PLOS | ONE

RESEARCH ARTICLE

A Limited-Memory BFGS Algorithm Based on a Trust-Region Quadratic Model for Large-Scale Nonlinear Equations

Yong Li[2], Gonglin Yuan[1]*, Zengxin Wei[1]

1 College of Mathematics and Information Science, Guangxi University, Nanning, Guangxi, P.R. China,
2 Department of Mathematics, Baise University, Baise, Guangxi, P. R. China

* glyuan@gxu.edu.cn

Citation: Li Y, Yuan G, Wei Z (2015) A Limited-Memory BFGS Algorithm Based on a Trust-Region Quadratic Model for Large-Scale Nonlinear Equations. PLoS ONE 10(5): e0120993. doi:10.1371/journal.pone.0120993

Academic Editor: Zi-Ke Zhang, Hangzhou Normal University, CHINA

Received: November 6, 2014

Accepted: February 9, 2015

Published: May 7, 2015

Data Availability Statement: All data are available and they are listed in the paper.

Funding: Program for Excellent Talents in Guangxi Higher Education Institutions (Grant No. 201261), Guangxi Higher Education Institutions (NO. YB2014389), Guangxi NSF (Grant No. 2012GXNSFAA053002) and China NSF (Grant No. 11261006 and 11161003).

Competing Interests: The authors have declared that no competing interests exist.

Abstract

In this paper, a trust-region algorithm is proposed for large-scale nonlinear equations, where the limited-memory BFGS (L-M-BFGS) update matrix is used in the trust-region subproblem to improve the effectiveness of the algorithm for large-scale problems. The global convergence of the presented method is established under suitable conditions. The numerical results of the test problems show that the method is competitive with the norm method.

Introduction

Consider

$$b(x) = 0, \ x \in \Re^n, \tag{1}$$

where $b{:}\Re^n \to \Re^n$ is a system of nonlinear equations and n denotes the large-scale dimension. Problems involving non-linear equations can be found in numerous applications in engineering, such as nonlinear fitting, function approximating, parameter estimating and leaning models [1–5]. Large-scale nonlinear equations are believed to be difficult to solve because the relations with x are complex and because the dimension is high. Currently, many effective algorithms exist for addressing (1), such as the trust-region method [6–11], Levenberg-Marquardt method [12–14], quasi-Newton method [15–19], and Gauss-Newton method [20–25]. Let θ be the norm function defined by $\theta(x) = \frac{1}{2}\|b(x)\|^2$, and suppose that $b(x)$ has a zero point. Then, (1) is equivalent to the following global optimization problem:

$$\min \theta(x), \ x \in \Re^n. \tag{2}$$

In this paper, we will focus on the solution of (2) via the trust-region (TR) methods. The traditional TR methods, at each iterative point x_k, obtain the trial step d_k using the following TR subproblem model:

$$\min \ p_k(d) = \frac{1}{2}\| \ b(x_k) + \nabla b(x_k)d \ \|^2 \ \textit{such that} \ \ \| \ d \ \| \leq \Delta, \tag{3}$$

where $\triangle > 0$ is the so-called TR radius and $\|\cdot\|$ denotes the Euclidean norm of vectors or their induced matrix. The TR method is very useful when the exact Jacobian or Hessian computation is inexpensive or possible. However, this case is rare in practice. It is not difficult to see that the Jacobian matrix $\nabla b(x)$ at every iteration must be computed, which obviously increases the workload and CPU time. Moreover, the global and superlinear convergence of the TR methods often requires that the nondegenerate assumption about $\nabla b(x^*)$ holds, where x^* is a solution of (1) and nondegeneracy means nonsingularity. The nondegeneracy of $\nabla b(x^*)$ seems to be too stringent of a requirement for the purpose of ensuring the global and superlinear convergence. To overcome the above drawbacks, Yuan et al. [9] presented a new TR subproblem model defined by

$$\min \ q_k(d) = \frac{1}{2}\| b(x_k) + B_k d \|^2 \ such \ that \ \ \| d \| \leq \Delta_k, \tag{4}$$

where the TR radius $\triangle_k = c^p \|b(x_k)\|$, $c \in (0, 1)$, p is a nonnegative integer and B_k is generated by the following BFGS formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k {y_k}^T}{{y_k}^T s_k}, \tag{5}$$

where $s_k = x_{k+1} - x_k$, $y_k = b(x_{k+1}) - b(x_k)$, x_{k+1} is the next iteration and B_0 is an initial symmetric, positive definite matrix. They established the global and superlinear convergence without nondegeneracy. The numerical results show that the given algorithm is competitive with the normal TR method up to a dimension of 600. The subproblem (4) can be also rewritten as follows

$$\begin{aligned} \min_{d\in\Re^n} q_k(d) &= \frac{1}{2}\| b(x_k) + \nabla b(x_k) d \|^2 \\ &= \frac{1}{2}\| b(x_k) \|^2 + b(x_k)^T \nabla b(x_k) d + \frac{1}{2} d^T \nabla b(x_k)^T \nabla b(x_k) d \\ s.t. &\quad \| d \| \leq \Delta_k. \end{aligned}$$

