# A New Trust Region Method with Simple Model for Large-Scale Optimization *

Qunyan Zhou[†], Wenyu Sun[‡] and Hongchao Zhang[§]

**Abstract**

In this paper a new trust region method with simple model for solving large-scale unconstrained nonlinear optimization is proposed. By employing generalized weak quasi-Newton equations, we derive several schemes to construct variants of scalar matrices as the Hessian approximation used in the trust region subproblem. Under some reasonable conditions, global convergence of the proposed algorithm is established in the trust region framework. The numerical experiments on solving the test problems with dimensions from 50 to 20000 in the CUTEr library are reported to show efficiency of the algorithm.

**Key words.** unconstrained optimization, Barzilai-Borwein method, weak quasi-Newton equation, trust region method, global convergence
**AMS(2010)** 65K05, 90C30

## 1. Introduction

In this paper, we consider the unconstrained optimization problem

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

where $f : R^n \to R$ is continuously differentiable with dimension $n$ relatively large. Trust region methods are a class of powerful and robust globalization

*This work was supported by National Natural Science Foundation of China (11171159, 11401308), the Natural Science Fund for Universities of Jiangsu Province (13KJB110007), the Fund of Jiangsu University of Technology (KYY13012).

[†]School of Mathematics and Physics, Jiangsu University of Technology, Changzhou 213001, Jiangsu, China. Email: zhouqunyan@jsut.edu.cn

[‡]Corresponding author. School of Mathematical Sciences, Jiangsu Key Laboratory for NSLSCS, Nanjing Normal University, Nanjing 210023, China. Email: wysun@njnu.edu.cn

[§]Department of Mathematics, Louisiana State University, USA. Email: hozhang@math.lsu.edu

methods for solving (1.1). The trust region method can be traced back to Levenberg (1944) [14] and Marquardt (1963) [18] for solving nonlinear least-squares problems (see [32]). However, the modern versions of trust region methods were first proposed by Winfield (1969, 1973) [36, 37] and Powell (1970, 1975) [23, 24]. Since the region method usually have strong global and local convergence properties (see [4, 22, 32]), it has attracted extensive research in the optimization field, e.g., Byrd, Schnabel and Schultz [3, 27], Di and Sun [9], Moré [20], Toint [34] and Yuan [38]. More recently, a comprehensive review monograph on the trust region methods was given by Conn, Gould and Toint [4]. The trust region method is also an iterative method, in which the following trust region subproblem

$$\begin{aligned} &\min \ q_k(s) = f_k + g_k^T s + \frac{1}{2} s^T B_k s \\ &\text{s.t.} \quad \|s\| \le \Delta_k \end{aligned} \tag{1.2}$$

needs to be solved at each iteration, where $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, $B_k \in R^{n\times n}$ is the Hessian or the Hessian approximation of $f$ at the current iterate $x_k$, and $\Delta_k > 0$ is the trust region radius. Suppose $s_k$ is an exact or an approximate solution of the trust region subproblem (1.2). The ratio $\rho_k$ between the actual function value reduction $f_k - f(x_k + s_k)$ and the predicted function value reduction $q_k(0) - q_k(s_k)$ plays an important role on deciding whether the trial step $s_k$ would be acceptable or not and how the trust region radius would be adjusted for the next iteration. Essentially, the trust region method is certain generalization of the line search techniques. It is more flexible and often more efficient than the Levenberg-Marquardt method. At the same time, in some extent, it is equivalent to certain regularization approaches.

Solving the trust region subproblem (1.2) is a critical step in the trust region algorithm. Many authors have studied this issue and proposed several methods (see [21, 22, 32]) for solving the trust region subproblem efficiently. However, as the problem dimension getting larger and larger, the standard quasi-Newton trust region methods turns out to be less efficient, since even full storage of the Hessian or its approximations becomes prohibitive, no mention the need of solving the subproblem (1.2). Several approaches had been done to deal with these difficulties. For example, Schubert [26] gave a modification of a quasi-Newton method with sparse Jacobian; Liu and Nocedal [16] applied a limited memory BFGS approximation of the Hessian ; Toint [33] discussed an efficient sparsity with Newton-type method; Wang and Yuan [35] and Hager [13] suggested subspace approaches to approximately solving the trust region subproblems.

The goal of this paper is to develop a nonmonotone trust-region method based on a simple model. Our method is a gradient based first-order method, which could be also regarded as a generalization of the Barzilai-Borwein (BB) two-point gradient method [1, 5, 6, 25, 32, 42]. Because of its simplicity, low memory requirement and only first order information is used, it is very suitable for solving large-scale optimization problems. In our method, the Hessian approximation $B_k$ is taken as a real positive definite scalar matrix $\gamma_k I$ for some $\gamma_k \geq 0$, which will inherit certain quasi-Newton properties. Then, the subproblem (1.2) can be written as

$$\begin{aligned} &\min \ q_k(s) = f_k + g_k^T s + \frac{1}{2}\gamma_k s^T s, \\ &\text{s.t.} \quad \|s\| \leq \Delta_k. \end{aligned} \tag{1.3}$$

Suppose $\|g_k\| \neq 0$. The solution of (1.3) can be easily solved as follows:

(i) if $\|g_k\| \leq \gamma_k \Delta_k$, $s_k = -\frac{1}{\gamma_k} g_k$;

(ii) if $\|g_k\| > \gamma_k \Delta_k$, the optimal solution $s_k$ of (1.3) will be on the boundary of the trust region [32], i.e., $s_k$ is the solution of the following problem

$$\begin{aligned} &\min \ q_k(s) = f_k + g_k^T s + \frac{1}{2}\gamma_k s^T s, \\ &\text{s.t.} \quad \|s\| = \Delta_k. \end{aligned} \tag{1.4}$$

From (1.4), we have the solution $s_k = -\frac{\Delta_k}{\|g_k\|} g_k$. Hence, in general, the solution of (1.3) can be written as

$$s_k = -\frac{1}{\tilde{\gamma}_k} g_k, \quad \text{where } \tilde{\gamma}_k = \max\left\{\gamma_k, \frac{\|g_k\|}{\Delta_k}\right\}.$$

So the trust region technique essentially provides an adaptive positive lower bound for the scalar $\gamma_k$ to prevent the stepsize $\|s_k\|$ from being unreasonably large. One the other hand, how to select the parameter $\gamma_k$ will play a key role for the success of the method. In this paper, motivated from the BB method, we propose a few strategies on determining the parameter $\gamma_k$.

As mentioned above, in this paper, to improve the efficiency of the trust region methods, a nonmonotone technique is applied into the framework of the trust region methods. In fact, the nonmonotone technique was originally developed with the purpose of overcoming the so called Maratos Effect [17], which could lead to the rejection of superlinear convergent steps since these steps could cause an increase in both the objective function value and the constraint violation. Later, Grippo et. al. [12] proposed a nonmonotone Newton method, in which a line search is performed so that the step size $\alpha_k$ satisfies the following condition:

$$f(x_k + \alpha_k d_k) \leq f_k^r + \beta \alpha_k g_k^T d_k, \tag{1.5}$$

where $\beta \in (0,1)$ and the reference function value $f_k^r = \max_{0 \leq j \leq m_k} \{f(x_{k-j})\}$ with $m_0 = 0, 0 \leq m_k \leq \min\{m_{k-1}+1, M\}$ for $k \geq 1$, and $M \geq 0$ being an integer parameter. By (1.5), the reference function value $f_k^r$ is guaranteed monotonically nonincreasing. Usually, (1.5) is called a maximum-value nonmonotone rule. Deng etc. [7] and Sun [30] generalized the above nonmonotone rule into the trust region framework. However, this rule has some drawbacks. For example, a good function value generated at some iterations could be completely thrown away due to the maximum principle in (1.5) for selecting reference function values. Motivated by this observation, Hager and Zhang [40] proposed another nonmonotone line search strategy, in which the maximum function value rule in (1.5) is replaced by a weighted average of function values of previous iterates. That is, their nonmonotone rule requires the decrease of a weighted average of the previous function values. More precisely, their method finds a step length $\alpha_k$ satisfying

$$f(x_k + \alpha_k d_k) \leq C_k + \beta \alpha_k g_k^T d_k, \tag{1.6}$$

where $\beta \in (0, 1)$,

$$C_k = \begin{cases} f(x_k), & k = 0; \\ \frac{\eta_{k-1} Q_{k-1} C_{k-1} + f(x_k)}{Q_k}, & k \geq 1, \end{cases} \quad Q_k = \begin{cases} 1, & k = 0; \\ \eta_{k-1} Q_{k-1} + 1, & k \geq 1, \end{cases} \tag{1.7}$$

with parameters $\eta_{k-1} \in [\eta_{\min}, \eta_{\max}], \eta_{\min} \in [0, 1)$ and $\eta_{\max} \in [\eta_{\min}, 1]$ Numerical results show that (1.6) is often superior to (1.5) (see [40]) for quasi-Newton type methods. In this paper we apply the weighted average nonmonotone scheme to the trust region framework.

