Hindawi Publishing Corporation
Advances in Operations Research
Volume 2009, Article ID 909753, 22 pages
doi:10.1155/2009/909753

Research Article

A Trust-Region-Based BFGS Method with Line Search Technique for Symmetric Nonlinear Equations

Gonglin Yuan,[1] Shide Meng,[2] and Zengxin Wei[1]

[1] *College of Mathematics and Information Science, Guangxi University, Nanning, Guangxi 530004, China*
[2] *Department of Mathematics and Computer Science, Yulin Teacher's College, Yulin, Guangxi 537000, China*

Correspondence should be addressed to Shide Meng, glyuan@yahoo.cn

Received 29 April 2009; Revised 19 August 2009; Accepted 28 October 2009

Recommended by Khosrow Moshirvaziri

A trust-region-based BFGS method is proposed for solving symmetric nonlinear equations. In this given algorithm, if the trial step is unsuccessful, the linesearch technique will be used instead of repeatedly solving the subproblem of the normal trust-region method. We establish the global and superlinear convergence of the method under suitable conditions. Numerical results show that the given method is competitive to the normal trust region method.

1. Introduction

Consider the following system of nonlinear equations:

$$g(x) = 0, \quad x \in \Re^n, \tag{1.1}$$

where $g : \Re^n \to \Re^n$ is continuously differentiable, and the Jacobian $\nabla g(x)$ of g is symmetric for all $x \in \Re^n$. Let ϑ be the norm function defined by $\vartheta(x) = 1/2\|g(x)\|^2$. Then the nonlinear equations (1.1) is equivalent to the following global optimization problem:

$$\min \vartheta(x), \quad x \in \Re^n. \tag{1.2}$$

There are two ways for nonlinear equations by numerical methods. One is the line search method and the other is the trust region method. For the line search method, the

following iterative formula is often used to solve (1.1):

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.3}$$

where x_k is the kth iteration point, α_k is a steplength, and d_k is search direction. To begin, we briefly review some methods for (1.1) by line search technique. First, we give some techniques for α_k. Brown and Saad [1] proposed the following line search method to obtain the stepsize α_k :

$$\vartheta(x_k + \alpha_k d_k) - \vartheta(x_k) \leq \sigma\alpha_k \nabla\vartheta(x_k)^T d_k, \tag{1.4}$$

where $\sigma \in (0,1)$. Based on this technique, Zhu [2] gave the nonmonotone line search technique:

$$\vartheta(x_k + \alpha_k d_k) - \vartheta\left(x_{l(k)}\right) \leq \sigma\alpha_k \nabla\vartheta(x_k)^T d_k, \tag{1.5}$$

$\|\vartheta(x_{l(k)})\| = \max_{0\leq j\leq m(k)}\{\|\vartheta(x_{k-j})\|\}$, $m(0) = 0$ and $m(k) = \min\{m(k-1)+1, M\}$, $k \geq 1$, and M is a nonnegative integer. From these two techniques (1.4) and (1.5), it is easy to see that the Jacobian matrix ∇g_k must be computed at every iteration, which will increase the workload especially for large-scale problems or this matrix is expensive to calculate. Considering these points, we [3] presented a new backtracking inexact technique to obtain the stepsize α_k:

$$\left\|g(x_k + \alpha_k d_k)\right\|^2 \leq \left\|g(x_k)\right\|^2 + \delta\alpha_k^2 g_k^T d_k, \tag{1.6}$$

where $\delta \in (0,1)$, $g_k = g(x_k)$, and d_k is a solution of the system of linear (1.15). We established the global convergence and the superlinear convergence of this method. The numerical results showed that the new line search technique is more effective than the normal methods. Li and Fukashima [4] proposed an approximate monotone line search technique to obtain the step-size α_k satisfying

$$\vartheta(x_k + \alpha_k d_k) - \vartheta(x_k) \leq -\delta_1\|\alpha_k d_k\|^2 - \delta_2\left\|\alpha_k g_k\right\|^2 + \epsilon_k\left\|g(x_k)\right\|^2, \tag{1.7}$$

where $\delta_1 > 0$ and $\delta_2 > 0$ are positive constants, $\alpha_k = r^{i_k}$, $r \in (0,1)$, i_k is the smallest nonnegative integer i such that (1.7), and ϵ_k satisfies

$$\sum_{k=0}^{\infty}\epsilon_k < \infty. \tag{1.8}$$

Combining the line search (1.7) with one special BFGS update formula, they got some better results (see [4]). Inspired by their idea, Wei [5] and Yuan [6–8] presented several approximate methods. Further work can be found in [9].

Second, we present some techniques for d_k. One of the most effective methods is Newton method. It normally requires a fewest number of function evaluations, and it is very good at handling ill-conditioning. However, its efficiency largely depends on the possibility

of solving a linear system efficiently which arises when computing the search d_k in each iteration:

$$\nabla g(x_k)d_k = -g(x_k). \tag{1.9}$$

Moreover, the exact solution of the system (1.9) could be too burdensome, or it is not necessary when x_k is far from a solution [10]. Inexact Newton methods [2, 3, 10] represent the basic approach underlying most of the Newton-type large-scale algorithms. At each iteration, the current estimate of the solution is updated by approximately solving the linear system (1.9) using an iterative algorithm. The inner iteration is typically "truncated" before the solution to the linear system is obtained. Griewank [11] firstly proposed the Broyden's rank one method for nonlinear equations and obtained the global convergence. At present, a lot of algorithms have been proposed for solving these two problems (1.1) and (1.2)(see [12–22] etc.).

Trust region method is a kind of important and efficient methods in the area of nonlinear optimization. This method can be traced back to the works of Levenberg [17] and Marquardt [18] on nonlinear least-squares problems and the work of Goldfeld et al. [23] for unconstrained optimization. Powell [24] was the first to establish the convergence result of trust region method for unconstrained optimization. Fletcher [25, 26] firstly proposed trust region algorithms for linearly constrained optimization problems and nonsmooth optimization problems, respectively. This method has been studied by many authors [15, 27–31] and has been applied to equality constrained problems [32–34]. Byrd et al. [35], Fan [36], Powell and Yuan [37], Vardi [38], Yuan [39, 40], Yuan et al. [41], and Zhang and Zhu [42] proposed various trust region algorithms for constrained optimization problems and established the convergence. Fan [36], Yuan [39], and Zhang [43] presented the trust region algorithms for nonlinear equations and got some results.

The normal trust-region subproblem for nonlinear equations is to find the trial step d_k such that

$$\begin{aligned} &\min\ q_k^*(d) = d^T\nabla g(x_k)g(x_k) + \frac{1}{2}d^T\nabla g(x_k)^T\nabla g(x_k)d \\ &\text{s.t. } \|d\| \le \Delta_k, \end{aligned} \tag{1.10}$$

where $\Delta_k > 0$ is a scalar called the trust region radium. Define the predicted descent of the objective function $g(x)$ at kth iteration by

$$\mathrm{Pred}_k^* = q_k^*(0) - q_k^*(d_k), \tag{1.11}$$

the actual descent of $g(x)$ by

$$\mathrm{Ared}_k^* = \vartheta(x_k) - \vartheta(x_k + d_k), \tag{1.12}$$

and the ratio of actual descent to predicted descent:

$$r_k^* = \frac{\mathrm{Ared}_k^*}{\mathrm{Pred}_k^*}. \tag{1.13}$$

For the normal trust region algorithm, if $r_k^* \geq \rho$ ($\rho \in (0,1)$, this case is called a successful iteration), the next iteration is $x_{k+1} = x_k + d_k$, and go to the next step; otherwise reduce the trust region radium Δ_k and solve this subproblem (1.10) repeatedly. Sometimes, we must do this work many times and compute the Jacobian matrix $\nabla g(x_k)$ and $\nabla g(x_k)^T \nabla g(x_k)$ at every time, which obviously increases the work time and workload, especially for large-scale problems. Even more detrimental, the trust region subproblem is not very easy (see [36, 39] etc.) to be solved for most of the practical problems.

