ISSN: 1017-060X (Print)

ISSN: 1735-8515 (Online)

Bulletin of the

Iranian Mathematical Society

Vol. 42 (2016), No. 4, pp. 819–837

Title:

A limited memory adaptive trust-region approach for large-scale unconstrained optimization

Author(s):

M. Ahookhosh, K. Amini, M. Kimiaei and M. R. Peyghami

Bull. Iranian Math. Soc.
Vol. 42 (2016), No. 4, pp. 819–837
Online ISSN: 1735-8515

A LIMITED MEMORY ADAPTIVE TRUST-REGION APPROACH FOR LARGE-SCALE UNCONSTRAINED OPTIMIZATION

M. AHOOKHOSH, K. AMINI*, M. KIMIAEI AND M. R. PEYGHAMI

(Communicated by Madjid Eshaghi Gordji)

Abstract. This study concerns with a trust-region-based method for solving unconstrained optimization problems. The approach takes the advantages of the compact limited memory BFGS updating formula together with an appropriate adaptive radius strategy. In our approach, the adaptive technique leads us to decrease the number of subproblems solving, while utilizing the structure of limited memory quasi-Newton formulas helps to handle large-scale problems. Theoretical analysis indicates that the new approach preserves the global convergence to a first-order stationary point under classical assumptions. Moreover, the superlinear and the quadratic convergence rates are also established under suitable conditions. Preliminary numerical experiments show the effectiveness of the proposed approach for solving large-scale unconstrained optimization problems.

Keywords: Unconstrained optimization, trust-region framework, compact quasi-Newton representation, limited memory technique, adaptive strategy.
MSC(2010): Primary: 90C30; Secondary: 65k05, 65k10.

1. Introduction

Over the past few decades large-scale unconstrained optimization gets lots of attention thanks to arising in many applications in the context of applied sciences such as biology, physics, geophysics, chemistry, engineering and industry. In general, an unconstrained optimization problem can be formulated as follows

$$\min f(x), \ x \in \mathbf{R}^n \tag{1.1}$$

Article electronically published on August 20, 2016.
Received: 15 November 2013, Accepted: 30 April 2015.
*Corresponding author.

where $f : \mathbb{R}^n \to \mathbb{R}$ is assumed to be continuously differentiable.

Motivation & history There exist lots of iterative schemes such as Newton, quasi-Newton, variable metric, gradient and conjugate gradient methods that have been introduced and developed for solving the unconstrained problem (1.1). In the most cases, they are required to exploit one of the general globalization techniques, i.e. line search and trust-region techniques, in order to guarantee the global convergence results (see [26]).

For a given x_k, a line search technique refers to a procedure that computes a step-size α_k along a specific direction d_k and generates a new point as $x_{k+1} = x_k + \alpha_k d_k$. Many line search strategies for determining the step-size have been proposed, for instance exact line search or Armijo, Wolfe and Goldstein inexact line searches (see [26]). On the other hand, a quadratic-based framework of trust-region technique computes a trial step d_k by solving the quadratic subproblem

$$\text{(1.2)} \qquad \begin{array}{ll} \text{minimize} & m_k(x_k + d) = f_k + g_k^T d + \frac{1}{2} d^T B_k d \\ \text{subject to} & d \in \mathbb{R}^n, \ \|d\| \leq \Delta_k, \end{array}$$

where $\|.\|$ denotes the Euclidean norm, $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, B_k is the exact Hessian, i.e. $G_k = \nabla^2 f(x_k)$, or its symmetric approximation and Δ_k is the trust-region radius. In order to evaluate an agreement between the model and the objective function and to accept or reject the trial step d_k, a criterion based on the actual and the model reductions is required. Traditional monotone trust-region methods do this by defining the ratio

$$\text{(1.3)} \qquad \rho_k = \frac{f(x_k) - f(x_k + d_k)}{m_k(x_k) - m_k(x_k + d_k)},$$

where the numerator and the denominator are called the actual and the predicted reductions, respectively. It is clear that the predicted reduction is always positive. The trial step d_k is accepted whenever ρ_k is greater than a positive constant. In this case, if ρ_k is greater than a constant $\mu_2 > 0$, then the new point $x_{k+1} = x_k + d_k$ is accepted and the iterate is called very successful. But if $\rho_k \geq \mu_1$ for $0 < \mu_1 < \mu_2$, then the new point is generated by $x_{k+1} = x_k + d_k$ and the iterate is called successful and the trust-region radius is also updated appropriately based on amount of ρ_k. Otherwise, the trial step is rejected and the quadratic subproblem (1.2) would be resolved at the current point with the reduced trust-region radius.

On the basis of literature reviews, one can realize that the traditional trust-region methods are very sensitive to the initial trust-region radius Δ_0 and its updating scheme. This fact leads the researchers to work on finding appropriate procedures for initial trust-region radius as well as its updating rule. In 1997, Sartenaer in [30] proposed an approach to determine the initial trust-region radius by monitoring the agreement between the model and the objective function along the steepest descent direction. A possible drawback of this approach is the dependency of the parameters to the problem information. Recently, Gould et al. in [18] examined the sensitivity of the traditional trust-region methods to the parameters related to the step acceptance test and trust-region radius update with extensive numerical experiments. Despite their comprehensive tests on a large number of test problems, they did not claim to find the best parameters for updating scheme. In 2002, motivated by a problem in the field of neural network, the first adaptive trust-region radius was proposed by Zhang et al. in [35] in which the information of the current iteration was used more effectively to introduce an adaptive scheme. They introduced the following adaptive trust-region radius

$$\Delta_k = c^{p_k} \|g_k\| \, \|\hat{B}_k^{-1}\|, \tag{1.4}$$

where $c \in (0,1)$ is a constant, $p_k \in \mathbb{N} \cup \{0\}$ and the matrix $\hat{B}_k = B_k + E_k$ is a safely positive definite matrix based on Schnabel and Eskow modified Cholesky factorization, see [31]. As numerical results show, their method works very well on small-scale unconstrained optimization problems, but the situation dramatically changes for the large-scale and even medium-scale problems because of the computation of the inverse matrix $\hat{B}_k^{-1}$. Subsequently, Shi and Guo in [33] proposed another interesting adaptive radius by

$$\Delta_k = -c^{p_k} \frac{g_k^T q_k}{q_k^T \hat{B}_k q_k} \|q_k\|, \tag{1.5}$$

where $c \in (0,1)$ is a constant, $p_k \in \mathbb{N} \cup \{0\}$, and $\hat{B}_k$ is generated by the procedure

$$\hat{B}_k = B_k + iI,$$

where i is the smallest nonnegative integer so that $q_k^T \hat{B}_k q_k > 0$ and q_k satisfies the well-known angle condition

$$-\frac{g_k^T q_k}{\|g_k\| . \|q_k\|} \geq \tau, \tag{1.6}$$

in which $\tau \in (0,1]$ is a constant. An important advantage of this method is its ability for selecting an appropriate q_k in order to make a more robust method. For instance, Shi and Guo proposed $\tilde{q}_k = -g_k$ and $\hat{q}_k = -B_k^{-1} g_k$. Preliminary numerical results and theoretical analysis showed that their method is promising for solving medium-scale problems without any need of exploring for appropriate initial trust-region radius. For more references about the adaptive trust-region radius, see also [1–4, 6].

