Journal of Inequalities and Applications
a SpringerOpen Journal

RESEARCH Open Access

A quasi-Newton algorithm for large-scale nonlinear equations

Linghua Huang*

*Correspondence:
linghuahuang@163.com
School of Information and Statistics, Guangxi University of Finance & Economics, Nanning, Guangxi 530003, P.R. China

Abstract

In this paper, the algorithm for large-scale nonlinear equations is designed by the following steps: (i) a conjugate gradient (CG) algorithm is designed as a sub-algorithm to obtain the initial points of the main algorithm, where the sub-algorithm's initial point does not have any restrictions; (ii) a quasi-Newton algorithm with the initial points given by sub-algorithm is defined as main algorithm, where a new nonmonotone line search technique is presented to get the step length α_k. The given nonmonotone line search technique can avoid computing the Jacobian matrix. The global convergence and the $1+q$-order convergent rate of the main algorithm are established under suitable conditions. Numerical results show that the proposed method is competitive with a similar method for large-scale problems.

Keywords: nonlinear equations; large-scale; conjugate gradient; quasi-Newton method; global convergence

1 Introduction

Consider the following nonlinear equations:

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

Here $e : \Re^n \to \Re^n$ is continuously differentiable and n denotes large-scale dimension. The large-scale nonlinear equations are difficult to solve since the relations of the variables x are complex and the dimension is larger. Problem (1.1) can model many real-life problems, such as engineering problems, dimensions of mechanical linkages, concentrations of chemical species, cross-sectional properties of structural elements, etc. If the Jacobian $\nabla e(x)$ of e is symmetric, then problem (1.1) is called a system of symmetric nonlinear equations. Let p be the norm function with $p(x) = \frac{1}{2}\|e(x)\|^2$, where $\|\cdot\|$ is the Euclidean norm. Then (1.1) is equivalent to the following global optimization models:

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

In fact, there are many actual problems that can convert to the above problems (1.2) (see [1–9] etc.) and have similar models (see [10–27] etc.). The iterative formula for (1.1) is

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

where α_k is a step length and d_k is a search direction. Now let us review some methods for α_k and d_k, respectively:

(i) Li and Fukashima [28] proposed an approximately monotone technique for α_k:

$$p(x_k + \alpha_k d_k) - p(x_k) \leq -\delta_1 \|\alpha_k d_k\|^2 - \delta_2 \|\alpha_k e_k\|^2 + \epsilon_k \|e_k\|^2, \tag{1.3}$$

where $e_k = e(x_k)$, $\delta_1 > 0, \delta_2 > 0$ are positive constants, $\alpha_k = r^{i_k}, r \in (0,1), i_k$ is the smallest nonnegative integer i satisfying (1.3) and ϵ_k such that

$$\sum_{k=0}^{\infty} \epsilon_k < \infty. \tag{1.4}$$

(ii) Gu *et al.* [29] presented a descent line search technique:

$$p(x_k + \alpha_k d_k) - p(x_k) \leq -\delta_1 \|\alpha_k d_k\|^2 - \delta_2 \|\alpha_k e_k\|^2. \tag{1.5}$$

(iii) Brown and Saad [30] gave the following technique to obtain α_k:

$$p(x_k + \alpha_k d_k) - p(x_k) \leq \beta \alpha_k \nabla p(x_k)^T d_k, \tag{1.6}$$

where $\beta \in (0,1)$ and $\nabla p(x_k) = \nabla e(x_k) e(x_k)$.

(iv) Based on this technique, Zhu [31] proposed a nonmonotone technique:

$$p(x_k + \alpha_k d_k) - p(x_{l(k)}) \leq \beta \alpha_k \nabla p(x_k)^T d_k, \tag{1.7}$$

$p(x_{l(k)}) = \max_{0 \leq j \leq m(k)} \{p(x_{k-j})\}$, $m(0) = 0$, and $m(k) = \min\{m(k-1) + 1, M\}$, $k \geq 1$, and M is a nonnegative integer.

(v) Yuan and Lu [32] gave a new technique:

$$p(x_k + \alpha_k d_k) - p(x_k) \leq \beta \alpha_k^2 e(x_k)^T d_k, \tag{1.8}$$

and some convergence results are obtained.

Next we present some techniques for the calculation of d_k. At present, there exist many well-known methods for d_k, such as the Newton method, the trust region method, and the quasi-Newton method, etc.

(i) The Newton method has the following form to get d_k:

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

This method is regarded as one of the most effective methods. However, its efficiency largely depends on the possibility to efficiently solve (1.9) at each iteration. Moreover, the exact solution of the system (1.9) could be too burdensome when the iterative point x_k is far from the exact solution [33]. In order to overcome this drawback, inexact quasi-Newton methods are often used.

(ii) The quasi-Newton method is of the form

$$B_k d_k + e_k = 0, \tag{1.10}$$

where B_k is generated by a quasi-Newton update formula, where the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update formula is one of the well-known quasi-Newton formulas with

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k {y_k}^T}{{y_k}^T s_k}, \tag{1.11}$$

where $s_k = x_{k+1} - x_k, y_k = e_{k+1} - e_k$, and $e_{k+1} = e(x_{k+1})$. Set H_k to be the inverse of B_k, then the inverse formula of (1.11) has the following form:

$$\begin{aligned} H_{k+1} &= H_k - \frac{{y_k}^T (s_k - H_k y_k) s_k s_k^T}{({y_k}^T s_k)^2} + \frac{(s_k - H_k y_k) s_k^T + s_k (s_k - H_k y_k)^T}{({y_k}^T s_k)^2} \\ &= \left(I - \frac{s_k {y_k}^T}{{y_k}^T s_k} \right) H_k \left(I - \frac{y_k s_k^T}{{y_k}^T s_k} \right) + \frac{s_k s_k^T}{{y_k}^T s_k}. \end{aligned} \tag{1.12}$$

There exist many quasi-Newton methods (see [31, 32, 34–39]) representing the basic approach underlying most of the Newton-type large-scale algorithms.