In order to alleviate the above bad situation that traditional algorithms have to compute Jacobian matrix $\nabla g(x_k)$ and $\nabla g(x_k)^T \nabla g(x_k)$ at each and every iteration while repeatedly resolving the trust region subproblem, in this paper, we would like to rewrite the following trust-region subproblem as

$$\begin{aligned} &\min\ q_k(d) = g(x_k)^T d + \frac{1}{2} d^T B_k d \\ &\text{s.t. } \|d\| \leq \Delta_k, \end{aligned} \tag{1.14}$$

where matrix B_k is the approximation to the Jacobian matrix of $g(x)$ at x_k. Due to the boundness of the region $\{d \mid \|d\| \leq \Delta_k\}$, (1.14) has a solution regardless of $B_k's$ definiteness (see [43]). This implies that it is valid to adopt a BFGS update formula to generate B_k for trust region methods and the BFGS update is presented as follows:

$$B_{k+1} = B_k + \frac{y_k y_k^T}{s_k^T y_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}, \tag{1.15}$$

where $y_k = g_{k+1} - g_k$, $s_k = x_{k+1} - x_k$. Define the predicted descent of the objective function $g(x)$ at kth iteration by

$$P\text{red}_k = q_k(0) - q_k(d_k), \tag{1.16}$$

the actual descent of $g(x)$ by

$$A\text{red}_k = \left\|g(x_k)\right\|^2 - \left\|g(x_k + d_k)\right\|^2, \tag{1.17}$$

and the ratio of actual descent to predicted descent:

$$r_k = \frac{\left\|g(x_k)\right\|^2 - \left\|g(x_k + d_k)\right\|^2}{q_k(0) - q_k(d_k)}. \tag{1.18}$$

If $r_k \geq \rho$ ($\rho \in (0,1)$, called a successful iteration), the next iteration is $x_{k+1} = x_k + d_k$. Otherwise, we use a search technique to obtain the steplength λ_k and let the next iteration be $x_{k+1} = x_k + \lambda_k d_k$. Motivated by the idea of the paper [4], we propose the following linesearch technique to obtain λ_k:

$$\left\|g(x_k + \lambda_k d_k)\right\|^2 - \left\|g_k\right\|^2 \leq -\sigma_1 \left\|\lambda_k g_k\right\|^2 - \sigma_2 \|\lambda_k d_k\|^2 + \sigma_3 \lambda_k d_k^T g_k, \tag{1.19}$$

where σ_1, σ_2, and σ_3 are some positive constants. In Section 3, we will show (1.19) is well-defined. Here and throughout this paper, $\|\cdot\|$ denotes the Euclidian norm of vectors or its induced matrix norm. $g(x_k)$ is replaced by g_k.

In the next section, the proposed algorithm for solving (1.1) is given. The global and superlinear convergence of the presented algorithm are stated in Section 3 and Section 4, respectively. The numerical results of the method are reported in Section 5.

2. Algorithms

Algorithm 2.1.

Initial: choose ρ, $r \in (0,1)$, $0 < \tau_1 < \tau_2 < 1 < \tau_3$, $\sigma_1,\ \sigma_2,\ \sigma_3 > 0$, $\Delta_{\min} > 0$, $x_0 \in \Re^n$. Let $k := 0$;

Step 1: Let $\Delta_k = \Delta_{\min}$;

Step 2: If $\|g_k\| = 0$, stop. Otherwise go to Step 3;

Step 3: Solve the subproblem (1.14) with $\Delta = \Delta_k$ to get d_k;

Step 4: If

$$r_k \triangleq \frac{\|g(x_k)\|^2 - \|g(x_k + d_k)\|^2}{q_k(0) - q_k(d_k)} < \rho, \tag{2.1}$$

Go to Step 5; Otherwise Let $x_{k+1} = x_k + d_k$, $\Delta_{k+1} \in [\|d_k\|, \tau_3\|d_k\|]$, and go to Step 6;

Step 5: Let k be the smallest nonnegative integer i such that (1.19) holds for $\lambda = r^i$. Let $\lambda_k = r^{i_k}$ and $x_{k+1} = x_k + \lambda_k d_k$, $\Delta_{k+1} \in [\tau_1\|d_k\|, \tau_2\|d_k\|]$;

Step 6**:** Update B_k to get B_{k+1} by (1.15). Let $k := k + 1$. Go to Step 2.

Here we also give a normal trust-region method for (1.1) and call it Algorithm 2.2.

Algorithm 2.2 (the normal Trust-Region Algorithm [44]).

Initial: Given a starting point $x_0 \in \Re^n$, $\Delta_0 > 0$ is the initial trust region radium, an upper bound of trust region radius Δ', $0 < \Delta_0 \le \Delta'$. Set $0 < \mu < 1,\ 0 < \eta_1 < \eta_2 < 1 < \eta_3$, $k := 0$.

Step 1: If $\|g_k\| = 0$, stop. Otherwise, go to Step 2.

Step 2: Solve the trust-region subproblem (1.10) to obtain d_k.

Step 3: Let

$$r_k = \frac{\vartheta(x_k) - \vartheta(x_k + d_k)}{q_k^*(0) - q_k^*(d_k)}, \tag{2.2}$$

if $r_k < \eta_1$, set $\Delta_{k+1} = \eta_1\Delta_k$; If $r_k > \eta_2$ and $\|d_k\| = \Delta_k$, let $\Delta_{k+1} = \min\{\eta_3\Delta_k, \Delta'\}$; Otherwise, let $\Delta_{k+1} = \Delta_k$.

Step 4: If $r_k > \mu$, let $x_{k+1} = x_k + d_k$ and go to Step 5; otherwise, let $x_{k+1} = x_k$, go to Step 2.

Step 5: Set $k := k + 1$. Go to Step 1.

Remark 2.3. By $y_k = g_{k+1} - g_k$, we have the following approximate relations:

$$y_k = g_{k+1} - g_k \approx \nabla g_{k+1} s_k. \tag{2.3}$$

Since B_{k+1} satisfies the secant equation $B_{k+1}s_k = y_k$ and ∇g_{k+1} is symmetric, we have approximately

$$B_{k+1}s_k \approx \nabla g_{k+1}s_k = \nabla g_{k+1}^T s_k. \tag{2.4}$$

This means that B_{k+1} approximates ∇g_{k+1} along direction s_k.

3. The Global Convergence

In this section, we will establish the global convergence of Algorithm 2.1. Let Ω be the level set defined by

$$\Omega = \{x \mid \|g(x)\| \leq \|g(x_0)\|\}, \tag{3.1}$$

which is bounded.

Assumption 1. (A) g is continuously differentiable on an open convex set Ω_1 containing Ω.

(B) The Jaconbian of g is symmetric and bounded on Ω_1 and there exists a positive constant M such that

$$\|\nabla g(x)\| \leq M \quad \forall x \in \Omega_1. \tag{3.2}$$

(C) ∇g is positive definite on Ω_1; that is, there is a constant $m > 0$ such that

$$m\|d\|^2 \leq d^T \nabla g(x) d \quad \forall x \in \Omega_1,\ d \in \Re^n. \tag{3.3}$$

(D) $\vartheta(x)$ is differentiable and its gradient satisfies

$$\|\nabla \vartheta(x) - \nabla \vartheta(y)\| \leq L\|x - y\|, \quad \forall x, y \in \Omega_1, \tag{3.4}$$

where L is the Lipschitz constant. By Assumptions 1(A) and 1(B), it is not difficult to get the following inequality:

$$\|y_k\| \leq M\|s_k\|. \tag{3.5}$$

According to Assumptions 1(A) and 1(C), we have

$$s_k^T y_k = s_k^T \nabla g(\xi) s_k \geq m\|s_k\|^2, \tag{3.6}$$

where $\xi = x_k + \vartheta_0(x_{k+1} - x_k)$, $\vartheta_0 \in (0,1)$, which means that the update matrix B_k is always positive definite. By (3.5) and (3.6), we have

$$\frac{s_k^T y_k}{\|s_k\|^2} \geq m, \quad \frac{\|y_k\|^2}{s_k^T y_k} \leq \frac{M^2}{m}. \tag{3.7}$$

Lemma 3.1 ([see Theorem 2.1 in [45]]). *Suppose that Assumption 1 holds. Let B_k be updated by BFGS formula* (1.15) *and let B_0 be symmetric and positive definite. For any $k \geq 0$, s_k and y_k satisfy* (3.7). *Then there exist positive constants β_1, β_2, and β_3 such that, for any positive integer $\tilde{k}$*

$$\beta_1\|d_k\|^2 \leq d_k^T B_k d_k \leq \beta_2\|d_k\|^2, \qquad \beta_1\|d_k\| \leq \|B_k d_k\| \leq \beta_3\|d_k\| \tag{3.8}$$

hold for at least $\lceil \tilde{k}/2 \rceil$ value of $k \in \{1, 2, \ldots, \tilde{k}\}$.