Yuan et al. [8] consider the case with the symmetric Jacobian matrix $\nabla b(x_k)$ and propose the following TR subproblem model:

$$\begin{aligned} \min_{d\in\Re^n} q_k(d) &= \frac{1}{2}\| b(x_k) + \nabla b(x_k) d \|^2 = \frac{1}{2}\| b(x_k) \|^2 + b(x_k)^T B_k d + \frac{1}{2} d^T B'_k d \\ s.t. &\quad \| d \| \leq \Delta_k, \end{aligned}$$

where B'_k is defined by the special BFGS update:

$$B'_{k+1} = B'_k - \frac{B'_k s_k s_k^T B'_k}{s_k^T B'_k s_k} + \frac{\delta_k {\delta_k}^T}{{\delta_k}^T s_k}, \tag{6}$$

where $\delta_k = b(x_k + y_k) - b(x_k)$. About the above TR methods, the TR subproblems models will be repeatedly computed if the calculated trial step d_k does not satisfy the given conditions, which obviously increase the workload. To avoid this drawback, Yuan et al. [10] give the following TR model:

$$\begin{aligned} \min_{d\in\Re^n} q_k^m(d) &= b(x_k)^T \nabla b(x_k) d + \frac{1}{2} d^T \nabla b(x_k)^T \nabla b(x_k) d \\ s.t. &\quad \| \nabla b(x_k) d \| \leq \Delta_k^m, \end{aligned}$$

where $\triangle_k^m = \max\{\|b(x_k)\|, \triangle_k\}$. In this model, the next point is $x_{k+1} = x_k + d_k$ if the trial step

d_k satisfies the successful iteration; otherwise, the next point is $x_{k+1} = x_k + \alpha_k d_k$, where $\alpha_k > 0$ is the so-called steplength designed by

$$\| b(x_k + \alpha_k d_k) \|^2 \leq \| b(x_k) \|^2 + \delta \alpha_k^2 b(x_k)^T d_k, \tag{7}$$

where $\delta \in (0, 1)$ and the technique was proposed by Yuan and Lu [16]. However, the above model can not ensure that the number of searching $\alpha_k > 0$ is small. Many TR model methods (see [7, 12] etc.) have been developed for solving nonlinear equations. However, these TR methods require the matrix information (the Jacobian matrix or the BFGS update matrix), which will increase the computational complexity. This observation motivate us find another approach to generate the matrix information requiring minimal memory.

Based on the problem (4), we will use the limited-memory BFGS (L-M-BFGS) method instead of the BFGS update because the former often provides a fast rate of linear convergence and requires minimal memory. The L-BFGS method is an adaptation of the standard BFGS method [26]. The implementation is identical to that of the normal BFGS method, the difference between the L-M-BFGS method and the BFGS method is that the inverse Hessian approximation is not explicitly formed, but defined by a small number of BFGS updates. At every iteration x_k, the L-M-BFGS method stores a small number (say, m) of correction pairs $\{s_i, y_i\}(i = k-1, \ldots, k-m)$ to obtain H_{k+1}, instead of storing the matrices H_k, where

$$s_k = x_{k+1} - x_k, \quad y_k = b(x_{k+1}) - b(x_k),$$

with H_k being the inverse of B_k. The L-M-BFGS update formula is defined as

$$\begin{aligned} H_{k+1} &= V_k^T[V_{k-1}^T H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T] V_k + \rho_k s_k s_k^T \\ &= V_k^T V_{k-1}^T H_{k-1} V_{k-1} + V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T V_k + \rho_k s_k s_k^T \\ &= \cdots \\ &= [V_k^T \cdots V_{k-m+1}^T] H_{k-m+1} [V_{k-m+1} \cdots V_k] + \rho_{k+m-1} [V_{k-1}^T \\ &\cdots \quad V_{k-m+2}^T] s_{k-m+1} s_{k-m+1}^T [V_{k-m+2} \ldots V_{k-1}] + \cdots + \rho_k s_k s_k^T, \end{aligned} \tag{8}$$

where $\rho_k = \frac{1}{y_k^T s_k}$ and $V_k = I - \rho_k y_k s_k^T$. It has been proved that the L-M-BFGS method is very suitable for large-scale systems of nonlinear equations [17]. Therefore, combining the L-M-BFGS update (8) and (4), we design the L-M-BFGS TR subproblem model as follows

$$\min \; q_k(d) = \frac{1}{2} \| b(x_k) + H_k^{-1} d \|^2 \; \textit{such that} \; \| d \| \leq \Delta_k, \tag{9}$$

where $\triangle_k = c^p \| b(x_k) \|$.

This paper is organized as follows. In the next section, the algorithm is discussed. In Section 3, the global convergence is established. Finally, the numerical results are reported in Section 4.

The L-M-BFGS Trust-Region Method

The algorithm is given as follows.

- Algorithm 1.

