\|^2 \\ & -\delta_2\|\alpha d_k\|^2;\end{aligned} \tag{10}$$

it is straightforward to see that as $\alpha \to 0$, the right-hand side of the above inequality goes to $(1+\epsilon_k)\|h_k\|^2$. Then it is not difficult to see that the sequence $\{x_k\}$ generated by one algorithm with line search (8) is approximately norm descent. In order to ensure the sequence $\{x_k\}$ is norm descent, Gu et al. [6] presented the following line search.

(v) Monotone Descent Line Search (Gu et al. [6]). The stepsize α_k is determined by

$$\begin{aligned}\|h(x_k + \alpha_k d_k)\|^2 - \|h_k\|^2 \le & -\delta_1\|\alpha_k h_k\|^2 \\ & -\delta_2\|\alpha_k d_k\|^2,\end{aligned} \tag{11}$$

where α_k, δ_1, and δ_2 are similar to (8).

In the following, we present some techniques for d_k.

(i) Newton Method. The search direction d_k is defined by

$$\nabla h(x_k)\, d_k = -h(x_k). \tag{12}$$

Newton method is one of the most effective methods since it normally requires a fewest number of function evaluations and is very good at handling ill-conditioning. However, its efficiency largely depends on the possibility to efficiently solve a linear system (12) which arises when computing. Moreover, the exact solution of the system (12) could be too burdensome or is not necessary when x_k is far from a solution [7]. Thus the quasi-Newton methods are proposed.

(ii) Quasi-Newton Method. The search direction d_k is defined by

$$B_k d_k + h_k = 0, \tag{13}$$

where B_k is the quasi-Newton update matrix. The quasi-Newton methods represent the basic approach underlying most of the Newton-type large-scale algorithms (see [3, 4, 8], etc.), where the famous BFGS method is one of the most effective quasi-Newton methods, generated by the following formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k {y_k}^T}{{y_k}^T s_k}, \tag{14}$$

where $s_k = x_{k+1} - x_k$ and $y_k = h_{k+1} - h_k$ with $h_k = h(x_k)$ and $h_{k+1} = h(x_k + \alpha_k d_k)$. By (11) and (14), Yuan and Yao [9] proposed a BFGS method for nonlinear equations and some good results were obtained. Denote $H_k = B_k^{-1}$, and then (14) has the inverse update formula represented by

$$\begin{aligned}H_{k+1} &= H_k - \frac{{y_k}^T(s_k - H_k y_k)s_k s_k^T}{({y_k}^T s_k)^2} \\ &\quad + \frac{(s_k - H_k y_k)s_k^T + s_k(s_k - H_k y_k)^T}{({y_k}^T s_k)^2} \\ &= \left(I - \frac{s_k {y_k}^T}{{y_k}^T s_k}\right) H_k \left(I - \frac{y_k s_k^T}{{y_k}^T s_k}\right) + \frac{s_k s_k^T}{{y_k}^T s_k}.\end{aligned} \tag{15}$$

Unfortunately, both the Newton method and the quasi-Newton method require many space to store $n \times n$ matrix at every iteration, which will prevent the efficiency of the algorithm for problems, especially for large-scale problems. Therefore low storage matrix information method should be necessary.

(iii) Limited Memory Quasi-Newton Method. The search direction d_k is defined by

$$d_k = -H_k h_k, \tag{16}$$

where H_k is generated by limited memory quasi-Newton method, where the famous limited memory quasi-Newton method is the so-called limited memory BFGS method. The L-BFGS method is an adaptation of the BFGS method for large-scale problems (see [10] in detail), which often requires minimal storage and provides a fast rate of linear convergence. The L-BFGS method has the following form:

$$\begin{aligned} H_{k+1} &= V_k^T H_k V_k + \rho_k s_k s_k^T \\ &= V_k^T \left[V_{k-1}^T H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T\right] V_k + \rho_k s_k s_k^T \\ &= \cdots \\ &= \left[V_k^T \cdots V_{k-\widetilde{m}+1}^T\right] H_{k-\widetilde{m}+1} \left[V_{k-\widetilde{m}+1} \cdots V_k\right] \\ &\quad + \rho_{k-\widetilde{m}+1} \left[V_{k-1}^T \cdots V_{k-\widetilde{m}+2}^T\right] \\ &\quad \times s_{k-\widetilde{m}+1} s_{k-\widetilde{m}+1}^T \left[V_{k-\widetilde{m}+2} \cdots V_{k-1}\right] + \cdots + \rho_k s_k s_k^T, \end{aligned} \tag{17}$$

where $\rho_k = 1/s_k^T y_k$, $V_k = I - \rho_k y_k s_k^T$, $\widetilde{m} > 0$ is an integer, and I is the unit matrix. Formula (17) shows that matrix H_k is obtained by updating the basic matrix $H_0 \widetilde{m}$ times using BFGS formula with the previous $\widetilde{m}$ iterations. By (17), together with (7) and (8), Yuan et al. [11] presented the L-BFGS method for nonlinear equations and got the global convergence. At present, there are many papers proposed for (1) (see [6, 12–15], etc.).