Although the proposed adaptive radius of Shi and Guo has some advantages such as decreasing in the total computational costs by decreasing the number of subproblems solving and determining a good initial radius, it suffers from some drawbacks as well. First of all, it can be easily seen that $\tilde{q}_k = -g_k$ does not generate an appropriate radius (see [33]), and the computation of $\hat{q}_k = -B_k^{-1} g_k$ requires the computation of the inverse matrix B_k^{-1} or solving a linear system of equations that causes being inappropriate for large-scale problems. Secondly, the process of generating $\hat{B}_k$ guarantees that the denominator of (1.5) is bounded away from zero, however, if the numerator term $-g_k^T q_k$ be close to zero, then this case causes a tiny trust-region radius which possibly increases the total number of iterations. Thirdly, numerical experiments have shown that when the ratio is so close to 1, for very successful iterations, the method does not necessarily enlarge the trust-region radius sufficiently. In addition, the procedure of constructing $\hat{B}_k$ is almost unusual and sometimes costly. Finally, the necessity of storing B_k for computing the term $q_k^T \hat{B}_k q_k$ in (1.5) and the term $d_k^T B_k d_k$ for computing the predicted reduction may cause the method to be unsuitable for large-scale problems.

It is known that the limited memory quasi-Newton methods are the customized version of the quasi-Newton methods for large-scale optimization problems. Besides, their implementations are almost identical to that of the quasi-Newton methods, however, the Hessian and the inverse Hessian approximations are not explicitly formed on them. Instead, they are defined based on information of some small number of previous iterations to reduce the required memory. As another advantage, some limited memory quasi-Newton formula, such as the compact limited memory BFGS updating formula (see [10]), can preserve positive definite property based on some mild conditions. Due to these remarkable advantages of the limited memory quasi-Newton formulas, they are widely utilized for large-scale optimization problems, see [8, 10, 19–21, 24, 25].

Content. In this paper, we propose an improved version of the adaptive trust-region radius (1.5) to attain better performances when the number of variables of underlying function is large. More specifically, we first substitute (1.5) by a modified formula to overcome the above-mentioned disadvantages. We then take the advantages of the compact limited memory BFGS formula in order to calculate the term $d_k^T B_k d_k$ in a low-cost way. It is also clear that the method can preserve positive definiteness of B_k based on some conditions, so it can avoid of calculating $\hat{B}_k$. The analysis of the new approach shows that it inherits both the stability of adaptive trust-region approaches and the effectiveness of the limited memory BSGS. We also investigate the global convergence to first-order stationary points of the proposed method and provide the superlinear and quadratic convergence rates. To show the efficiency of the proposed method, some numerical results are reported.

The remainder of this paper is organized as follows. In Section 2, we describe the motivation behind the proposed algorithm and outline the algorithm. Section 3 is devoted to investigating the global, superlinear and quadratic convergence properties of the algorithm. Numerical results are provided in Section

4 to show the well promising behaviour of the proposed method for solving large-scale unconstrained optimization problems. Finally, some conclusions are given in Section 5.

2. THE NEW ALGORITHM: MOTIVATION AND STRUCTURE

In this section, we first verify some alternatives to overcome disadvantages of the trust-region radius (1.5) which mentioned in the previous section. We then construct our trust-region-based algorithm and establish some of its properties.

It can be easily seen that if one exploits a positive definite quasi-Newton formula for B_k, then there is no need to define $\hat{B}_k$ in (1.5) due to the fact that $q_k^T B_k q_k > 0$ for every arbitrary non-zero vector q_k. Therefore, the method exempts from the generating $\hat{B}_k$. As mentioned before, $\tilde{q}_k = -g_k$ does not generate an appropriate trust-region radius, and the calculation of $\hat{q}_k = -B_k^{-1} g_k$ imposes a remarkable computational cost to the method. Thus, it is needed to find a new q_k with less computational cost satisfying (1.6). For this purpose, the following q_k is employed throughout the paper

$$q_k = -H_k g_k, \tag{2.1}$$

which clearly has less computational cost in comparison to the cost of calculating $\hat{q}_k = -B_k^{-1} g_k$. In order to show $q_k = -H_k g_k$ satisfies (1.6), it is sufficient that the spectral condition number κ_k of H_k, the ratio λ_1/λ_n of the largest to the smallest eigenvalues of H_k, is bounded above independent of k. Let θ_k denotes the angle between $q_k = -H_k g_k$ and $-g_k$, so the fact that $\|H_k g_k\| \leq \lambda_1 \|g_k\|$ and $g_k^T H_k g_k \geq \lambda_n \|g_k\|^2$ implying

$$\sin\left(\frac{\pi}{2} - \theta_k\right) = \cos(\theta_k) = \frac{g_k^T H_k g_k}{\|g_k\|.\|H_k g_k\|} \geq \frac{\lambda_n \|g_k\|^2}{\lambda_1 \|g_k\|^2} = \kappa_k^{-1}.$$

From the inequality $sin(x) \leq x$, it can be concluded that

$$\theta_k \leq \frac{\pi}{2} - \kappa_k^{-1}.$$

Therefore, if spectral condition number κ_k is bounded above for any $k \in \mathbb{N}$, then θ_k is bounded away from $\pi/2$, i.e. the angle condition (1.6) is satisfied, for example see [16].

It is known that the compact limited memory BFGS formula can be used to decrease the computational costs of the calculation $q_k^T B_k q_k$, $d_k^T B_k d_k$ and $H_k g_k$ (see [10, 20]). Besides, it can remain positive definite if the curvature condition $y_k^T s_k > 0$ holds for all previous iterations, where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. We also increase the trust-region radius more than that defined in (1.5) when the procedure encounters with very successful iterations. It seems that these changes possibly lead to an approach needing less number of iterations and function evaluations.

It is clear that traditional trust-region methods and the radius defined in (1.5) requires to store the $n \times n$ matrix B_k needing lots amount of memory, especially for large-scale problems. Hence it is worth to explore a way to avoid storing this matrix. The pioneering work to find such methods proposed by Nocedal in [25] called the limited memory quasi-Newton method. On the basis of the interesting features of limited memory quasi-Newton methods, they have been widely used in various fields of optimization, see [8, 10, 19–21, 24, 25] and references therein.