The organization of the paper is as follows. In Section 2, the algorithm of our nonmonotone trust region method with simple model is presented. Based on the generalized weak quasi-Newton equations, several schemes for computing the scalar matrix $B_k = \gamma_k I$ are proposed in Section 3. We establish the global convergence of our algorithm in Section 4. Numerical experiments based on the CUTEr [2] problem library are reported in Section 5 to show the efficiency of our algorithm. Finally, some concluding remarks are given in Section 6.

## 2. Algorithm

Let $s_k$ be the solution of the simple model subproblem (1.3). The predicted reduction of the objective function value by the model is defined as

$$\text{Pred}(s_k) = q_k(0) - q_k(s_k),$$

and the actual reduction of the objective function value is given as

$$\text{Ared}(s_k) = C_k - f(x_k + s_k).$$

Define the ratio

$$\rho_k = \frac{\text{Ared}(s_k)}{\text{Pred}(s_k)} = \frac{C_k - f(x_k + s_k)}{q_k(0) - q_k(s_k)}, \tag{2.1}$$

where $C_k$ is computed by (1.7). Same as the standard trust region strategy, if $\text{Ared}(s_k)$ is satisfactory compared with $\text{Pred}(s_k)$, we will finish the current iteration by taking $x_{k+1} = x_k + s_k$ and adjust the trust region radius properly; otherwise, we will resolve the subproblem (1.3) again at the iterate $x_k$ with a reduced trust region radius.

Now we state the trust region method with simple model as follows.

**Algorithm 2.1** (Trust Region Method with Simple Model (TRMSM))

Step 0. Given $x_0 \in R^n,\ 0 < \Delta_0,\ 0 < \mu < \nu_1 < \nu_2 < 1,\ c_1 \in (0,1),$ $1 < c_3 < c_2,\ 0 < \gamma_{\max},\ \eta_{\min} \in [0,1)$ and $\eta_{\max} \in [\eta_{\min}, 1]$. Set $k = 0,\ \gamma_0 = 1,\ C_0 = f(x_0)$.

Step 1. If $\|g_k\|$ is sufficiently small, stop.

Step 2. Solve the subproblem (1.3) for $s_k$.

Step 3. Compute $\text{Ared}(s_k)$, $\text{Pred}(s_k)$ and $\rho_k$.

Step 4. If $\rho_k < \mu$, set $\Delta_k = c_1 \Delta_k$ and go to Step 2.

Step 5. Set $x_{k+1} = x_k + s_k$. Compute $\Delta_{k+1}$ by

$$\Delta_{k+1} = \begin{cases} c_2 \Delta_k, & \text{if} \quad \rho_k \geq \nu_2 \ \text{ and } \ \|s_k\| = \Delta_k, \\ c_3 \Delta_k, & \text{if} \quad \rho_k \geq \nu_1, \\ \Delta_k, & \text{otherwise.} \end{cases}$$

Step 6. Compute $\gamma_{k+1} = \gamma^*_{k+1}$ or $\gamma_{k+1} = \gamma^{**}_{k+1}(\theta)$ with $\theta \geq 0$ according to the expressions in section 3. Set $\gamma_{k+1} = \max\{0, \min\{\gamma_{k+1}, \gamma_{\max}\}\}$.

Step 7. Choose $\eta_k \in [\eta_{\min}, \eta_{\max}]$ and compute $C_{k+1}$. Set $k := k+1$, and go to Step 1.

**Remark 2.2** The purpose of Step 6 is to avoid uphill directions and to keep the sequence $\{\gamma_k\}$ uniformly bounded. In fact, for all $k$, we have

$$0 \leq \gamma_k \leq \gamma_{\max}. \tag{2.2}$$

## 3. Several Schemes to Determine Scalar $\gamma_k$

In this section, we propose several strategies on how to determine the scalar $\gamma_k$ in Algorithm 2.1, which would be a critical issue for the success of our algorithm. It is well known that the classic quasi-Newton equation

$$B_{k+1} s_k = y_k, \tag{3.1}$$

where $s_k = x_{k+1} - x_k, y_k = g_{k+1} - g_k$, can be derived from the following quadratic model

$$q_{k+1}(s) = f_{k+1} + g_{k+1}^T s + \frac{1}{2} s^T B_{k+1} s. \tag{3.2}$$

The quadratic model (3.2) is an approximation of the objective function at $x_{k+1}$ and satisfies the following three interpolation conditions:

$$q_{k+1}(0) = f_{k+1}, \tag{3.3}$$

$$\nabla q_{k+1}(0) = g_{k+1}, \tag{3.4}$$

$$\nabla q_{k+1}(-s_k) = g_k, \tag{3.5}$$

where $f_{k+1} = f(x_{k+1})$ and $g_{k+1} = \nabla f(x_{k+1})$. It is usually difficult to satisfy the quasi-Newton equation (3.1) with a nonsingular scalar matrix [11]. Hence, we need some alternative conditions that can maintain the accumulated curvature information along the negative gradient as correct as possible. Zhang and Xu [41] proposed some modified quasi-Newton equations. Dennis and Wolkowicz [8] introduced a weaker form by projecting the quasi-Newton equation (3.1) in the direction $s_k$, that is requiring

$$s_k^T B_{k+1} s_k = s_k^T y_k. \tag{3.6}$$

Following this idea: starting from some weak quasi-Newton equation, we would derive a scalar matrix $B_k = \gamma_k I$ used in our trust region model (1.3). If we restrict $B_{k+1}$ to be a scalar matrix $\overline{\gamma}_{k+1} I$, then the weak quasi-Newton equation (3.6) gives

$$\overline{\gamma}_{k+1} = \frac{s_k^T y_k}{s_k^T s_k}, \tag{3.7}$$

which is in fact just the same BB parameter proposed in [1]. Furthermore, if we left-multiply $y_k$ in (3.1), then we have

$$y_k^T B_{k+1} s_k = y_k^T y_k. \tag{3.8}$$

Setting $B_{k+1} = \hat{\gamma}_{k+1} I$, the previous weak quasi-Newton equation (3.8) gives

$$\hat{\gamma}_{k+1} = \frac{y_k^T y_k}{s_k^T y_k}, \tag{3.9}$$

which is another often used BB parameter [1].

Motivated by the BB method, we further discuss two new generalized weak quasi-Newton equations, which can be regarded as extensions of (3.6). Based on these new proposed quasi-Newton equations, we propose two new classes of scalar matrices as the Hessian approximation used in our model (1.3).

### 3.1. Scheme I

By only using the iterate difference $s_k$ and the corresponding gradient difference $y_k$, (3.7) and (3.9) just utilize the data from the two most recent iterates to construct the Hessian approximation, and they are known as the two-point first-order methods. In order for $\gamma_{k+1}I$ to carry more accurate curvature information of the Hessian, we would like to use multi-point information from more than two points and consider some generalization of the weak quasi-Newton equation.

Consider a differentiable curve $x(\tau) \in R^n$ and the derivative of $g(x(\tau)) := \nabla f(x(\tau))$ at some point $x(\tau^*)$, which can be obtained by the chain rule

$$\left.\frac{dg(x(\tau))}{d\tau}\right|_{\tau=\tau^*} = G(x(\tau))\left.\frac{dx(\tau)}{d\tau}\right|_{\tau=\tau^*}, \tag{3.10}$$

where $\tau$ is a parameter and $G(x(\tau))$ is the Hessian matrix at $x(\tau)$. We are interested in deriving a relation that will be satisfied by the approximation of the Hessian at $x_{k+1}$. Assume that $x(\tau)$ passes through $x_{k+1}$ and choose $\tau^*$ so that $x(\tau^*) = x_{k+1}$. Then it follows from (3.10) that

$$G(x_{k+1})\frac{dx(\tau^*)}{d\tau} = \frac{dg(x(\tau^*))}{d\tau}. \tag{3.11}$$

Now, we try to find a new weak quasi-Newton equation which uses the information at the most recent three iterates $x_{k-1}, x_k$ and $x_{k+1}$. Consider $x(\tau)$ as the interpolating polynomial of degree 2 satisfying

$$x(\tau_j) = x_{k+j-1}, \quad j = 0, 1, 2,$$

and define

$$\frac{dx(\tau_2)}{d\tau} = r_k, \quad \frac{dg(x(\tau_2))}{d\tau} = w_k.$$

Applying (3.11), the quasi-Newton equation will be generalized to

$$B_{k+1}r_k = w_k, \tag{3.12}$$

and a generalized weak quasi-Newton equation can be derived as

$$r_k^T B_{k+1} r_k = r_k^T w_k. \tag{3.13}$$

Furthermore, if $B_{k+1}$ in (3.13) is set to be a scalar matrix $\gamma_{k+1}^* I$, then we have

$$\gamma_{k+1}^* = \frac{r_k^T w_k}{r_k^T r_k}, \tag{3.14}$$

which can be regarded as a generalized BB parameter.

