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

Gonglin Yuan,¹ Shide Meng,² and Zengxin Wei¹

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.

Copyright © 2009 Gonglin Yuan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Consider the following system of nonlinear equations:

\[ g(x) = 0, \quad x \in \mathbb{R}^n, \tag{1.1} \]

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

\[ \min \theta(x), \quad x \in \mathbb{R}^n. \tag{1.2} \]

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):

\[ x_{k+1} = x_k + \alpha_k d_k, \quad (1.3) \]

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

\[ \vartheta(x_k + \alpha_k d_k) - \vartheta(x_k) \leq \sigma \alpha_k \nabla \vartheta(x_k)^T d_k, \quad (1.4) \]

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

\[ \vartheta(x_k + \alpha_k d_k) - \vartheta(x_k) \leq \sigma \alpha_k \nabla \vartheta(x_k)^T d_k, \quad (1.5) \]

\[ \| \vartheta(x_k) \| = \max_{0 \leq j \leq m(k)} \| \vartheta(x_k) \|, \quad m(0) = 0 \text{ and } m(k) = \min\{m(k-1) + 1, M\}, k \geq 1, \text{ and } M \]

is a nonnegative integer. From these two techniques (1.4) and (1.5), it is easy to see that the Jacobian matrix \( \nabla g_k \) 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 \( \alpha_k \):

\[ \| g(x_k + \alpha_k d_k) \|^2 \leq \| g(x_k) \|^2 + \delta \alpha_k^2 g_k^T d_k, \quad (1.6) \]

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 Fukushima [4] proposed an approximate monotone line search technique to obtain the stepsize \( \alpha_k \) satisfying

\[ \vartheta(x_k + \alpha_k d_k) - \vartheta(x_k) \leq -\delta_1 \| \alpha_k d_k \|^2 - \delta_2 \| \alpha_k g_k \|^2 + \varepsilon_k \| g(x_k) \|^2, \quad (1.7) \]

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 \( \varepsilon_k \) satisfies

\[ \sum_{k=0}^{\infty} \varepsilon_k < \infty. \quad (1.8) \]

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

$$\min q_k^*(d) = d^T\nabla g(x_k)g(x_k) + \frac{1}{2}d^T\nabla g(x_k)^T\nabla g(x_k)d $$ \tag{1.10}

subjected to $\|d\| \leq \Delta_k$,

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

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

the actual descent of $g(x)$ by

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

and the ratio of actual descent to predicted descent:

$$r_k^* = \frac{\text{Ared}_k^*}{\text{Pred}_k^*}. \tag{1.13}$$
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 \( x_{k+1} = x_k + \lambda_k d_k \) and go to the next step; otherwise reduce the trust region radius \( \Delta_k \) 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

\[
\min \; q_k (d) = g(x_k)^T d + \frac{1}{2} d^T B_k d
\]

s.t. \( \|d\| \leq \Delta_k \),

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:

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

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 \( k \)th iteration by

\[
P_{\text{red}} = q_k(0) - q_k(d_k),
\]

the actual descent of \( g(x) \) by

\[
A_{\text{red}} = \|g(x_k)\|^2 - \|g(x_k + d_k)\|^2,
\]

and the ratio of actual descent to predicted descent:

\[
r_k = \frac{\|g(x_k)\|^2 - \|g(x_k + d_k)\|^2}{q_k(0) - q_k(d_k)}.
\]

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 \( \lambda_k \) 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 \( \lambda_k \):

\[
\|g(x_k + \lambda_k d_k)\|^2 - \|g_k\|^2 \leq -\sigma_1 \|\lambda_k g_k\|^2 - \sigma_2 \|\lambda_k d_k\|^2 + \sigma_3 \lambda_k d_k^T g_k,
\]
where $\sigma_1$, $\sigma_2$, and $\sigma_3$ 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 $\rho$, $r \in (0, 1)$, $0 < \tau_1 < \tau_2 < 1 < \tau_3$, $\sigma_1$, $\sigma_2$, $\sigma_3 > 0$, $\Delta_{\min} > 0$, $x_0 \in \mathbb{R}^n$.