Considering the subproblem (1.14), we give the following assumption similar to (1.14). Similar to [2], the following assumption is needed.

Assumption 2. B_k is a good approximation to ∇g_k, that is,

$$\|(\nabla g_k - B_k)d_k\| \leq \varepsilon_0\|g_k\|, \tag{3.9}$$

and d_k satisfies

$$\|g_k + B_k d_k\| \leq \varepsilon_1\|g_k\|, \tag{3.10}$$

where $\varepsilon_0 \in (0,1)$ is a small quantity, and $\varepsilon_1 > 0$, $\varepsilon_0 + \varepsilon_1 \in (0,1)$.

Lemma 3.2. *Let Assumption 2 hold. Then d_k is descent direction for $\vartheta(x)$ at x_k, that is,*

$$\nabla\vartheta(x_k)^T d_k < 0. \tag{3.11}$$

Proof. Let r_k be the residual associated with d_k so that $g_k + B_k d_k = r_k$:

$$\begin{aligned}
\nabla\vartheta(x_k)^T d_k &= g(x_k)^T \nabla g(x_k) d_k \\
&= g(x_k)^T \left[(\nabla g(x_k) - B_k)d_k + (r_k - g(x_k))\right] \\
&= g(x_k)^T (\nabla g(x_k) - B_k)d_k + g(x_k)^T r_k - g(x_k)^T g(x_k).
\end{aligned} \tag{3.12}$$

So we have

$$\nabla\vartheta(x_k)^T d_k + \|g(x_k)\|^2 = g(x_k)^T (\nabla g(x_k) - B_k)d_k + g(x_k)^T r_k. \tag{3.13}$$

Therefore, taking the norm in the right-hand side of the above equality, we have that from Assumption 2

$$\begin{aligned}\nabla\vartheta(x_k)^T d_k &\le \|g(x_k)\|\|(\nabla g(x_k) - B_k)d_k\| + \|g(x_k)\|\|r_k\| - \|g(x_k)\|^2 \\ &\le -(1-\varepsilon_0-\varepsilon_1)\|g(x_k)\|^2.\end{aligned} \tag{3.14}$$

Hence, for $\varepsilon_0 + \varepsilon_1 \in (0,1)$, the lemma is satisfied. □

According to the above lemma, it is easy to deduce that the norm function $\vartheta(x)$ is descent, which means that $\|g_{k+1}\| \le \|g_k\|$ is true.

Lemma 3.3. *Let $\{x_k\}$ be generated by Algorithm 2.1 and suppose that Assumption 2 holds. Then $\{x_k\} \subset \Omega$. Moreover, $\{\|g_k\|\}$ converges.*

Proof. By Lemma 3.2, we have $\|g_{k+1}\| \le \|g_k\|$. Then we conclude from Lemma 3.3 in [46] that $\{\|g_k\|\}$ converges. Moreover, we have for all k

$$\|g_{k+1}\| \le \|g_k\| \le \|g_{k-1}\| \le \cdots \le \|g(x_0)\|. \tag{3.15}$$

This implies that $\{x_k\} \subset \Omega$. □

Lemma 3.4. *Let Assumption 1 hold. Then the following inequalities*

$$g_k^T d_k \le -\beta_1\|d_k\|^2, \qquad \|g_k\|^2 \ge \beta_1^2\|d_k\|^2 \tag{3.16}$$

$$-\frac{1}{\beta_1}\|g_k\|^2 \le g_k^T d_k \tag{3.17}$$

hold.

Proof. Since the update matrix B_k is positive definite. Then, problem (1.14) has a unique solution d_k, which together with some multiplier $\alpha_k \ge 0$ satisfies the following equations:

$$\begin{aligned}B_k d_k + \alpha_k d_k &= -g_k, \\ \alpha_k(\|d_k\| - \Delta_k) &= 0.\end{aligned} \tag{3.18}$$

From (3.18), we can obtain

$$d_k^T B_k d_k + g_k^T d_k = -\alpha_k\|d_k\|^2 \le 0, \tag{3.19}$$

$$\alpha_k = \frac{-g_k^T d_k - d_k^T B_k d_k}{\|d_k\|^2}. \tag{3.20}$$

By (3.19) and (3.8), we get (3.16), which also imply that the inequality (3.17) holds. □

The next lemma will show that (1.19) is reasonable, and then Algorithm 2.1 is well defined.

Lemma 3.5. *Let Assumptions 1(D) and 2 hold. Then there exists a step-size λ_k such that* (1.19) *in a finite number of backtracking steps.*

Proof. From Lemma 3.8 in [1] we have that in a finite number of backtracking steps, λ_k must satisfy

$$\left\|g(x_k+\alpha_k d_k)\right\|^2-\left\|g(x_k)\right\|^2\le \delta\lambda_k g(x_k)^T\nabla g(x_k)d_k,\quad \delta\in(0,1). \tag{3.21}$$

By (3.12) and (3.14), let $\beta_0=(1-\varepsilon_0-\varepsilon_1)$, and we have

$$\begin{aligned} g(x_k)^T\nabla g(x_k)d_k &\le -\beta_0\left\|g_k\right\|^2=-\frac{\beta_0}{3}\left\|g_k\right\|^2-\frac{\beta_0}{3}\left\|g_k\right\|^2-\frac{\beta_0}{3}\left\|g_k\right\|^2\\ &\le -\frac{\beta_0}{3}\left\|g_k\right\|^2-\frac{\beta_0}{3}\beta_1^2\|d_k\|^2+\frac{\beta_0}{3}\beta_1 g_k^T d_k, \end{aligned} \tag{3.22}$$

where the last inequality follows (3.16) and (3.17). By $\lambda_k\le 1$, let $\sigma_1\in(0,(\beta_0/3)\delta)$, $\sigma_2\in(0,(\beta_0/3)\beta_1^2\delta)$, $\sigma_3\in(0,(\beta_0/3)\beta_1\delta)$, then we obtain (1.19). The proof is complete. □

Lemma 3.6. *Let $\{x_k\}$ be generated by the Algorithm 2.1. Suppose that Assumptions 1 and 2 hold. Then one has*

$$\sum_{k=0}^{\infty}\left(-g_k^T d_k\right)<\infty,\quad \sum_{k=0}^{\infty}d_k^T B_k d_k<\infty. \tag{3.23}$$

In particular, one has

$$\lim_{k\to\infty}\left(-g_k^T d_k\right)=0,\quad \lim_{k\to\infty}d_k^T B_k d_k=0. \tag{3.24}$$

Proof. By (3.8) and (3.19), we have

$$q_k(d_k)=g_k^T d_k+\frac{1}{2}d_k^T B_k d_k\le\frac{1}{2}g_k^T d_k\le-\frac{1}{2}d_k^T B_k d_k. \tag{3.25}$$

From Step 4 of Algorithm 2.1, if $r_k\ge\rho$ is true, we get

$$\left\|g(x_{k+1})\right\|^2-\left\|g(x_k)\right\|^2\le q_k(d_k)\le\frac{1}{2}g_k^T d_k\le-\frac{1}{2}d_k^T B_k d_k, \tag{3.26}$$

otherwise, if $r_k<\rho$ is true, by Step 5 of Algorithm 2.1, (3.8), and (3.26), we can obtain

$$\begin{aligned} \left\|g(x_{k+1})\right\|^2-\left\|g(x_k)\right\|^2 &\le -\sigma_1\left\|\lambda_k g_k\right\|^2-\sigma_2\|\lambda_k d_k\|^2+\sigma_3\lambda_k d_k^T g_k\\ &\le \sigma_3\lambda_k d_k^T g_k\le-\sigma_3\lambda_k d_k^T B_k d_k. \end{aligned} \tag{3.27}$$