Here, we briefly describe the compact limited memory BFGS formulae proposed by Byrd et al. in [10]. For a positive integer constant m_1, let matrices S_k and Y_k be defined as follows:

$$S_k = [s_{k-m}, \cdots, s_{k-1}],\ Y_k = [y_{k-m}, \cdots, y_{k-1}], \tag{2.2}$$

where $m = \min\{k, m_1\}$. Suppose that D_k be the $m \times m$ diagonal matrix

$$D_k = \operatorname{diag}\left[s_{k-m}^T y_{k-m}, \cdots, s_{k-1}^T y_{k-1}\right] \tag{2.3}$$

and let L_k be $m \times m$ lower triangular matrix

$$(L_k)_{i,j} = \begin{cases} s_{k-m+i-1}^T y_{k-m+j-1} & \text{if } i > j; \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Then, using relations (2.2)–(2.4), the compact representation of BFGS formula proposed by Byrd et al. in [10] can be expressed as follows:

$$B_k = B_k^{(0)} - \left[\begin{array}{cc} Y_k & B_k^{(0)} S_k \end{array}\right] \left[\begin{array}{cc} -D_k & L_k^T \\ L_k & S_k^T B_k^{(0)} S_k \end{array}\right]^{-1} \left[\begin{array}{c} Y_k^T \\ S_k^T B_k^{(0)} \end{array}\right], \tag{2.5}$$

where the basic matrix $B_k^{(0)}$ is defined as $B_k^{(0)} = \sigma_k I$, for some positive scalar σ_k. Defining $A_k = [Y_k, S_k]$ and using the way in [20], we can easily write

$$B_k = \sigma_k I - A_k \left[\begin{array}{cc} I & 0 \\ 0 & \sigma_k \end{array}\right] \left[\begin{array}{cc} -D_k & L_k^T \\ L_k & \sigma_k S_k^T S_k \end{array}\right]^{-1} \left[\begin{array}{cc} I & 0 \\ 0 & \sigma_k \end{array}\right] A_k^T.$$

Consequently, using the Cholesky factorization, this formula can be stated as

$$B_k = \sigma_k I - A_k \left[\begin{array}{cc} I & 0 \\ 0 & \sigma_k \end{array}\right] \left[\begin{array}{cc} -D_k^{\frac{1}{2}} & D_k^{-\frac{1}{2}} L_k^T \\ 0 & J_k^T \end{array}\right]^{-1} \left[\begin{array}{cc} D_k^{\frac{1}{2}} & 0 \\ -L_k D_k^{-\frac{1}{2}} & J_k \end{array}\right]^{-1} \left[\begin{array}{cc} I & 0 \\ 0 & \sigma_k \end{array}\right] A_k^T, \tag{2.6}$$

where J_k is a lower triangular matrix satisfying $J_k^T J_k = \sigma_k S_k^T S_k + L_k D_k^{-1} L_k^T$. Notice that the matrix D_k is a positive definite diagonal matrix, so $D_k^{1/2}$ exists and is a positive definite diagonal matrix as well. Moreover, from positive definiteness of the matrix D_k, one can easily conclude that $\sigma_k S_k^T S_k + L_k D_k^{-1} L_k^T$ is also positive definite and therefore the matrix J_k evidently exists. Under curvature assumption $s_k^T y_k > 0$ for each k, it is not difficult to show that the matrix B_k generated by the formula (2.6) is positive definite. In the rest of the paper, we use (2.6) to approximate exact Hessian $G_k = \nabla^2 f(x_k)$, however, we will never form it explicitly.

It is obvious that the computation of the term $H_k g_k$ in (2.1) is so costly, especially when the number of variables is large. Therefore, this formula can be calculated according to the compact limited memory technique which is customized for the approximate inverse Hessian, i.e. H_k. As in [10, 20], the compact limited memory representation of the approximate inverse Hessian H_k can be constructed by

(2.7)

$$H_k = \sigma_k^{-1} I + A_k \left[\begin{array}{cc} 0 & \sigma_k^{-1} I \\ R_k^{-T} & 0 \end{array}\right] \left[\begin{array}{cc} D_k + \sigma_k^{-1} Y_k^T Y_k & -I \\ -I & 0 \end{array}\right] \left[\begin{array}{cc} 0 & R_k^{-1} \\ \sigma_k^{-1} I & 0 \end{array}\right] A_k^T,$$

where

$$(R_k)_{i,j} = \begin{cases} s_{k-m+i-1}^T y_{k-m+i-1}, & \text{if } i \le j; \\ 0, & \text{otherwise.} \end{cases}$$

Since the matrix R_k is an $m \times m$ upper triangular matrix, for any arbitrary vector η_k, it is easy to compute the vector $\theta_k = R_k^{-1} \eta_k$ by solving an upper triangular system $R_k \theta_k = \eta_k$. Thanks to the updating formula of H_k in (2.7), for an arbitrary vector $v_k \in \mathbb{R}^n$, we compute the vector $H_k v_k$ by the following scheme:

Scheme 1: Calculation of $H_k v_k$ based on the formula (2.7)

Step 1. Determine $\xi = A_k^T v_k$ and partition it as $\xi = \left[\begin{array}{c} \xi_1 \\ \xi_2 \end{array}\right]$ where $\xi_1, \xi_2 \in \mathbf{R}^m$.
Step 2. Solve $R_k w = \xi_2$.
Step 3. Set $\beta = (D_k + \sigma_k^{-1} Y_k^T Y_k) w - \sigma_k^{-1} \xi_1$ and solve $R_k^T \gamma = \beta$.
Step 4. Set $H_k v_k = \sigma_k^{-1} [v_k + Y_k w] + S_k \gamma$.

Using the fact that D_k and R_k are respectively diagonal and triangular matrices, we observe that the Scheme 1 consists of $2mn + \frac{m(m+1)}{2}$ multiplications for Step 1, $m^2 + 3m + \frac{m(m+1)}{2}$ multiplications for Step 2 and $2mn + n$ multiplications for Step 3. Thus Scheme 1 requires $4mn + 2m^2 + 4m + n$ multiplications for computing the term $H_k v_k$.

As mentioned in the implementation of the Shi and Guo's algorithm in [33], we need to calculate the term $v_k^T B_k v_k$ in the subproblem (1.2) as well as the trust-region radius, respectively. One knows that direct computation of these formulas is expensive for large-scale problems. As discussed in [10, 20], these formula can be calculated more efficiently if one applies the compact version of limited memory quasi-Newton updates. Based on the formula (2.6), for an arbitrary vector $v_k \in \mathbb{R}^n$, we establish an effective scheme to compute $v_k^T B_k v_k$ as follows:

Scheme 2: Calculation of $v_k^T B_k v_k$ based on the formula (2.6)

Step 1. Compute $\xi = A_k^T v_k$, and let $\xi = \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}$ where $\xi_1, \xi_2 \in \mathbf{R}^m$.

Step 2. Solve two following systems

$$D_k^{\frac{1}{2}} t_1 = \xi_1 \text{ and } J_k t_2 = \sigma_k \xi_2 + L_k (D_k^{-\frac{1}{2}} t_1).$$

Step 3. Set $v_k^T B_k v_k = \sigma_k v_k^T v_k + t_1^T t_1 - t_2^T t_2$.

In Scheme 2, it is observed that the vector ξ is available from Step 1 of Scheme 1, so we do not need to recompute it. The lower triangular $2m \times 2m$ linear system in Step 2 needs $m^2 + 3m$ multiplications. Finally, Step 3 requires only $2(n + m)$ multiplications. Hence Procedure 2 requires $2n + 2m^2 + 5m$ multiplications to compute the scalar $v_k^T B_k v_k$, for an arbitrary vector v_k.