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

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

(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 $x_{k+1} = x_k + \lambda_k d_k$, $\Delta_{k+1} \in [\|d_k\|, \tau_3 \|d_k\|]$;

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 $x_0 \in \mathbb{R}^n$, $\Delta_0 > 0$ is the initial trust region radius, an upper bound of trust region radius $\Delta'$, $0 < \Delta_0 \leq \Delta'$. 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

$$r_k = \frac{\bar{\delta}(x_k) - \bar{\delta}(x_k + d_k)}{\bar{q}_k(0) - \bar{q}_k(d_k)}$$

(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:

\[
y_k = g_{k+1} - g_k \approx \nabla g_{k+1} s_k.
\] (2.3)

Since \( B_{k+1} \) satisfies the secant equation \( B_{k+1} s_k = y_k \) and \( \nabla g_{k+1} \) is symmetric, we have approximately

\[
B_{k+1} s_k \approx \nabla g_{k+1} s_k = \nabla g_{k+1}^T s_k.
\] (2.4)

This means that \( B_{k+1} \) approximates \( \nabla g_{k+1} \) along direction \( s_k \).

3. The Global Convergence

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

\[
\Omega = \{ x \mid \| g(x) \| \leq \| g(x_0) \| \},
\] (3.1)

which is bounded.

Assumption 1. (A) \( g \) is continuously differentiable on an open convex set \( \Omega_1 \) containing \( \Omega \).

(B) The Jacobian of \( g \) is symmetric and bounded on \( \Omega_1 \) and there exists a positive constant \( M \) such that

\[
\| \nabla g(x) \| \leq M \quad \forall x \in \Omega_1.
\] (3.2)

(C) \( \nabla g \) is positive definite on \( \Omega_1 \); that is, there is a constant \( m > 0 \) such that

\[
m\| d \|^2 \leq d^T \nabla g(x) d \quad \forall x \in \Omega_1, \quad d \in \mathbb{R}^n.
\] (3.3)

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

\[
\| \nabla \vartheta(x) - \nabla \vartheta(y) \| \leq L \| x - y \|, \quad \forall x, y \in \Omega_1,
\] (3.4)

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

\[
\| y_k \| \leq M \| s_k \|.
\] (3.5)

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

\[
s_k^T y_k = s_k^T \nabla g(x) s_k \geq m \| s_k \|^2,
\] (3.6)
where \( \xi = x_k + \delta_0(x_{k+1} - x_k), \ \delta_0 \in (0, 1), \) which means that the update matrix \( B_k \) is always positive definite. By (3.5) and (3.6), we have
\[
\frac{s_k^T y_k}{\|s_k\|^2} \geq m, \quad \|y_k\|^2 \leq \frac{M^2}{m}.
\] (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 \( \beta_1, \beta_2, \) and \( \beta_3 \) such that, for any positive integer \( k \)
\[
\beta_1 \|d_k\|^2 \leq d_k^T B_k d_k \leq \beta_2 \|d_k\|^2, \quad \beta_1 \|d_k\| \leq \|B_k d_k\| \leq \beta_3 \|d_k\|
\] (3.8)

hold for at least \( \lceil k/2 \rceil \) 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 \( \nabla g_k \), that is,
\[
\| (\nabla g_k - B_k)d_k \| \leq \varepsilon_0 \|g_k\|,
\] (3.9)
and \( d_k \) satisfies
\[
\|g_k + B_k d_k\| \leq \varepsilon_1 \|g_k\|,
\] (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 \( \delta(x) \) at \( x_k \), that is,
\[
\nabla \delta(x_k)^T d_k < 0.
\] (3.11)

**Proof.** Let \( r_k \) be the residual associated with \( d_k \) so that \( g_k + B_k d_k = r_k \):
\[
\nabla \delta(x_k)^T d_k = g(x_k)^T \nabla g(x_k) d_k
\]
\[
= g(x_k)^T [(\nabla g(x_k) - B_k)d_k + (r_k - g(x_k))] \] (3.12)
\[
= g(x_k)^T (\nabla g(x_k) - B_k)d_k + g(x_k)^T r_k - g(x_k)^T g(x_k).
\]