By Lemma 3.5, we know that (1.19) can be satisfied in a finite number of backtracking steps, which means that there exists a constant $\lambda^* \in (0,1)$ satisfying $\lambda^* \le \lambda_k$ for all k. By (3.26) and (3.27), we have

$$\left\|g(x_{k+1})\right\|^2 - \left\|g(x_k)\right\|^2 \le \rho_1 g_k^T d_k \le -\rho_1 d_k^T B_k d_k \le -\rho_1\beta_1\|d_k\|^2 < 0, \tag{3.28}$$

where $\rho_1 = \min\{1/2, \sigma_3\lambda^*\}$. According to (3.28), we get

$$\begin{aligned}\sum_{k=0}^{\infty} d_k^T B_k d_k &\le \sum_{k=0}^{\infty}\left(-g_k^T d_k\right) \le \frac{1}{\rho_1}\sum_{k=0}^{\infty}\left(\left\|g(x_k)\right\|^2 - \left\|g(x_{k+1})\right\|^2\right)\\ &= \frac{1}{\rho_1}\lim_{N\to\infty}\sum_{k=0}^{N}\left(\left\|g(x_k)\right\|^2 - \left\|g(x_{k+1})\right\|^2\right)\\ &= \frac{1}{\rho_1}\lim_{N\to\infty}\left(\left\|g(x_0)\right\|^2 - \left\|g(x_{N+1})\right\|^2\right),\end{aligned} \tag{3.29}$$

and by Lemma 3.3, we know that $\{\|g_k\|\}$ is convergent. Therefore, we deduce that (3.23) holds. According to (3.23), it is easy to deduce (3.24). The proof is complete. □

Lemma 3.7. *Suppose that Assumptions 1 and 2 hold. There are positive constants $b_1 \le b_2$, and b_3 such that for any k, if $\|d_k\| \ne \Delta_{\min}$, then the following inequalities hold:*

$$b_1\left\|g_k\right\| \le \|d_k\| \le b_2\left\|g_k\right\|, \quad \alpha_k \le b_3. \tag{3.30}$$

Proof. We will prove this lemma in the following two cases.

Case 1 ($\|d_k\| < \Delta_k$). By (3.18), we have $\alpha_k = 0$ and $B_k d_k = -g_k$. Together with (3.8) and (3.19), we get

$$\begin{aligned}\beta_1\|d_k\|^2 \le d_k^T B_k d_k = -d_k^T g_k \le \|d_k\|\left\|g_k\right\|,\\ \left\|-g_k\right\| = \left\|g_k\right\| = \|B_k d_k\| \le \beta_3\|d_k\|.\end{aligned} \tag{3.31}$$

Then (3.30) holds with $b_1 = 1/\beta_3 \le b_2 = 1/\beta_1$ and $b_3 = 0$.

Case 2 ($\|d_k\| = \Delta_k$). From (3.19) and (3.8), we have

$$\beta_1\|d_k\|^2 \le d_k^T B_k d_k \le -g_k^T d_k \le \left\|g_k\right\|\|d_k\|. \tag{3.32}$$

Then, we get $\|d_k\| \le 1/\beta_1 \ \|g_k\|$. By (3.10) and (3.8), it is easy to deduce that

$$(1-\varepsilon_1)\left\|g_k\right\| \le \|B_k d_k\| \le \beta_3\|d_k\|. \tag{3.33}$$

So we obtain $\|d_k\| \geq (1-\varepsilon_1)/\beta_3 \ \|g_k\|$. Using (3.20), we have

$$\alpha_k = \frac{-g_k^T d_k - d_k^T B_k d_k}{\|d_k\|^2} \leq \frac{\|g_k\|}{\|d_k\|} \leq \frac{\beta_3}{1-\varepsilon_1}. \tag{3.34}$$

Therefore, (3.30) holds. The proof is complete. □

In the next theorem, we establish the global convergence of Algorithm 2.1.

Theorem 3.8. *Let $\{x_k\}$ be generated by Algorithm 2.1 and the conditions in Assumptions 1 and 2 hold. Then one has*

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{3.35}$$

Proof. By Lemma 3.6, we have

$$\lim_{k\to\infty} -g_k^T d_k = \lim_{k\to\infty} d_k^T B_k d_k = 0. \tag{3.36}$$

Combining (3.8) and (3.36), we get

$$\lim_{k\to\infty} \|d_k\| = 0. \tag{3.37}$$

Together with (3.30), we obtain (3.35). The proof is complete. □

4. The Superlinear Convergence Analysis

In this section, we will present the superlinear convergence of Algorithm 2.1.

Assumption 3. ∇g is Hölder continuous at x^*; that is, for every x in a neighborhood of x^*, there are positive constants M_1 and γ such that

$$\|\nabla g(x) - \nabla g(x^*)\| \leq M_1 \|x - x^*\|^\gamma, \tag{4.1}$$

where x^* stands for the unique solution of (1.1) in Ω_1.

Lemma 4.1. *Let $\{x_k\}$ be generated by Algorithm 2.1 and the conditions in Assumptions 1 and 2 hold. Then, for any fixed $\gamma > 0$, one has*

$$\sum_{k=0}^{\infty} \|x_k - x^*\|^\gamma < \infty. \tag{4.2}$$

Moreover, one has

$$\sum_{k=0}^{\infty} \chi_k(\gamma) < \infty, \tag{4.3}$$

where $\chi_k(\gamma) = \max\{\|x_k - x^*\|^\gamma, \|x_{k+1} - x^*\|^\gamma\}$.

Proof. Using Assumption 1, we can have the following inequality:

$$m\|x - x^*\| \le \|g(x)\| = \|g(x) - g(x^*)\| \le M\|x - x^*\|, \quad x \in \Omega_1. \tag{4.4}$$

By (3.8) and (3.30), we have

$$\begin{aligned} -\beta_2\|d_k\|^2 \le -d_k^T B_k d_k \le -\beta_1\|d_k\|^2, \\ -b_2^2\|g_k\|^2 \le -\|d_k\|^2 \le -b_1^2\|g_k\|^2. \end{aligned} \tag{4.5}$$

Together with (3.28), we get

$$\begin{aligned} \|g_{k+1}\|^2 - \|g_k\|^2 &\le \rho_1 g_k^T d_k \le -\rho_1 d_k^T B_k d_k \\ &\le -\rho_1\beta_1\|d_k\|^2 \\ &\le -\rho_1\beta_1 b_1^2\|g_k\|^2, \end{aligned} \tag{4.6}$$

and let $\rho_0 = \min\{\rho_1\beta_1 b_1^2,\ \rho\} \in (0,1)$. Suppose that there exists a positive integer k_0, as $k \ge k_0$, (3.8) holds. Then we obtain

$$\|g_{k+1}\|^2 \le \|g_k\|^2 - \rho_0\|g_k\|^2 = (1-\rho_0)\|g_k\|^2 \le \cdots \le (1-\rho_0)^{k-k_0+1}\|g_{k_0}\|^2 = c_0 c_1^k, \tag{4.7}$$

where $c_0 = (1-\rho_0)^{1-k_0}\|g_0\|^2$, $c_1 = (1-\rho_0) \in (0,1)$. This together with (4.4) shows that $\|x_{k+1} - x^*\|^2 \le m^{-2}c_0 c_1^k$ holds for all k large enough. Therefore, for any γ, we have (4.2). Notice that $\chi_k(\gamma) \le \|x_k - x^*\|^\gamma + \|x_{k+1} - x^*\|^\gamma$; from (4.2), we can get (4.3). □

Lemma 4.2. *Let Assumptions 1, 2, and 3 hold. Then, for all k sufficiently large, there exists a positive constant* M_2 *such that*

$$\|y_k - \nabla g(x^*)s_k\| \le M_2\chi_k\|s_k\|, \tag{4.8}$$

where $\chi_k = \max\{\|x_k - x^*\|^\gamma, \|x_{k+1} - x^*\|^\gamma\}$.