Now, we describe the k-th step of our novel scheme. In the first, we define

$$\beta_k = -\frac{g_k^T q_k}{q_k^T B_k q_k} \|q_k\|,$$

where q_k is an arbitrary vector satisfying (1.6), for example $q_k = -g_k$ or $q_k = -H_k g_k$, and B_k is defined by the compact limited memory BFGS formula (2.6). Note that the formula (2.6) leads to a positive definite matrix B_k and consequently $q_k^T B_k q_k > 0$, for any arbitrary non-zero vector q_k. We now define s_k based on the definition of β_k as follows

$$s_k := \begin{cases} \|g_0\| & \text{if } k = 0; \\ \beta_k & \text{if } k \geq 1,\ \mu_1 \leq \rho_{k-1} < \mu_2; \\ c_1 \beta_k & \text{if } k \geq 1,\ \rho_{k-1} \geq \mu_2 \end{cases} \tag{2.8}$$

where $c_1 > 1$, $0 < \mu_1 \leq \mu_2 \leq 1$, then compute Δ_k, the adaptive trust-region radius, by

$$\Delta_k = c^{p_k} s_k \tag{2.9}$$

in which p_k is the smallest integer in $\mathbb{N} \cup \{0\}$ that guarantee $\rho_k \geq \mu_1$. In view of our discussion, a new limited memory trust-region algorithm with adaptive radius is outlined in the following:

Algorithm 1: LMATR (Limited memory trust-region algorithm with adaptive radius)

```
Input: $x_0 \in \mathbb{R}^n$, $B_0 \in \mathbb{R}^{n\times n}$, $k_{max}$, $0 < \mu_1 \leq \mu_2 \geq 1$, $0 < c_1 \leq < 1$,
       $c_2 \geq 1$, $\epsilon > 0$;
Output: $x_b$; $f_b$;
1  begin
2      $\Delta_0 \leftarrow \|g_0\|$; $k \leftarrow 0$; $p \leftarrow 0$;
3      while $\|g_k\| \geq \epsilon$ & $k \leq k_{max}$ do
4          compute $q_k$ by (2.1) using Scheme 1;
5          compute $q_k^T B_k q_k$ using Scheme 2;
6          compute $s_k$ using (2.8);
7          solve the subproblem (1.2) to specify $d_k$;
8          $\hat{x}_{k+1} \leftarrow x_k + d_k$; compute $f(\hat{x}_{k+1})$;
9          compute $d_k^T B_k d_k$ using Scheme 2;
10         determine $\rho_k$ using (1.3);
11         while $\rho_k < \mu_1$ do
12             $p \leftarrow p + 1$;
13             solve the subproblem (1.2) to specify $d_k$;
14             $\hat{x}_{k+1} \leftarrow x_k + d_k$; compute $\hat{f}_{k+1} = f(\hat{x}_{k+1})$;
15             determine $\rho_k$ using (1.3);
16         end
17         $x_{k+1} \leftarrow \hat{x}_{k+1}$; $f_{k+1} = \hat{f}_{k+1}$; $m \leftarrow \min\{k, m_1\}$;
18         update $S_k$, $Y_k$, $D_k$, $L_k$, $Y_k^T Y_k$ and $R_k$;
19         $k \leftarrow k + 1$;
20     end
21     $x_b \leftarrow x_k$; $f_b \leftarrow f_k$;
22 end
```

The loop starts from Line 3 and ends at line 20 is called the outer loop, and the loop starts from Line 11 and ends at line 16 is called the inner loop.

3. **Convergence analysis**

This section is devoted to analyzing the global convergence of the proposed algorithm. We first give some properties of the algorithm and then investigate its global convergence to first-order stationary points. The local superlinear and quadratic convergence rates of the proposed algorithm are also established in the sequel.

To establish the global convergence property, we assume that the decrease on the model m_k is at least as much as a fraction of that obtained by Cauchy's point, i.e., there exists a constant $\beta \in (0, 1)$ such that, for all k,

$$m_k(x_k) - m_k(x_k + d_k) \geq \beta \|g_k\| \min \left[\Delta_k, \frac{\|g_k\|}{\|B_k\|} \right]. \tag{3.1}$$

This inequality is called the sufficient reduction condition and have been investigated by many authors when they extended some inexact methods for approximately solving the subproblem (1.2), for example see [12, 26, 27]. Inequality (3.1) implies that $d_k \neq 0$ whenever $g_k \neq 0$. Furthermore, throughout the paper, we also consider the following two assumptions in order to analyse the convergence properties of Algorithm 1:

(H1) The objective function $f(x)$ is twice continuously differentiable and has a lower bound on the upper level set $L(x_0) = \{x \in \mathbb{R}^n | f(x) \leq f(x_0),\ x_0 \in \mathbb{R}^n\}$.

(H2) The approximate Hessian matrix B_k is uniformly bounded, i.e., there exists a constant $M > 0$ such that $\|B_k\| \leq M$, for all $k \in \mathbb{N} \cup \{0\}$.

Remark 3.1. Suppose that the objective function $f(x)$ is a twice continuously differentiable function and the level set $L(x_0)$ is bounded. Then (H1) implies that $\|\nabla^2 f(x)\|$ is uniformly continuous and bounded above on the open bounded convex set Ω, containing $L(x_0)$. As a result, there exists a constant $L > 0$ such that $\|\nabla^2 f(x)\| \leq L$, for all $x \in \Omega$. Therefore, using the mean value theorem, it can be concluded that, for all $x, y \in \Omega$,

$$\|g(x) - g(y)\| \leq L\|x - y\|,$$

which leads to the fact that $f(x)$ is Lipschitz continuous in the open bounded convex set Ω.

In the next two lemmas, it will be proved that the inner loop of Algorithm 1 will be stopped after the finite number of steps.

Lemma 3.2. *Suppose that (H2) holds and the sequence $\{x_k\}$ is generated by Algorithm 1. Then, we have*

$$|f(x_k + d_k) - m_k(x_k + d_k))| = O(\|d_k\|^2).$$

Proof. Taylor's expansion along with Remark 3.1 and the definition of $m_k(d)$ imply that

$$\begin{aligned} |f(x_k + d_k) - m_k(x_k + d_k))| &\leq | - d_k^T G_k d_k + d_k^T B_k d_k| + O(\|d_k\|^2) \\ &= |d_k^T (B_k - G_k) d_k| + O(\|d_k\|^2) \\ &\leq (L + M)\|d_k\|^2 + O(\|d_k\|^2) \\ &= O(\|d_k\|^2). \end{aligned}$$

This completes the proof of the lemma. $\square$

Lemma 3.3. *Suppose that (H2) holds and the sequence $\{x_k\}$ is generated by Algorithm 1. Then, the inner loop of Algorithm 1 terminates after the finite number of steps.*