How to determine $r_k$ and $w_k$? To determine their expressions, the values of $\tau_0$, $\tau_1$ and $\tau_2$ are needed. Without loss of generality, we set $\tau_0 = -1$, $\tau_1 = 0$ and $\tau_2 = 1$. Then, the Newton interpolating polynomial $x(\tau)$ is given as

$$\begin{aligned} x(\tau) &= x[\tau_0] + x[\tau_0, \tau_1](\tau - \tau_0) + x[\tau_0, \tau_1, \tau_2](\tau - \tau_0)(\tau - \tau_1) \\ &= x[-1] + x[-1, 0](\tau + 1) + x[-1, 0, 1](\tau + 1)\tau \end{aligned} \tag{3.15}$$

(see [28, 31]). Here,

$$x[0] = x(0) = x_k, \;\; x[-1] = x(-1) = x_{k-1},$$

$$x[-1, 0] = \frac{x[0] - x[-1]}{0 - (-1)} = x(0) - x(-1) = x_k - x_{k-1},$$

and

$$x[-1, 0, 1] = \frac{x[0, 1] - x[-1, 0]}{1 - (-1)} = \frac{1}{2}(x_{k+1} - 2x_k + x_{k-1}).$$

Thus, it follows from (3.15) that

$$x(\tau) = x_{k-1} + (\tau + 1)(x_k - x_{k-1}) + \frac{\tau(\tau + 1)}{2}(x_{k+1} - 2x_k + x_{k-1}).$$

So

$$\left.\frac{dx(\tau)}{d\tau}\right|_{\tau=\tau_2} = x_k - x_{k-1} + \frac{3}{2}(x_{k+1} - 2x_k + x_{k-1}) = \frac{3}{2}s_k - \frac{1}{2}s_{k-1},$$

i.e.,

$$r_k = \frac{3}{2}s_k - \frac{1}{2}s_{k-1}. \tag{3.16}$$

Using the same arguments as above, $w_k$ can be derived as

$$w_k = \frac{3}{2}y_k - \frac{1}{2}y_{k-1}. \tag{3.17}$$

So, (3.14) with (3.16)-(3.17) would give a proper formula for determining the scalar $r^*_{k+1}$.

### 3.2. Scheme II

In the following, we generalize (3.6) in another way. One can view the weak quasi-Newton equation (3.6) as a projection of the quasi-Newton equation in a direction $v$ such that $v^T B_{k+1} s_k = v^T y_k \neq 0$. The choice of $v$ may influence the quality of the curvature information provided by the weak quasi-Newton equation. Hence, Yuan [39] introduced a weak quasi-Newton equation directly derived from an interpolation emphasizing more on function values rather than from the projection of the quasi-Newton equation. More precisely, the information of two successive function values is used in his weak quasi-Newton equation and the quadratic model function (3.2) is required to satisfy the interpolation conditions (3.3), (3.4) and

$$q_{k+1}(-s_k) = f_k, \tag{3.18}$$

which replaces (3.5) used in constructing the model (3.2). Moreover, it is pointed out in [29, 32] that the interpolation condition (3.18) is very important in the iterative procedure because it emphasizes more use of the function value information, and that some non-quadratic function models (e.g., conic function models) also possess this property (see [9]). By using (3.2), (3.3), (3.4) and (3.18), one can get

$$s_k^T B_{k+1} s_k = 2(f_k - f_{k+1} + s_k^T g_{k+1}) \tag{3.19}$$

which is just the weak quasi-Newton equation proposed in [39].

Now, by considering a weighted combination of the weak quasi-Newton equations (3.6) and (3.19), we have another generalized weak quasi-Newton equation

$$\begin{aligned} s_k^T B_{k+1} s_k &= (1-\theta) s_k^T y_k + \theta[2(f_k - f_{k+1}) + 2 s_k^T g_{k+1}] \\ &= s_k^T y_k + \theta[2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k], \end{aligned} \tag{3.20}$$

where $\theta \geq 0$ is the weight parameter. If $B_{k+1}$ is set to be a scalar matrix $\gamma_{k+1}^{**}(\theta) I$, then (3.20) yields

$$\gamma_{k+1}^{**}(\theta) = \frac{s_k^T y_k + \theta[2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k]}{s_k^T s_k}. \tag{3.21}$$

It is easy to see that formula (3.21) would be reduced to (3.7) if $f$ is quadratic on the line segment between $x_k$ and $x_{k+1}$. For general nonlinear functions, since both the function values and gradient information at $x_k$ and $x_{k+1}$ are used in (3.21), we might expect that the formula (3.21) will be better than the standard BB formula (3.7). Essentially, (3.21) can be also regarded as an extension of the BB formula (3.21).

The following theorem gives the property of $\gamma_{k+1}^{**}(\theta)$.

**Theorem 3.1** *Suppose that $f$ is sufficiently smooth. If $\|s_k\|$ is small enough, then we have*

$$\begin{aligned} & s_k^T G_{k+1} s_k - \gamma_{k+1}^{**}(\theta) s_k^T s_k \\ = & \left(\frac{1}{2} - \frac{\theta}{6}\right) T_{k+1} \otimes s_k^3 - \left(\frac{1}{6} - \frac{\theta}{12}\right) V_{k+1} \otimes s_k^4 + \mathcal{O}(\|s_k\|^5), \end{aligned} \tag{3.22}$$

*where $G_{k+1} \in R^{n\times n}$ is the Hessian of $f$ at $x_{k+1}$, $\otimes$ is an appropriate tensor product, $T_{k+1} \in R^{n\times n\times n}$ and $V_{k+1} \in R^{n\times n\times n\times n}$ are the tensors of $f$ at $x_{k+1}$ satisfying*

$$T_{k+1} \otimes s_k^3 = \sum_{i,j,l=1}^{n} \frac{\partial^3 f(x_{k+1})}{\partial x^i \partial x^j \partial x^l} s_k^i s_k^j s_k^l$$

*and*

$$V_{k+1} \otimes s_k^4 = \sum_{i,j,l,m=1}^{n} \frac{\partial^4 f(x_{k+1})}{\partial x^i \partial x^j \partial x^l \partial x^m} s_k^i s_k^j s_k^l s_k^m.$$

**Proof.** Using the Taylor formula, we get

$$f_k = f_{k+1} - g_{k+1}^T s_k + \frac{1}{2} s_k^T G_{k+1} s_k - \frac{1}{6} T_{k+1} \otimes s_k^3 + \frac{1}{24} V_{k+1} \otimes s_k^4 + \mathcal{O}(\|s_k\|^5) \quad (3.23)$$

and

$$g_k^T s_k = g_{k+1}^T s_k - s_k^T G_{k+1} s_k + \frac{1}{2} T_{k+1} \otimes s_k^3 - \frac{1}{6} V_{k+1} \otimes s_k^4 + \mathcal{O}(\|s_k\|^5). \quad (3.24)$$

Then from (3.21), (3.23) and (3.24) we have

$$\begin{aligned} s_k^T G_{k+1} s_k - \gamma_{k+1}^{**}(\theta) s_k^T s_k &= s_k^T G_{k+1} s_k - s_k^T y_k - \theta[2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k] \\ &= \left(\frac{1}{2} - \frac{\theta}{6}\right) T_{k+1} \otimes s_k^3 - \left(\frac{1}{6} - \frac{\theta}{12}\right) V_{k+1} \otimes s_k^4 + \mathcal{O}(\|s_k\|^5). \end{aligned}$$

This completes the proof. □

By the formula (3.21), if $\theta = 0$, we have

$$\gamma_{k+1}^{**}(0) = \frac{s_k^T y_k}{s_k^T s_k} \quad (3.25)$$

which is the BB formula $\overline{\gamma}_{k+1}$ given in (3.7). Then, it follows from (3.22) that

$$s_k^T G_{k+1} s_k - \gamma_{k+1}^{**}(0) s_k^T s_k = \frac{1}{2} T_{k+1} \otimes s_k^3 - \frac{1}{6} V_{k+1} \otimes s_k^4 + \mathcal{O}(\|s_k\|^5). \quad (3.26)$$

From this equation (3.26) and the Theorem 3.1, it is reasonable to believe that if the parameter $\theta$ is chosen such that

$$\left|\frac{1}{2} - \frac{\theta}{6}\right| < \frac{1}{2} \quad \text{and} \quad \left|\frac{1}{6} - \frac{\theta}{12}\right| < \frac{1}{6},$$

i.e., $0 < \theta < 4$, then $\gamma_{k+1}^{**}(\theta) s_k^T s_k$ may capture the second order curvature $s_k^T G_{k+1} s_k$ with a higher precision than $\overline{\gamma}_{k+1} s_k^T s_k$ does.