Initial: Given the constants ρ, $c \in (0, 1)$, $p = 0$, $\epsilon > 0$, $x_0 \in \Re^n$, $H_0 \in \Re^n \times \Re^n$ is symmetric and positive definite. Set $k := 0$;
Step 1: If $\|b_k\| < \epsilon$, stop;
Step 2: Solve the TR subproblem (9) with $\triangle = \triangle_k$ to obtain d_k;

Step 3: Calculate the following ratio of the actual reduction over the predicted reduction:

$$r_k^p = \frac{\theta(x_k + d_k) - \theta(x_k)}{q_k(d_k) - q_k(0)}. \quad (1)$$

If $r_k^p < \rho$, then we let $p = p + 1$, go to Step 2. Otherwise, go to Step 4;
Step 4: Set $x_{k+1} = x_k + d_k$, $y_k = b_{k+1} - b_k$. Use (8) to generate the updated matrix H_{k+1} with positive definiteness.
Step 5: Let $k := k + 1$, $p = 0$, and go to Step 1.

Remark 1. (i) In Algorithm 1, the "Step 2-Step 3-Step 2" procedure is called the inner cycle.

(ii) It is well known that the positive definiteness of the update matrix H_k is very important when analyzing the convergence of the algorithm. Byrd et al. [27] showed that the limited-memory BFGS matrix has this property if the curvature $(s_k)^T y_k > 0$ is satisfied. Similarly, Powell [28] proposed that y_k should be as follows:

$$y_k = \begin{cases} y_k, & \text{if } s_k^T y_k \geq 0.2 s_k^T B_k s_k, \\ \theta_k y_k + (1 - \theta_k) B_k s_k, & \text{otherwise,} \end{cases}$$

where $\theta_k = \frac{0.8 s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T y_k}$, B_k is an approximation of $\nabla b(x_k)$ and $B_k = H_k^{-1}$. Therefore, Step 4 of Algorithm 1 is reasonable.

To compare with the normal TR method, another algorithm is given and we call it Algorithm 2.

- Algorithm 2.(BFGS Trust-Region Method)

Step 2: Solve the TR subproblem (4) with $\triangle = \triangle_k$ to obtain d_k;
Step 4: Set $x_{k+1} = x_k + d_k$, $y_k = b_{k+1} - b_k$. Use (5) to generate the updated matrix $H_{k+1} = B_{k+1}^{-1}$ with positive definiteness.

Remark 2. It is easy to see that the difficult step of these two algorithms is the Step 2. The following *Dogleg* method [29] is used to solve the TR subproblem models (4) and (9) to obtain d_k.

- At the kth iteration:

set $\varpi_k = H_k b(x_k)$;
If $\|\varpi_k\| \leq \triangle_k$, let $d_k = -\varpi_k$; Otherwise, set
$\varsigma_k = \frac{\|H_k^{-1} b(x_k)\|^2}{b(x_k)^T H_k^{-1} H_k^{-1} H_k^{-1} b(x_k)}$, $\varrho_k = -\varsigma_k H_k^{-1} b(x_k)$, $\tau_k = -\varpi_k$:
If $\varsigma_k \geq 0$ and $\varsigma_k \leq 1$ hold, set $d_k = \varsigma_k \varrho_k$; If $\varsigma_k \geq 1$ and $\varsigma_k \leq 2$ hold, set $d_k = \varrho_k + (\varsigma_k - 1)(\tau_k - \varrho_k)$.

Convergence Analysis

This section will provide some convergence results under the following assumptions.

Assumption A (i) Let the level set Ω

$$\Omega = \{x | \theta(x) \leq \theta(x_0)\} \quad (1)$$

be bounded, and let $b(x)$ be twice continuously differentiable on an open convex set Ω_1 containing Ω.

(ii) The relation

$$\| [\nabla b(x_k) - H_k^{-1}]b_k \| = O(\| d_k \|) \tag{2}$$

holds.

Assumption A(i) implies that there exists $M_1 > 0$ such that

$$\| \nabla b(x_k)^T \nabla b(x_k) \| \leq M_1, \ \forall \ k. \tag{3}$$

Similar to Moré [30], Yuan et al. [9], and Yuan and Sun [11], we can obtain the following lemma.

Lemma 0.1 *Let the predicted reduction be*

$$Pred_k(d_k) = q_k(d_k) - q_k(0) \tag{4}$$

and let d_k be the solution of (9). Then, we have

$$Pred_k(d_k) \leq -\frac{1}{2} \| H_k^{-1} b_k \| \min \left\{ \Delta_k, \frac{\| H_k^{-1} b_k \|}{\| H_k^{-1} \|^2} \right\}. \tag{5}$$

Proof. For any $\alpha \in [0, 1]$, by the property of the solution of (9) named d_k, we obtain

$$\begin{aligned} Pred_k(d_k) &\leq Pred_k\left(-\alpha \frac{\Delta_k}{\| H_k^{-1} b_k \|} H_k^{-1} b_k \right) \\ &= \frac{1}{2}\alpha^2 \Delta_k^2 (H_k^{-1} H_k^{-1} b_k)^T (H_k^{-1} H_k^{-1} b_k) / \| H_k^{-1} b_k \|^2 - \alpha \Delta_k \| H_k^{-1} b_k \| \\ &\leq \frac{1}{2}\alpha^2 \Delta_k^2 \| H_k^{-1} H_k^{-1} \| - \alpha \Delta_k \| H_k^{-1} b_k \| . \end{aligned}$$