Proof. By contrary, we assume that an infinite cycle happens in Step 2 of Algorithm 1. Therefore, setting $p = p_k$ implies

$$\Delta_k^p \to 0, \text{ as } p \to \infty. \tag{3.2}$$

Since the current iteration x_k is not the optimum solution of problem (1.1), then there exists a positive constant ϵ so that $\|g_k\| \geq \epsilon$. This fact and (1.6) suggest that

$$m_k(x_k) - m_k(x_k + d_k^p) \geq \beta\epsilon \min\left[\Delta_k^p, \frac{\epsilon}{M}\right], \tag{3.3}$$

where d_k^p is the solution of the subproblem (1.2) corresponding to $p \in \mathbb{N} \cup \{0\}$. Now, by Lemma 3.2, (3.2) and (3.3), we obtain

$$\begin{aligned}\left|\frac{f(x_k) - f(x_k + d_k^p)}{m_k(x_k) - m_k(x_k + d_k^p)} - 1\right| &= \left|\frac{f(x_k + d_k^p) - m_k(x_k + d_k^p))}{m_k(x_k) - m_k(x_k + d_k^p)}\right| \\ &\leq \frac{O(\|d_k^p\|^2)}{\beta\epsilon \min\left[\Delta_k^p, \frac{\epsilon}{M}\right]} \leq \frac{O((\Delta_k^p)^2)}{\beta\epsilon\ \Delta_k^p} \to 0, \ (p \to \infty).\end{aligned}$$

Therefore, there exists a sufficiently large constant p_1 such that for all $p \geq p_1$, the inequality

$$\rho_k = \frac{f(x_k) - f(x_k + d_k^p)}{m_k(x_k) - m_k(x_k + d_k^p)} \geq \mu_1$$

holds. This fact indicates that there exists a finite nonnegative integer p_k, which is a contradiction with (3.2). Therefore, the inner loop of Algorithm 1 terminates after the finite number of steps. $\square$

In order to establish the global convergence to first-order critical points, we require that the following condition holds

$$m_k(x_k) - m_k(x_k + d_k) \geq \frac{c^{p_k}}{2M}\left[\frac{g_k^T q_k}{\|q_k\|}\right]^2, \text{ for all } k \in \mathbb{N} \cup \{0\}. \tag{3.4}$$

Hence the subsequent lemma plays an important role in proving the global convergence of the sequence $\{x_k\}$ generated by Algorithm 1.

Lemma 3.4. *Suppose that (H2) holds, the sequence $\{x_k\}$ is generated by Algorithm 1 and d_k is a solution of the subproblem (1.2). Then (3.4) holds.*

Proof. By setting

$$\hat{d}_k = -c^{p_k}\frac{g_k^T q_k}{q_k^T B_k q_k} q_k,$$

it is clear that $\|\hat{d}_k\| \leq \Delta_k$, i.e., $\hat{d}_k$ is a feasible point of the subproblem (1.2). On the other hand, from (1.6), we have $g_k^T q_k < 0$. Consequently, using these facts and the assumption (H2), we obtain

$$\begin{aligned} m_k(x_k) - m_k(x_k + d_k) \geq m_k(x_k) - m_k(x_k + \hat{d}_k) &= -g_k^T \hat{d}_k - \frac{1}{2}\hat{d}_k^T B_k \hat{d}_k \\ &= c^{p_k} \frac{g_k^T q_k}{q_k^T B_k q_k}\left[g_k^T q_k - \frac{1}{2}c^{p_k} g_k^T q_k\right] \\ &= c^{p_k} \frac{(g_k^T q_k)^2}{q_k^T B_k q_k}\left[1 - \frac{1}{2}c^{p_k}\right] \\ &\geq c^{p_k} \frac{(g_k^T q_k)^2}{q_k^T B_k q_k}\left[1 - \frac{1}{2}\right] \\ &= \frac{1}{2}c^{p_k} \ \frac{(g_k^T q_k)^2}{q_k^T B_k q_k} \\ &\geq \frac{c^{p_k}}{2M}\left[\frac{g_k^T q_k}{\|q_k\|}\right]^2, \end{aligned}$$

completing the proof. □

We here establish the global convergence property to first-order stationary points of Algorithm 1 under the mentioned assumptions.

Theorem 3.5. *Suppose that (H1) and (H2) hold and q_k satisfies (1.6). Then, Algorithm 1 either stops at a stationary point of (1.1) or generates an infinite sequence $\{x_k\}$ such that*

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{3.5}$$

Proof. If Algorithm 1 stops at a stationary point of the problem (1.1), then we have nothing to do. Otherwise, we first prove that

$$\lim_{k\to\infty} -\frac{g_k^T q_k}{\|q_k\|} = 0. \tag{3.6}$$

By contrary, we suppose that there exists a constant $\epsilon_0 > 0$ and an infinite subset $K \subseteq \mathbb{N} \cup \{0\}$ such that

$$-\frac{g_k^T q_k}{\|q_k\|} \geq \epsilon_0 \ \forall k \in K. \tag{3.7}$$

Lemma 3.4 and (3.7) imply that

$$f_k - f(x_k + d_k) \geq \mu_1(m_k(x_k) - m_k(x_k + d_k)) \geq \frac{\mu_1}{2M}c^{p_k}\left[\frac{g_k^T q_k}{\|q_k\|}\right]^2 \geq \frac{1}{2M}\mu_1 c^{p_k}\epsilon_0^2.$$

Taking an infinite summation from both sides of this inequality and using assumption (H1), we get

$$\frac{1}{2M}\mu_1\epsilon_0^2\sum_{k=0}^{\infty}c^{p_k} \leq \sum_{k=0}^{\infty}(f_k - f_{k+1}) = f_0 - \lim_{k\to\infty} f(x_k) < \infty.$$

This fact leads to

$$\lim_{k\to\infty} c^{p_k} = 0.$$

This equality clearly implies that $p_k \to \infty$ as $k \to \infty$, which is a contradiction with Lemma 3.3. Therefore, (3.6) holds.

Using (3.6) and the fact that the vector q_k satisfies (1.6), we obtain

$$0 \leq \tau\|g_k\| \leq -\frac{g_k^T q_k}{\|g_k\|.\|q_k\|}\ \|g_k\| = -\frac{g_k^T q_k}{\|q_k\|} \to 0,\ (k\to\infty).$$

Therefore, (3.5) holds and the proof is completed. □

Theorem 1 is the key step in investigating the convergence analysis of Algorithm 1. It obviously implies that, if the generated sequence $\{x_k\}$ of Algorithm 1 has limit points, then all of the limit points satisfies the first-order necessary condition.

In what follows, we will verify the superlinear and the quadratic convergence rates of Algorithm 1 under some classical conditions that have been widely used in nonlinear optimization literatures, see [26].

Theorem 3.6. *Suppose that assumptions (H1) and (H2) hold, the sequence $\{x_k\}$ is generated by Algorithm 1 converges to x^*, $d_k = -B_k^{-1}g_k$, the matrix $G(x) = \nabla^2 f(x)$ is continuous in a neighborhood $N(x^*,\epsilon)$ of x^*, and B_k satisfies the following condition*

$$\lim_{k\to\infty}\frac{\|[B_k - G(x^*)]d_k\|}{\|d_k\|} = 0.$$

Then, the sequence $\{x_k\}$ converges to x^* superlinearly. Moreover, if $B_k = G(x_k)$ and $G(x)$ is Lipschitz continuous in a neighborhood $N(x^*,\epsilon)$, then the sequence $\{x_k\}$ converges to x^* quadratically.