In order to effectively solve large-scale nonlinear equations and possess good theory property, based on the above discussions of α_k and d_k, we will combine (11) and (16) and present a L-BFGS method for (1) since (11) can make the norm function be descent and (16) need less low storage. The main attributes of the new algorithm are stated as follows.

(i) A L-BFGS method with (11) is presented.

(ii) The norm function is descent.

(iii) The global convergence is established under appropriate conditions.

(iv) Numerical results show that the given algorithm is more competitive than the normal algorithm for large-scale nonlinear equations.

This paper is organized as follows. In the next section, the backtracking inexact L-BFGS algorithm is stated. Section 3 will present the global convergence of the algorithm under some reasonable conditions. Numerical experiments are done to test the performance of the algorithms in Section 4.

2. Algorithms

This section will state the L-BFGS method in association with the new backtracking line search technique (11) for solving (1).

Algorithm 1.

Step 0. Choose an initial point $x_0 \in \mathfrak{R}^n$, an initial symmetric positive definite matrix $H_0 \in \mathfrak{R}^{n\times n}$, positive constants δ_1, δ_2, constants $r, \rho \in (0,1)$, and a positive integer m_1. Let $k := 0$.

Step 1. Stop if $\|h_k\| = 0$.

Step 2. Determine d_k by (16).

Step 3. If

$$\|h(x_k + d_k)\| \le \rho \|h(x_k)\|, \tag{18}$$

then take $\alpha_k = 1$ and go to Step 5. Otherwise go to Step 4.

Step 4. Let i_k be the smallest nonnegative integer i such that (11) holds for $\alpha = r^i$. Let $\alpha_k = r^{i_k}$.

Step 5. Let the next iterative be $x_{k+1} = x_k + \alpha_k d_k$.

Step 6. Let $\widetilde{m} = \min\{k+1, m_1\}$. Put $s_k = x_{k+1} - x_k = \alpha_k d_k$ and $y_k = h_{k+1} - h_k$. Update H_0 for $\widetilde{m}$ times to get H_{k+1} by (17).

Step 7. Let $k := k + 1$. Go to Step 1.

In the following, to conveniently analyze the global convergence, we assume that the algorithm updates B_k (the inverse of H_k) with the basically bounded and positive definite matrix B_0 (H_0's inverse). Then Algorithm 1 with B_k has the following steps.

Algorithm 2.

Step 2. Determine d_k by

$$B_k d_k = -h_k. \tag{19}$$

Step 6. Let $\widetilde{m} = \min\{k+1, m_1\}$. Put $s_k = x_{k+1} - x_k = \alpha_k d_k$ and $y_k = h_{k+1} - h_k$. Update B_0 for $\widetilde{m}$ times; that is, for $l = k - \widetilde{m} + 1, \ldots, k$ compute

$$B_k^{l+1} = B_k^l - \frac{B_k^l s_l s_l^T B_k^l}{s_l^T B_k^l s_l} + \frac{y_l {y_l}^T}{{y_l}^T s_l}, \tag{20}$$

where $s_l = x_{l+1} - x_l$, $y_l = h_{l+1} - h_l$, and $B_k^{k-\widetilde{m}+1} = B_0$ for all k.

Remark 3. Algorithms 1 and 2 are mathematically equivalent. Throughout this paper, Algorithm 2 is given only for the purpose of analysis, so we only discuss Algorithm 2 in theory. In the experiments, we implement Algorithm 1.

3. Global Convergence

Define the level set Ω by

$$\Omega = \{x \mid \|h(x)\| \le \|h(x_0)\|\}. \tag{21}$$

In order to establish the global convergence of Algorithm 2, similar to [4, 11], we need the following assumptions.

Assumption A. g is continuously differentiable on an open convex set Ω_1 containing Ω. Moreover the Jacobian of g is symmetric, bounded, and positive definite on Ω_1; namely, there exist positive constants $M \ge m > 0$ satisfying

$$\begin{gathered} \|\nabla h(x)\| \le M \quad \forall x \in \Omega_1, \\ m\|d\|^2 \le d^T \nabla h(x)\, d \quad \forall x \in \Omega_1,\ d \in \mathfrak{R}^n. \end{gathered} \tag{22}$$

Assumption B. B_k is a good approximation to ∇g_k; that is,

$$\left\|(\nabla h_k - B_k)\, d_k\right\| \le \epsilon \left\|h_k\right\|, \tag{23}$$

where $\epsilon \in (0,1)$ is a small quantity.

Remark 4. Assumption A implies

$$\left\|y_k\right\| \le M \left\|s_k\right\|, \qquad {y_k}^T s_k \ge m\left\|s_k\right\|^2. \tag{24}$$

The relations in (24) can ensure that B_{k+1} generated by (20) inherits symmetric and positive definiteness of B_k. Thus, (19) has a unique solution for each k. Moreover, the following lemma holds.