Thus, the inequality

$$-Pred_k(d_k) \geq \max_{0 \leq \alpha \leq 1} [\alpha \Delta_k \| H_k^{-1} b_k \| - \frac{1}{2}\alpha^2 \Delta_k^2 \| H_k^{-1} \|^2] \geq \frac{1}{2} \| H_k^{-1} b_k \| \min \left\{ \Delta_k, \frac{\| H_k^{-1} b_k \|}{\| H_k^{-1} \|^2} \right\}$$

holds. This completes the proof.

Lemma 0.2 *Let d_k be the solution of (9), and let the actual reduction $Ared_k(d_k)$ be*

$$Ared_k(d_k) = \theta(x_k + d_k) - \theta(x_k). \tag{6}$$

Suppose that Assumption A holds and that the sequence $\{x_k, d_k\}$ is generated by Algorithm 1. Then, the inner cycle of Algorithm 1 will stop in a finite number of steps.

Proof. We will establish this lemma by contradiction. Suppose that the inner cycle of Algorithm 1 infinitely circles at x_k, namely, $p \to \infty$, $r_k^p < \rho$ and $c^p \to 0$ hold. It is easy to obtain $\|b_k\| \geq \epsilon$, or the algorithm stops. Thus, we can conclude that $\|d_k\| \leq \triangle_k = c^p \|b_k\| \to 0$ is true.

By the definitions of d_k, $Ared_k(d_k)$, $Pred_k(d_k)$, and Assumption A, we obtain

$$\begin{aligned} | Ared_k(d_k) - Pred_k(d_k) | &= | \theta(x_k + d_k) - q_k(d_k) | \\ &= \frac{1}{2} | \| b_k + \nabla b(x_k) d_k + O(\| d_k \|^2) \|^2 - \| b_k + H_k^{-1} d_k \|^2 | \\ &= | b_k^T \nabla b(x_k) d_k - b_k^T H_k^{-1} d_k + O(\| d_k \|^2) + O(\| d_k \|^4) | \\ &\leq \| [\nabla b(x_k) - H_k^{-1}] b_k \| \| d_k \| + O(\| d_k \|^2) + O(\| d_k \|^4) \\ &= O(\| d_k \|^2). \end{aligned} \tag{7}$$

where the second equality follows Taylor's formula. It follows from Lemma 0.1 and 7) that

$$| r_k^p - 1 | = \frac{| Ared_k(d_k) - Pred_k(d_k) |}{| Pred_k(d_k) |} \leq \frac{2O(\| d_k \|^2)}{\Delta_k \| B_k b_k \|} \to 0.$$

Thus, if p is sufficiently large, we have $r_k^p \geq \rho$. This relation contradicts $r_k^p < \rho$. The proof is complete.

The above lemma shows that the given algorithm is well-defined.

Theorem 0.1 *Let Assumption A hold, and let* $\{x_k\}$ *be generated by Algorithm 1. Then,* $\{x_k\} \subset \Omega$, *and* $\{\theta(x_k)\}$ *converges. In particular, Algorithm 1 either stops in a finite number of steps or generates an infinite sequence* $\{x_k\}$ *such that*

$$\lim_{k \to \infty} \| b_k \| = 0. \tag{8}$$

Proof. By the definition of Algorithm 1, we obtain $r_k^p \geq \rho > 0$. Combining this with Lemma 0.1 generates

$$\theta(x_{k+1}) \leq \theta(x_k) \leq \cdots \leq \theta(x_0). \tag{9}$$

Thus, $\{x_k\} \subset \Omega$ holds. Considering $\theta(x_k) \geq 0$, we then conclude that $\{\theta(x_k)\}$ converges.

If Algorithm 1 does not stop after a finite number of steps, again by (9) and $\theta(x_k) \geq 0$, we easily conclude that

$$\theta(x_k) \to 0, \ k \to \infty,$$

which shows that $b(x_k) \to 0$, $k \to \infty$. The proof is complete.

Theorem 0.1 proves that the sequence $\{x_k\}$ of Algorithm 1 converges such that $\|b(x_k)\| \to 0$ without the assumption that $\nabla b(x^*)$ is nondegenerate, where x^* is a cluster point of $\{x_k\}$.