The earliest nonmonotone line search framework was developed by Grippo, Lampariello, and Lucidi in [40] for Newton's methods. Many subsequent papers have exploited nonmonotone line search techniques of this nature (see [41–44] etc.), which shows that the nonmonotone technique works well in many cases. Considering these points, Zhu [31] proposed the nonmonotone line search (1.7). From (1.7), we can see that the Jacobian matrix $\nabla e(x)$ must be computed at every iteration. Computing the Jacobian matrix $\nabla e(x)$ may be expensive if n is large and for any n at every iteration. Thus, one might prefer to remove the matrix, leading to a new nonmonotone technique.

Inspired by the above observations, we make a study of inexact quasi-Newton methods with a new nonmonotone technique for solving smooth nonlinear equations. In the kth iteration of our algorithm, the following new nonmonotone technique is used to obtain α_k:

$$p(x_k + \alpha_k d_k) \le p(x_{l(k)}) + \alpha_k \sigma e(x_k)^T d_k, \tag{1.13}$$

where $\sigma \in (0,1)$ is a constant and d_k is a solution of (1.10). Comparing with (1.7), the new technique (1.13) does not compute the Jacobian matrix $\nabla e(x)$. Then the storage and workload can be saved in theory. In Section 3, we will state the technique (1.13) is well defined.

It is well known that the initial point plays an important role in an algorithm. For example, the local superlinear convergence needs the iteration point x lies in the neighborhood of the optimal solution x^*, if the choice of the point x is correct then the Newton method can get the optimal solution x^* just need one step, moreover, the correct initial point can speed up the efficiency of an algorithm. The nonlinear conjugate gradient method is one of the most effective line search methods for unconstrained optimization problems due to its simplicity and low memory requirement, especially for large-scale problems. Many scholars have made many studies and obtained lots of achievements on the CG methods or other similar new methods (see [45–49] etc.), where the results of [46] are especially interesting. It has been proved that the numerical performance of the CG methods is very interesting for large-scale problems in different application fields. These considerations prompt us to design a CG algorithm (sub-algorithm) for solving large-scale nonlinear equations, where the terminated iteration point of the CG algorithm was used as the initial point of the given algorithm (main algorithm). Then there exist two advantages from this process: one is that we can use the CG's characteristic to get a better initial point and another is that the good convergent results of the main algorithm can be preserved. The main attributes of this paper are stated as follows:

- A sub-algorithm is designed to get the initial point of the main algorithm.
- A new nonmonotone line search technique is presented, moreover, the Jacobian matrix ∇e_k must not be computed at every iteration.
- The given method possesses the sufficient descent property for the normal function $p(x)$.
- The global convergence and the $1+q$-order convergent rate of the new method are established under suitable conditions.
- Numerical results show that this method is more effective than other similar methods.

We organize the paper as follows. In Section 2, the algorithms are stated. Convergent results are established in Section 3 and numerical results are reported in Section 4. In the last section, our conclusion is given. Throughout this paper, we use these notations: $\|\cdot\|$ is the Euclidean norm, $e(x_k)$ and $g(x_{k+1})$ are replaced by e_k and g_{k+1}, respectively.

2 Algorithm

In this section, we will design a sub-algorithm and the main algorithm, respectively. These two algorithms are listed as follows.

Initial point algorithm (sub-algorithm)

```
Step 0: Given any $x_0 \in \Re^n, \delta_1, \delta_2 \in (0,1), \epsilon_k > 0$, $r \in (0,1), \epsilon \in [0,1)$, let $k := 0$.
Step 1: If $\|e_k\| \le \epsilon$, stop. Otherwise let $d_k = -e_k$ and go to next step.
Step 2: Choose $\epsilon_{k+1}$ satisfies (1.4) and let $\alpha_k = 1, r, r^2, r^3, \dots$ until (1.3) holds.
Step 3: Let $x_{k+1} = x_k + \alpha_k d_k$.
Step 4: If $\|e_{k+1}\| \le \epsilon$, stop.
Step 5: Compute $d_{k+1} = -e_{k+1} + \beta_k d_k$, set $k = k+1$ and go to Step 2.
```

Remark (i) β_k of Step 5 is a scalar and different β_k will determine different CG methods.

(ii) From Step 2 and [28], it is easy to deduce that there exists α_k such that (1.3). Thus, this sub-algorithm is well defined.

In the following, we will state the main algorithm. First, assume that the terminated point of sub-algorithm is x_{sup}, then the given algorithm is defined as follows.

Algorithm 1 (Main algorithm) *Step* 0: *Choose* $x_{sup} \in \Re^n$ *as the initial point, an initial symmetric positive definite matrix* $B_0 \in \Re^{n\times n}$, *and constants* $r, \sigma \in (0,1), \epsilon_{\text{main}} < \epsilon$, *a positive integer* $M > 0$, $m(k) = 0$, *let* $k := 0$;

```
Step 1: Stop if $\|e_k\| \le \epsilon_{\text{main}}$. Otherwise solve (1.10) to get $d_k$.
Step 2: Let $\alpha_k = 1, r, r^2, r^3, \dots$ until (1.13) holds.
Step 3: Let the next iterative be $x_{k+1} = x_k + \alpha_k d_k$.
Step 4: Update $B_k$ by quasi-Newton update formula and ensure the update matrix $B_{k+1}$
    is positive definite.
Step 5: Let $k := k+1$. Go to Step 1.
```

Remark Step 4 of Algorithm 1 can ensure that B_k is always positive definite. This means that (1.10) has a unique solution d_k. By positive definiteness of B_k, it is easy to obtain $e_k^T d_k < 0$. In the following sections, we only concentrate to the convergence of the main algorithm.

3 Convergence analysis

Let Ω be the level set with

$$\Omega = \{x | \|e(x)\| \le \|e(x_0)\|\}. \tag{3.1}$$

Similar to [31, 32, 50], the following assumptions are needed to prove the global convergence of Algorithm 1.