Proof. From Theorem 3.8 and (4.4), it is not difficult to get $x_k \to x^*$. Then (4.1) holds for all k large enough. Using the mean value theorem, for all k sufficiently large, we have

$$\begin{aligned}\left\|y_k - \nabla g(x^*)s_k\right\| &= \left\|\nabla g(x_k + t_0(x_{k+1} - x_k))s_k - \nabla g(x^*)s_k\right\| \\ &\le \left\|\nabla g(x_k + t_0(x_{k+1} - x_k)) - \nabla g(x^*)\right\|\|s_k\| \\ &\le M_1\|x_k + t_0(x_{k+1} - x_k) - x^*\|^{\gamma}\|s_k\| \\ &\le M_2\chi_k\|s_k\|,\end{aligned} \tag{4.9}$$

where $M_2 = M_1(2t_0 + 1)$, $t_0 \in (0, 1)$. Therefore, the inequality of (4.8) holds. □

Lemma 4.3. *Let Assumptions 1, 2, and 3 hold and let x_k be generated by Algorithm 2.1. Denote $Q = \nabla g(x^*)^{-1/2}$, $H_k = B_k^{-1}$. Then, for all large k, there are positive constants e_i, $i = 1, 2, 3, 4$, and $\eta \in (0, 1)$ such that*

$$\left\|B_{k+1} - \nabla g(x^*)\right\|_{Q,F} \le (1 + e_1\chi_k)\left\|B_k - \nabla g(x^*)\right\|_{Q,F} + e_2\chi_k, \tag{4.10}$$

$$\left\|H_{k+1} - \nabla g(x^*)^{-1}\right\|_{Q^{-1},F} \le \left(\sqrt{1 - \eta\varpi_k^2} + e_3\chi_k\right)\left\|H_k - \nabla g(x^*)^{-1}\right\|_{Q^{-1},F} + e_4\chi_k, \tag{4.11}$$

where $\|A\|_{Q,F} = \|Q^T AQ\|_F$, $\|\cdot\|_F$ is the Frobenius norm of a matrix and ϖ_k is defined as follows:

$$\varpi_k = \frac{\left\|Q^{-1}\left(H_k - \nabla g(x^*)^{-1}\right)y_k\right\|}{\left\|H_k - \nabla g(x^*)^{-1}\right\|_{Q^{-1},F}\|Qy_k\|}. \tag{4.12}$$

In particular, $\{\|B_k\|\}_F$ and $\{\|H_k\|\}_F$ are bounded.

Proof. From (1.15), we have

$$\begin{aligned}\left\|B_{k+1} - \nabla g(x^*)\right\|_{Q,F} &= \left\|B_k - \nabla g(x^*) + \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}\right\|_{Q,F} \\ &\le (1 + e_1\tau_k)\left\|B_k - \nabla g(x^*)\right\|_{Q,F} + e_2\chi_k,\end{aligned} \tag{4.13}$$

where the last inequality follows the inequality (49) of [47]. Hence, (4.10) holds. By (4.8), in a way similar to that of [46], we can prove that (4.11) holds and $\|B_k\|$ and $\|H_k\|$ are bounded. The proof is complete. □

Lemma 4.4. *Let $\{x_k\}$ be generated by Algorithm 2.1 and the conditions in Assumptions 1, 2 and 3 hold. Then*

$$\lim_{k\to\infty} \frac{\left\|(B_k - \nabla g(x^*))s_k\right\|}{\|s_k\|} = 0, \tag{4.14}$$

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

Proof. In a similar way to [46], it is not difficult to obtain

$$\lim_{k\to\infty}\frac{\left\|Q^{-1}\left(H_k-\nabla g(x^*)^{-1}\right)y_k\right\|}{\|Qy_k\|}=0. \tag{4.15}$$

On the other hand, we have

$$\begin{aligned}\left\|Q^{-1}\left(H_k-\nabla g(x^*)^{-1}\right)y_k\right\| &= \left\|Q^{-1}H_k(\nabla g(x^*)-B_k)\nabla g(x^*)^{-1}y_k\right\|\\ &\geq \left\|Q^{-1}H_k(\nabla g(x^*)-B_k)s_k\right\|\\ &\quad-\left\|Q^{-1}H_k(\nabla g(x^*)-B_k)\left(s_k-\nabla g(x^*)^{-1}y_k\right)\right\|\\ &\geq \left\|Q^{-1}H_k(\nabla g(x^*)-B_k)s_k\right\|\\ &\quad-\left\|Q^{-1}\right\|\|H_k\|\left(\|\nabla g(x^*)\|+\|B_k\|\right)\left\|\nabla g(x^*)^{-1}\left(y_k-\nabla g(x^*)s_k\right)\right\|\\ &\geq \left\|Q^{-1}H_k(\nabla g(x^*)-B_k)s_k\right\|\\ &\quad-M_2\chi_k\left\|Q^{-1}\right\|\|H_k\|\left(\|\nabla g(x^*)\|+\|B_k\|\right)\left\|\nabla g(x^*)^{-1}\right\|\|s_k\|\\ &= \left\|Q^{-1}H_k(\nabla g(x^*)-B_k)s_k\right\|-o(\|s_k\|),\end{aligned} \tag{4.16}$$

where the last inequality follows from (4.8). We know that $\{\|B_k\|\}$ and $\{\|H_k\|\}$ are bounded, and $\{H_k\}$ is positive definite. By (3.5), we get

$$\|Qy_k\|\leq M\|Q\|\|s_k\|. \tag{4.17}$$

Combining (4.15) and (4.17), we conclude that (4.14) holds. The proof is complete. □

Theorem 4.5. *Let the conditions in Assumptions 1, 2 and 3 hold. If $\varepsilon_1\to 0$ in (3.10). Then the sequence $\{x_k\}$ generated by Algorithm 2.1 converges to x^* superlinearly for $\lambda_k=1$.*

Proof. For all $x_k\in\Omega_1$, we get

$$\begin{aligned}\frac{\|g_{k+1}\|}{\|d_k\|} &= \frac{\left\|g_k+B_kd_k+(\nabla g_k-B_k)d_k+O\left(\|d_k\|^2\right)\right\|}{\|d_k\|}\\ &\leq \frac{\|g_k+B_kd_k\|}{\|g_k\|}\frac{\|g_k\|}{\|d_k\|}+\frac{\|(\nabla g_k-B_k)d_k\|}{\|d_k\|}+\frac{O\left(\|d_k\|^2\right)}{\|d_k\|}\\ &\leq \varepsilon_1\frac{\|g_k\|}{\|d_k\|}+\frac{\|(\nabla g_k-B_k)d_k\|}{\|d_k\|}+O(\|d_k\|),\end{aligned} \tag{4.18}$$

where the last inequality follows (3.10). By (3.5), we have

$$\|g_k\| \leq \|g_{k+1} - g_k\| + \|g_{k+1}\| \leq M\|d_k\| + \|g_{k+1}\|. \tag{4.19}$$

Dividing both sides by $\|d_k\|$, we get

$$\frac{\|g_k\|}{\|d_k\|} \leq M + \frac{\|g_{k+1}\|}{\|d_k\|}. \tag{4.20}$$

Substituting this into (4.18), we can obtain

$$\frac{\|g_{k+1}\|}{\|d_k\|} \leq \varepsilon_1 \left(M + \frac{\|g_{k+1}\|}{\|d_k\|} \right) + \frac{\|(\nabla g_k - B_k)d_k\|}{\|d_k\|} + O(\|d_k\|), \tag{4.21}$$

which means that

$$\frac{\|g_{k+1}\|}{\|d_k\|} \leq \frac{(M\varepsilon_1 + \|(\nabla g_k - B_k)d_k\|/\|d_k\| + O(\|d_k\|))}{(1 - \varepsilon_1)}. \tag{4.22}$$

Since $\varepsilon_1 \to 0$, and $\|d_k\| \to 0$ as $k \to \infty$, by (4.14) and (3.10), we have

$$\lim_{k\to\infty} \frac{\|g_{k+1}\|}{\|d_k\|} = 0. \tag{4.23}$$

Using (3.16), we get

$$\lim_{k\to\infty} \frac{\|g_{k+1}\|}{\|g_k\|} = 0. \tag{4.24}$$

Considering (4.4), we have

$$\lim_{k\to\infty} \frac{\|x_k + d_k - x^*\|}{\|x_k - x^*\|} = 0. \tag{4.25}$$

Therefore, we get the result of the superlinear convergence. □

5. Numerical Results

In this section, we test the proposed BFGS trust-region method on symmetric nonlinear equations and compare it with Algorithm 2.2. The following problems with various sizes will be solved.