Numerical Results

This section will report numerical results obtained using Algorithm 1. The test functions with initial points x_0 are listed as follows:

$$b(x) = (f_1(x), f_2(x), \cdots, f_n(x))^T.$$

Function 1. Exponential function 2

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

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

Function 2. Trigonometric function

$$f_i(x) = 2(n + i(1 - \cos(x_i)) - \sin(x_i) - \sum_{k=1}^{n} \cos(x_k))(2\sin(x_i) - \cos(x_i)), \ i = 1, 2, \cdots, n$$

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

Function 3. Singular function

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

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

Function 4. Logarithmic function

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

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

Function 5. Broyden Tridiagonal function [[31], pp. 471–472]

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

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

Function 6. Trigexp function

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

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

Function 7. Strictly convex function 1 [[32], p. 29]

$b(x)$ is the gradient of $h(x) = \sum_{i=1}^{n}(e^{x_i} - x_i)$.

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

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

Function 8. Variable dimensional function

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

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

Function 9. Discrete boundary value problem [33].

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

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

Function 10. The discretized two-point boundary value problem that is similar to the problem in [34]

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

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 $x = (50, 0, 50, 0, \cdots)$.

All codes for the experiments were written in MATLAB r2009a and run on a PC with an E5507@2.27 GHz CPU and 2.99 GB of memory using the Windows XP operating system. The parameters were set as $\rho = 0.0001$, $\epsilon = 10^{-5}$, $c = 0.1$, $\gamma = 0.7$, where B_0 and H_0 are unit matrices and $m = 6$. In the inner iterations of Algorithm 1 and Algorithm 2, the trial step will be accepted when $p > 5$. We also stop the program if the number of iterations is larger than one thousand or when $\theta(x) < 10^{-5}$. The columns of the tables have the following meaning:

Dim: the dimension of the problem. NG: the number of norm function evaluations.
NI: the total number of iterations. GN: the normal value of $\theta(x)$ when the program stops.
NaN: fails to find the final values of $\theta(x)$.

From Table 1, it is clear that the new algorithm performs quite well in terms of solving these problems and that the dimension does not have an obvious influence on the performance of the presented method. Algorithm 2 can also successfully solve some of these problems. However, Algorithm 2 fails to solve problems 3 and 6.

To clearly demonstrate these algorithms' efficiencies, a tool developed by Dolan and Moré [35] is used to analyze them. In this tool, the so-called (cumulative) distribution function ϱ_s is defined by

$$\varrho_s(\kappa) = \frac{1}{n_p} size\{p \in P : \pi_{p,s} \leq \kappa\},$$

Table 1. Numerical results using Algorithms 1 and 2.

		Algorithm 1		Algorithm 2	
Functions	Dim	NI/NG	GN	NI/NG	GN
1	800	0/1	8.348973e-006	0/1	8.348973e-006
	1000	0/1	6.676674e-006	0/1	6.676674e-006
	2000	0/1	3.335834e-006	0/1	3.335834e-00
2	800	7/18	1.766154e-006	8/19	4.764396e-006
	1000	7/18	1.578116e-006	8/19	4.228277e-006
	2000	6/17	8.507453e-006	8/19	2.609172e-006
3	800	82/108	5.997218e-006	1000/1716	3.453004e+018
	1000	68/89	8.159424e-006	1000/1636	3.372369e+019
	2000	95/152	8.892174e-006	1000/1701	1.757490e+022
4	800	5/6	6.440159e-008	5/6	6.440159e-008
	1000	5/6	7.954854e-008	5/6	7.954854e-008
	2000	5/6	1.553487e-007	5/6	1.553487e-007
5	800	61/67	9.546109e-006	47/83	7.073607e-006
	1000	58/64	9.424256e-006	35/66	5.881295e-006
	2000	61/67	9.412218e-006	44/75	8.483566e-006
6	800	62/73	6.368574e-008	6/22	NaN
	1000	59/75	1.493209e-007	6/22	NaN
	2000	71/92	2.616014e-007	6/22	NaN
7	800	6/7	1.665516e-009	5/6	1.718325e-006
	1000	6/7	2.070362e-009	5/6	2.139904e-006
	2000	6/7	4.094910e-009	5/6	4.247951e-006
8	800	1/2	0.000000e+000	1/2	0.000000e+000
	1000	1/2	0.000000e+000	1/2	0.000000e+000
	2000	1/2	0.000000e+000	1/2	0.000000e+000
9	800	2/3	3.679023e-006	4/10	1.336824e-006
	1000	2/3	2.358640e-006	4/10	9.011723e-007
	2000	2/3	5.917002e-007	2/3	3.637130e-006
10	800	15/21	1.212419e-007	32/40	6.985962e-006
	1000	15/21	1.391409e-007	34/42	5.778079e-006
	2000	15/21	1.824225e-007	33/44	5.520880e-006

doi:10.1371/journal.pone.0120993.t001

where p is a problem, s is its solver, n_p is the number of problems, n_s is the number of existing solvers, P is the test set, κ denotes the spending time or the iteration number, and $\pi_{p,s}$ is the performance ratio given by

$$\pi_{p,s} = \frac{\kappa_{p,s}}{\min\{\kappa_{p,s} : s \in S\}},$$

where S is the set of solvers. According to the above formulas and the rules given in [35], the performance profile plot of Table 1 is given in the following two figures.