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

(ii) The Jacobian of e is symmetric, bounded, and positive definite on Ω_1, *i.e.*, there exist positive constants $M^* \ge m_* > 0$ such that

$$\|\nabla e(x)\| \le M^* \quad \forall x \in \Omega_1 \tag{3.2}$$

and

$$m_* \|d\|^2 \le d^T \nabla e(x) d \quad \forall x \in \Omega_1, d \in \Re^n. \tag{3.3}$$

Assumption B B_k is a good approximation to ∇e_k, *i.e.*,

$$\|(\nabla e_k - B_k) d_k\| \le \epsilon_* \|e_k\|, \tag{3.4}$$

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

Considering Assumption B and using the von Neumann lemma, we deduce that B_k is also bounded (see [31]).

Lemma 3.1 *Let Assumption* B *hold. Then d_k is a descent direction of $p(x)$ at x_k, i.e.,*

$$\nabla p(x_k)^T d_k \le -(1 - \epsilon_*) \|e(x_k)\|^2. \tag{3.5}$$

Proof By using (1.10), we get

$$\begin{aligned} \nabla p(x_k)^T d_k &= e(x_k)^T \nabla e(x_k) d_k \\ &= e(x_k)^T \big[\big(\nabla e(x_k) - B_k\big) d_k - e(x_k)\big] \\ &= e(x_k)^T \big(\nabla e(x_k) - B_k\big) d_k - e(x_k)^T e(x_k). \end{aligned} \tag{3.6}$$

Thus, we have

$$\begin{aligned} \nabla p(x_k)^T d_k + \|e(x_k)\|^2 &= e(x_k)^T \big(\nabla e(x_k) - B_k\big) d_k \\ &\le \|e(x_k)\| \big\|\big(\nabla e(x_k) - B_k\big) d_k\big\|. \end{aligned}$$

It follows from (3.4) that

$$\begin{aligned}\nabla p(x_k)^T d_k &\le \|e(x_k)\| \|(\nabla e(x_k) - B_k)d_k\| - \|e(x_k)\|^2 \\ &\le -(1-\epsilon_*)\|e(x_k)\|^2. \end{aligned} \tag{3.7}$$

The proof is complete. □

The following lemma shows that the line search technique (1.13) is reasonable, then Algorithm 1 is well defined.

Lemma 3.2 *Let Assumptions* A *and* B *hold. Then Algorithm* 1 *will produce an iteration* $x_{k+1} = x_k + \alpha_k d_k$ *in a finite number of backtracking steps.*

Proof From Lemma 3.5 in [32] we have in a finite number of backtracking steps

$$p(x_k + \alpha_k d_k) \le p(x_k) + \alpha_k \sigma e(x_k)^T d_k,$$

from which, in view of the definition of $p(x_{l(k)}) = \max_{0\le j\le m(k)}\{p(x_{k-j})\} \ge p(x_k)$, we obtain (1.13). Thus we conclude the result of this lemma. The proof is complete. □

Now we establish the global convergence theorem of Algorithm 1.

Theorem 3.1 *Let Assumptions* A *and* B *hold, and* $\{\alpha_k, d_k, x_{k+1}, e_{k+1}\}$ *be generated by Algorithm* 1. *Then*

$$\lim_{k\to\infty} \|e_k\| = 0. \tag{3.8}$$

Proof By the acceptance rule (1.13), we have

$$p(x_{k+1}) - p(x_{l(k)}) \le \sigma \alpha_k e_k^T d_k < 0. \tag{3.9}$$

Using $m(k+1) \le m(k)+1$ and $p(x_{k+1}) \le p(x_{l(k)})$, we obtain

$$p(x_{l(k+1)}) \le \max\{p(x_{l(k)}), p(x_{k+1})\} = p(x_{l(k)}).$$

This means that the sequence $\{p(x_{l(k)})\}$ is decreasing for all k. Then $\{p(x_{l(k)})\}$ is convergent. Based on Assumptions A and B, similar to Lemma 3.4 in [32], it is not difficult to deduce that there exist constants $b_1 \ge b_2 > 0$ such that

$$b_2\|d_k\|^2 \le d_k^T B_k d_k = -e_k^T d_k \le b_1\|d_k\|^2. \tag{3.10}$$

By (1.13) and (3.10), for all $k > M$, we get

$$\begin{aligned} p(x_{l(k)}) &= p(x_{l(k)-1} + \alpha_{l(k)-1} d_{l(k)-1}) \\ &\le \max_{0\le j\le m(l(k)-1)}\{p(x_{l(k)-j-1})\} + \sigma \alpha_{l(k)-1} g_{l(k)-1}^T d_{l(k)-1} \\ &\le \max_{0\le j\le m(l(k)-1)}\{p(x_{l(k)-j-1})\} - \sigma b_2 \alpha_{l(k)-1}\|d_{l(k)-1}\|^2. \end{aligned} \tag{3.11}$$

Since $\{p(x_{l(k)})\}$ is convergent, from the above inequality, we have

$$\lim_{k\to\infty} \alpha_{l(k)-1}\|d_{l(k)-1}\|^2 = 0.$$

This implies that either

$$\lim_{k\to\infty} \inf d_{l(k)-1} = 0 \tag{3.12}$$

or

$$\lim_{k\to\infty} \inf \alpha_{l(k)-1} = 0. \tag{3.13}$$

If (3.12) holds, following [40], by induction we can prove that

$$\lim_{k\to\infty} \|d_{l(k)-j}\| = 0 \tag{3.14}$$

and

$$\lim_{k\to\infty} p(x_{l(k)-j}) = \lim_{k\to\infty} p(x_{l(k)})$$

for any positive integer j. As $k \geq l(k) \geq k - M$ and M is a positive constant, by

$$x_k = x_{k-M-1} + \alpha_{k-M-1}d_{k-M-1} + \cdots + \alpha_{l(k)-1}d_{l(k)-1}$$

and (3.14), it can be derived that

$$\lim_{k\to\infty} p(x_{l(k)}) = \lim_{k\to\infty} p(x_k). \tag{3.15}$$

According to (3.10) and the rule for accepting the step $\alpha_k d_k$,

$$p(x_{k+1}) - p(x_{l(k)}) \leq \alpha_k \sigma e_k^T d_k \leq \alpha_k \sigma b_2 \|d_k\|^2. \tag{3.16}$$