Lemma 5 (see Theorem 2.1 in [16] or see Lemma 3.4 of [11]). *Let Assumption A hold and let $\{\alpha_k, d_k, x_{k+1}, g_{k+1}\}$ be generated by Algorithm 2. Then, for any $r_0 \in (0,1)$ and $k \ge 0$, there are positive constants β_j, $j = 1,2,3$; the following relations*

$$\beta_2\left\|s_i\right\|^2 \le s_i^T B_i s_i \le \beta_3\left\|s_i\right\|^2, \qquad \left\|B_i s_i\right\| \le \beta_1 \left\|s_i\right\| \tag{25}$$

hold for at least $\lceil r_0 k\rceil$ values of $i \in [0,k]$.

By Assumption B, similar to [4, 9, 11, 15], it is easy to get the following lemma.

Lemma 6. *Let Assumption B hold and let $\{\alpha_k, d_k, x_{k+1}, h_{k+1}\}$ be generated by Algorithm 2. Then d_k is a descent direction for $\theta(x)$ at x_k; that is, $\nabla\theta(x_k)^T d_k < 0$ holds.*

Based on the above lemma, by Assumption B, similar to Lemma 3.8 in [2], we can get the following lemma.

Lemma 7. *Let Assumption B hold and let $\{\alpha_k, d_k, x_{k+1}, h_{k+1}\}$ be generated by Algorithm 2. Then $\{x_k\} \subset \Omega$. Moreover, $\{\|h_k\|\}$ converges.*

Lemma 8. *Let Assumptions A and B hold. Then, in a finite number of backtracking steps, Algorithm 2 will produce an iterate $x_{k+1} = x_k + \alpha_k d_k$.*

Proof. It is sufficient for us to prove that the line search (11) is reasonable. By Lemma 3.8 in [2], we can deduce that, in a finite number of backtracking steps, α_k is such that

$$\begin{aligned}\left\|h(x_k + \alpha_k d_k)\right\|^2 - \left\|h(x_k)\right\|^2 &\le \delta\alpha_k h(x_k)^T \nabla h(x_k)\, d_k,\\ &\qquad \delta \in (0,1).\end{aligned} \tag{26}$$

By (19), we get

$$\begin{aligned}\nabla\theta(x_k)^T d_k &= h(x_k)^T \nabla h(x_k)\, d_k\\ &= h(x_k)^T \left[(\nabla h(x_k) - B_k)\, d_k - h(x_k)\right]\\ &= h(x_k)^T (\nabla h(x_k) - B_k)\, d_k - h(x_k)^T h(x_k).\end{aligned} \tag{27}$$

Thus

$$\begin{aligned}\nabla\theta(x_k)^T d_k + \left\|h_k\right\|^2 &\le h(x_k)^T (\nabla h(x_k) - B_k)\, d_k\\ &\le \left\|h(x_k)\right\| \left\|(\nabla h(x_k) - B_k)\, d_k\right\|.\end{aligned} \tag{28}$$

By Assumption B, we have

$$\begin{aligned}\nabla\theta(x_k)^T d_k &\le \left\|h(x_k)\right\| \left\|(\nabla h(x_k) - B_k)\, d_k\right\| - \left\|h(x_k)\right\|^2\\ &\le -(1-\epsilon)\left\|h(x_k)\right\|^2.\end{aligned} \tag{29}$$

Using (19) again and $\alpha_k \le 1$, we obtain

$$\begin{aligned}\alpha_k h(x_k)^T \nabla h(x_k)\, d_k &\le -\alpha_k (1-\epsilon)\left\|h(x_k)\right\|^2\\ &= -\frac{(1-\epsilon)}{2\alpha_k}\left\|\alpha_k h_k\right\|^2 - \frac{(1-\epsilon)}{2\alpha_k}\left\|\alpha_k B_k d_k\right\|^2\\ &\le -\frac{(1-\epsilon)}{2}\left\|\alpha_k h_k\right\|^2 - \frac{\beta_2^2(1-\epsilon)}{2}\left\|\alpha_k d_k\right\|^2.\end{aligned} \tag{30}$$

Setting $\delta_1 \in (0, \delta((1-\epsilon)/2))$ and $\delta_2 \in (0, \delta(\beta_2^2(1-\epsilon)/2))$ implies (11). This completes the proof. □

Remark 9. The above lemma shows that Algorithm 2 is well defined. By a way similar to Lemma 3.2 and Corollary 3.4 in [5], it is not difficult to deduce that

$$\alpha_k \ge \frac{\beta_2 r}{\beta_1^2\sigma_1 + \sigma_2 + M^2} \tag{31}$$

holds; we do not prove it anymore. Now we establish the global convergence theorem.

Theorem 10. *Let Assumptions A and B hold. Then the sequence $\{x_k\}$ generated by Algorithm 2 converges to the unique solution x^* of* (1).