In the following, let us further consider several possible choices of $\theta$ and the corresponding formulas for $\gamma_{k+1}^{**}(\theta)$:

(1) Setting $\theta = 1$, then

$$\gamma_{k+1}^{**}(1) = \frac{s_k^T y_k + 2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k}{s_k^T s_k}. \tag{3.27}$$

The resulting matrix $\gamma_{k+1}^{**}(1)I$ satisfies the weak quasi-Newton equation (3.19). By (3.22), we have

$$s_k^T G_{k+1} s_k - \gamma_{k+1}^{**}(1) s_k^T s_k = \frac{1}{3} T_{k+1} \otimes s_k^3 - \frac{1}{12} V_{k+1} \otimes s_k^4 + \mathcal{O}(\|s_k\|^5). \tag{3.28}$$

(2) Setting $\theta = 2$, then

$$\gamma_{k+1}^{**}(2) = \frac{s_k^T y_k + 4(f_k - f_{k+1}) + 2(g_k + g_{k+1})^T s_k}{s_k^T s_k}. \tag{3.29}$$

It follows from (3.22) that

$$s_k^T G_{k+1} s_k - \gamma_{k+1}^{**}(2) s_k^T s_k = \frac{1}{6} T_{k+1} \otimes s_k^3 + \mathcal{O}(\|s_k\|^5). \tag{3.30}$$

(3) Setting $\theta = 3$, then

$$\gamma_{k+1}^{**}(3) = \frac{s_k^T y_k + 6(f_k - f_{k+1}) + 3(g_k + g_{k+1})^T s_k}{s_k^T s_k}. \tag{3.31}$$

By (3.22), we obtain that

$$s_k^T G_{k+1} s_k - \gamma_{k+1}^{**}(3) s_k^T s_k = \frac{1}{12} V_{k+1} \otimes s_k^4 + \mathcal{O}(\|s_k\|^5). \tag{3.32}$$

## 4. Convergence Analysis

In this section, we discuss the global convergence of Algorithm 2.1. This analysis follows the convergence analysis framework of the trust region methods (see [4, 22, 32]). But it is based on the simple model and the average-value nonmonotone technique proposed in [40].

**Lemma 4.1** *Suppose $\|g_k\| \neq 0$. The solution $s_k$ of the simple model (1.3) satisfies*

$$\text{Pred}(s_k) = q_k(0) - q_k(s_k) \geq \frac{1}{2}\|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\gamma_k}\right\}. \tag{4.1}$$

**Proof.** If $\|g_k\| \leq \gamma_k \Delta_k$, then $s_k = -\frac{1}{\gamma_k} g_k$. Hence, we have

$$\begin{aligned}
\text{Pred}(s_k) &= -g_k^T \left(-\frac{1}{\gamma_k} g_k\right) - \frac{1}{2}\left(-\frac{1}{\gamma_k} g_k\right)^T \gamma_k I \left(-\frac{1}{\gamma_k} g_k\right) \\
&= \frac{\|g_k\|^2}{\gamma_k} - \frac{1}{2}\frac{\|g_k\|^2}{\gamma_k} \\
&= \frac{1}{2}\frac{\|g_k\|^2}{\gamma_k}. \qquad (4.2)
\end{aligned}$$

If $\|g_k\| > \gamma_k \Delta_k$, then $s_k = -\frac{\Delta_k}{\|g_k\|} g_k$. Hence, we have

$$\begin{aligned}
\text{Pred}(s_k) &= -g_k^T \left(-\frac{\Delta_k}{\|g_k\|} g_k\right) - \frac{1}{2}\left(-\frac{\Delta_k}{\|g_k\|} g_k\right)^T \gamma_k I \left(-\frac{\Delta_k}{\|g_k\|} g_k\right) \\
&= \Delta_k \|g_k\| - \frac{1}{2}\gamma_k \Delta_k^2 \\
&> \Delta_k \|g_k\| - \frac{1}{2}\Delta_k \|g_k\| \\
&= \frac{1}{2}\Delta_k \|g_k\|. \qquad (4.3)
\end{aligned}$$

It follows from (4.2) and (4.3) that (4.1) holds. □

**Lemma 4.2** *Let $\{x_k\}$ be the sequence generated by Algorithm 2.1, then we have*

$$f_{k+1} \leq C_{k+1} \leq C_k, \quad \textit{for any } k \geq 0. \qquad (4.4)$$

**Proof.** The proof is similar to Lemma 3.1 in [19] and Theorem 2.2 in [40]. □

**Lemma 4.3** *Algorithm 2.1 is well defined, i.e., if $\|g_k\| \neq 0$, the Algorithm 2.1 will not cycle infinitely between Step 2 and Step 4.*

**Proof.** We prove this lemma by way of contradiction. Suppose that Algorithm 2.1 cycles infinitely between Step 2 and Step 4. We define the cycling index at the iteration $k$ by $k(i)$, then we have

$$\rho_{k(i)} < \mu, \quad \text{for all } i = 1, 2, \cdots, \qquad (4.5)$$

and $\triangle_{k(i)} \to 0$ as $i \to \infty$.

Because $f$ is continuously differentiable and $\|s_{k(i)}\| \leq \triangle_{k(i)}$, by the Taylor's theorem, for $i$ large enough we have

$$\begin{aligned}
&| f_k - f(x_k + s_{k(i)}) - (q_k(0) - q_k(s_{k(i)})) | \\
&= \left| -\int_0^1 s_{k(i)}^T g(x_k + t s_{k(i)}) dt - f_k + f_k + s_{k(i)}^T g(x_k) + \frac{1}{2}\gamma_k s_{k(i)}^T s_{k(i)} \right| \\
&= \left| \frac{1}{2}\gamma_k s_{k(i)}^T s_{k(i)} - \int_0^1 s_{k(i)}^T [g(x_k + t s_{k(i)}) - g(x_k)] dt \right| \\
&\leq \mathcal{O}(\gamma_k \triangle_{k(i)}^2) + o(\triangle_{k(i)}). \qquad (4.6)
\end{aligned}$$

Then, it follows from (4.1), (4.6) and (2.2) that

$$\left|\frac{f_k - f(x_k + s_{k(i)})}{q_k(0) - q_k(s_{k(i)})} - 1\right| \leq \frac{\gamma_{\max}\mathcal{O}(\triangle_{k(i)}^2) + o(\triangle_{k(i)})}{\frac{1}{2}\|g_k\| \min\{\Delta_{k(i)}, \frac{\|g_k\|}{\gamma_{\max}}\}} \to 0 \quad \text{as } i \to \infty,$$

which implies

$$\lim_{i\to\infty} \frac{f_k - f(x_k + s_{k(i)})}{q_k(0) - q_k(s_{k(i)})} = 1. \tag{4.7}$$

Combining (2.1), (4.7) and Lemma 4.2, we obtain

$$\rho_{k(i)} = \frac{C_k - f(x_k + s_{k(i)})}{q_k(0) - q_k(s_{k(i)})} \geq \frac{f_k - f(x_k + s_{k(i)})}{q_k(0) - q_k(s_{k(i)})}.$$

This inequality together with (4.7) imply that $\rho_{k(i)} \geq \mu$ for sufficiently large $i$, which contradicts (4.5). □

**Theorem 4.4** *Suppose that $f$ is bounded below. Let $\{x_k\}$ be the sequence generated by Algorithm 2.1, then we have*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{4.8}$$

**Proof.** We show (4.8) by way of contradiction. Suppose that there exists a constant $\tau > 0$ such that

$$\|g_k\| \geq \tau, \quad \text{for all } k. \tag{4.9}$$

It follows from $\rho_k \geq \mu$, (2.1) and Lemma 4.1 that

$$f_{k+1} \leq C_k - \mu \text{Pred}(s_k) \leq C_k - \frac{1}{2}\mu\|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\gamma_k}\right\}. \tag{4.10}$$

Combining the definition of $C_k$, (4.10) and (2.2), we can obtain

$$\begin{aligned} C_{k+1} &= \frac{\eta_k Q_k C_k + f_{k+1}}{Q_{k+1}} \\ &\leq \frac{\eta_k Q_k C_k + C_k - \frac{1}{2}\mu\|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\gamma_k}\}}{Q_{k+1}} \\ &= C_k - \frac{\frac{1}{2}\mu\|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\gamma_{\max}}\}}{Q_{k+1}}, \end{aligned}$$

which together with (4.9) imply that

$$C_k - C_{k+1} \geq \frac{\frac{1}{2}\mu\tau \min\{\Delta_k, \frac{\tau}{\gamma_{\max}}\}}{Q_{k+1}}. \tag{4.11}$$