This means

$$\lim_{k\to\infty} \alpha_k \|d_k\|^2 = 0,$$

which implies that

$$\lim_{k\to\infty} \alpha_k = 0 \tag{3.17}$$

or

$$\lim_{k\to\infty} \|d_k\| = 0. \tag{3.18}$$

If equation (3.18) holds, since B_k is bounded, then $\|e_k\| = \|B_k d_k\| \leq \|B_k\| \|d_k\| \to 0$ holds. The conclusion of this lemma holds. If (3.17) holds. Then acceptance rule (1.13) means that, for all large enough k, $\alpha_k' = \frac{\alpha_k}{r}$ such that

$$p(x_k + \alpha_k' d_k) - p(x_k) \geq p(x_k + \alpha_k' d_k) - p(x_{l(k)}) > \sigma \alpha_k' e_k^T d_k. \tag{3.19}$$

Since

$$p(x_k + \alpha_k' d_k) - p(x_k) = \alpha_k' \nabla p(x_k)^T d_k + o(\alpha_k' \|d_k\|). \tag{3.20}$$

Using (3.19) and (3.20) in [32], we have

$$\nabla p(x_k)^T d_k = e_k^T \nabla e(x_k) d_k \leq \delta^* e_k^T d_k,$$

where $\delta^* > 0$ is a constant and $\sigma < \delta^*$. So we get

$$[\delta^* - \sigma] \alpha_k' e_k^T d_k + o(\alpha_k' \|d_k\|) \geq 0. \tag{3.21}$$

Note that $\delta^* - \sigma > 0$ and $e_k^T d_k < 0$, we have from dividing (3.21) by $\alpha_k' \|d_k\|$

$$\lim_{k \to \infty} \frac{e_k^T d_k}{\|d_k\|} = 0. \tag{3.22}$$

By (3.10), we have

$$\lim_{k \to \infty} \|d_k\| = 0. \tag{3.23}$$

Consider $\|e_k\| = \|B_k d_k\| \leq \|B_k\| \|d_k\|$ and the bounded B_k again, we complete the proof. □

Lemma 3.3 (see Lemma 4.1 in [31]) *Let e be continuously differentiable, and* $\nabla e(x)$ *be nonsingular at* x^* *which satisfies* $e(x^*) = 0$. *Let*

$$a \equiv \left\{ \|\nabla e(x^*)\| + \frac{1}{2c}, 2c \right\}, c = \|\nabla e(x^*)^{-1}\|. \tag{3.24}$$

If $\|x_k - x^*\|$ *sufficiently small, then the inequality*

$$\frac{1}{a} \|x_k - x^*\| \leq \|e(x_k)\| \leq a \|x_k - x^*\| \tag{3.25}$$

holds.

Theorem 3.2 *Let the assumptions in Lemma* 3.3 *hold. Assume that there exists a sufficiently small* $\varepsilon_0 > 0$ *such that* $\|B_k - \nabla e(x_k)\| \leq \varepsilon_0$ *for each k. Then the sequence* $\{x_k\}$ *converges to* x^* *superlinearly for* $\alpha_k = 1$. *Moreover, if e is q-order smooth at* x^* *and there is a neighborhood U of* x^* *satisfying for any* $x_k \in U$,

$$\|[B_k - \nabla e(x^*)](x_k - x^*)\| \leq \eta \|x_k - x^*\|^{1+q}, \tag{3.26}$$

then $x_k \to x^*$ *with order at least* $1 + q$, *where* η *is a constant.*

Proof Since g is continuously differentiable and $\nabla e(x)$ is nonsingular at x^*, there exists a constant $\gamma > 0$ and a neighborhood U of x^* satisfying

$$\max\{\|\nabla e(y)\|, \|\nabla e(y)^{-1}\|\} \leq \gamma,$$

where $\nabla e(y)$ is nonsingular for any $y \in U$. Consider the following equality when $\alpha_k = 1$:

$$\begin{aligned} &B_k(x_{k+1} - x^*) + [\nabla e(x_k)(x_k - x^*) - B_k(x_k - x^*)] \\ &\quad + [e(x_k) - e(x^*) - \nabla e(x_k)(x_k - x^*)] \\ &= e(x_k) + B_k d_k = 0, \end{aligned} \tag{3.27}$$

the second term and the third term are $o(\|x_k - x^*\|)$. By the von Neumann lemma, and considering that $\nabla e(x_k)$ is nonsingular, B_k is also nonsingular. For any $y \in U$ and $\nabla e(y)$ being nonsingular and $\max\{\|\nabla e(y)\|, \|\nabla e(y)^{-1}\|\} \leq \gamma$, then we obtain from Lemma 3.3

$$\|x_{k+1} - x^*\| = o(\|x_k - x^*\|) = o(\|e(x_k)\|), \quad \text{as } k \to \infty,$$

this means that the sequence $\{x_k\}$ converges to x^* superlinearly for $\alpha_k = 1$.

If e is q-order smooth at x^*, then we get

$$e(x_k) - e(x^*) - \nabla e(x_k)(x_k - x^*) = O(\|x_k - x^*\|^{q+1}).$$

Consider the second term of (3.27) as $x_k \to x^*$, and use (3.26), we can deduce that the second term of (3.27) is also $O(\|x_k - x^*\|^{q+1})$. Therefore, we have

$$\|x_{k+1} - x^*\| = O(\|x_k - x^*\|^{q+1}), \quad \text{as } x_k \to x^*.$$

The proof is complete. □

4 Numerical results

In this section, we report results of some numerical experiments with the proposed method. The test functions have the following form:

$$e(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T,$$

where these functions have the associated initial guess x_0. These functions are stated as follows.