Proof. Lemma 7 implies that $\{\|h_k\|\}$ converges. If

$$\lim_{k\to\infty}\left\|h_k\right\| = 0, \tag{32}$$

then every accumulation point of $\{x_k\}$ is a solution of (1). Assumption A means that (1) has only one solution. Moreover, since Ω is bounded, $\{x_k\} \subseteq \Omega$ has at least one accumulation point. Therefore $\{x_k\}$ itself converges to the unique solution of (1). Therefore, it suffices to verify (32).

If (18) holds for infinitely many k's, then (32) is trivial. Otherwise, if (18) holds for only finitely many k's, we conclude that Step 3 is executed for all k sufficiently large. By (11), we have

$$\delta_1\left\|\alpha_k h_k\right\|^2 + \delta_2\left\|s_k\right\|^2 \le \left\|h_k\right\|^2 - \left\|h_{k+1}\right\|^2. \tag{33}$$

Since $\{\|h_k\|\}$ is bounded, by adding these inequalities, we get

$$\sum_{k=0}^{\infty}\left\|\alpha_k h_k\right\|^2 < \infty, \qquad \sum_{k=0}^{\infty}\left\|\alpha_k d_k\right\|^2 < \infty. \tag{34}$$

Then we have

$$\lim_{k\to\infty}\left\|\alpha_k h_k\right\| = 0, \tag{35}$$

which together with (31) implies (32). This completes the proof. □

4. Numerical Results

This section reports numerical results with Algorithm 1 and normal BFGS algorithm. The test problems with the associated initial guess x_0 are listed with

$$h(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T. \tag{36}$$

Problem 1. Exponential function 1:

$$\begin{aligned} f_1(x) &= e^{x_1 - 1} - 1, \\ f_i(x) &= i\left(e^{x_i - 1} - x_i\right), \quad i = 2, 3, \ldots, n. \end{aligned} \tag{37}$$

Initial guess: $x_0 = (1/n^2, 1/n^2, \ldots, 1/n^2)^T$.

Problem 2. Exponential function 2:

$$\begin{aligned} f_1(x) &= e^{x_1} - 1, \\ f_i(x) &= \frac{i}{10}\left(e^{x_i} + x_{i-1} - i\right), \quad i = 2, 3, \ldots, n. \end{aligned} \tag{38}$$

Initial guess: $x_0 = (1/n^2, 1/n^2, \ldots, 1/n^2)^T$.

Problem 3. Trigonometric function:

$$\begin{aligned} f_i(x) = 2&\left(n + i(1 - \cos x_i) - \sin x_i - \sum_{j=1}^{n} \cos x_j\right) \\ &\times (2\sin x_i - \cos x_i), \quad i = 1, 2, 3, \ldots, n. \end{aligned} \tag{39}$$

Initial guess: $x_0 = (101/100n, 101/100n, \ldots, 101/100n)^T$.

Problem 4. Singular function:

$$\begin{aligned} f_1(x) &= \frac{1}{3}x_1^3 + \frac{1}{2}x_2^2, \\ f_i(x) &= -\frac{1}{2}x_i^2 + \frac{i}{3}x_i^3 + \frac{1}{2}x_{i+1}^2, \quad i = 2, 3, \ldots, n-1, \\ f_n(x) &= -\frac{1}{2}x_n^2 + \frac{n}{3}x_n^3. \end{aligned} \tag{40}$$

Initial guess: $x_0 = (1, 1, \ldots, 1)^T$.

Problem 5. Logarithmic function:

$$f_i(x) = \ln(x_i + 1) - \frac{x_i}{n}, \quad i = 1, 2, 3, \ldots, n. \tag{41}$$

Initial guess: $x_0 = (1, 1, \ldots, 1)^T$.

Problem 6. Broyden tridiagonal function [17, pages 471-472]:

$$\begin{aligned} f_1(x) &= (3 - 0.5x_1)x_1 - 2x_2 + 1, \\ f_i(x) &= (3 - 0.5x_i)x_i - x_{i-1} + 2x_{i+1} + 1, \\ &\qquad i = 2, 3, \ldots, n-1, \\ f_n(x) &= (3 - 0.5x_n)x_n - x_{n-1} + 1. \end{aligned} \tag{42}$$

Initial guess: $x_0 = (-1, -1, \ldots, -1)^T$.

Problem 7. Trigexp function [17, page 473]:

$$\begin{aligned} f_1(x) &= 3x_1^3 + 2x_2 - 5 + \sin(x_1 - x_2)\sin(x_1 + x_2), \\ f_i(x) &= -x_{i-1}e^{x_{i-1} - x_i} + x_i\left(4 + 3x_i^2\right) + 2x_{i+1} \\ &\quad + \sin(x_i - x_{i+1})\sin(x_i + x_{i+1}) - 8, \\ &\qquad i = 2, 3, \ldots, n-1, \\ f_n(x) &= -x_{n-1}e^{x_{n-1} - x_n} + 4x_n - 3. \end{aligned} \tag{43}$$

Initial guess: $x_0 = (0, 0, \ldots, 0)^T$.