Figs 1 and 2 show that the performance of these methods is a function of *NI* and *NG*, respectively, in Table 1. It is clear that the given method performs better because it has a higher probability of being the optimal solver. Fig 1 shows that Algorithm 1 is superior to Algorithm 2 for $\kappa \leq 14$, approximately. From Fig 2, it is clear that the given method can successfully solve

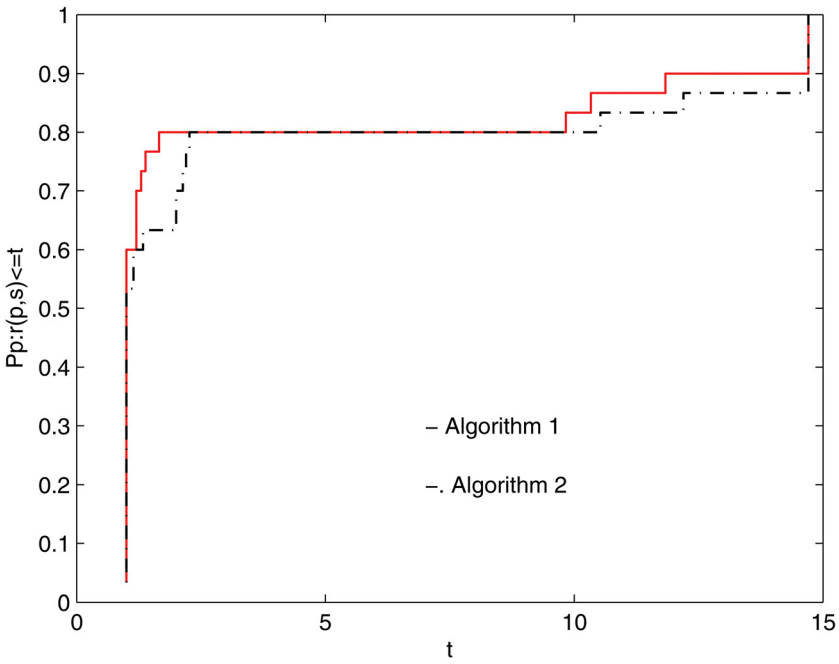

Fig 1. Performance profiles of these methods(NI).

doi:10.1371/journal.pone.0120993.g001

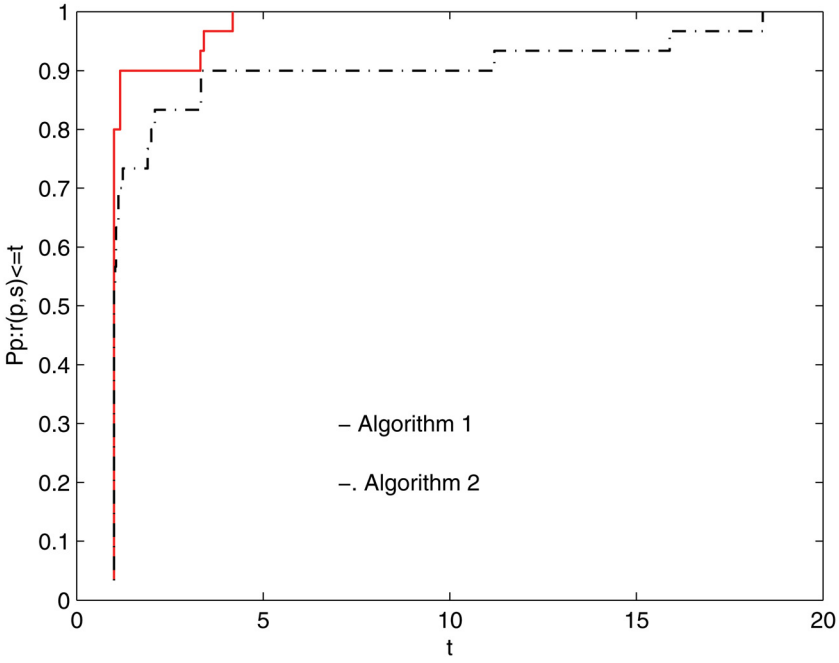

Fig 2. Performance profiles of these methods(NG).

doi:10.1371/journal.pone.0120993.g002

100% of the test problems at $\kappa \approx 5$ and that Algorithm 2 successfully solves the test problems at $\kappa \approx 17$. Overall, the numerical results are interesting.

Supporting Information:Legends of Figs 1 and 2

Dolan and Moré [35] developed a new tool to analyze the efficiency of algorithms. They introduced the notion of a performance profile for evaluating and comparing the performance of the set of solvers S on a test set P. Assuming that n_s solvers and n_p problems exist, for each problem p and solver s, they defined $\kappa_{p,s}$ as the computing time (the number of function evaluations or some other metric) required to solve problem p using solver s.

Requiring a baseline for comparison, they compared the performance on problem p using solver s with the best performance by any solver on this problem, giving the the performance ratio

$$\pi_{p,s} = \frac{\kappa_{p,s}}{\min\{\kappa_{p,s} : s \in S\}}.$$

Suppose that a parameter $\pi_M \geq \pi_{p,s}$ for all p, s is chosen, and $\pi_{p,s} = \pi_M$ if and only if solver s does not solve problem p.