Function 1 Exponential function 2

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

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

Function 2 Trigonometric function

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

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

Function 3 Logarithmic function

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

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

Function 4 Broyden tridiagonal function [[51], pp. 471-472]

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

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

Function 5 Trigexp function [[51], p. 473]

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

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

Function 6 Strictly convex function 1 [[52], p. 29]. $e(x)$ is the gradient of $h(x) = \sum_{i=1}^{n}(e^{x_i} - x_i)$.

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

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

Function 7 Strictly convex function 2 [[52], p. 30]
$e(x)$ is the gradient of $h(x) = \sum_{i=1}^{n} \frac{i}{10}(e^{x_i} - x_i)$.

$$f_i(x) = \frac{i}{10}\big(e^{x_i} - 1\big), \quad i = 1,2,3,\ldots,n.$$

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

Function 8 Variable dimensioned function

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

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

Function 9 Discrete boundary value problem [53].

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

Initial guess: $x_0 = (h(h-1), h(2h-1), \ldots, h(nh-1))$.

Function 10 The discretized two-point boundary value problem similar to the problem in [53]

$$
e(x) = Ax + \frac{1}{(n+1)^2} F(x) = 0,
$$

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

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

and $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))^T$ with $F_i(x) = \sin x_i - 1, i = 1,2,\ldots,n$, and $x_0 = (50, 0, 50, 0, \ldots)$.

In the experiments, all codes were written in MATLAB r2009a and run on a PC with G1620T@2.40 GHz CPU processor and 4.0 GB memory and Windows XP operation system. In order to compare the performance the given algorithm with CG's initial points (called new method with CG), we also do the experiment with only the main algorithm with initial points x_0 (called the normal method). Aslam Noor *et al.* [54] presented a variational iteration technique for nonlinear equations, where the so-called VIM1 method has the better numerical performance. The VIM1 method has the following iteration form:

$$x_{k+1} = x_k - \left[\nabla e - \mathbf{diag}(\beta_1 e_1, \beta_2 e_2, \ldots, \beta_n e_n)\right]^{-1}(x_k)e(x_k),$$

where $\beta_i \in (0,1)$ for $i = 1,2,\ldots,n$. In their paper, only low dimension problems (two variables) are tested. In this experiment, we also give the numerical results of this method for large-scale nonlinear equations to compare with our proposed algorithm.

The parameters were chosen as $r = 0.1$, $\sigma = 0.9, M = 12, \epsilon = 10^{-4}$, and $\epsilon_{\text{main}} = 10^{-5}$. In order to ensure the positive definiteness of B_k, in Step 4 of the main algorithm: if $y_k^T s_k > 0$, update B_k by (1.11), otherwise let $B_{k+1} = B_k$. This program will also be stopped if the iteration number of main algorithm is larger than 200. Since the line search cannot always ensure these descent conditions $d_k^T e_k < 0$ and $d_k^T \nabla e(x_k) e_k < 0$, an uphill search direction may occur in numerical experiments. In this case, the line search rule maybe fails. In order to avoid this case, the stepsize α_k will be accepted if the searching time is larger than six in the inner circle for the test problems.

In the sub-algorithm, the CG formula is used by the following Polak-Ribière-Polyak (PRP) method [55, 56]

$$d_k = \begin{cases} -e_k + \frac{e_k^T(e_k - e_{k-1})}{\|e_{k-1}\|^2} d_{k-1} & \text{if } k \geq 1, \\ -e_k & \text{if } k = 0. \end{cases} \tag{4.1}$$

For the line search technique, (1.3) is used and the largest search number of times is ten, where $\delta_1 = \delta_2 = 10^{-7}$, and $\epsilon_k = \frac{1}{NI^2}$ (*NI* is the iteration number). The sub-algorithm will also stopped if the iteration number is larger than 150. The iteration number, the function evaluations, and the CPU time of the sub-algorithm are added to the main algorithm for new method with CG. The meaning of the items of the columns of Table 1 is:

Dim: the dimension.

NI: the number of iterations.

NG: the number of function evaluations.

cpu time: the cpu time in seconds.

GF: the final norm function evaluations $p(x)$ when the program is stopped.

GD: the final norm evaluations of search direction d_k.

fails: fails to find the final values of $p(x)$ when the program is stopped.

From Tables 1-2, it is easy to see that the number of iterations and the number of function evaluations of the new method with CG are less than those of the normal method for these test problems. Moreover, the cpu time and the final function norm evaluations of the new method with CG are more competitive than those of the normal method. For the VIM1 method, the results of Problems 1-7 are very interesting, but it fails for Problems 8-10. Moreover, it is not difficult to find that more CUP time is needed for this method. The main reason maybe lies in the computation of the Jacobian matrix at every iteration.

The tool of Dolan and Moré [57] is used to analyze the efficiency of these three algorithms.

Figures 1-3 show that the performance of these methods are relative to NI, NG, and cpu time of Tables 1-2, respectively. The numerical results indicate that the proposed method