Problem 8. Strictly convex function 1 [18, page 29]: $h(x)$ is the gradient of $h(x) = \sum_{i=1}^{n}(e^{x_i} - x_i)$. Consider

$$f_i(x) = e^{x_i} - 1, \quad i = 1, 2, 3, \ldots, n. \tag{44}$$

Initial guess: $x_0 = (1/n, 2/n, \ldots, 1)^T$.

Problem 9. Linear function-full rank:

$$f_i(x) = x_i - \frac{2}{n}\sum_{j=1}^{n} x_j + 1. \tag{45}$$

Initial guess: $x_0 = (100, 100, \ldots, 100)^T$.

Problem 10. Penalty function:

$$\begin{aligned} f_i(x) &= \sqrt{10^{-5}}(x_i - 1), \quad i = 1, 2, 3, \ldots, n-1, \\ f_n(x) &= \left(\frac{1}{4n}\right)\sum_{j=1}^{n} x_j^2 - \frac{1}{4}. \end{aligned} \tag{46}$$

Initial guess: $x_0 = (1/3, 1/3, \ldots, 1/3)^T$.

Problem 11. Variable dimensioned function:

$$\begin{aligned} f_i(x) &= x_i - 1, \quad i = 1, 2, 3, \ldots, n-2, \\ f_{n-1}(x) &= \sum_{j=1}^{n-2} j(x_j - 1), \\ f_n(x) &= \left(\sum_{j=1}^{n-2} j(x_j - 1)\right)^2. \end{aligned} \tag{47}$$

Initial guess: $x_0 = (1 - (1/n), 1 - (2/n), \ldots, 0)^T$.

Problem 12. Tridiagonal system [19]:

$$\begin{aligned} f_1(x) &= 4\left(x_1 - x_2^2\right), \\ f_i(x) &= 8x_i\left(x_i^2 - x_{i-1}\right) - 2(1 - x_i) \\ &\quad + 4\left(x_i - x_{i+1}^2\right), \quad i = 2, 3, \ldots, n-1, \\ f_n(x) &= 8x_n\left(x_n^2 - x_{n-1}\right) - 2(1 - x_n). \end{aligned} \tag{48}$$

Initial guess: $x_0 = (12, 12, \ldots, 12)$.