From Lemma 4.2, we know that $f_k \leq C_k$ for all $k \geq 0$ and $\{C_k\}$ is monotonically nonincreasing. Then, it follows from the assumption $f$ being bounded below that $\{C_k\}$ is convergent. So we have from (4.11) that

$$\sum_{k=0}^{\infty} \frac{\min\{\Delta_k, \frac{\tau}{\gamma_{\max}}\}}{Q_{k+1}} < \infty. \tag{4.12}$$

By (1.7) and the fact that $\eta_k \in [\eta_{\min}, \eta_{\max}] \subset [0, 1]$, we have

$$Q_{k+1} = 1 + \sum_{i=0}^{k} \prod_{m=0}^{i} \eta_{k-m} \leq k + 2. \tag{4.13}$$

Combining (4.12) and (4.13) yields

$$\sum_{k=0}^{\infty} \frac{\min\{\Delta_k, \frac{\tau}{\gamma_{\max}}\}}{k+2} < \infty.$$

Because $\sum_{k=0}^{\infty} \frac{1}{k+2}$ is divergent, the above inequality implies that there exists an infinite index set $\mathcal{T}$ such that

$$\lim_{k\to\infty,\, k\in\mathcal{T}} \Delta_k = 0. \tag{4.14}$$

Without loss of generality, we can assume that for all $k \in \mathcal{T}$ there are more than one inner cycles performed in the loop between Step 2 and Step 4 at the $k$-th iterate. So, the solution $\tilde{s}_k$ of the following subproblem

$$\begin{aligned} \min\ & q_k(s) = f_k + g_k^T s + \frac{1}{2}\gamma_k s^T s \\ \text{s.t.}\quad & \|s\| \leq \frac{\Delta_k}{c_1},\ k \in \mathcal{T} \end{aligned}$$

is not accepted at the $k$-th iterate for all $k \in \mathcal{T}$, which means

$$\rho_k = \frac{C_k - f(x_k + \tilde{s}_k)}{q_k(0) - q_k(\tilde{s}_k)} < \mu,\ k \in \mathcal{T}. \tag{4.15}$$

On the other hand, by (4.9), (4.14) and a similar proof of Lemma 4.3, we can have $\rho_k \geq \mu$ for all sufficiently large $k \in \mathcal{T}$, which contradicts (4.15). Hence, we have (4.8) holds. □

## 5. Numerical Results

In this section, to analyze the effectiveness of Algorithm 2.1, we perform Algorithm 2.1 on a set of 56 nonlinear unconstrained optimization problems in the CUTEr [2] library with dimensions ranging from 50 to 20000. The codes are written in Fortran 95 using double precision. All tests are performed on an ASUS laptop (Intel Core2 Duo, 2.93GHz, 2GRAM) under Fedora 8 Linux using the gfortran compiler (version 4.1.2) with default options.

For all our tests, the parameters in Algorithm 2.1 are as follows: $\Delta_0 = \|g_0\|, \mu = 0.1, \nu_1 = 0.5, \nu_2 = 0.75, c_1 = 0.5, c_2 = 2, c_3 = 1.5, \gamma_{\max} = 10^6, \eta_k = 1$. All together, the following six algorithms are tested and compared:

(1) GBB: The BB method with nonmonotone line search [25];

(2) TRMSM1: Algorithm 2.1 with $\gamma_{k+1} = \gamma^{**}_{k+1}(0)$;

(3) TRMSM2: Algorithm 2.1 with $\gamma_{k+1} = \gamma^{*}_{k+1}$ using (3.16)-(3.17);

(4) TRMSM3: Algorithm 2.1 with $\gamma_{k+1} = \gamma^{**}_{k+1}(1)$;

(5) TRMSM4: Algorithm 2.1 with $\gamma_{k+1} = \gamma^{**}_{k+1}(2)$;

(6) TRMSM5: Algorithm 2.1 with $\gamma_{k+1} = \gamma^{**}_{k+1}(3)$.

To be consistent, the same nonmonotone technique (1.5) is employed in all the testing algorithms . The stopping condition for all the testing algorithms is

$$\|g_k\|_\infty \le 10^{-5}(1 + |f(x_k)|).$$

In addition, the algorithm is stopped if the number of iterations exceed 10000. And in such case, we claim fail of this algorithm.

The numerical results of GBB, TRMSM1 and TRMSM2 are given in Table 1, and the numerical results of TRMSM3, TRMSM4 and TRMSM5 are given in Table 2. These two tables include the name of the problem (Function), the dimension of the problem ($n$), the number of function evaluations (NF), the number of iterations (Iter) and the final objective function value (Fval). The sign " $-$ " means the algorithm fails because the number of iterations exceeds 10000. By these results, we can see all the testing problems are successfully solved by TRMSM2 and TRMSM5. The performance profiles [10] using different metrics are given in Figure 1 and Figure 2. In this comparison, we delete the problems "CHNROSNB", "FLETCBV3" and'MODBEALE", since for these 3 problems different algorithms converge to different minimizers.

Figure 1 shows the performance profiles of GBB, TRMSM1 and TRMSM2. By Figure 1, we can see that for this set of testing problems, TRMSM1 often uses less number of iterations and function evaluations than GBB method. Although both GBB and TRMSM1 search along the negative gradient direction and use the same nonmonotone technique, they are different in the way of choosing the steplength due to the differences of trust-region method and line-search method. On the other hand, we can obviously see from Figure 1 that TRMSM2, which is based on our generalized weak quasi-Newton equation (3.13) using more function and gradient information, performs slightly better than TRMSM1.

Figure 2 displays the performance profiles of TRMSM1, TRMSM3, TRMSM4 and TRMSM5. By Figure 2, we can see that TRMSM3, TRMSM4 and TRMSM5 are more effective than TRMSM1. This is not surprising since we have seen from the analysis in Section 3 that the new simple model in TRMSM3, TRMSM4 and TRMSM5 can asymptotically provide better Hessian approximations than the model used in TRMSM1. One the other hand, we can also see from Figure 2 that TRMSM5 gives the overall best performance for this set of testing problems among all the four comparing algorithms. However, it seems still hard to draw a conclusion on which algorithm would perform best for a more general set of testing problems.

## 6. Conclusion

In this paper, we propose a new trust region method with simple model for solving large-scale unconstrained optimization with objective function continuously differentiable. The underlying ideas are to use the generalized weak quasi-Newton equations as well as the scalar matrix approximation of the Hessian to generate a trust region algorithm with simple model. The nonmonotone technique based on weighted average function values [40] is combined with the trust region method to improve the algorithm's efficiency. Both the theoretical analysis and our preliminary numerical results indicate that our proposed new methods could be promising alternatives of the existing methods for solving smooth large-scale unconstrained optimization.