So we have
\[
\nabla \delta(x_k)^T d_k + \|g(x_k)\|^2 = g(x_k)^T (\nabla g(x_k) - B_k)d_k + g(x_k)^T r_k.
\] (3.13)
Therefore, taking the norm in the right-hand side of the above equality, we have that from Assumption 2

\[
\nabla \vartheta(x_k)^T d_k \leq \|g(x_k)\| \left(\nabla g(x_k) - B_k\right) d_k + \|g(x_k)\| \|r_k\| - \|g(x_k)\|^2 \leq -(1 - \varepsilon_0 - \varepsilon_1)\|g(x_k)\|^2.
\]

(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 \(\|g_{k+1}\| \leq \|g_k\|\) 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 \(\|g_{k+1}\| \leq \|g_k\|\). Then we conclude from Lemma 3.3 in [46] that \(\{\|g_k\|\}\) converges. Moreover, we have for all \(k\)

\[
\|g_{k+1}\| \leq \|g_k\| \leq \|g_{k-1}\| \leq \cdots \leq \|g(x_0)\|.
\]

(3.15)

This implies that \(\{x_k\} \subset \Omega\). \(\square\)

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

\[
g_k^T d_k \leq -\beta_1 \|d_k\|^2, \quad \|g_k\|^2 \geq \beta_2 \|d_k\|^2
\]

(3.16)

\[
\frac{1}{\beta_1} \|g_k\|^2 \leq g_k^T d_k
\]

(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 \(\alpha_k \geq 0\) satisfies the following equations:

\[
B_k d_k + \alpha_k d_k = -g_k,
\]

\[
\alpha_k (\|d_k\| - \Delta_k) = 0.
\]

(3.18)

From (3.18), we can obtain

\[
d_k^T B_k d_k + g_k^T d_k = -\alpha_k \|d_k\|^2 \leq 0,
\]

(3.19)

\[
\alpha_k = \frac{-g_k^T d_k - d_k^T B_k d_k}{\|d_k\|^2}.
\]

(3.20)

By (3.19) and (3.8), we get (3.16), which also imply that the inequality (3.17) holds. \(\square\)
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 \( \lambda_k \) 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, \( \lambda_k \) must satisfy

\[
\|g(x_k + a_k d_k)\|^2 - \|g(x_k)\|^2 \leq \delta \lambda_k g(x_k)^T \nabla g(x_k) d_k, \quad \forall \delta \in (0, 1). \tag{3.21}
\]

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

\[
g(x_k)^T \nabla g(x_k) d_k \leq -\beta_0 \|g_k\|^2 = -\frac{\beta_0}{3} \|g_k\|^2 - \frac{\beta_0}{3} \|g_k\|^2 - \frac{\beta_0}{3} \|g_k\|^2
\]

\[
\leq -\frac{\beta_0}{3} \|g_k\|^2 - \frac{\beta_0}{3} \|g_k\|^2 - \frac{\beta_0}{3} \|g_k\|^2 + \frac{\beta_0}{3} \beta_1 g_k^T d_k,
\]

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

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

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

In particular, one has

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

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

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

From Step 4 of Algorithm 2.1, if \( r_k \geq \rho \) is true, we get

\[
\|g(x_{k+1})\|^2 - \|g(x_k)\|^2 \leq q_k(d_k) \leq \frac{1}{2} g_k^T d_k \leq -\frac{1}{2} d_k^T B_k d_k. \tag{3.26}
\]

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

\[
\|g(x_{k+1})\|^2 - \|g(x_k)\|^2 \leq -\sigma_1 \|g_k\|^2 - \sigma_2 \|d_k\|^2 + \sigma_3 \lambda_k d_k^T g_k \leq \sigma_3 \lambda_k d_k^T g_k \leq -\sigma_3 \lambda_k d_k^T B_k d_k. \tag{3.27}
\]
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 $\lambda^* \leq \lambda_k$ for all $k$. By (3.26) and (3.27), we have