Problem 1. The discretized two-point boundary value problem like the problem in [48] is

$$g(x) \triangleq Ax + \frac{1}{(n+1)^2} F(x) = 0, \tag{5.1}$$

where A is the $n \times n$ tridiagonal matrix given by

$$A = \begin{bmatrix} 8 & -1 & & & \\ -1 & 8 & -1 & & \\ & -1 & 8 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & -1 \\ & & & & -1 & 8 \end{bmatrix}, \tag{5.2}$$

and $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))^T$ with $F_i(x) = \sin x_i - 1,\ i = 1, 2, \ldots, n$.

Problem 2. Unconstrained optimization problem is

$$\min\ f(x), \quad x \in R^n, \tag{5.3}$$

with Engval function [49] $f : R^n \to R$ defined by

$$f(x) = \sum_{i=2}^{n} \left[\left(x_{i-1}^2 + x_i^2 \right)^2 - 4x_{i-1} + 3 \right]. \tag{5.4}$$

The related symmetric nonlinear equation is

$$g(x) \triangleq \frac{1}{4} \nabla f(x) = 0, \tag{5.5}$$

where $g(x) = (g_1(x), g_2(x), \ldots, g_n(x))^T$ with

$$\begin{aligned} g_1(x) &= x_1 \left(x_1^2 + x_2^2 \right) - 1, \\ g_i(x) &= x_i \left(x_{i-1}^2 + 2x_i^2 + x_{i+1}^2 \right) - 1, \quad i = 2, 3, \ldots, n-1, \\ g_n(x) &= x_n \left(x_{n-1}^2 + x_n^2 \right). \end{aligned} \tag{5.6}$$

In the experiments, the parameters in Algorithm 2.1 were chosen as $\tau_1 = 0.5$, $\tau_2 = 0.9$, $\tau_3 = 3$, $r = 0.1, \Delta_{\min} = \|g_0\|$, $B_0 = I, \rho = 0.25$, $\sigma_1 = \sigma_2 = 10^{-5}$, and $\sigma_3 = 0.9$. We obtain d_k from subproblem (1.14) by the well-known *Dogleg* method. The parameters in Algorithm 2.2 were

Table 1: Test Results For Problem 1.

(a) (Small-scales). Test results for Algorithm 2.1.

x0	(1,. . .,1)	(60,. . .,60)	(600,. . .,600)	(−1, . . . , −1)	(−60, . . . , −60)	(−600, . . . , −600)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 10$	13/24/ 2.406624e-07	14/25/ 2.272840e-07	17/30/ 3.104130e-07	13/24/ 2.449361e-07	14/25/ 2.398188e-07	17/30/ 4.593832e-07
$n = 50$	48/101/ 2.189696e-07	49/102/ 4.009098e-07	50/103/ 2.147571e-07	48/101/ 2.181267e-07	49/102/ 4.008911e-07	50/103/ 2.120250e-07
$n = 99$	82/171/ 6.794811e-07	89/188/ 6.345939e-07	91/190/ 7.804790e-07	82/171/ 8.358725e-07	89/188/ 6.367964e-07	91/190/ 7.801889e-07
x0	(1,0,1,0,. . .)	(60,0,60,0,. . .)	(600,0,600,0,. . .)	(−1, 0, −1, 0, . . .)	(−60, 0, −60, 0, . . .)	(−600, 0, −600, 0, . . .)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 10$	21/42/ 7.364467e-07	22/43/ 3.922363e-07	22/45/ 4.894966e-07	21/44/ 3.463471e-08	22/43/ 3.860638e-07	22/45/ 4.895404e-07
$n = 50$	72/153/ 9.350290e-07	86/181/ 4.420131e-07	88/185/ 7.620218e-07	70/151/ 6.776281e-07	86/181/ 4.420083e-07	49/83/ 8.003368e-07
$n = 99$	73/156/ 9.013346e-07	88/185/ 7.631881e-07	88/191/ 6.856481e-07	74/161/ 9.918464e-07	88/185/ 7.368909e-07	88/191/ 6.856897e-07

(b) (Large-scales). Test results for Algorithm 2.1.

x0	(1,. . .,1)	(60,. . .,60)	(600,. . .,600)	(−1, . . . , −1)	(−60, . . . , −60)	(−600, . . . , −600)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 200$	83/178/ 9.096568e-7	106/225/ 9.483206e-7	117/250/ 8.796828e-7	85/180/ 7.376219e-7	106/225/ 9.263058e-7	117/250/ 8.779599e-7
$n = 500$	85/180/ 8.830573e-7	103/218/ 9.825658e-7	115/244/ 9.765194e-7	83/178/ 7.659650e-7	103/218/ 9.796118e-7	115/244/ 9.755827e-7
$n = 1000$	76/165/ 8.611337e-7	96/207/ 8.301215e-7	105/224/ 9.957816e-7	76/165/ 8.587066e-7	96/207/ 8.291876e-7	105/224/ 9.925005e-7
x0	(1,0,1,0,. . .)	(60,0,60,0,. . .)	(600,0,600,0,. . .)	(−1, 0, −1, 0, . . .)	(−60, 0, −60, 0, . . .)	(−600, 0, −600, 0, . . .)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 200$	68/149/ 8.780047e-7	91/194/ 7.484521e-7	101/216/ 9.790557e-7	69/150/ 9.770900e-7	91/194/ 7.275693e-7	101/216/ 9.559911e-7
$n = 500$	72/155/ 9.797645e-07	96/205/ 9.993161e-7	106/225/ 8.916405e-7	72/155/ 9.886969e-7	97/206/ 7.492841e-7	106/225/ 8.921008e-7
$n = 1000$	69/152/ 9.919863e-7	93/200/ 6.930976e-7	106/227/ 8.119328e-7	69/152/ 9.948500e-7	93/200/ 6.946308e-7	106/227/ 8.123102e-7

(c) (Small-scales). Test results for Algorithm 2.2.

x0	(1,. . .,1)	(60,. . .,60)	(600,. . .,600)	(−1, . . . , −1)	(−60, . . . , −60)	(−600, . . . , −600)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 10$	54/107/ 8.039519e-7	67/133/ 7.624248e-7	74/147/ 8.167466e-7	54/107/ 8.061366e-7	67/133/ 7.624560e-7	74/147/ 8.167469e-7
$n = 50$	58/115/ 9.602663e-7	72/143/ 7.684310e-007	79/157/ 8.876868e-7	58/115/ 9.603892e-7	72/143/ 7.684327e-007	79/157/ 8.876870e-7
$n = 99$	60/119/ 7.614838e-7	73/145/ 8.350445e-7	80/159/ 9.679851e-7	60/119/ 7.615091e-7	73/145/ 8.350450e-7	80/159/ 9.679851e-7
x0	(1,0,1,0,. . .)	(60,0,60,0,. . .)	(600,0,600,0,. . .)	(−1, 0, −1, 0, . . .)	(−60, 0, −60, 0, . . .)	(−600, 0, −600, 0, . . .)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 10$	52/103/ 7.605486e-7	64/127/ 9.929883e-7	72/143/ 7.732628e-7	52/103/ 7.646868e-7	64/127/ 9.930747e-7	72/143/ 7.732660e-7
$n = 50$	56/111/ 8.896898e-7	69/137/ 9.690007e-7	77/153/ 8.223484e-7	56/111/ 8.899175e-7	69/137/ 9.690048e-7	77/153/ 8.223488e-7
$n = 99$	57/113/ 9.598124e-7	71/141/ 7.734909e-7	78/155/ 8.965851e-7	57/113/ 9.598763e-7	71/141/ 7.734918e-7	78/155/ 8.965852e-7

(d) (Large-scales). Test results for Algorithm 2.2.