Table 1

Nr.	Dim	Algorithm 1			Normal BFGS algorithm		
		NI/NG	GN	Time	NI/NG	GN	Time
Problem 1	500	24/25	$9.908338e-05$	$2.901619e+01$	23/87	NaN	$1.950013e+00$
	1000	9/10	$5.697414e-05$	$5.366434e+01$	23/87	NaN	$1.124767e+01$
	1500	9/10	$3.363290e-05$	$1.689803e+02$	NI > 1000	Inf	$8.094892e+01$
	2000	9/10	$2.706880e-05$	$3.880213e+02$	NI > 1000	Inf	$1.550962e+02$
Problem 2	500	8/16	$3.660478e-05$	$6.926444e+00$	60/390	NaN	$7.269647e+00$
	1000	8/16	$6.406401e-05$	$4.513109e+01$	40/314	NaN	$2.857938e+01$
	1500	9/17	$7.655321e-05$	$1.689959e+02$	33/258	NaN	$7.102726e+01$
	2000	9/17	$9.651851e-05$	$3.881617e+02$	29/226	NaN	$1.410717e+02$
Problem 3	500	18/33	$7.747640e-06$	$2.110694e+01$	NI > 1000	$5.905028e+04$	$1.220396e+02$
	1000	17/32	$8.381440e-05$	$1.270472e+02$	NI > 1000	$1.577282e+05$	$5.830069e+02$
	1500	17/32	$6.658002e-05$	$3.975217e+02$	NI > 1000	$2.929920e+05$	$1.645561e+03$
	2000	17/32	$5.689277e-05$	$9.154451e+02$	NI > 1000	$4.345902e+05$	$3.776363e+03$
Problem 4	500	809/3134	$9.614778e-05$	$1.113785e+03$	9/73	NaN	$1.107607e+00$
	1000	960/3894	$9.865549e-05$	$8.674373e+03$	8/65	NaN	$5.725237e+00$
	1500	197/695	$8.562547e-05$	$5.509394e+03$	8/65	NaN	$1.584970e+01$
	2000	220/676	$9.847745e-05$	$1.418851e+04$	8/65	NaN	$3.584903e+01$
Problem 5	500	6/7	$4.925073e-06$	$4.305628e+00$	6/7	$4.925073e-06$	$7.488048e-01$
	1000	6/7	$6.747222e-06$	$2.725337e+01$	6/7	$6.747222e-06$	$4.477229e+00$
	1500	6/7	$8.176417e-06$	$8.352294e+01$	6/7	$8.176417e-06$	$1.340049e+01$
	2000	6/7	$9.391335e-06$	$1.925364e+02$	6/7	$9.391335e-06$	$3.001459e+01$
Problem 6	500	96/97	$9.183793e-05$	$1.272812e+02$	65/66	$9.473359e-05$	$8.346053e+00$
	1000	17/18	$7.206980e-05$	$1.263140e+02$	63/64	$8.700852e-05$	$4.623870e+01$
	1500	17/18	$6.663689e-05$	$3.956965e+02$	63/64	$9.605270e-05$	$1.402917e+02$
	2000	17/18	$6.136239e-05$	$9.107026e+02$	64/65	$8.724048e-05$	$3.194588e+02$
Problem 7	500	14/15	$7.551200e-05$	$1.533490e+01$	52/53	$9.761790e-05$	$6.333641e+00$
	1000	15/16	$5.530847e-05$	$1.084051e+02$	55/63	$8.871231e-05$	$3.985826e+01$
	1500	14/15	$8.445286e-05$	$3.119708e+02$	60/68	$7.722045e-05$	$1.336617e+02$
	2000	15/16	$4.705617e-05$	$7.805354e+02$	13/63	NaN	$5.564556e+01$
Problem 8	500	6/7	$4.600878e-05$	$4.196427e+00$	6/7	$2.375490e-05$	$7.020045e-01$
	1000	6/7	$6.434846e-05$	$2.658257e+01$	6/7	$3.327487e-05$	$4.461629e+00$
	1500	6/7	$7.851881e-05$	$8.319533e+01$	6/7	$4.062329e-05$	$1.335369e+01$
	2000	6/7	$9.049762e-05$	$1.928484e+02$	6/7	$4.683284e-05$	$2.985859e+01$
Problem 9	500	2/10	$2.027911e-10$	$5.460035e-01$	NI > 1000	$4.514037e+117$	$1.037407e+01$
	1000	2/10	$2.666034e-10$	$2.698817e+00$	NI > 1000	$6.383813e+117$	$2.861058e+01$
	1500	2/10	$1.379742e-09$	$8.330453e+00$	NI > 1000	$7.818542e+117$	$5.319634e+01$
	2000	2/10	$1.701735e-09$	$1.942212e+01$	NI > 1000	$9.028075e+117$	$8.375694e+01$
Problem 10	500	435/2865	$5.154286e-05$	$5.901362e+02$	41/210	NaN	$3.806424e+00$
	1000	637/4215	$5.701191e-05$	$5.749432e+03$	67/425	NaN	$3.689424e+01$
	1500	303/1914	$4.003848e-05$	$8.526469e+03$	63/491	NaN	$1.146919e+02$
	2000	473/3281	$3.558910e-05$	$3.074095e+04$	64/499	NaN	$2.622845e+02$
Problem 11	500	1/2	$0.000000e+00$	$1.092007e-01$	1/2	$0.000000e+00$	$1.560010e-01$
	1000	1/2	$0.000000e+00$	$7.956051e-01$	1/2	$0.000000e+00$	$6.864044e-01$
	1500	1/2	$0.000000e+00$	$2.277615e+00$	1/2	$0.000000e+00$	$2.324415e+00$
	2000	1/2	$0.000000e+00$	$5.101233e+00$	1/2	$0.000000e+00$	$5.210433e+00$
Problem 12	500	260/800	$1.654459e-05$	$3.509398e+02$	8/51	NaN	$7.800050e-01$
	1000	324/1053	$8.997051e-05$	$2.902289e+03$	8/51	NaN	$5.226033e+00$
	1500	254/829	$1.257117e-05$	$7.123224e+03$	8/51	NaN	$1.583410e+01$
	2000	372/1353	$9.331122e-05$	$2.414117e+04$	8/51	NaN	$3.046700e+01$

Table 1: Continued.

Nr.	Dim	Algorithm 1			Normal BFGS algorithm		
		NI/NG	GN	Time	NI/NG	GN	Time
Problem 13	500	96/209	$9.194537e-05$	$1.287632e+02$	10/74	NaN	$1.185608e+00$
	1000	53/89	$5.718492e-05$	$4.531049e+02$	10/74	NaN	$6.021639e+00$
	1500	NI > 1000	$1.180195e+10$	$2.835008e+04$	10/74	NaN	$1.790891e+01$
	2000	54/132	$3.919816e-05$	$3.337298e+03$	10/74	NaN	$4.031066e+01$
Problem 14	500	18/68	$5.826508e-05$	$2.076373e+01$	16/115	NaN	$6.240040e-01$
	1000	18/68	$8.239959e-05$	$1.357989e+02$	16/115	NaN	$3.354021e+00$
	1500	19/69	$5.002681e-05$	$4.532921e+02$	16/115	NaN	$9.765663e+00$
	2000	19/69	$5.774900e-05$	$1.042820e+03$	16/115	NaN	$2.148134e+01$
Problem 15	500	8/9	$6.870902e-05$	$6.910844e+00$	22/23	$8.899350e-05$	$2.667617e+00$
	1000	7/8	$7.203786e-05$	$3.561503e+01$	16/17	$9.828207e-05$	$1.174688e+01$
	1500	7/8	$4.809670e-05$	$1.122739e+02$	13/14	$9.654107e-05$	$2.903179e+01$
	2000	7/8	$3.609946e-05$	$2.583845e+02$	11/12	$9.465949e-05$	$5.519315e+01$
Problem 16	500	0/1	$0.000000e+00$	$0.000000e+00$	0/1	$0.000000e+00$	$0.000000e+00$
	1000	0/1	$0.000000e+00$	$0.000000e+00$	0/1	$0.000000e+00$	$0.000000e+00$
	1500	0/1	$0.000000e+00$	$1.560010e-02$	0/1	$0.000000e+00$	$0.000000e+00$
	2000	0/1	$0.000000e+00$	$1.560010e-02$	0/1	$0.000000e+00$	$1.560010e-02$