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

(3.28)

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

$$
\sum_{k=0}^{\infty} d_k^T B_k d_k \leq \sum_{k=0}^{\infty} (-g_k^T d_k) \leq \frac{1}{\rho_1} \sum_{k=0}^{\infty} \left(\|g(x_k)\|^2 - \|g(x_{k+1})\|^2\right)
$$

$$
= \frac{1}{\rho_1} \lim_{N \to \infty} \sum_{k=0}^{N} \left(\|g(x_k)\|^2 - \|g(x_{k+1})\|^2\right)
$$

$$
= \frac{1}{\rho_1} \lim_{N \to \infty} \left(\|g(x_0)\|^2 - \|g(x_{N+1})\|^2\right),
$$

(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. \qed

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

$$
b_1 \|g_k\| \leq \|d_k\| \leq b_2 \|g_k\|, \quad \alpha_k \leq b_3.
$$

(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

$$
\beta_1 \|d_k\|^2 \leq d_k^T B_k d_k = -d_k^T g_k \leq \|d_k\| \|g_k\|, \quad \|g_k\| = \|B_k d_k\| \leq \beta_3 \|d_k\|.
$$

(3.31)

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

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

$$
\beta_1 \|d_k\|^2 \leq d_k^T B_k d_k \leq -g_k^T d_k \leq \|g_k\| \|d_k\|.
$$

(3.32)

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

$$
(1 - \varepsilon_1) \|g_k\| \leq \|B_k d_k\| \leq \beta_3 \|d_k\|.
$$

(3.33)
So we obtain \( \|d_k\| \geq (1 - \varepsilon_1)/\beta_3 \|g_k\| \). Using (3.20), we have

\[
\alpha_k = \frac{-g_k^T d_k - d_k^T B_k d_k}{\|d_k\|^2} \leq \frac{\|g_k\|}{\|d_k\|} \leq \frac{\beta_3}{1 - \varepsilon_1}.
\]

(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

\[
\lim_{k \to \infty} \|g_k\| = 0.
\]

(3.35)

**Proof.** By Lemma 3.6, we have

\[
\lim_{k \to \infty} -g_k^T d_k = \lim_{k \to \infty} d_k^T B_k d_k = 0.
\]

(3.36)

Combining (3.8) and (3.36), we get

\[
\lim_{k \to \infty} \|d_k\| = 0.
\]

(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.** \( \nabla g \) is Hölder continuous at \( x^* \), that is, for every \( x \) in a neighborhood of \( x^* \), there are positive constants \( M_1 \) and \( \gamma \) such that

\[
\| \nabla g(x) - \nabla g(x^*) \| \leq M_1 \| x - x^* \|^\gamma,
\]

(4.1)

where \( x^* \) stands for the unique solution of (1.1) in \( \Omega_1 \).

**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

\[
\sum_{k=0}^{\infty} \|x_k - x^*\|^\gamma < \infty.
\]

(4.2)
Moreover, one has

\[ \sum_{k=0}^{\infty} \chi_k(\gamma) < \infty, \quad (4.3) \]

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

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

\[ m\| x - x^* \| \leq \| g(x) \| = \| g(x) - g(x^*) \| \leq M \| x - x^* \|, \quad x \in \Omega. \quad (4.4) \]

By (3.8) and (3.30), we have

\[ -\beta_2 \| d_k \|^2 \leq -d_k^T B_k d_k \leq -\beta_1 \| d_k \|^2, \quad (4.5) \]

Together with (3.28), we get

\[ \| g_{k+1} \|^2 - \| g_k \|^2 \leq \rho_1 g_k^T d_k \leq -\rho_1 d_k^T B_k d_k \leq -\rho_1 \beta_1 \| d_k \|^2 \leq -\rho_1 \beta_1 \| g_k \|^2, \quad (4.6) \]