Table 1 Numerical results

		New method with CG				Normal method (only main algorithm)			
P	Dim	NI/NG	GF	GD	cpu time	NI/NG	GF	GD	cpu time
1	1000	1/1	6.676674e–006	1.335335e–005	0.000000e+000	0/2	6.676674e–006	1.335335e–005	0.000000e+000
	2000	1/1	3.335834e–006	6.671668e–006	0.000000e+000	0/2	3.335834e–006	6.671668e–006	0.000000e+000
	3000	1/1	2.223334e–006	4.446667e–006	1.560010e–002	0/2	2.223334e–006	4.446667e–006	3.120020e–002
2	1000	12/17	1.570352e–007	1.214954e–007	1.544410e+000	199/2879	1.624268e–004	1.228551e+000	1.338801e+002
	2000	200/2927	8.144022e–005	3.138945e–001	8.647135e+002	199/2928	8.144022e–005	3.138945e–001	8.680832e+002
	3000	200/2326	5.434381e–005	2.481121e–003	1.614626e+003	199/2327	5.434381e–005	2.481121e–003	1.622785e+003
3	1000	8/8	4.194859e–006	8.389718e–006	1.560010e–002	115/1009	5.838535e–008	1.171251e–007	7.996611e+001
	2000	8/8	7.775106e–006	1.555021e–005	7.800050e–002	117/1040	1.161670e–007	2.328056e–007	5.662368e+002
	3000	9/9	1.614597e–012	3.231630e–012	1.553770e+001	137/1362	1.739498e–007	3.484891e–007	2.141410e+003
4	1000	92/165	5.632576e–006	5.669442e–006	2.215214e+000	199/285	3.703283e+001	1.196051e+001	1.356897e+002
	2000	87/156	6.245922e–006	6.085043e–006	1.502290e+001	199/230	3.637504e+000	9.570779e+000	9.591097e+002
	3000	94/169	6.678153e–006	6.437585e–006	4.731510e+001	199/234	2.639260e+002	1.904865e+001	3.096277e+003
5	1000	22/51	8.288299e–006	6.268946e–007	2.106014e+000	199/2570	3.195300e+004	4.649782e+006	1.779971e+001
	2000	21/50	4.114462e–006	1.943351e–006	5.179233e+000	199/2652	6.395300e+004	1.354972e+005	8.327333e+001
	3000	21/51	9.843373e–006	8.504719e–006	1.597450e+001	199/2853	9.595300e+004	1.135356e+005	1.314776e+002
6	1000	9/11	5.984185e–012	1.197596e–011	7.176046e–001	6/9	6.069722e–007	1.118437e–006	4.149627e+000
	2000	9/11	1.505191e–006	3.010383e–006	7.800050e–002	6/9	1.210931e–006	2.231765e–006	2.898499e+001
	3000	9/11	2.251571e–006	4.503142e–006	1.404009e–001	6/9	1.814891e–006	3.345093e–006	9.300780e+001
7	1000	200/602	1.208240e–003	4.697005e+000	3.424222e+001	199/573	3.156137e+005	3.893380e+004	1.378113e+002
	2000	200/760	1.612671e+001	9.034319e+000	2.420200e+002	199/644	1.014481e+006	3.131576e+005	9.872367e+002
	3000	200/693	5.570227e–003	9.501149e+001	7.698181e+002	199/743	9.357473e+007	2.087488e+007	3.087119e+003
8	1000	2/2	0.000000e+000	0.000000e+000	0.000000e+000	1/3	0.000000e+000	0.000000e+000	6.552042e–001
	2000	2/2	0.000000e+000	0.000000e+000	0.000000e+000	1/3	0.000000e+000	0.000000e+000	4.820431e+000
	3000	2/2	0.000000e+000	0.000000e+000	6.240040e–002	1/3	0.000000e+000	0.000000e+000	1.538170e+001
9	1000	67/118	7.138941e–006	1.820053e–005	3.010819e+000	2/5	2.358640e–006	4.611203e–006	1.404009e+000
	2000	70/124	6.342724e–006	1.607326e–005	2.062333e+001	2/5	5.917002e–007	1.169969e–006	9.703262e+000
	3000	74/131	7.447187e–006	1.799920e–005	6.450641e+001	2/5	2.632811e–007	5.225655e–007	3.084140e+001
10	1000	26/49	2.044717e–008	3.900140e–008	2.359983e+002	121/125	7.382123e–006	1.467673e–005	4.987196e+002
	2000	24/47	9.030382e–006	2.717060e–006	1.847286e+003	121/125	7.454090e–006	1.481981e–005	3.852538e+003
	3000	27/51	6.468831e–009	1.138377e–008	6.632227e+003	121/125	7.523322e–006	1.495745e–005	1.299774e+004

Table 2 Numerical results of VIM1 method

P	Dim	NI/NG	GF	cpu time	P	Dim	NI/NG	GF	cpu time
1	1000	1/1	6.676674e–006	1.560010e–002	6	1000	5/5	4.591162e–011	9.656462e+000
	2000	1/1	3.335834e–006	0.000000e+000		2000	5/5	9.140464e–011	7.439688e+001
	3000	1/1	2.223334e–006	3.120020e–002		3000	5/5	1.368978e–010	2.484628e+002
2	1000	18/18	2.840705e–007	5.494355e+001	7	1000	5/5	4.058902e–006	9.656462e+000
	2000	27/27	2.532474e–006	6.315544e+002		2000	6/6	1.983880e–017	8.993458e+001
	3000	22/22	9.781547e–007	1.669476e+003		3000	6/6	6.708054e–017	3.007543e+002
3	1000	5/5	5.430592e–007	9.578461e+000	8	1000	fails		
	2000	5/5	5.619751e–007	7.435008e+001		2000	fails		
	3000	5/5	5.870798e–007	2.484160e+002		3000	fails		
4	1000	4/4	4.559227e–009	1.243328e+001	9	1000	fails		
	2000	4/4	9.082664e–009	1.026487e+002		2000	fails		
	3000	4/4	1.360090e–008	3.708768e+002		3000	fails		
5	1000	9/9	2.648764e–006	3.196460e+001	10	1000	fails		
	2000	9/9	2.649263e–006	2.529244e+002		2000	fails		
	3000	9/9	2.649430e–006	8.258849e+002		3000	fails		

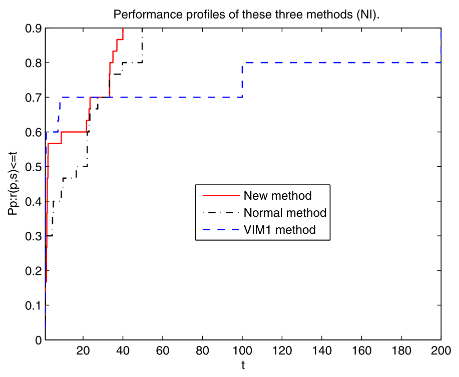

Figure 1 Performance profiles of these three methods (NI).