x0	(1,. . .,1)	(60,. . .,60)	(600,. . .,600)	(−1, . . . , −1)	(−60, . . . , −60)	(−600, . . . , −600)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 200$	61/121/ 8.110467e-7	74/147/ 8.917908e-7	82/163/ 7.610549e-7	61/121/ 8.110534e-7	74/147/ 8.917909e-7	82/163/ 7.610549e-7
$n = 500$	62/123/ 9.526492e-7	76/151/ 7.712044e-7	83/165/ 8.958279e-7	62/123/ 9.526504e-7	76/151/ 7.712044e-7	83/165/ 8.958279e-7
$n = 1000$	63/125/ 9.938699e-7	77/153/ 8.049274e-7	84/167/ 9.351920e-7	63/125/ 9.938703e-7	77/153/ 8.049274e-7	84/167/ 9.351920e-7
x0	(1,0,1,0,. . .)	(60,0,60,0,. . .)	(600,0,600,0,. . .)	(−1, 0, −1, 0, . . .)	(−60, 0, −60, 0, . . .)	(−600, 0, −600, 0, . . .)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 200$	59/117/ 7.503172e-7	72/143/ 8.249912e-7	79/157/ 9.576414e-7	59/117/ 7.503296e-7	72/143/ 8.249914e-7	79/157/ 9.576414e-7
$n = 500$	60/119/ 8.811245e-7	73/145/ 9.701366e-7	81/161/ 8.285552e-7	60/119/ 8.811269e-7	73/145/ 9.701367e-7	81/161/ 8.285552e-7
$n = 1000$	61/121/ 9.191890e-7	75/149/ 7.444393e-7	82/163/ 8.649128e-7	61/121/ 9.191896e-7	75/149/ 7.444393e-7	82/163/ 8.649128e-7

chosen as $\Delta' = \Delta_0 = \|g_0\|, \eta_1 = 0.25$, $\eta_2 = 0.75$, $\mu = 0.01$, and $\eta_3 = 2$. Since the matrices $\nabla g(x_k)^T \nabla g(x_k)$ will be singular, we solve (1.10) by *Extreme Minimization with 2—Dimension Subspace Method* to obtain d_k. The program was coded in $MATLAB$ 6.5.1. We stopped the iteration when the condition $\|g(x)\| \leq 10^{-6}$ was satisfied. If the iteration number is larger than one thousand, we also stop this program and this method is considered to be failed. For Algorithm 2.1, Tables 1(a) and 1(b) and Tables 2(a) and 2(b) show the performance of the method need to solve Problem 1 and Problem 2, respectively. For Algorithm 2.2, Tables 1(c) and 1(d) and Tables 2(c) and 2(d) show the performance of the normal trust region method need to solve Problem 1 and Problem 2, respectively. The columns of the tables have the following meaning:

Dim: the dimension of the problem,

NI: the total number of iterations,

NG: the number of the function evaluations,

EG: the norm of the function evaluations.

From Tables 1(a)–2(d), it is not difficult to see that the proposed method performs better than the normal method does. Furthermore, the performance of Algorithm 2.1 hardly changes with the dimension increasing. Overall, the given method is competitive to the normal trust region method.

6. Discussion

We give a trust-region-based BFGS method and establish its convergent results in this paper. The numerical results show that this method is promising. In fact, this problem (1.1) can come from unconstrained optimization problem and an equality constrained optimization problem (for details see [4]). There are some other practical problems, such as the saddle point problem, the discretized two-point boundary value problem, and the discretized elliptic boundary value problem, take the form of (1.1) with symmetric Jacobian (see, e.g., Chapter 1 in [50]). This presented method can also extend to solve the normal nonlinear equations.

Table 2: Test Results For Problem 2.

(a) (Small-scales). Test results for Algorithm 2.1.

x0	(0.5,. . .,0.5)	(1,. . .,1)	(3,. . .,3)	(−0.75, . . . , −0.75)	(−2, . . . , −2)	(−3, . . . , −3)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 10$	25/44/ 9.720971e-07	21/32/ 4.889567e-07	92/103/ 2.475812e-08	21/32/ 8.691255e-07	46/59/ 5.860956e-07	86/105/ 6.348374e-08
$n = 50$	37/56/ 9.950345e-07	39/56/ 8.776379e-07	113/139/ 9.587026e-07	40/63/ 6.984106e-07	69/96/ 9.523480e-07	103/125/ 9.404211e-07
$n = 99$	42/59/ 9.725361e-07	41/60/ 7.374460e-07	113/135/ 7.909796e-07	40/55/ 8.380367e-07	117/489/ 9.805302e-07	97/129/ 7.975248e-07
x0	(0.5,0,0.5,0,. . .)	(1,0,1,0,. . .)	(3,0,3,0,. . .)	(−0.75, 0, −0.75, 0, . . .)	(−2, 0, −2, 0, . . .)	(−3, 0, −3, 0, . . .)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 10$	24/35/ 4.711749e-07	21/30/ 3.147507e-07	44/65/ 3.529113e-07	27/48/ 4.004367e-07	39/76/ 8.503415e-07	29/42/ 7.623619e-07
$n = 50$	36/57/ 8.776354e-07	36/57/ 8.287552e-07	54/77/ 8.491652e-07	41/64/ 9.492805e-07	42/69/ 9.029472e-07	58/77/ 9.752703e-07
$n = 99$	36/61/ 8.265146e-07	37/56/ 9.507706e-07	60/93/ 5.373087e-07	42/73/ 8.247653e-07	50/79/ 9.217390e-07	62/88/ 8.307004e-07

(b) (Large-scales). Test results for Algorithm 2.1

x0	(0.5,. . .,0.5)	(1,. . .,1)	(3,. . .,3)	(−0.75, . . . , −0.75)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 200$	40/57/ 7.464372e-007	41/58/ 4.921097e-007	112/130/ 6.229759e-007	40/65/ 7.785598e-007
$n = 500$	36/57/ 7.887407e-007	41/60/ 3.538433e-007	113/135/ 8.871522e-007	38/69/ 9.785814e-007
$n = 1000$	42/65/ 7.382939e-007	40/59/ 7.463210e-007	120/146/ 6.044161e-007	40/69/ 4.563405e-007
x0	(0.5,0,0.5,0,. . .)	(1,0,1,0,. . .)	(3,0,3,0,. . .)	(−0.75, 0, −0.75, 0, . . .)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 200$	41/64/ 6.671246e-007	36/61/ 9.977774e-007	63/94/ 8.153527e-007	39/64/ 9.737674e-007
$n = 500$	42/67/ 9.154342e-007	37/58/ 8.340650e-007	49/74/ 8.277585e-007	37/60/ 6.328648e-007
$n = 1000$	43/62/ 7.874632e-007	40/61/ 8.997602e-007	55/76/ 8.830280e-007	41/68/ 8.430165e-007

(c) (Small-scales). Test results for Algorithm 2.2.

x0	(0.5,. . .,0.5)	(1,. . .,1)	(3,. . .,3)	(−0.75, . . . , −0.75)	(−2, . . . , −2)	(−3, . . . , −3)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 10$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$
$n = 50$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$
$n = 99$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$
x0	(0.5,0,0.5,0,. . .)	(1,0,1,0,. . .)	(3,0,3,0,. . .)	(−0.75, 0, −0.75, 0, . . .)	(−2, 0, −2, 0, . . .)	(−3, 0, −3, 0, . . .)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 10$	196/391/ 9.528159e-007	204/407/ 9.939962e-007	213/425/ 9.975163e-007	$NI > 1000$	$NI > 1000$	$NI > 1000$
$n = 50$	199/397/ 9.791882e-007	208/415/ 9.592104e-007	217/433/ 9.634499e-007	$NI > 1000$	$NI > 1000$	$NI > 1000$
$n = 99$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$

(d) (Large-scales). Test results for Algorithm 2.2.

x0	(0.5,. . .,0.5)	(1,. . .,1)	(3,. . .,3)	(−0.75, . . . , −0.75)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 200$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$
$n = 500$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$
$n = 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$	$NI > 1000$
x0	(0.5,0,0.5,0,. . .)	(1,0,1,0,. . .)	(3,0,3,0,. . .)	(−0.75, 0, −0.75, 0, . . .)
Dim	NI/NG/EG	NI/NG/EG	NI/NG/EG	NI/NG/EG
$n = 200$	200/399/ 9.537272e-007	208/415/ 9.804205e-007	217/433/ 9.775482e-007	$NI > 1000$
$n = 500$	201/401/ 9.425775e-007	209/417/ 9.430954e-007	217/433/ 9.908579e-007	$NI > 1000$
$n = 1000$	202/403/ 9.503816e-007	209/417/ 9.824140e-007	218/435/ 9.470469e-007	$NI > 1000$

Acknowledgments

The authrs are very grateful to anonymous referees and the editors for their valuable suggestions and comments, which improve their paper greatly. This work is supported by China NSF Grands 10761001 and the Scientific Research Foundation of Guangxi University (Grant no. X081082).