Proof. The proof is similar to what stated in the proof of Theorems 4.3 and 4.4 in [33] and therefore the details are omitted. □

4. Preliminary numerical experiments

In this section, we report some numerical results of LMATR on a large set of standard test problems taken from the references [5, 23] in which problem's dimensions are varied from 500 to 10000. To show the efficiency of LMATR, we compare its results with those obtained by the limited memory quasi-Newton algorithm (LMQNM), proposed in page 48 of [26], the limited memory version of the traditional trust-region algorithm (LMTTR) and the limited memory version of the adaptive trust-region algorithm of Shi and Guo [33] (LMTRS) with $q_k = -H_k g_k$.

All these algorithms take advantages of the compact limited memory BFGS with $m_1 = 5$ which is rewritten in MATLAB code due to L-BFGS-B Fortran code in [37], which is based on the literatures [9, 22, 36]. Furthermore, all the algorithms are coded with double precision format in MATLAB 7.4 and the trust-region subproblems are solved by a modified Steihaug-Toint procedure (see [12]), which employs the compact limited memory BFGS. Similar to Bastin et al. in [7], the Steihaug-Toint algorithm terminates at $x_k + d$ when

$$\|\nabla f(x_k) + B_k d\| \leq \min\left[\frac{1}{10}, \|g_k\|^{\frac{1}{2}}\right] \|g_k\| \text{ or } \|d\| = \Delta_k,$$

holds. Moreover, all the algorithms are stopped whenever the total number of iterations exceeds 20000 or the condition

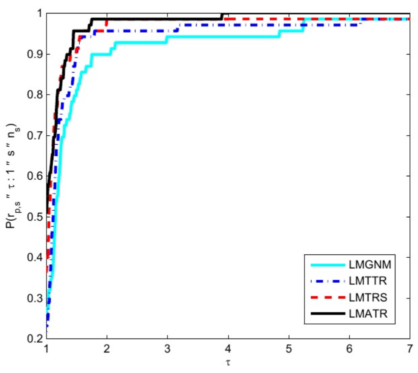
$$\|g_k\| \leq 10^{-5}$$

is satisfied. We set $\mu_1 = 0.05$ and $\mu_2 = 0.9$ for the algorithms LMATR, LMTTR and LMTRS. Besides, we exploit $c = 0.2$ and $c_1 = 1.55$ for LMATR, and the parameters of LMTRS are chosen as done in [33]. Similar to [12], LMTTR employs $\Delta_0 = \frac{1}{10}\|g_0\|$ and updates the trust-region radius by

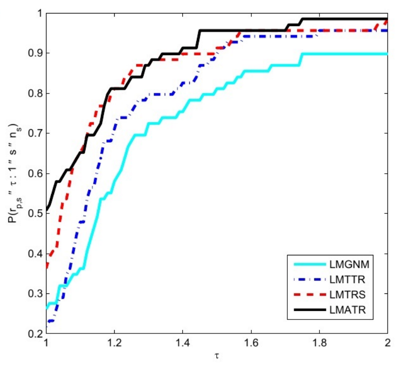
$$\Delta_{k+1} = \begin{cases} \alpha_1 \|d_k\| & \text{if } \rho_k < \mu_1; \\ \Delta_k & \text{if } \mu_1 \leq \rho_k < \mu_2; \\ \max\{\alpha_2 \|d_k\|,\ \Delta_k\} & \text{if } \rho_k \geq \mu_2, \end{cases}$$

where $\alpha_1 = 0.25$ and $\alpha_2 = 3.5$. During the run of the algorithms, we make sure that all of the codes converge to the same point. Therefore, we have just provided those results in which all the algorithms are convergent to an identical point. The results are summarized in Table 1.

Thanks to the structure of Algorithm 1 (LMATR) and the other presented algorithms, it is easy to see that the total number of iterations, N_i, are identical to the total number of gradient evaluations, N_g. Therefore, in Table 1, we only report the number of iterations, the number of function evaluations, N_f as efficiency measures. From Table 1, it can be seen that, in the most cases, both the number of iterations and function evaluations of Algorithm 1 is remarkably less than the others.