**Table 1.** Numerical results of GBB, TRMSM1 and TRMSM2

| Function | $n$ | GBB | | | TRMSM1 | | | TRMSM2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | NF | Iter | Fval | NF | Iter | Fval | NF | Iter | Fval |
| ARGLINA | 200 | 4 | 3 | 2.00E+02 | 3 | 2 | 2.00E+02 | 3 | 2 | 2.00E+02 |
| ARWHEAD | 5000 | 62 | 17 | 0.00E+00 | 26 | 11 | 0.00E+00 | 29 | 14 | 0.00E+00 |
| BDQRTIC | 5000 | 227 | 82 | 2.00E+04 | 268 | 170 | 2.00E+04 | 220 | 146 | 2.00E+04 |
| BOX | 10000 | 171 | 26 | -1.86E+03 | 220 | 136 | -1.86E+03 | 117 | 72 | -1.86E+03 |
| BROWNAL | 200 | 63 | 15 | 1.38E-09 | 26 | 10 | 1.47E-09 | 27 | 11 | 1.47E-09 |
| BROYDN7D | 5000 | – | – | – | 4010 | 2560 | 1.86E+03 | 3735 | 2389 | 1.87E+03 |
| BRYBND | 5000 | 73 | 37 | 5.57E-12 | 45 | 33 | 1.02E-10 | 42 | 30 | 1.31E-11 |
| CHNROSNB | 50 | 958 | 864 | 3.24E-11 | 1268 | 879 | 3.92E+00 | 1015 | 783 | 3.93E+00 |
| COSINE | 10000 | – | – | – | 13 | 11 | -1.00E+04 | 13 | 11 | -1.00E+04 |
| CRAGGLVY | 5000 | 4402 | 3534 | 1.78E+03 | 1539 | 1048 | 1.69E+03 | 187 | 134 | 1.69E+03 |
| CURLY10 | 10000 | 158 | 134 | -1.00E+06 | 226 | 162 | -1.00E+06 | 169 | 118 | -1.00E+06 |
| CURLY20 | 10000 | 360 | 304 | -1.00E+06 | 473 | 296 | -1.00E+06 | 385 | 249 | -1.00E+06 |
| CURLY30 | 10000 | 938 | 791 | -1.00E+06 | 380 | 236 | -1.00E+06 | 387 | 240 | -1.00E+06 |
| DIXMAANA | 3000 | 16 | 8 | 1.00E+00 | 11 | 8 | 1.00E+00 | 12 | 9 | 1.00E+00 |
| DIXMAANB | 3000 | 18 | 8 | 1.00E+00 | 11 | 7 | 1.00E+00 | 12 | 8 | 1.00E+00 |
| DIXMAANC | 3000 | 23 | 10 | 1.00E+00 | 13 | 8 | 1.00E+00 | 14 | 9 | 1.00E+00 |
| DIXMAAND | 3000 | 26 | 10 | 1.00E+00 | 15 | 9 | 1.00E+00 | 16 | 10 | 1.00E+00 |
| DIXMAANE | 3000 | 56 | 48 | 1.00E+00 | 283 | 280 | 1.00E+00 | 294 | 291 | 1.00E+00 |
| DIXMAANF | 3000 | 483 | 375 | 1.00E+00 | 396 | 392 | 1.00E+00 | 239 | 235 | 1.00E+00 |
| DIXMAANG | 3000 | 579 | 468 | 1.00E+00 | 266 | 261 | 1.00E+00 | 281 | 276 | 1.00E+00 |
| DIXMAANH | 3000 | 1061 | 769 | 1.00E+00 | 410 | 404 | 1.00E+00 | 283 | 277 | 1.00E+00 |
| DIXMAANI | 3000 | 60 | 52 | 1.00E+00 | 622 | 401 | 1.00E+00 | 641 | 413 | 1.00E+00 |
| DIXMAANJ | 3000 | 209 | 198 | 1.01E+00 | 125 | 121 | 1.00E+00 | 132 | 128 | 1.00E+00 |
| DIXMAANL | 3000 | 210 | 159 | 1.00E+00 | 123 | 117 | 1.00E+00 | 104 | 98 | 1.00E+00 |
| DIXON3DQ | 10000 | 3254 | 2537 | 3.92E-03 | 4498 | 2838 | 4.80E-03 | 4713 | 2983 | 3.88E-03 |
| DQDRTIC | 5000 | 63 | 39 | 2.44E-13 | 34 | 26 | 1.15E-13 | 31 | 23 | 1.01E-12 |
| EDENSCH | 2000 | 42 | 18 | 1.20E+04 | 32 | 24 | 1.20E+04 | 29 | 21 | 1.20E+04 |
| EG2 | 1000 | 20 | 10 | -9.37E+02 | 14 | 5 | -9.99E+02 | 14 | 5 | -9.99E+02 |
| ENGVAL1 | 5000 | 30 | 14 | 5.55E+03 | 20 | 12 | 5.55E+03 | 22 | 14 | 5.55E+03 |
| FLETCBV2 | 5000 | 2 | 2 | -5.00E-01 | 2 | 2 | -5.00E-01 | 2 | 2 | -5.00E-01 |
| FLETCBV3 | 5000 | 5 | 5 | -1.74E+09 | 9 | 9 | -9.75E+04 | 9 | 9 | -9.65E+04 |
| FLETCHCR | 1000 | 858 | 805 | 2.19E-12 | 1064 | 893 | 1.65E-13 | 955 | 721 | 5.40E-11 |
| FMINSRF2 | 5625 | 6889 | 3477 | 1.00E+00 | 1445 | 1064 | 1.00E+00 | 1054 | 744 | 1.00E+00 |
| FMINSURF | 5625 | 6828 | 3399 | 1.00E+00 | 1710 | 1127 | 1.00E+00 | 1238 | 815 | 1.00E+00 |
| FREUROTH | 5000 | 99 | 43 | 6.08E+05 | 133 | 81 | 6.08E+05 | 184 | 114 | 6.08E+05 |
| GENROSE | 500 | 5481 | 3560 | 1.00E+00 | 5917 | 3740 | 1.00E+00 | 5387 | 3411 | 1.00E+00 |
| LIARWHD | 5000 | 340 | 82 | 4.40E-20 | 163 | 95 | 1.53E-16 | 118 | 68 | 1.17E-15 |
| MODBEALE | 20000 | 1244 | 634 | 5.56E-11 | – | – | – | 1319 | 998 | 3.03E+00 |
| MOREBV | 5000 | 50 | 21 | 2.58E-09 | 35 | 26 | 2.89E-09 | 43 | 27 | 2.02E-09 |
| NONDIA | 5000 | 95 | 20 | 3.70E-09 | 45 | 19 | 1.52E-09 | 33 | 13 | 4.51E-09 |
| PENALTY1 | 1000 | – | – | – | 146 | 91 | 9.69E-03 | 202 | 129 | 9.69E-03 |
| PENALTY2 | 200 | 66 | 2 | 4.71E+13 | 23 | 2 | 4.71E+13 | 23 | 2 | 4.71E+13 |
| POWELLSG | 5000 | 156 | 126 | 6.72E-05 | 212 | 134 | 4.69E-05 | 179 | 114 | 3.42E-05 |
| SCHMVETT | 5000 | 26 | 22 | -1.50E+04 | 14 | 12 | -1.50E+04 | 23 | 21 | -1.50E+04 |
| SENSORS | 100 | 57 | 43 | -2.09E+03 | 22 | 18 | -2.11E+03 | 23 | 19 | -2.11E+03 |
| SINQUAD | 5000 | 50 | 22 | -6.75E+06 | 33 | 21 | -6.76E+06 | 38 | 25 | -6.76E+06 |
| SPARSQUR | 10000 | 92 | 35 | 2.27E-07 | 43 | 28 | 3.01E-07 | 38 | 23 | 2.39E-07 |
| SROSENBR | 5000 | 58 | 19 | 1.22E-08 | 33 | 17 | 6.25E-09 | 51 | 29 | 2.00E-12 |
| TOINTGOR | 50 | 106 | 89 | 1.37E+03 | 138 | 99 | 1.37E+03 | 109 | 99 | 1.37E+03 |
| TOINTGSS | 5000 | 27 | 26 | 1.00E+01 | 3 | 2 | 1.00E+01 | 3 | 2 | 1.00E+01 |
| TOINTPSP | 50 | 254 | 185 | 2.26E+02 | 255 | 174 | 2.26E+02 | 210 | 146 | 2.56E+02 |
| TOINTQOR | 50 | 47 | 35 | 1.18E+03 | 33 | 29 | 1.18E+03 | 34 | 30 | 1.18E+03 |
| TQUARTIC | 5000 | 5708 | 898 | 5.33E-06 | – | – | – | 8847 | 5608 | 5.96E-04 |
| TRIDIA | 5000 | 2839 | 2777 | 1.24E-12 | 3651 | 2772 | 2.96E-12 | 3674 | 3056 | 1.15E-11 |
| VAREIGVL | 50 | 38 | 31 | 2.75E-12 | 32 | 29 | 1.48E-13 | 29 | 26 | 5.57E-11 |
| WOODS | 4000 | 329 | 287 | 1.57E-10 | 709 | 474 | 1.02E-10 | 525 | 394 | 1.04E-09 |