References

[1] 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.
[2] 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.
[3] 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.
[4] 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.
[5] Z. Wei, G. Yuan, and Z. Lian, "An approximate Gauss-Newton-based BFGS method for solving symmetric nonlinear equations," *Guangxi Sciences*, vol. 11, no. 2, pp. 91–99, 2004.
[6] G. Yuan and X. Li, "An approximate Gauss-Newton-based BFGS method with descent directions for solving symmetric nonlinear equations," *OR Transactions*, vol. 8, no. 4, pp. 10–26, 2004.
[7] G. Yuan and X. Li, "A new nonmonotone conjugate gradient method for symmetric nonlinear equations," *Guangxi Sciences*, vol. 16, no. 2, pp. 109–112, 2009 (Chinese).
[8] G. Yuan, Z. Wei, and X. Lu, "A modified Gauss-Newton-based BFGS method for symmetric nonlinear equations," *Guangxi Sciences*, vol. 13, no. 4, pp. 288–292, 2006.
[9] 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.
[10] S. G. Nash, "A survey of truncated-Newton methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 45–59, 2000.
[11] A. Griewank, "The 'global' convergence of Broyden-like methods with a suitable line search," *Journal of the Australian Mathematical Society. Series A*, vol. 28, no. 1, pp. 75–92, 1986.
[12] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, Mass, USA, 1995.
[13] W. Cheng, Y. Xiao, and Q.-J. Hu, "A family of derivative-free conjugate gradient methods for large-scale nonlinear systems of equations," *Journal of Computational and Applied Mathematics*, vol. 224, no. 1, pp. 11–19, 2009.
[14] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, MPS/SIAM Series on Optimization, SIAM, Philadelphia, Pa, USA, 2000.

[15] J. E. Dennis Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall Series in Computational Mathematics, Prentice Hall, Englewood Cliffs, NJ, USA, 1983.
[16] R. Fletcher, *Practical Methods of Optimization*, A Wiley-Interscience Publication, John Wiley & Sons, New York, NY, USA, 2nd edition, 1987.
[17] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164–166, 1944.
[18] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, pp. 431–441, 1963.
[19] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research, Springer, New York, NY, USA, 1999.
[20] N. Yamashita and M. Fukushima, "On the rate of convergence of the Levenberg-Marquardt method," *Computing*, vol. 15, pp. 239–249, 2001.
[21] G. Yuan, Z. Wang, and Z. Wei, "A rank-one fitting method with descent direction for solving symmetric nonlinear equations," *International Journal of Communications, Network and System Sciences*, no. 6, pp. 555–561, 2009.
[22] Y. Yuan and W. Sun, *Optimization Theory and Algorithm*, Scientific Publisher House, Beijing, China, 1997.
[23] S. M. Goldfeld, R. E. Quandt, and H. F. Trotter, "Maximization by quadratic hill-climbing," *Econometrica*, vol. 34, pp. 541–551, 1966.
[24] M. J. D. Powell, "Convergence properties of a class of minimization algorithms," in *Nonlinear Programming, 2*, O. L. Mangasarian, R. R. Meyer, and S. M. V, Eds., pp. 1–27, Academic Press, New York, NY, USA, 1974.
[25] R. Fletcher, "An algorithm for solving linearly constrained optimization problems," *Mathematical Programming*, vol. 2, pp. 133–165, 1972.
[26] R. Fletcher, "A model algorithm for composite nondifferentiable optimization problems," *Mathematical Programming Study*, no. 17, pp. 67–76, 1982.
[27] P. T. Boggs, A. J. Kearsley, and J. W. Tolle, "A practical algorithm for general large scale nonlinear optimization problems," *SIAM Journal on Optimization*, vol. 9, no. 3, pp. 755–778, 1999.
[28] J. Nocedal and Y. Yuan, "Combining trust region and line search techniques," in *Advances in Nonlinear Programming*, vol. 14, pp. 153–175, Kluwer Acadmic Publishers, Dordrecht, The Netherlands, 1998.
[29] T. Steihaug, "The conjugate gradient method and trust regions in large scale optimization," *SIAM Journal on Numerical Analysis*, vol. 20, no. 3, pp. 626–637, 1983.
[30] Y. Yuan, "A review of trust region algorithms for optimization," in *Proceedings of the 4th International Congress on Industrial & Applied Mathematics (ICIAM '99)*, pp. 271–282, Oxford University Press, Oxford, UK, 2000.
[31] Y. Yuan, "On the truncated conjugate gradient method," *Mathematical Programming*, vol. 87, no. 3, pp. 561–573, 2000.
[32] M. R. Celis, J. E. Dennis, and R. A. Tapia, "A trust region strategy for nonlinear equality constrained optimization," in *Numerical Optimization, 1984*, P. R. Boggs, R. H. Byrd, and R. B. Schnabel, Eds., pp. 71–82, SIAM, Philadelphia, Pa, USA, 1985.
[33] X. Liu and Y. Yuan, "A global convergent, locally superlinearly convergent algorithm for equality constrained optimization," Research Report ICM-97–84, Inst. Comp. Math. Sci/Eng. Computing, Chinese Academy of Sciences, Beijing, China.
[34] A. Vardi, "A trust region algorithm for equality constrained minimization: convergence properties and implementation," *SIAM Journal on Numerical Analysis*, vol. 22, no. 3, pp. 575–579, 1985.
[35] R. H. Byrd, R. B. Schnabel, and G. A. Shultz, "A trust region algorithm for nonlinearly constrained optimization," *SIAM Journal on Numerical Analysis*, vol. 24, no. 5, pp. 1152–1170, 1987.
[36] J.-Y. Fan, "A modified Levenberg-Marquardt algorithm for singular system of nonlinear equations," *Journal of Computational Mathematics*, vol. 21, no. 5, pp. 625–636, 2003.
[37] M. J. D. Powell and Y. Yuan, "A trust region algorithm for equality constrained optimization," *Mathematical Programming*, vol. 49, no. 2, pp. 189–211, 1990.
[38] A. Vardi, "A trust region algorithm for equality constrained minimization: convergence properties and implementation," *SIAM Journal on Numerical Analysis*, vol. 22, no. 3, pp. 575–591, 1985.
[39] Y. Yuan, "Trust region algorithm for nonlinear equations," *Information*, vol. 1, pp. 7–21, 1998.
[40] Y. Yuan, "On a subproblem of trust region algorithms for constrained optimization," *Mathematical Programming*, vol. 47, no. 1, pp. 53–63, 1990.

[41] 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.
[42] J. Z. Zhang and D. T. Zhu, "Projected quasi-Newton algorithm with trust region for constrained optimization," *Journal of Optimization Theory and Applications*, vol. 67, no. 2, pp. 369–393, 1990.
[43] J. Zhang and Y. Wang, "A new trust region method for nonlinear equations," *Mathematical Methods of Operations Research*, vol. 58, no. 2, pp. 283–298, 2003.
[44] Y. J. Wang and N. H. Xiu, *Theory and Algorithms for Nonlinear Programming*, Shanxi Science and Technology Press, Xian, China, 2004.
[45] 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.
[46] J. E. Dennis Jr. and J. J. Moré, "A characterization of superlinear convergence and its application to quasi-Newton methods," *Mathematics of Computation*, vol. 28, pp. 549–560, 1974.
[47] A. Griewank and Ph. L. Toint, "Local convergence analysis for partitioned quasi-Newton updates," *Numerische Mathematik*, vol. 39, no. 3, pp. 429–448, 1982.
[48] 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.
[49] E. Yamakawa and M. Fukushima, "Testing parallel variable transformation," *Computational Optimization and Applications*, vol. 13, no. 1–3, pp. 253–274, 1999.
[50] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, USA, 1970.