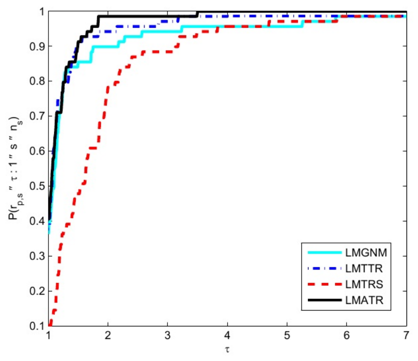


(A) $\tau = 7$

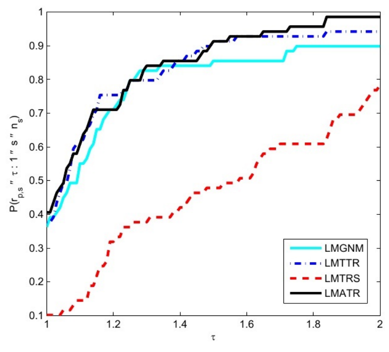


(B) $\tau = 2$

Figure 1. Performance profile for the number of iterations

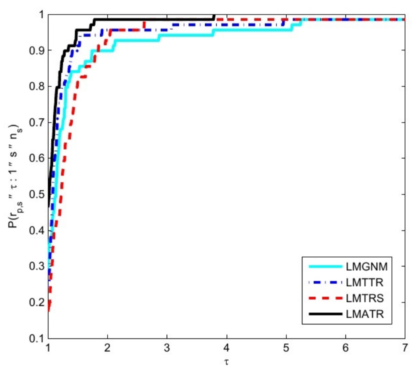


(A) $\tau = 7$

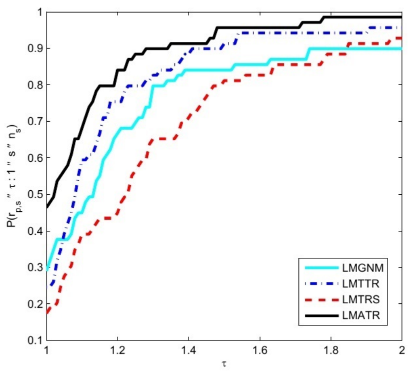


(B) $\tau = 2$

Figure 2. Performance profile for the number of function evaluations



(A) $\tau = 7$



(B) $\tau = 2$

Figure 3. Performance profile for $N_f + 3N_i$

Table 1. Numerical results

Problem name	Dim	LMQNM	LMTTR	LMTRS	LMATR
		n_i / n_f	n_i / n_f	n_i / n_f	n_i / n_f
POWER	500	4273/4397	6515/6860	6707/8061	6164/6543
Hager	500	35/40	36/41	38/49	38/49
Raydan 1	500	165/176	146/155	144/178	143/157
Gen. White and Holst	500	6423/9262	6390/8445	5764/15836	5577/9099
BIGGSB1	500	1409/1449	1368/1447	1365/1668	1521/1598
Gen. Rosenbrock	1000	6892/9686	6248/8041	5983/13247	5735/8362
Ext. White and Holst	1000	76/111	66/100	46/123	57/94
Partial per. quad.	1000	188/217	170/193	169/229	164/187
DIXON3DQ	1000	5327/5671	4692/4971	4849/6009	5156/5494
FLETCHCR	1000	5333/6271	5814/6423	7173/9008	4687/5559
TRIDIA	1000	871/895	868/924	790/920	631/669
Quad. QF2	1000	1322/1681	289/308	278/342	273/289
VARDIM	1000	77/112	81/114	84/245	81/111
ENGVAL1	1000	25/32	22/27	23/42	20/26
Diagonal 2	1000	145/160	150/162	140/167	141/164
Ext. tridiag. 2	1000	27/31	29/32	28/43	27/30
Ext. Penalty	1000	69/71	61/76	62/140	56/71
ARWHEAD	1000	14/16	9/15	9/48	10/17
EG2	1000	153/241	26/39	21/101	20/29
CUBE	1000	1263/2081	1903/2235	1646/6606	1280/2328
Ext. Maratos	1000	153/232	649/689	112/423	105/217
Ext. Powell	5000	69/77	71/94	49/198	64/112
Ext. Wood	5000	178/236	50/73	34/102	49/77
Per. quad.	5000	597/623	608/637	575/702	592/631
Per. tridiag. quad.	5000	557/583	635/676	591/693	555/595
Broyden tridiag.	5000	68/87	52/58	51/74	33/40
Almost per. quad.	5000	580/597	629/661	599/710	592/623
NONDQUAR	5000	1834/2036	2096/2542	2292/3949	2155/2746
EDENSCH	5000	26/30	20/25	21/39	21/24
Quad. QF1	5000	587/604	607/633	588/696	598/640
Ext. Fre. and Roth	5000	20/23	18/23	17/45	19/26
Ext. tridiag. 1	5000	27/31	28/31	24/44	32/38
BDEXP	5000	31/32	36/37	31/32	31/32
HIMMELBG	5000	2/3	37/38	2/3	2/3
QUARTC	5000	31/32	31/32	28/32	26/28
LIARWHD	5000	34/42	39/55	33/96	31/40
Ext. PSC1	5000	21/24	12/14	12/26	14/18
Ext. BD1	5000	14/17	15/19	14/17	14/17
Ext. quad. pen. QP1	5000	34/36	28/35	27/61	24/31
Ext. DENSCHNB	5000	10/12	10/11	9/13	9/11
Ext. DENSCHNF	5000	16/18	13/18	14/38	13/22
SINCOS	5000	21/24	12/14	12/26	14/18
COSINE	5000	17/20	14/17	15/26	15/18
Raydan 2	5000	8/9	7/8	8/9	8/9
Diag. 4	5000	4/6	6/9	8/23	7/11
Diag. 7	5000	11/14	7/8	7/9	7/10
Diag. 8	5000	6/8	6/7	7/11	5/7
NONSCOMP	5000	52/60	41/50	636/2466	34/40
DIXMAANE	6000	332/349	363/386	288/339	336/360
DIXMAANF	6000	257/268	252/262	209/241	245/259
DIXMAANG	6000	240/247	325/349	225/267	289/307
DIXMAANI	6000	2417/2478	2568/2744	811/965	3157/3376
DIXMAANH	9000	498/586	249/258	233/278	268/283
DIXMAANA	9000	8/11	9/10	9/17	10/13
DIXMAANB	9000	7/10	7/8	8/16	9/12
DIXMAANC	9000	9/12	10/12	10/20	9/12
DIXMAAND	9000	10/14	9/11	11/24	13/17
DIXMAANJ	9000	284/298	227/236	447/513	388/406
DIXMAANK	9000	376/382	403/414	414/458	349/364
DIXMAANL	9000	1656/1714	353/364	330/384	316/326
DQDRTIC	10000	10/14	18/30	15/33	14/23
Diag. 5	10000	5/6	7/8	5/6	5/6
NONDIA	10000	11/13	9/19	11/61	9/19
Ext.TET	10000	10/12	10/11	10/18	9/12
Ext. Beale	10000	18/20	18/19	16/27	19/23
Full Hessian FH3	10000	4/6	4/11	4/35	4/11
Gen. PSC1	10000	44/46	38/41	39/56	33/38
Ext. Himmelblau	10000	13/16	11/14	12/24	10/13
Ext. quad. exp. EP1	10000	6/8	4/8	4/25	4/9

Although the proposed algorithm is not the best algorithm in some problems, it has better computational performances. We also take the advantages of the performance profile of Dolan and Moré in [13], which is a statistical tools to compare the efficiency of algorithms. Therefore, we illustrate the results of Table 1 in Figures 1-3 according to the total number of iterations, the total number of function evaluations and the efficiency measure $N_f + 3N_i$, respectively.

From Figure 1, it is observed that LMATR attains the most wins among the other considered algorithms. More precisely, it solves about 51% of test functions efficiently and faster than the others. We also see that LMTRS performs better than LMQNM and LMTTR regarding of the total number of iterations. Moreover, considering the ability of completing the run successfully, LMATR is the best algorithm among the others, where its figure grows up faster than the other algorithms. This means that whenever the proposed algorithm is not the best algorithm, it performs close to the best algorithm. Figure 2 shows that LMATR, LMQNM and LMTTR are so competitive regarding to the total number of function evaluations, however they perform better than LMTRS. Furthermore, the results of LMATR have the most wins in about 40% of test functions. We also use the efficiency measure $N_f + 3N_i$ in Figure 3, where it show that LMATR outperforms the others. Our preliminary computational experiments show that LMATR is promising for solving large-scale unconstrained optimization problems.

5. **Concluding remarks**

We present an iterative scheme for solving large-scale unconstrained optimization problems based on a trust-region framework equipped with an adaptive radius and the compact limited memory BFGS. As it is known, using an appropriate adaptive radius can decrease the total number of subproblems solving in trust-region methods. We describe some disadvantages of the adaptive trust-region radius of Shi and Guo in [33], especially for solving large problems. To overcome these drawbacks, we propose some reformations for this radius. Moreover, the limited memory quasi-Newton schemes have been developed to cope with large-scale optimization problems. We therefore unify this two interesting ideas into a trust-region algorithm to decrease the computational cost compared with the traditional trust-region framework.

From the theoretical analysis point of view, the proposed algorithm inherits the global convergence of traditional trust-region algorithms to the first-order stationary points under classical assumptions. The superlinear and the quadratic convergence rates are also established. Finally, our preliminary numerical experiments on the set of standard test problems point out that the proposed algorithm is efficient for solving large-scale unconstrained optimization problems.

Acknowledgement

The authors are grateful for the valuable comments and suggestions of anonymous referees.