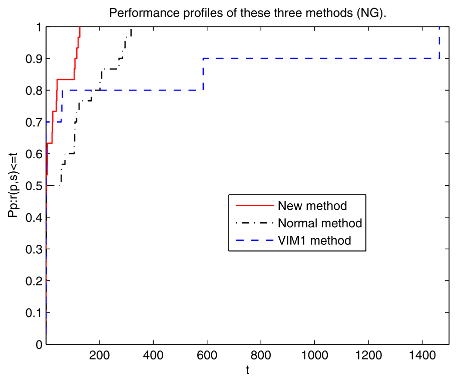

Figure 2 Performance profiles of these three methods (NG).

performs best among these three methods. To this end, we think that the enhancement of this proposed method is noticeable.

5 Conclusion

In this paper, we focus on two algorithms solved a class of large-scale nonlinear equations. At the first step, a CG algorithm, called a sub-algorithm, was used as the initial points of the main algorithm. Then a quasi-Newton algorithm with the initial points done by a CG sub-algorithm was defined as the main algorithm. In order to avoid computing the

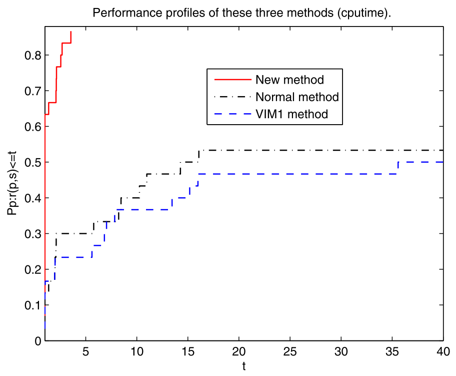

Figure 3 Performance profiles of these three methods (cpu time).

Jacobian matrix, a nonmonotone line search technique was used in the algorithms. The convergence results are established and numerical results are reported.

According to the numerical performance, it is clear that the CG technique is very effective for large-scale nonlinear equations. This observation inspires us to design the CG methods to directly solve nonlinear equations in the future.

Competing interests
The author declares to have no competing interests.

Acknowledgements
Only the author contributed in writing this paper. The author thanks the referees and the Editor for their valuable comments, which greatly improved the paper.

Received: 24 November 2016 Accepted: 18 January 2017 Published online: 03 February 2017

References

1. Chen, B, Shu, H, Coatrieux, G, Chen, G, Sun, X, Coatrieux, J: Color image analysis by quaternion-type moments. J. Math. Imaging Vis. **51**, 124-144 (2015)
2. Fu, Z, Ren, K, Shu, J, Sun, X, Huang, F: Enabling personalized search over encrypted outsourced data with efficiency improvement. IEEE Trans. Parallel Distrib. Syst. (2015). doi:10.1109/TPDS.2015.2506573
3. Gu, B, Sheng, VS: A robust regularization path algorithm for ν-support vector classification. IEEE Trans. Neural Netw. Learn. Syst. (2016). doi:10.1109/TNNLS.2016.2527796
4. Gu, B, Sheng, VS, Tay, KY, Romano, W, Li, S: Incremental support vector learning for ordinal regression. IEEE Trans. Neural Netw. Learn. Syst. **26**, 1403-1416 (2015)
5. Guo, P, Wang, J, Li, B, Lee, S: A variable threshold-value authentication architecture for wireless mesh networks. J. Internet Technol. **15**, 929-936 (2014)
6. Li, J, Li, X, Yang, B, Sun, X: Segmentation-based image copy-move forgery detection scheme. IEEE Trans. Inf. Forensics Secur. **10**, 507-518 (2015)
7. Pan, Z, Zhang, Y, Kwong, S: Efficient motion and disparity estimation optimization for low complexity multiview video coding. IEEE Trans. Broadcast. **61**, 166-176 (2015)
8. Shen, J, Tan, H, Wang, J, Wang, J, Lee, S: A novel routing protocol providing good transmission reliability in underwater sensor networks. J. Internet Technol. **16**, 171-178 (2015)
9. Xia, Z, Wang, X, Sun, X, Wang, Q: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **27**, 340-352 (2015)
10. Fu, Z, Wu, X, Guan, C, Sun, X, Ren, K: Towards efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans. Inf. Forensics Secur. (2016). doi:10.1109/TIFS.2016.2596138
11. Gu, B, Sun, X, Sheng, VS: Structural minimax probability machine. IEEE Trans. Neural Netw. Learn. Syst. (2016). doi:10.1109/TNNLS.2016.2544779
12. Ma, T, Zhou, J, Tang, M, Tian, Y, Al-Dhelaan, A, Al-Rodhaan, M, Lee, S: Social network and tag sources based augmenting collaborative recommender system. IEICE Trans. Inf. Syst. **98**, 902-910 (2015)
13. Ren, Y, Shen, J, Wang, JN, Han, J, Lee, S: Mutual verifiable provable data auditing in public cloud storage. J. Internet Technol. **16**, 317-323 (2015)
14. Yuan, G: Modified nonlinear conjugate gradient methods with sufficient descent property for large-scale optimization problems. Optim. Lett. **3**, 11-21 (2009)
15. Yuan, G, Duan, X, Liu, W, Wang, X, et al.: Two new PRP conjugate gradient algorithms for minimization optimization models. PLoS ONE **10**, e0140071 (2015)
16. Yuan, G, Lu, X: A modified PRP conjugate gradient method. Ann. Oper. Res. **166**, 73-90 (2009)
17. Yuan, G, Lu, X, Wei, Z: A conjugate gradient method with descent direction for unconstrained optimization. J. Comput. Appl. Math. **233**, 519-530 (2009)