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

\[ \| g_{k+1} \|^2 \leq \| g_k \|^2 - \rho_0 \| g_k \|^2 \leq (1 - \rho_0) \| g_k \|^2 \leq \cdots \leq (1 - \rho_0)^{k-k_0+1} \| g_{k_0} \|^2 = c_0 c_1^k, \quad (4.7) \]

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 \( \| x_{k+1} - x^* \|^2 \leq m^2 c_0 c_1^k \) holds for all \( k \) large enough. Therefore, for any \( \gamma \), we have (4.2). Notice that \( \chi_k(\gamma) \leq \| x_k - x^* \|^\gamma + \| x_{k+1} - x^* \|^\gamma \); 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

\[ \| y_k - \nabla g(x^*) s_k \| \leq M_2 \chi_k \| s_k \|, \quad (4.8) \]

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

$$
\|y_k - \nabla g(x^*) s_k\| = \|\nabla g(x_k + t_0(x_{k+1} - x_k))s_k - \nabla g(x^*) s_k\|
\leq \|\nabla g(x_k + t_0(x_{k+1} - x_k)) - \nabla g(x^*)\| \|s_k\|
\leq M_1 \|x_k + t_0(x_{k+1} - x_k) - x^*\|^T \|s_k\|
\leq M_2 \chi_k \|s_k\|,
$$

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

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 $\gamma_i$, $i = 1, 2, 3, 4$, and $\eta \in (0, 1)$ such that

$$
\|B_{k+1} - \nabla g(x^*)\|_{Q,F} \leq (1 + \gamma_1 \chi_k) \|B_k - \nabla g(x^*)\|_{Q,F} + \gamma_2 \chi_k, \tag{4.10}
$$

$$
\|H_{k+1} - \nabla g(x^*)^{-1}\|_{Q^{-1,F}} \leq \left(1 - \eta \gamma_4 \chi_k^2 + \gamma_3 \chi_k\right) \|H_k - \nabla g(x^*)^{-1}\|_{Q^{-1,F}} + \gamma_4 \chi_k, \tag{4.11}
$$

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

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

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

Proof. From (1.15), we have

$$
\|B_{k+1} - \nabla g(x^*)\|_{Q,F} = \left\|B_k - \nabla g(x^*) + \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T s_k}\right\|_{Q,F}
\leq (1 + \gamma_1 \tau_k) \|B_k - \nabla g(x^*)\|_{Q,F} + \gamma_2 \chi_k,
$$

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. \qed

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

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

where $s_k = x_{k+1} - x_k$. 
Let the conditions in Assumptions 1, 2 and 3 hold. If Theorem 4.5.

In a similar way to Proof.

On the other hand, we have

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

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 \(\epsilon_1 \to 0\) in (3.10). Then the sequence \(\{x_k\}\) generated by Algorithm 2.1 converges to \(x^*\) superlinearly for \(\lambda_k = 1\).

Proof. For all \(x_k \in \Omega_i\), we get

\[
\|g_{k+1}\| = \frac{\|g_k + B_k d_k + (\nabla g_k - B_k) d_k + O(\|d_k\|^2)\|}{\|d_k\|} \leq \frac{\|g_k + B_k d_k\|}{\|g_k\|} \frac{\|g_k\|}{\|d_k\|} + \frac{\|\nabla g_k - B_k\|}{\|d_k\|} \frac{\|d_k\|}{\|d_k\|} + O(\|d_k\|). \tag{4.18}
\]
where the last inequality follows (3.10). By (3.5), we have

\[ \|g_k\| \leq \|g_{k+1} - g_k\| + \|g_{k+1}\| \leq M\|d_k\| + \|g_{k+1}\|. \]  

(4.19)

Dividing both sides by \(\|d_k\|\), we get

\[ \frac{\|g_k\|}{\|d_k\|} \leq M + \frac{\|g_{k+1}\|}{\|d_k\|}. \]  

(4.20)

Substituting this into (4.18), we can obtain

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

(4.21)

which means that

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

(4.22)