Problem 13. Five-diagonal system [19]:

$$\begin{aligned} f_1(x) &= 4\left(x_1 - x_2^2\right) + x_2 - x_3^2, \\ f_2(x) &= 8x_2\left(x_2^2 - x_1\right) - 2\left(1 - x_2\right) + 4\left(x_2 - x_3^2\right) + x_3 - x_4^2, \\ f_i(x) &= 8x_i\left(x_i^2 - x_{i-1}\right) - 2\left(1 - x_i\right) + 4\left(x_i - x_{i+1}^2\right) + x_{i-1}^2 \\ &\quad - x_{i-2} + x_{i+1} - x_{i+2}^2, \quad i = 3, 4, \dots, n-2, \\ f_{n-1}(x) &= 8x_{n-1}\left(x_{n-1}^2 - x_{n-2}\right) - 2\left(1 - x_{n-1}\right) \\ &\quad + 4\left(x_{n-1} - x_n^2\right) + x_{n-2}^2 - x_{n-3}, \\ f_n(x) &= 8x_n\left(x_n^2 - x_{n-1}\right) - 2\left(1 - x_n\right) + x_{n-1}^2 - x_{n-2}. \end{aligned} \tag{49}$$

Initial guess: $x_0 = (-2, -2, \dots, -2)$.

Problem 14. Extended Freudentein and Roth function (n is even) [20]: for $i = 1, 2, \dots, n/2$

$$\begin{aligned} f_{2i-1}(x) &= x_{2i-1} + \left(\left(5 - x_{2i}\right)x_{2i} - 2\right)x_{2i} - 13, \\ f_{2i}(x) &= x_{2i-1} + \left(\left(1 + x_{2i}\right)x_{2i} - 14\right)x_{2i} - 29. \end{aligned} \tag{50}$$

Initial guess: $x_0 = (6, 3, 6, 3, \dots, 6, 3)$.

Problem 15. Discrete boundry value problem [21]:

$$\begin{aligned} f_1(x) &= 2x_1 + 0.5h^2\left(x_1 + t\right)^3 - x_2, \\ f_i(x) &= 2x_i + 0.5h^2\left(x_i + ti\right)^3 - x_{i-1} + x_{i+1}, \\ &\qquad i = 2, 3, \dots, n-1, \\ f_n(x) &= 2x_n + 0.5h^2\left(x_n + tn\right)^3 - x_{n-1}, \\ &\qquad t = \frac{1}{n+1}. \end{aligned} \tag{51}$$

Initial guess: $x_0 = (t(t-1), t(2t-1), \dots, t(nt-1))$.

Problem 16. Troesch problem [22]:

$$\begin{aligned} f_1(x) &= 2x_1 + \varrho h^2 \sin t\left(\varrho x_1\right) - x_2, \\ f_i(x) &= 2x_i + \varrho h^2 \sin t\left(\varrho x_i\right) - x_{i-1} - x_{i+1}, \\ &\qquad i = 2, 3, \dots, n-1, \\ f_n(x) &= 2x_n + \varrho h^2 \sin t\left(\varrho x_n\right) - x_{n-1}, \\ &\qquad t = \frac{1}{n+1}, \quad \varrho = 10. \end{aligned} \tag{52}$$

Initial guess: $x_0 = (0, 0, \dots, 0)$.

In the experiments, the parameters in Algorithm 1 and the normal BFGS method were chosen as $r = 0.1$, $\rho = 0.5$, $m_1 = 6$, $\delta_1 = \delta_2 = 0.001$, H_0 and is the unit matrix. All codes were written in MATLAB r2013b and run on PC with

6600@2.40 GHz Core 2 CPU processor and 4.00 GB memory and Windows 7 operation system. We stopped the program when the condition $\|h(x)\| \leq 10^{-4}$ was satisfied. Since the line search cannot always ensure the descent condition $d_k^T h_k < 0$, uphill search direction may occur in the numerical experiments. In this case, the line search rule maybe fails. In order to avoid this case, the stepsize α_k will be accepted if the searching time is larger than eight in the inner circle for the test problems. We also stop this program if the iteration number arrived at 1000. The columns of the tables have the following meaning.

Dim: the dimension. NI: the total number of iterations.

NG: the number of the norm function evaluations. Time: the CPU time in second.

GN: the normal value of $\| h(x) \|$ when the program stops.