18. Yuan, G, Wei, Z: New line search methods for unconstrained optimization. J. Korean Stat. Soc. **38**, 29-39 (2009)
19. Yuan, G, Wei, Z: The superlinear convergence analysis of a nonmonotone BFGS algorithm on convex objective functions. Acta Math. Sin. Engl. Ser. **24**(1), 35-42 (2008)
20. Yuan, G, Wei, Z: Convergence analysis of a modified BFGS method on convex minimizations. Comput. Optim. Appl. **47**, 237-255 (2010)
21. Yuan, G, Wei, Z: A trust region algorithm with conjugate gradient technique for optimization problems. Numer. Funct. Anal. Optim. **32**, 212-232 (2011)
22. Yuan, G, Wei, Z: The Barzilai and Borwein gradient method with nonmonotone line search for nonsmooth convex optimization problems. Math. Model. Anal. **17**, 203-216 (2012)
23. Yuan, G, Wei, Z, Wang, Z: Gradient trust region algorithm with limited memory BFGS update for nonsmooth convex minimization. Comput. Optim. Appl. **54**, 45-64 (2013)
24. Yuan, G, Wei, Z, Wu, Y: Modified limited memory BFGS method with nonmonotone line search for unconstrained optimization. J. Korean Math. Soc. **47**, 767-788 (2010)
25. Yuan, G, Wei, Z, Zhao, Q: A modified Polak-Ribière-Polyak conjugate gradient algorithm for large-scale optimization problems. IIE Trans. **46**, 397-413 (2014)
26. Yuan, G, Zhang, M: A modified Hestenes-Stiefel conjugate gradient algorithm for large-scale optimization. Numer. Funct. Anal. Optim. **34**, 914-937 (2013)
27. Zhang, Y, Sun, X, Baowei, W: Efficient algorithm for K-barrier coverage based on integer linear programming. China Communications **13**, 16-23 (2016)
28. Li, D, Fukushima, M: A global and superlinear convergent Gauss-Newton-based BFGS method for symmetric nonlinear equations. SIAM J. Numer. Anal. **37**, 152-172 (1999)
29. Gu, G, Li, D, Qi, L, Zhou, S: Descent directions of quasi-Newton methods for symmetric nonlinear equations. SIAM J. Numer. Anal. **40**, 1763-1774 (2002)
30. Brown, PN, Saad, Y: Convergence theory of nonlinear Newton-Krylov algorithms. SIAM J. Optim. **4**, 297-330 (1994)
31. Zhu, D: Nonmonotone backtracking inexact quasi-Newton algorithms for solving smooth nonlinear equations. Appl. Math. Comput. **161**, 875-895 (2005)
32. Yuan, G, Lu, X: A new backtracking inexact BFGS method for symmetric nonlinear equations. Comput. Math. Appl. **55**, 116-129 (2008)
33. Nash, SG: A surey of truncated-Newton matrices. J. Comput. Appl. Math. **124**, 45-59 (2000)
34. Dembao, RS, Eisenstat, SC, Steinaug, T: Inexact Newton methods. SIAM J. Numer. Anal. **19**, 400-408 (1982)
35. Griewank, A: The 'global' convergence of Broyden-like methods with a suitable line search. J. Aust. Math. Soc. Ser. B, Appl. Math **28**, 75-92 (1986)
36. Ypma, T: Local convergence of inexact Newton methods. SIAM J. Numer. Anal. **21**, 583-590 (1984)
37. Yuan, G, Wei, Z, Lu, X: A BFGS trust-region method for nonlinear equations. Computing **92**, 317-333 (2011)
38. Yuan, G, Wei, Z, Lu, S: Limited memory BFGS method with backtracking for symmetric nonlinear equations. Math. Comput. Model. **54**, 367-377 (2011)
39. Yuan, G, Yao, S: A BFGS algorithm for solving symmetric nonlinear equations. Optimization **62**, 82-95 (2013)
40. Grippo, L, Lampariello, F, Lucidi, S: A nonmonotone line search technique for Newton's method. SIAM J. Numer. Anal. **23**, 707-716 (1986)
41. Birgin, EG, Martinez, JM, Raydan, M: Nonmonotone spectral projected gradient methods on convex sets. SIAM J. Optim. **10**, 1196-1211 (2000)
42. Han, J, Liu, G: Global convergence analysis of a new nonmonotone BFGS algorithm on convex objective functions. Comput. Optim. Appl. **7**, 277-289 (1997)
43. Liu, G, Peng, J: The convergence properties of a nonmonotonic algorithm. J. Comput. Math. **1**, 65-71 (1992)
44. Zhou, J, Tits, A: Nonmonotone line search for minimax problem. J. Optim. Theory Appl. **76**, 455-476 (1993)
45. Yuan, G: A new method with descent property for symmetric nonlinear equations. Numer. Funct. Anal. Optim. **31**, 974-987 (2010)
46. Yuan, G, Meng, Z, Li, Y: A modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimizations and nonlinear equations. J. Optim. Theory Appl. **168**, 129-152 (2016)
47. Yuan, G, Lu, S, Wei, Z: A new trust-region method with line search for solving symmetric nonlinear equations. Int. J. Comput. Math. **88**, 2109-2123 (2011)
48. Yuan, G, Wei, Z, Li, G: A modified Polak-Ribière-Polyak conjugate gradient algorithm for nonsmooth convex programs. J. Comput. Appl. Math. **255**, 86-96 (2014)
49. Yuan, G, Zhang, M: A three-terms Polak-Ribière-Polyak conjugate gradient algorithm for large-scale nonlinear equations. J. Comput. Appl. Math. **286**, 186-195 (2015)
50. Yuan, G, Lu, X, Wei, Z: BFGS trust-region method for symmetric nonlinear equations. J. Comput. Appl. Math. **230**, 44-58 (2009)
51. Gomez-Ruggiero, M, Martinez, J, Moretti, A: Comparing algorithms for solving sparse nonlinear systems of equations. SIAM J. Sci. Comput. **23**, 459-483 (1992)
52. Raydan, M: The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. SIAM J. Optim. **7**, 26-33 (1997)
53. Moré, J, Garbow, B, Hillström, K: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**, 17-41 (1981)
54. Aslam Noor, M, Waseem, M, Inayat Noor, K, Al-Said, E: Variational iteration technique for solving a system of nonlinear equations. Optim. Lett. **7**, 991-1007 (2013)
55. Polak, E, Ribière, G: Note sur la convergence de directions conjugees. Rev. Franaise Informat. Recherche Opérationnelle **3**, 35-43 (1969)
56. Polyak, E: The conjugate gradient method in extremal problems. USSR Comput. Math. Math. Phys. **9**, 94-112 (1969)
57. Dolan, ED, Moré, JJ: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201-213 (2002)