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

\[ \lim_{k \to \infty} \frac{\|g_{k+1}\|}{\|d_k\|} = 0. \]  

(4.23)

Using (3.16), we get

\[ \lim_{k \to \infty} \frac{\|g_{k+1}\|}{\|g_k\|} = 0. \]  

(4.24)

Considering (4.4), we have

\[ \lim_{k \to \infty} \frac{\|x_k + d_k - x^*\|}{\|x_k - x^*\|} = 0. \]  

(4.25)

Therefore, we get the result of the superlinear convergence.

\[ \square \]

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

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

where \( A \) is the \( n \times n \) tridiagonal matrix given by

\[
A = \begin{bmatrix}
8 & -1 & & & \\
-1 & 8 & -1 & & \\
& -1 & 8 & -1 & \\
& & \ddots & \ddots & \ddots \\
& & & -1 & 8
\end{bmatrix}, \quad (5.2)
\]

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

\[ \min f(x), \; x \in \mathbb{R}^n, \quad (5.3) \]

with Engval function [49] \( f : \mathbb{R}^n \to \mathbb{R} \) defined by

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

The related symmetric nonlinear equation is

\[ g(x) \triangleq \frac{1}{4} \nabla f(x) = 0, \quad (5.5) \]

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

\[
g_1(x) = x_1 \left( x_1^2 + x_2^2 \right) - 1, \\
g_i(x) = x_i \left( x_{i-1}^2 + 2x_i^2 + x_{i+1}^2 \right) - 1, \quad i = 2, 3, \ldots, n-1, \quad (5.6) \\
g_n(x) = x_n \left( x_{n-1}^2 + x_n^2 \right).
\]

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.

<table>
<thead>
<tr>
<th>x₀</th>
<th>(1,..,1)</th>
<th>(60,..,60)</th>
<th>(600,..,600)</th>
<th>(−1,..,−1)</th>
<th>(−60,..,−60)</th>
<th>(−600,..,−600)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
<tr>
<td>n = 10</td>
<td>13/24/</td>
<td>14/25/</td>
<td>17/30/</td>
<td>13/24/</td>
<td>14/25/</td>
<td>17/30/</td>
</tr>
<tr>
<td>n = 50</td>
<td>48/101/</td>
<td>49/102/</td>
<td>50/103/</td>
<td>48/101/</td>
<td>49/102/</td>
<td>50/103/</td>
</tr>
<tr>
<td>n = 99</td>
<td>82/171/</td>
<td>89/188/</td>
<td>91/190/</td>
<td>82/171/</td>
<td>89/188/</td>
<td>91/190/</td>
</tr>
<tr>
<td>x₀</td>
<td>(1,0,1,0,..)</td>
<td>(60,0,60,0,..)</td>
<td>(600,0,600,0,..)</td>
<td>(−1,0,−1,0,..)</td>
<td>(−60,0,−60,0,..)</td>
<td>(−600,0,−600,0,..)</td>
</tr>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
<tr>
<td>n = 10</td>
<td>21/42/</td>
<td>22/43/</td>
<td>22/45/</td>
<td>21/44/</td>
<td>22/43/</td>
<td>22/45/</td>
</tr>
<tr>
<td>n = 50</td>
<td>72/153/</td>
<td>86/181/</td>
<td>88/185/</td>
<td>70/151/</td>
<td>86/181/</td>
<td>49/183/</td>
</tr>
<tr>
<td>n = 99</td>
<td>73/156/</td>
<td>88/185/</td>
<td>88/191/</td>
<td>74/161/</td>
<td>88/185/</td>
<td>88/191/</td>
</tr>
</tbody>
</table>
| (b) (Large-scales). Test results for Algorithm 2.1.