References

[1] M. Ahookhosh and K. Amini, A nonmonotone trust region method with adaptive radius for unconstrained optimization problems, *Comput. Math. Appl.* **60** (2010), no. 3, 411–422.

[2] M. Ahookhosh, H. Esmaeili and M. Kimiaei, An effective trust-region-based approach for symmetric nonlinear systems, *Int. J. Comput. Math.* **90** (2013), no. 3, 671–690.

[3] K. Amini and M. Ahookhosh, A hybrid of adjustable trust-region and nonmonotone algorithms for unconstrained optimization, *Appl. Math. Model.* **38** (2014), no. 9-10, 2601–2612.

[4] K. Amini and M. Ahookhosh, Combination adaptive trust region method by nonmonotone strategy for unconstrained nonlinear programming, *Asia-Pac. J. Oper. Res.* **28** (2011), no. 5, 585–600.

[5] N. Andrei, An unconstrained optimization test functions collection, *Adv. Model. Optim.* **10** (2008), no. 1, 147–161.

[6] D. Ataee Tarzanagh, M. R. Peyghami and H. Mesgarani, A new nonmonotone trust region method for unconstrained optimization equipped by an efficient adaptive radius, *Optim. Methods Softw.* **29** (2014), no. 4, 819–836.

[7] F. Bastin, V. Malmedy, M. Mouffe, Ph. L. Toint and D. Tomanos, A retrospective trust-region method for unconstrained optimization, *Math. Programming* **123** (2008), no. 2, 395–418.

[8] J. V. Burke, A. Wiegmann and L. Xu, Limited memory BFGS updating in a trust-region framework, *SIAM Journal on Optimization.* submitted, 2008.

[9] R. H. Byrd, P. Lu and J. Nocedal, A limited memory algorithm for bound constrained optimization, *SIAM J. Sci. Comput.* **16** (1995), no. 5, 1190–1208.

[10] R. Byrd, J. Nocedal and R. Schnabel, Representation of quasi-Newton matrices and their use in limited memory methods, *Math. Programming* **63** (1994), no. 2, 129–156.

[11] A. R. Conn, N. I. M. Gould and Ph. L. Toint, LANCELOT. A Fortran package for large-scale nonlinear optimization (release A), Springer Series in Computational Mathematics, 17, Springer-Verlag, Berlin, 1992.

[12] A. R. Conn, N. I. M. Gould and Ph. L. Toint, Trust-Region Methods, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2000.

[13] E. Dolan and J. J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.* **21** (2002), no. 2, 201–213.

[14] H. Esmaeili and M. Kimiaei, An efficient adaptive trust-region method for systems of nonlinear equations, *Int. J. Comput. Math.* **92** (2015), no. 1, 151–166.

[15] H. Esmaeili and M. Kimiaei, A new adaptive trust-region method for system of nonlinear equations, *Appl. Math. Model.* **38** (2014), no. 11–12, 3003–3015.

[16] R. Fletcher, Practical Method of Optimization, Wiley, NewYork, 2000.

[17] N. Gould, S. Lucidi, M. Roma and Ph. L. Toint, Solving the trust-region subproblem using the Lanczos method, *SIAM J. Optim.* **9** (1999), no. 2, 504–525.

[18] N. I. M. Gould, D. Orban and A. Sartenaer and P. L. Toint, Sensitivity of trust-region algorithms to their parameters, *4OR* **3** (2005), no. 3, 227–241.

[19] D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Programming* **45** (1989), no. 3, 503–528.

[20] L. Kaufman, Reduced storage, quasi-Newton trust region approaches to function optimization, *SIAM J. Optim.* **10** (1999), no. 1, 56–69.

[21] J. L. Morales and J. Nocedal, Enriched methods for large-scale unconstrained optimization, *Comput. Optim. Appl.* **21** (2002), no. 2, 143–154.

[22] J. L. Morales and J. Nocedal, L-BFGS-B: Remark on Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization, to appear in *ACM Trans. Math. Software*, 2011.
[23] J. J. Moré, B. S. Garbow and K. E. Hillström, Testing Unconstrained Optimization Software, *ACM Trans. Math. Software* **7** (1981) 17–41.
[24] S. G. Nash and J. Nocedal, A numerical study of the limited memory BFGS method and the truncated Newton method for large scale optimization, *SIAM J. Optim.* **1** (1991), no. 3, 358–372.
[25] J. Nocedal, Updating quasi-Newton matrices with limited storage, *Mathematics of Computation.* **35** (1980), no. 151, 773–782.
[26] J. Nocedal and S. J. Wright, Numerical Optimization, Springer, New York, 2006.
[27] J. Nocedal and Y. Yuan, Combining trust region and line search techniques, in: Y. Yuan (Ed.), Advanced in nonlinear programming, 153–175, Kluwer Acad. Publ., Dordrecht, 1998.
[28] M. J. D. Powell, A new algorithm for unconstrained optimization, In: Rosen JB, Mangasarian OL, Ritter K (eds) Nonlinear programming, 31–66, Academic Press, New York, 1970.
[29] M. J. D. Powell, On the global convergence of trust region algorithms for unconstrained minimization, *Math. Programming* **29** (1984) 297–303.
[30] A. Sartenaer, Automatic determination of an initial trust region in nonlinear programming, *SIAM J. Sci. Statist. Comput.* **18** (1997), no. 6, 1788–1803.
[31] R. B. Schnabel and E. Eskow, A new modified Cholesky factorization, *SIAM J. Sci. Statist. Comput.* **11** (1990), no. 6, 1136–1158.
[32] G. A. Schultz, R. B. Schnabel and R. H. Byrd, A family of trust-region-based algorithms for unconstrained minimization with strong global convergence, *SIAM J. Numer. Anal.* **22** (1985), no. 1, 47–67.
[33] Z. J. Shi and J. H. Guo, A new trust region method with adaptive radius, *Comput. Optim. Appl.*, **213** (2008), no. 2, 509–520.
[34] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, *SIAM J. Numer. Anal.* **20** (1983), no. 3, 626–637.
[35] X. S. Zhang, J. L. Zhang and L. Z. Liao, An adaptive trust region method and its convergence, *Sci. China Ser. A* **45** (2002), no. 5, 620–631.
[36] C. Zhu, R. H. Byrd and J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization, *ACM Trans. Math. Software* **23** (1997), no. 4, 550–560.
[37] http://users.eecs.northwestern.edu/ nocedal/lbfgsb.html

(Masoud Ahookhosh) Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria.
E-mail address: `masoud.ahookhosh@univie.ac.at`

(Keyvan Amini) Department of Mathematics, Razi University, Kermanshah, Iran.
E-mail address: `kamini@razi.ac.ir`

(Morteza Kimiaei) Department of Mathematics, Asadabad Branch, Islamic Azad University, Asadabad, Iran.
E-mail address: `morteza.Kimiaei@gmail.com`

(M. Reza Peyghami) Department of Mathematics, K. N. Toosi University of Technology, P.O. Box 16315-1618, Tehran, Iran.
E-mail address: `peyghami@kntu.ac.ir`