The performance of solver s on any given problem might be of interest. More importantly, one would like to obtain an overall assessment of the performance of the solver. With this motivation, these authors defined

$$\varrho_s(\kappa) = \frac{1}{n_p} size\{p \in P : \pi_{p,s} \leq \kappa\}.$$

Specifically, $\varrho_s(\kappa)$ is the probability for solver $s \in S$ that a performance ratio $\pi_{p,s}$ is within a factor $\kappa \in \Re$ of the best possible ratio. Then, function ϱ_s is the (cumulative) distribution function for the performance ratio. The performance profile $\varrho_s : \Re \mapsto [0, 1]$ for a solver is a nondecreasing, piecewise constant function and is continuous from the right at each breakpoint. The value of $\varrho_s(1)$ is the probability that the solver would outperform the remaining solvers.

According to the above rules, we know that one solver whose performance profile plot is on the top right outperforms the remaining solvers. Similar legends can be found in [10, 36–38].

Conclusion

1. In this paper, a TR model combining with the L-M-BFGS technique is presented for nonlinear equations and the global convergence is established. Fact, this work is an extension of the paper [9] and the difference between these two papers is the update matrix: this paper chooses the L-M-BFGS formula and the paper [9] chooses the normal BFGS formula.
2. Since the L-M-BFGS technique requires minimal storage and is suitable for certain classes of large scale optimization problems, so we extend it to large scale nonlinear equations. Numerical results turn out the given algorithm is competitive to similar method for large-scale nonlinear equations.
3. It is well known that the conjugate gradient methods are effective techniques for large-scale optimization problems since they do not need the matrix information, which motivates us to use the conjugate gradient technique for large scale nonlinear equations. We think the numerical performance is also very interesting.

Acknowledgments

This work is supported by the QFRC visitor funds of the University of Technology Sydney, study abroad funds of Guangxi talents in China, Guangxi Colleges and Universities Key Laboratory of Mathematics and Its Applications, the Program for Excellent Talents in Guangxi Higher Education Institutions (Grant No. 201261), Guangxi Higher Education Institutions (NO.YB2014389), the Guangxi NSF (Grant No. 2012GXNSFAA053002) and the China NSF (Grant No. 11261006 and 11161003). The authors thank the referees for their valuable comments which greatly improve our paper.

Author Contributions

Conceived and designed the experiments: YL GY ZW. Performed the experiments: YL GY ZW. Analyzed the data: YL GY ZW. Contributed reagents/materials/analysis tools: YL GY ZW. Wrote the paper: GY YL ZW.

References

1. Gu B and Sheng VS. Feasibility and finite convergence analysis for accurate on-line v-support vector learning. IEEE Transactions on Neural Networks and Learning Systems, 2013, 24(8):1304–1315.
2. Li J,Li XL,Yang B,Sun XM. Segmentation-based Image Copy-move Forgery Detection Scheme, IEEE Transactions on Information Forensics and Security, 2015, 10(3):507–518. doi: 10.1109/TIFS.2014.2381872
3. Wen XZ,Shao L, Fang W, and Xue Y. Efficient Feature Selection and Classification for Vehicle Detection. IEEE Transactions on Circuits and Systems for Video Technology, 2015. doi: 10.1109/TCSVT.2014.2358031
4. Zhang H, Wu J, Nguyen TM, Sun MX. Synthetic Aperture Radar Image Segmentation by Modified Student's t-Mixture Model, IEEE Transaction on Geoscience and Remote Sensing, 2014, vol. 52, no. 7, pp. 4391–4403. doi: 10.1109/TGRS.2013.2281854
5. Fu ZJ. Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing, IEICE Transactions on Communications, Vol.E98-B, No. 1, pp.190–200. doi: 10.1587/transcom.E98.B.190
6. Conn AR, Gould NIM, and Toint PL, Trust region method, Society for Industrial and Applied Mathematics, Philadelphia PA, 2000.
7. Yuan Y, Trust region algorithm for nonlinear equations, Information, 1 (1998), pp. 7–21.
8. Yuan G, Lu X, and Wei Z, BFGS trust-region method for symmetric nonlinear equations, Journal of Computational and Applied Mathematics, 230 (2009), pp. 44–58. doi: 10.1016/j.cam.2008.10.062
9. Yuan G, Lu X, and Wei Z, A BFGS trust-region method for nonlinear equations, Computing, 92 (2011), pp. 317–333. doi: 10.1007/s00607-011-0146-z
10. Yuan G, Lu S, and Wei Z, A new trust-region method with line search for solving symmetric nonlinear equations, International Journal of Computer Mathematics, 88 (2011), pp. 2109–2123. doi: 10.1080/00207160.2010.526206
11. Yuan Y and Sun W, Optimization theory and methods, Science Press, Beijing, 1997.
12. Fan JY, A modified Levenberg-Marquardt algorithm for singular system of nonlinear equations, Journal of Computational Mathematics, 21 (2003), pp. 625–636.
13. Yamashita N and Fukushima M, On the rate of convergence of the Levenberg-Marquardt Method, Computing, 15 (2001), pp. 239–249.
14. Zhang J and Wang Y, A new trust region method for nonlinear equations, Mathematical Methods of Operations Research, 58 (2003), pp. 283–298. doi: 10.1007/s001860300302
15. Griewank A, The 'global' convergence of Broyden-like methods with a suitable line search, Journal of the Australian Mathematical Society, Series B, 28 (1986), pp. 75–92. doi: 10.1017/S0334270000005208
16. Yuan G and Lu X, A new backtracking inexact BFGS method for symmetric nonlinear equations, Computers and Mathematics with Applications, 55 (2008) pp. 116–129. doi: 10.1016/j.camwa.2006.12.081
17. Yuan G, Wei Z, and Lu S, Limited memory BFGS method with backtracking for symmetric nonlinear equations, Mathematical and Computer Modelling, 54 (2011), pp. 367–377. doi: 10.1016/j.mcm.2011.02.021