<table>
<thead>
<tr>
<th>x₀</th>
<th>(1,..,1)</th>
<th>(60,..,60)</th>
<th>(600,..,600)</th>
<th>(−1,..,−1)</th>
<th>(−60,..,−60)</th>
<th>(−600,..,−600)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
<tr>
<td>n = 500</td>
<td>85/180/</td>
<td>103/218/</td>
<td>115/244/</td>
<td>83/178/</td>
<td>103/218/</td>
<td>115/244/</td>
</tr>
<tr>
<td>n = 1000</td>
<td>76/165/</td>
<td>96/207/</td>
<td>105/224/</td>
<td>76/165/</td>
<td>96/207/</td>
<td>105/224/</td>
</tr>
<tr>
<td>x₀</td>
<td>(1,0,1,0,..)</td>
<td>(60,0,60,0,..)</td>
<td>(600,0,600,0,..)</td>
<td>(−1,0,−1,0,..)</td>
<td>(−60,0,−60,0,..)</td>
<td>(−600,0,−600,0,..)</td>
</tr>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
<tr>
<td>n = 200</td>
<td>68/149/</td>
<td>91/194/</td>
<td>101/216/</td>
<td>69/150/</td>
<td>91/194/</td>
<td>101/216/</td>
</tr>
<tr>
<td>n = 500</td>
<td>72/155/</td>
<td>96/205/</td>
<td>106/225/</td>
<td>72/155/</td>
<td>97/206/</td>
<td>106/225/</td>
</tr>
</tbody>
</table>
| (c) (Small-scales). Test results for Algorithm 2.2.

<table>
<thead>
<tr>
<th>x₀</th>
<th>(1,..,1)</th>
<th>(60,..,60)</th>
<th>(600,..,600)</th>
<th>(−1,..,−1)</th>
<th>(−60,..,−60)</th>
<th>(−600,..,−600)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
<tr>
<td>n = 10</td>
<td>54/107/</td>
<td>67/133/</td>
<td>74/147/</td>
<td>54/107/</td>
<td>67/133/</td>
<td>74/147/</td>
</tr>
<tr>
<td>n = 50</td>
<td>58/115/</td>
<td>72/143/</td>
<td>79/157/</td>
<td>58/115/</td>
<td>72/143/</td>
<td>79/157/</td>
</tr>
<tr>
<td>n = 99</td>
<td>60/119/</td>
<td>73/145/</td>
<td>80/159/</td>
<td>60/119/</td>
<td>73/145/</td>
<td>80/159/</td>
</tr>
<tr>
<td>x₀</td>
<td>(1,0,1,0,..)</td>
<td>(60,0,60,0,..)</td>
<td>(600,0,600,0,..)</td>
<td>(−1,0,−1,0,..)</td>
<td>(−60,0,−60,0,..)</td>
<td>(−600,0,−600,0,..)</td>
</tr>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
<tr>
<td>n = 10</td>
<td>52/103/</td>
<td>64/127/</td>
<td>72/143/</td>
<td>52/103/</td>
<td>64/127/</td>
<td>72/143/</td>
</tr>
<tr>
<td>n = 50</td>
<td>56/111/</td>
<td>69/137/</td>
<td>77/153/</td>
<td>56/111/</td>
<td>69/137/</td>
<td>77/153/</td>
</tr>
<tr>
<td>n = 99</td>
<td>57/115/</td>
<td>71/141/</td>
<td>78/155/</td>
<td>57/113/</td>
<td>71/141/</td>
<td>78/155/</td>
</tr>
</tbody>
</table>
| (d) (Large-scales). Test results for Algorithm 2.2.

<table>
<thead>
<tr>
<th>x₀</th>
<th>(1,..,1)</th>
<th>(60,..,60)</th>
<th>(600,..,600)</th>
<th>(−1,..,−1)</th>
<th>(−60,..,−60)</th>
<th>(−600,..,−600)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
<tr>
<td>n = 10</td>
<td>52/103/</td>
<td>64/127/</td>
<td>72/143/</td>
<td>52/103/</td>
<td>64/127/</td>
<td>72/143/</td>
</tr>
<tr>
<td>n = 50</td>
<td>56/111/</td>
<td>69/137/</td>
<td>77/153/</td>
<td>56/111/</td>
<td>69/137/</td>
<td>77/153/</td>
</tr>
<tr>
<td>n = 99</td>
<td>57/115/</td>
<td>71/141/</td>
<td>78/155/</td>
<td>57/113/</td>
<td>71/141/</td>
<td>78/155/</td>
</tr>
</tbody>
</table>
The numerical results show that this method is promising. In fact, this problem comes from unconstrained optimization problem and an equality constrained optimization 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.