**Table 2.** Numerical results of TRMSM3, TRMSM4 and TRMSM5

| Function | $n$ | TRMSM3 | | | TRMSM4 | | | TRMSM5 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | NF | Iter | Fval | NF | Iter | Fval | NF | Iter | Fval |
| ARGLINA | 200 | 3 | 2 | 2.00E+02 | 3 | 2 | 2.00E+02 | 3 | 2 | 2.00E+02 |
| ARWHEAD | 5000 | 26 | 11 | 0.00E+00 | 26 | 11 | 0.00E+00 | 27 | 12 | 1.11E-12 |
| BDQRTIC | 5000 | 195 | 129 | 2.00E+04 | 166 | 103 | 2.00E+04 | 235 | 139 | 2.00E+04 |
| BOX | 10000 | 211 | 131 | -1.86E+03 | 212 | 131 | -1.86E+03 | 228 | 142 | -1.86E+03 |
| BROWNAL | 200 | 25 | 9 | 1.47E-09 | 25 | 9 | 1.47E-09 | 25 | 9 | 1.47E-09 |
| BROYDN7D | 5000 | 3583 | 2369 | 1.86E+03 | 3948 | 2559 | 1.84E+03 | 3870 | 2464 | 1.85E+03 |
| BRYBND | 5000 | 48 | 36 | 2.59E-13 | 45 | 33 | 7.73E-12 | 40 | 28 | 1.66E-11 |
| CHNROSNB | 50 | 1334 | 984 | 6.12E-12 | 1352 | 981 | 1.37E-12 | 1460 | 980 | 1.12E-11 |
| COSINE | 10000 | 13 | 11 | -0.1E+04 | 12 | 10 | -1.00E+04 | 13 | 11 | -1.00E+04 |
| CRAGGLVY | 5000 | 146 | 108 | 1.69E+03 | 222 | 162 | 1.69E+03 | 150 | 110 | 1.69E+03 |
| CURLY10 | 10000 | 75 | 73 | -1.00E+06 | 126 | 84 | -1.00E+06 | 89 | 55 | -1.00E+06 |
| CURLY20 | 10000 | 175 | 112 | -1.00E+06 | 189 | 117 | -1.00E+06 | 712 | 452 | -1.00E+06 |
| CURLY30 | 10000 | 726 | 458 | -1.00E+06 | 698 | 452 | -1.00E+06 | 562 | 349 | -1.00E+06 |
| DIXMAANA | 3000 | 10 | 7 | 1.00E+00 | 11 | 8 | 1.00E+00 | 11 | 8 | 1.00E+00 |
| DIXMAANB | 3000 | 11 | 7 | 1.00E+00 | 11 | 7 | 1.00E+00 | 11 | 7 | 1.00E+00 |
| DIXMAANC | 3000 | 13 | 8 | 1.00E+00 | 13 | 8 | 1.00E+00 | 13 | 8 | 1.00E+00 |
| DIXMAAND | 3000 | 15 | 9 | 1.00E+00 | 15 | 9 | 1.00E+00 | 15 | 9 | 1.00E+00 |
| DIXMAANE | 3000 | 229 | 226 | 1.00E+00 | 252 | 249 | 1.00E+00 | 222 | 219 | 1.00E+00 |
| DIXMAANF | 3000 | 353 | 349 | 1.00E+00 | 219 | 215 | 1.00E+00 | 304 | 300 | 1.00E+00 |
| DIXMAANG | 3000 | 218 | 213 | 1.00E+00 | 306 | 301 | 1.00E+00 | 212 | 207 | 1.00E+00 |
| DIXMAANH | 3000 | 229 | 223 | 1.00E+00 | 249 | 243 | 1.00E+00 | 206 | 200 | 1.00E+00 |
| DIXMAANI | 3000 | 993 | 626 | 1.00E+00 | 823 | 629 | 1.00E+00 | 551 | 548 | 1.00E+00 |
| DIXMAANJ | 3000 | 181 | 177 | 1.00E+00 | 123 | 119 | 1.00E+00 | 106 | 102 | 1.00E+00 |
| DIXMAANL | 3000 | 115 | 109 | 1.00E+00 | 111 | 105 | 1.00E+00 | 128 | 122 | 1.00E+00 |
| DIXON3DQ | 10000 | 6830 | 4313 | 4.94E-03 | 8198 | 5214 | 5.12E-03 | 5144 | 3267 | 5.15E-03 |
| DQDRTIC | 5000 | 34 | 26 | 1.15E-13 | 34 | 26 | 1.15E-13 | 34 | 26 | 1.15E-13 |
| EDENSCH | 2000 | 29 | 21 | 1.20E+04 | 28 | 20 | 1.20E+04 | 26 | 18 | 1.20E+04 |
| EG2 | 1000 | 13 | 4 | -9.99E+02 | 13 | 4 | -9.99E+02 | 14 | 5 | -9.99E+02 |
| ENGVAL1 | 5000 | 22 | 14 | 5.55E+03 | 15 | 8 | 5.55E+03 | 21 | 13 | 5.55E+03 |
| FLETCBV2 | 5000 | 2 | 2 | -5.00E-01 | 2 | 2 | -5.00E-01 | 2 | 2 | -5.00E-01 |
| FLETCBV3 | 5000 | 9 | 9 | -9.62E+04 | 9 | 9 | -1.14E+05 | 8 | 8 | -4.96E+04 |
| FLETCHCR | 1000 | 1327 | 1027 | 1.26E-10 | 1439 | 1058 | 2.68E-11 | 879 | 647 | 4.98E-12 |
| FMINSRF2 | 5625 | 1184 | 905 | 1.00E+00 | 1071 | 743 | 1.00E+00 | 1013 | 720 | 1.00E+00 |
| FMINSURF | 5625 | 1506 | 1024 | 1.00E+00 | 1939 | 1245 | 1.00E+00 | 1513 | 1024 | 1.00E+00 |
| FREUROTH | 5000 | 66 | 38 | 6.08E+05 | 57 | 30 | 6.08E+05 | 60 | 37 | 6.08E+05 |
| GENROSE | 500 | 5977 | 3779 | 1.00E+00 | 5684 | 3599 | 1.00E+00 | 5621 | 3561 | 1.00E+00 |
| LIARWHD | 5000 | 145 | 84 | 1.19E-15 | 136 | 79 | 2.62E-08 | 144 | 83 | 6.10E-19 |
| MODBEALE | 20000 | 643 | 481 | 1.35E-11 | 2244 | 1542 | 3.03E+00 | 887 | 615 | 1.42E-11 |
| MOREBV | 5000 | 35 | 26 | 2.29E-09 | 34 | 22 | 2.73E-09 | 35 | 26 | 2.29E-09 |
| NONDIA | 5000 | 49 | 19 | 1.53E-09 | 61 | 26 | 3.66E-09 | 49 | 19 | 4.32E-08 |
| PENALTY1 | 1000 | 76 | 41 | 9.69E-03 | 74 | 39 | 9.69E-03 | 69 | 34 | 9.69E-03 |
| PENALTY2 | 200 | 23 | 2 | 4.71E+13 | 23 | 2 | 4.71E+13 | 23 | 2 | 4.71E+13 |
| POWELLSG | 5000 | 128 | 112 | 5.61E-06 | 107 | 99 | 7.81E-06 | 127 | 104 | 3.01E-05 |
| SCHMVETT | 5000 | 15 | 13 | -1.50E+04 | 50 | 37 | -1.50E+04 | 17 | 15 | -1.50E+04 |
| SENSORS | 100 | 22 | 18 | -2.11E+03 | 19 | 15 | -2.11E+03 | 20 | 16 | -2.10E+03 |
| SINQUAD | 5000 | 30 | 17 | -6.76E+06 | 30 | 18 | -6.76E+06 | 33 | 20 | -6.76E+06 |
| SPARSQUR | 10000 | 49 | 29 | 1.47E-07 | 48 | 29 | 1.45E-07 | 34 | 19 | 3.78E-07 |
| SROSENBR | 5000 | 42 | 23 | 1.30E-13 | 33 | 17 | 9.89E-10 | 32 | 16 | 2.50E-09 |
| TOINTGOR | 50 | 136 | 104 | 1.37E+03 | 114 | 101 | 1.37E+03 | 138 | 112 | 1.37E+03 |
| TOINTGSS | 5000 | 3 | 2 | 1.00E+01 | 3 | 2 | 1.00E+01 | 3 | 2 | 1.00E+01 |
| TOINTPSP | 50 | 271 | 194 | 2.26E+02 | 210 | 147 | 2.26E+02 | 198 | 145 | 2.26E+02 |
| TOINTQOR | 50 | 33 | 29 | 1.18E+03 | 33 | 29 | 1.18E+03 | 33 | 29 | 1.18E+03 |
| TQUARTIC | 5000 | – | – | – | – | – | – | 12026 | 7612 | 6.25E-04 |
| TRIDIA | 5000 | 4156 | 3388 | 2.22E-15 | 3151 | 2788 | 2.24E-11 | 3751 | 3218 | 8.70E-13 |
| VAREIGVL | 50 | 33 | 30 | 4.05E-09 | 33 | 30 | 3.40E-11 | 49 | 43 | 3.52E-11 |
| WOODS | 4000 | 494 | 332 | 4.08E-09 | 308 | 232 | 1.33E-09 | 374 | 266 | 1.88E-08 |