NaN: not-a-number, impling that the code fails to get a real value.

Inf: returning the IEEE arithmetic representation for positive infinity or infinity which is also produced by operations like dividing by zero.

From the numerical results in Table 1, it is not difficult to show that the proposed method is more successful than the normal BFGS method. We can see that there exist many problems which can not be successfully solved by the normal BFGS method. Moreover, the normal BFGS method fails to get real value for several problems. Then we can conclude that the presented method is more competitive than the normal BFGS method.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank Professor Gonglin Yuan of Guangxi University for his suggestions in the paper organization and help in the codes of the program, which save them much time to complete this paper. The authors also thank the referees for valuable comments and the editor for suggestions in the idea and the English of this paper, which improves the paper greatly. This work is supported by Guangxi NSF (Grant no. 2012GXNSFAA053002) and China NSF (Grant no. 11261006).

References

[1] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY. USA, 1970.

[2] P. N. Brown and Y. Saad, "Convergence theory of nonlinear Newton-Krylov algorithms," *SIAM Journal on Optimization*, vol. 4, no. 2, pp. 297–330, 1994.

[3] D. Zhu, "Nonmonotone backtracking inexact quasi-Newton algorithms for solving smooth nonlinear equations," *Applied Mathematics and Computation*, vol. 161, no. 3, pp. 875–895, 2005.

[4] G. Yuan and X. Lu, "A new backtracking inexact BFGS method for symmetric nonlinear equations," *Computers & Mathematics with Applications*, vol. 55, no. 1, pp. 116–129, 2008.

[5] D. Li and M. Fukushima, "A globally and superlinearly convergent Gauss-Newton-based BFGS method for symmetric nonlinear equations," *SIAM Journal on Numerical Analysis*, vol. 37, no. 1, pp. 152–172, 1999.

[6] G. Gu, D. Li, L. Qi, and S. Zhou, "Descent directions of quasi-Newton methods for symmetric nonlinear equations," *SIAM Journal on Numerical Analysis*, vol. 40, no. 5, pp. 1763–1774, 2002.

[7] S. G. Nash, "A survey of truncated-Newton methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 45–59, 2000.

[8] A. Griewank, "The "global" convergence of Broyden-like methods with a suitable line search," *Australian Mathematical Society B*, vol. 28, no. 1, pp. 75–92, 1986.

[9] G. Yuan and S. Yao, "A BFGS algorithm for solving symmetric nonlinear equations," *Optimization*, vol. 62, no. 1, pp. 85–99, 2013.

[10] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of quasi-Newton matrices and their use in limited memory methods," *Mathematical Programming*, vol. 63, no. 1–3, pp. 129–156, 1994.

[11] G. Yuan, Z. Wei, and S. Lu, "Limited memory BFGS method with backtracking for symmetric nonlinear equations," *Mathematical and Computer Modelling*, vol. 54, no. 1-2, pp. 367–377, 2011.

[12] G. Yuan, "A new method with descent property for symmetric nonlinear equations," *Numerical Functional Analysis and Optimization*, vol. 31, no. 7–9, pp. 974–987, 2010.

[13] G. Yuan and X. Li, "A rank-one fitting method for solving symmetric nonlinear equations," *Journal of Applied Functional Analysis*, vol. 5, no. 4, pp. 389–407, 2010.

[14] G. Yuan, X. Lu, and Z. Wei, "BFGS trust-region method for symmetric nonlinear equations," *Journal of Computational and Applied Mathematics*, vol. 230, no. 1, pp. 44–58, 2009.

[15] G. Yuan, Z. Wei, and X. Lu, "A BFGS trust-region method for nonlinear equations," *Computing*, vol. 92, no. 4, pp. 317–333, 2011.

[16] R. H. Byrd and J. Nocedal, "A tool for the analysis of quasi-Newton methods with application to unconstrained minimization," *SIAM Journal on Numerical Analysis*, vol. 26, no. 3, pp. 727–739, 1989.

[17] M. A. Gomes-Ruggiero, J. M. Martínez, and A. C. Moretti, "Comparing algorithms for solving sparse nonlinear systems of equations," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 2, pp. 459–483, 1992.

[18] M. Raydan, "The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem," *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 26–33, 1997.

[19] G. Y. Li, "Successive column correction algorithms for solving sparse nonlinear systems of equations," *Mathematical Programming*, vol. 43, no. 2, pp. 187–207, 1989.

[20] Y. Bing and G. Lin, "An efficient implementation of Merrill's method for sparse or partially separable systems of nonlinear equations," *SIAM Journal on Optimization*, vol. 1, no. 2, pp. 206–221, 1991.

[21] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Testing unconstrained optimization software," *ACM Transactions on Mathematical Software*, vol. 7, no. 1, pp. 17–41, 1981.

[22] S. M. Roberts and J. J. Shipman, "On the closed form solution of Troesch's problem," *Journal of Computational Physics*, vol. 21, no. 3, pp. 291–304, 1976.