\[ \Delta' = \Delta_0 = \|g_0\|, \eta_1 = 0.25, \eta_2 = 0.75, \mu = 0.01, \text{ and } \eta_3 = 2. \] 

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 \( \|g(x)\| \leq 10^{-8} \) 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>
<thead>
<tr>
<th>Dim</th>
<th>(1,...,1)</th>
<th>(60,...60)</th>
<th>(600,...600)</th>
<th>(−1,...,−1)</th>
<th>(−60,...,−60)</th>
<th>(−600,...,−600)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x0</td>
<td>(1,0,1,0,...)</td>
<td>(60,0,60,0,...)</td>
<td>(600,0,600,0,...)</td>
<td>(−1,0,−1,0,...)</td>
<td>(−60,0,−60,0,...)</td>
<td>(−600,0,−600,0,...)</td>
</tr>
<tr>
<td>Dim</td>
<td>(1,...,1)</td>
<td>(60,...,60)</td>
<td>(600,...,600)</td>
<td>(−1,...,−1)</td>
<td>(−60,...,−60)</td>
<td>(−600,...,−600)</td>
</tr>
<tr>
<td>n = 200</td>
<td>59/117/7.503172e−7</td>
<td>72/143/8.249912e−7</td>
<td>79/157/9.576414e−7</td>
<td>72/143/8.249912e−7</td>
<td>72/143/8.249912e−7</td>
<td>72/143/8.249912e−7</td>
</tr>
<tr>
<td>n = 1000</td>
<td>61/121/9.191890e−7</td>
<td>75/149/8.443936e−7</td>
<td>82/163/8.443936e−7</td>
<td>75/149/8.443936e−7</td>
<td>82/163/8.443936e−7</td>
<td>8.443936e−7</td>
</tr>
</tbody>
</table>

(d) (Large-scales). Test results for Algorithm 2.2.
Table 2: Test Results For Problem 2.

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

<table>
<thead>
<tr>
<th>x₀</th>
<th>(0.5,...,0.5)</th>
<th>(1,...,1)</th>
<th>(3,...,3)</th>
<th>(-0.75,...,-0.75)</th>
<th>(-2,...,-2)</th>
<th>(-3,...,-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>x₀</th>
<th>(0.5,0,5,0,...)</th>
<th>(1,0,1,0,...)</th>
<th>(3,0,3,0,...)</th>
<th>(-0.75,0,-0.75,...)</th>
<th>(-2,0,-2,0,...)</th>
<th>(-3,0,-3,0,...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>x₀</th>
<th>(0.5,...,0.5)</th>
<th>(1,...,1)</th>
<th>(3,...,3)</th>
<th>(-0.75,...,-0.75)</th>
<th>(-2,...,-2)</th>
<th>(-3,...,-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
<td>NI/NG/EG</td>
</tr>
<tr>
<td>n = 10</td>
<td>40/57/7.46237e-007</td>
<td>41/58/4.92102e-007</td>
<td>112/130/4.00136e-007</td>
<td>39/76/6.62995e-007</td>
<td>40/65/7.78554e-007</td>
<td>40/65/7.31669e-007</td>
</tr>
<tr>
<td>n = 1000</td>
<td>42/59/7.38293e-007</td>
<td>40/59/7.46321e-007</td>
<td>120/146/6.04416e-007</td>
<td>40/69/9.40421e-07</td>
<td>40/69/9.40421e-07</td>
<td>40/69/9.40421e-07</td>
</tr>
</tbody>
</table>

- Test results for Algorithm 2.1
- Test results for Algorithm 2.2
Acknowledgments

The authors 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 Grants 10761001 and the Scientific Research Foundation of Guangxi University (Grant no. X081082).

References