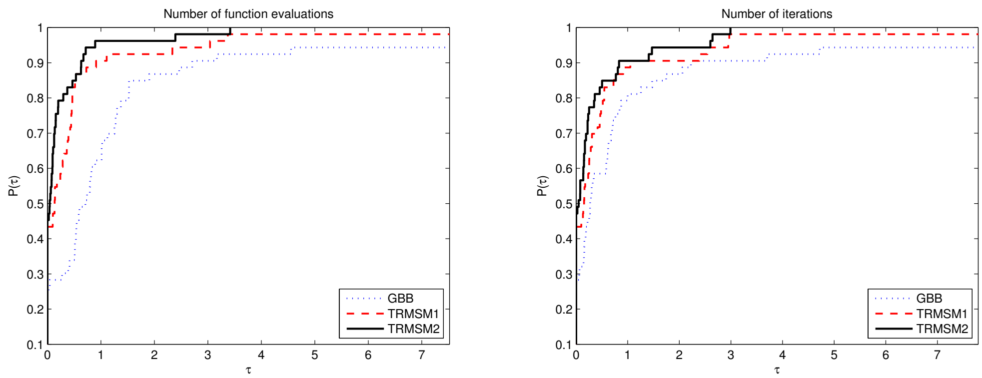


Figure 1: Comparison of GBB, TRMSM1 and TRMSM2

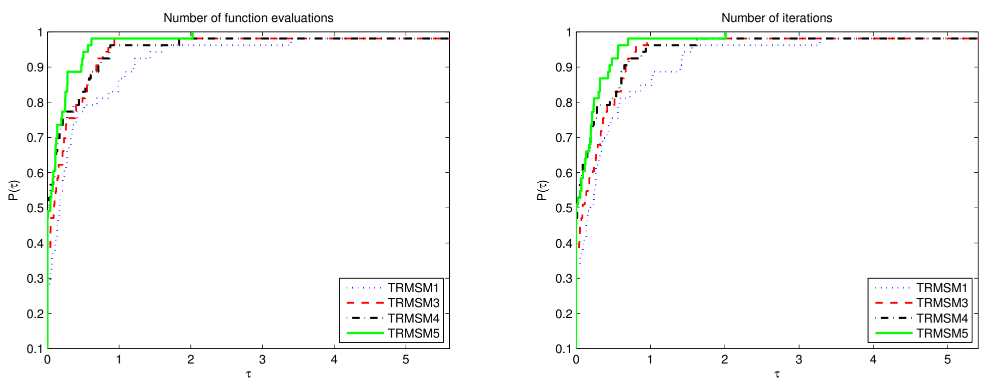


Figure 2: Comparison of TRMSM1, TRMSM3, TRMSM4 and TRMSM5

# References


[1] J. Barzilai, J.M. Borwein, Two point step size gradient method, *IMA J. Numer. Anal.* 8 (1988) 141-148.

[2] I. Bongartz, A.R. Conn, N.I. M. Gould, Ph.L. Toint, CUTE: Constrained and unconstrained testing environment, *ACM Transactions on Mathematical Software*, 21 (1995) 123-160.

[3] R.H. Byrd, R.B. Schnabel, and G.A. Schultz, Approximate solution of the trust regions problem by minimization over two-dimensional subspaces, *Math. Prog.* 40 (1988) 247-263.

[4] A.R. Conn, N.I.M. Gould, Ph.L. Toint, *Trust Region Methods*, SIAM, Philadelphia, (2000).

[5] Y.H. Dai and L.Z. Liao, R-linear convergence of the Barzilai and Borwein gradient method, *IMA J. Numerical Analysis* 22 (2002) 1-10.

[6] Y.H. Dai, J.Y. Yuan and Y.X. Yuan, Modified two-point stepsize gradient methods for unconstrained optimization, *Comput. Optim. Appl.* 22 (2002) 103-109.

[7] N.Y. Deng, Y. Xiao, F.J. Zhou, Nonmontonic trust region algorithm, *J. Optim. Theory and Appl.* 26 (1993) 259-285.

[8] J.E. Dennis, H. Wolkowicz, Sizing and least-change secant methods, *SIAM J. Numer. Anal.* 30 (1993) 1291-1314.

[9] S. Di and W. Sun, Trust region method for conic model to solve unconstrained optimization, *Optimization Methods and Software* 6 (1996) 237-263.

[10] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.* 91 (2002) 201-213.

[11] M. Farid, W.J. Leong, M.A. Hassan, A new two-step gradient-type method for large-scale unconstrained optimization, *Comput. Math. Appl.* 59 (2010) 3301-3307.

[12] L. Grippo, F. Lamparillo, S. Lucidi, A nonmonotone line search technique for Newton's method, *SIAM J. Numer. Anal.* 23 (1986) 707-716.

[13] W. W. Hager, Minimizing a quadratic over a sphere, *SIAM J. Optim.* 12 (2001) 188-208.

[14] K. Levenberg, A method for the solution of certain nonlinear problems in least squares, *Quarterly of Applied Mathematics* 2 (1944) 164-168.

[15] G.D. Li and Y. Yuan, Computing Celis-Dennis-Tapia step, *Journal of Computational Mathematics* 23 (2005) 463-478.

[16] D.C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Prog.* 45 (1989) 503-528.

[17] N. Maratos, Exact penalty function algorithms for finite dimensional and control optimization problems, Ph.D. thesis, Imperial College Sci. Tech., University of London, (1978).

[18] D.W. Marquardt, An algorithm for least-squares estimation of nonlinear inequalities, *SIAM J. Appl. Math.* 11 (1963) 431-441.

[19] J. Mo, C. Liu, S. Yan, A nonmonotone trust region method based on nonincreasing technique of weighted average of the successive function values, *J. Comput. Appl. Math.* 209 (2007) 97-108.

[20] J.J. Moré, Recent developments in algorithms and software for trust region methods, in *Mathematical Programming: The State of the Art*, Springer-Verlag, Berlin, (1983) 258-287.

[21] J.J. Moré, D.C. Sorensen, Computing a trust region step, *SIAM J. Sci. Stat. Comp.* 4 (1983) 553-572.

[22] J. Nocedal, S.J. Wright, *Numerical Optimization*, Springer, New York, (1999).

[23] M.J.D. Powell, A new algorithm for unconstrained optimization, In: J.B. Rosen, O.L. Mangasarian, and K. Ritter, eds., *Nonlinear Programming*, Academic Press, New York, 31–65, (1970).

[24] M.J.D. Powell, Convergence properties of a class of minimization algorithms, in: *Nonlinear Programming 2*, O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds., Academic Press, New York, (1975) 1-27.

[25] M. Raydan, The Barzilai and Borwein gradient method for large scale unconstrained minimization problem, *SIAM J. Optim.* 7 (1997) 26-33.

[26] L.K. Schubert, Modification of a quasi-Newton method for nonlinear equations with sparse Jacobian, *Math. Comput.* 24 (1970) 27-30.

[27] G.A. Schultz, R.B. Schnabel, and R.H. Byrd, A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties, *SIAM J. Numer. Anal.* 22 (1985) 47-67.

[28] A. Sidi, A new approach to vector-valued rational interpolation, *J. Approx. Theory* 130 (2004) 177-187.

[29] W. Sun, Optimization methods for non-quadratic model, *Asia Pacific J. Oper. Res.* 13 (1996) 43-63.

[30] W. Sun, Non-monotone trust region method for optimization, *Appl. Math. Comput.* 156 (2004) 159-174.

[31] W. Sun, Q. Du, J. Chen, *Computational Methods*, Higher Education Press, Beijing, (2007).

[32] W. Sun and Y. Yuan, *Optimization Theory and Methods: Nonlinear Programming*, Springer, New York, (2006).

[33] Ph.L. Toint, Towards an efficient sparsity exploiting Newton method for minimization, in *Sparse Matrices and Their Uses*, Academic Press, New York, (1981) 57-87.

[34] Ph.L. Toint, Global convergence of a class of trust region methods for nonconvex minimization in Hilbert space, *IMA J. Numer. Anal.* 8 (1988) 231-252.

[35] Z.H. Wang and Y. Yuan, A subspace implementation of quasi-Newton trust-region methods for unconstrained optimization, *Numerische Mathematik*, 104 (2006) 241-269.

[36] D. Winfield, *Function and Functional Optimization by Interpolation in Data Tables*, PhD thesis, Havard University, Cambridge, USA, (1969).

[37] D. Winfield, Function minimization by interpolation in a data table, *J. Inst. Math. Appl.*, 12 (1973) 339-347.

[38] Y. Yuan, On a subproblem of trust-region algorithm for constrained optimization, *Math. Prog.* 47 (1990) 53-63.

[39] Y. Yuan, A modified BFGS algorithm for unconstrained optimization, *IMA J. Numer. Anal.* 11 (1991) 325-332.

[40] H.C. Zhang, W.W. Hager, A nonmonotone line search technique and its application to unconstrained optimization, *SIAM J. Optim.* 14 (2004) 1043-1056.

[41] J.Z. Zhang, C.X. Xu, Properties and numerical performance of quasi-Newton methods with modified quasi-Newton equations, *J. Comput. Appl. Math.* 137 (2001) 269-278.

[42] Y. Zhang, W. Sun, L. Qi, A nomonotone filter Barzilai-Borwein method for optimization, *Asia Pacific J. Oper. Res.* 27 (2010) 55-69.