18. Yuan G and Yao S, A BFGS algorithm for solving symmetric nonlinear equations, Optimization, 62 (2013), pp. 85–99. doi: 10.1080/02331934.2011.564621
19. Zhu D, Nonmonotone backtracking inexact quasi-Newton algorithms for solving smooth nonlinear equations, Applied Mathematics and Computation, 161 (2005), pp. 875–895. doi: 10.1016/j.amc.2003.12.074
20. Bertsekas DP, Nonlinear programming, Athena Scientific, Belmont, 1995.
21. Gu G, Li D, Qi L, and Zhou S, Descent directions of quasi-Newton methods for symmetric nonlinear equations, SIAM Journal on Numerical Analysis, 40 (2002), pp. 1763–1774. doi: 10.1137/S0036142901397423
22. Levenberg K, A method for the solution of certain nonlinear problem in least squares, Quarterly of Applied Mathematics, 2 (1944), pp. 164–166.
23. Li D and Fukushima M, A global and superlinear convergent Gauss-Newton-based BFGS method for symmetric nonlinear equations, SIAM Journal on Numerical Analysis, 37 (1999), 152–172. doi: 10.1137/S0036142998335704
24. Marquardt DW, An algorithm for least-squares estimation of nonlinear inequalities, SIAM Journal on Applied Mathematics, 11 (1963), pp. 431–441. doi: 10.1137/0111030
25. Nocedal J and Wright SJ, Numerical optimization, Spinger, New York, 1999.
26. Byrd RH, Nocedal J, Schnabel RB, Representations of quasi-Newton matrices and their use in limited memory methods, Math. Prgram., 63 (1994), pp. 129–156. doi: 10.1007/BF01582063
27. Byrd RH, Lu PH, and Nocedal J, A limited memory algorithm for bound constrained optimization, SIAM J. Statist. Sci. Comput., 16 (1995), pp. 1190–1208. doi: 10.1137/0916069
28. Powell MJD, A fast algorithm for nonlinearly constrained optimization calculations, Numer. Anal., (1978), pp. 155–157.
29. Wang YJ and Xiu NH, Theory and algoithm for nonlinear programming, Shanxi Science and Technology Press, Xian, 2004.
30. Moré JJ, Recent development in algorithm and software for trust region methods, in: Bachem A., Grotschel M., Kortz B. (Eds.), Mathematical Programming: The State of the Art, Spinger-Verlag, Berlin, 1983, 258–285.
31. Gomez-Ruggiero M, Martinez JM, and Moretti A, Comparing algorithms for solving sparse nonlinear systems of equations, SIAM Journal on Scientific and Statistical Computing, 23 (1992), pp. 459–483. doi: 10.1137/0913025
32. Raydan M, The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem, SIAM Journal on Optimization, 7 (1997), pp. 26–33. doi: 10.1137/S1052623494266365
33. Moré JJ, Garbow BS, and Hillström KE, Testing unconstrained optimization software, ACM Transactions on Mathematical Software, 7 (1981), pp. 17–41. doi: 10.1145/355934.355936
34. Ortega JM and Rheinboldt WC, Iterative solution of nonlinear equations in several variables, Academic Press, New York, 1970.
35. Dolan ED and Moré JJ, Benchmarking optimization software with performance profiles, Mathematical Programming, 91 (2002), pp. 201–213. doi: 10.1007/s101070100263
36. Yuan G and Lu X, A modified PRP conjugate gradient method, Annals of Operations Research, 166 (2009), pp. 73–90. doi: 10.1007/s10479-008-0420-4
37. Yuan G, Lu X, and Wei Z, A conjugate gradient method with descent direction for unconstrained optimization, Journal of Computational and Applied Mathematics, 233 (2009), pp. 519–530. doi: 10.1016/j.cam.2009.08.001
38. Yuan G and Zhang M, A modified Hestenes-Stiefel conjugate gradient algorithm for large-scale optimization, Numerical Functional Analysis and Optimization, 34 (2013), pp. 914–937. doi: 10.1080/01630563.2013.777350