18 A Subspace Trust Region Method for Large Scale Unconstrained Optimization [*]

Zhouhong Wang Zaiwen Wen Yaxiang Yuan

State Key Laboratory of Scientific Engineering Computing
Institute of Computational Mathematics and Scientific/Engineering Computing
Academy of Mathematics and System Sciences, Chinese Academy of Sciences
POBox 2719, Beijing 100080, P.R.China

Abstract: A trust region algorithm for unconstrained large scale optimization is constructed by using a subspace technique. The new method can be viewed as an improvement on the limited memory quasi-Newton method with trust regions. Due to the special structures of the subspace trust region subproblems, the method can be applied to very large scale problems. Convergence properties of the method are analyzed. Numerical results for some typical large scale examples are reported and it shows that the new algorithm is efficient.

Keywords: subspace method, trust region techniques, limited memory storage,large scale optimization

1 Introduction

Consider the following unconstrained optimization problem:

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

where n is large and analytic expressions for the object f and its gradient g are available.

Large scale nonlinear optimization problems are very important in scientific and engineering computation, and such problems are normally difficult to solve because of the large number of variables, the limited memory storage, and a large amount of computation at each iteration. Various approaches have been proposed, such as combined CG-QN algorithm in [1], partitioned quasi-Newton method in [2] and the limited memory BFGS(L-BFGS) in [3; 4], etc. . [4; 5] showed that limited memory quasi-Newton methods are effective for solving large scale unconstrained optimization problems.

The purpose of this paper is to present a new trust region method using subspace technique for large scale unconstrained nonlinear optimization problems. It has been shown that the trust region method has good convergence properties in [6]

[*]Supported by Chinese NSF grant 10231060

and [7]. However, when the number of variables is large, the solution of the quadratic subproblem would be very cost. By analysis of the L-BFGS method, we find out that the quadratic subproblem with a trust region could be constructed in a subspace whose dimension is very small. Then the solution of the subproblem would need only a little effort even if the problem is very large.

L-BFGS method is a variation of standard BFGS method. Let x_k be the point at the k-th iteration, $g_k = \nabla f(x_k)$ and $\nabla^2 f(x_k)$ be the gradient and Hessian of f at the point x_k respectively. Denote

$$s_k = x_{k+1} - x_k, \quad y_k = g_{k+1} - g_k.$$

L-BFGS method choose a "basic matrix" γI and update it m times using the BFGS formula to get the current inverse Hessian approximation H_k by storing the most recent m pairs (s_i, y_i), $i = k-m, \cdots, k-1$. Denote

$$S = [s_{k-m}, \cdots, s_{k-1}], \quad Y = [y_{k-m}, \cdots, y_{k-1}].$$

By Theorem 2.2 in [8], the search direction d at the k-th iteration in L-BFGS can be represented as(where the subscript k is omitted for convenience)

$$\begin{aligned} d &= -Hg \\ &= -\left\{\gamma I + [S \quad \gamma Y] \begin{bmatrix} R^{-\mathrm{T}}(D + \gamma Y^{\mathrm{T}} Y) R^{-1} & -R^{-\mathrm{T}} \\ -R^{-1} & 0 \end{bmatrix} \begin{bmatrix} S^{\mathrm{T}} \\ \gamma Y^{\mathrm{T}} \end{bmatrix}\right\} g \\ &= -\gamma g + [S \quad \gamma Y] \begin{bmatrix} \mathbf{z_s} \\ \mathbf{z_y} \end{bmatrix} = [-g \quad S \quad Y] \begin{bmatrix} \gamma \\ \mathbf{z_s} \\ \gamma \mathbf{z_y} \end{bmatrix}, \end{aligned} \tag{1.2}$$

where

$$\begin{bmatrix} \mathbf{z_s} \\ \mathbf{z_y} \end{bmatrix} = -\begin{bmatrix} R^{-\mathrm{T}}(D + \gamma Y^{\mathrm{T}} Y) R^{-1} & -R^{-\mathrm{T}} \\ -R^{-1} & 0 \end{bmatrix} \begin{bmatrix} S^{\mathrm{T}} g \\ \gamma Y^{\mathrm{T}} g \end{bmatrix}.$$

By formula (1.2), we could see that the search direction obtained by the L-BFGS method belongs to the subspace expanded by $[-g, S, Y]$. When some step in the direction d_k is not satisfactory, the line-search algorithms choose a shorter step length, and the information about the Hessian of f in the subspace is not used. We can not say that the descent direction d_k in (1.2) is the best one, and we could expect to get better solution by searching in the subspace spanned by $[-g, S, Y]$ with the trust region techniques. Our method is different from the method proposed by [9], which searches in the whole space.

2 The Algorithm

In order to avoid numerical problems, define $\mathbf{A_k}$ as

$$\mathbf{A_k} = \left[\frac{-\mathbf{g_k}}{\|\mathbf{g_k}\|}, \frac{\mathbf{s_{k-m}}}{\|\mathbf{s_{k-m}}\|}, \cdots, \frac{\mathbf{s_{k-1}}}{\|\mathbf{s_{k-1}}\|}, \frac{\mathbf{y_{k-m}}}{\|\mathbf{y_{k-m}}\|}, \cdots, \frac{\mathbf{y_{k-1}}}{\|\mathbf{y_{k-1}}\|}\right]. \tag{2.1}$$

So $\mathbf{A_k}$ is a $n \times (2m+1)$ matrix. Denote the subspace spanned by the columns of $\mathbf{A_k}$ by $\mathbf{L}(\mathbf{A_k})$, and

$$F_k(\mathbf{z}) = \mathbf{f}(\mathbf{x_k} + \mathbf{A_k z}). \tag{2.2}$$

By Taylor's series theorem,

$$F_k(\mathbf{z}) \approx \mathbf{f}(\mathbf{x_k}) + \mathbf{z}^{\mathrm{T}}\mathbf{A}_{\mathbf{k}}^{\mathrm{T}}\mathbf{g}_{\mathbf{k}} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{z}^{\mathrm{T}}\mathbf{A}_{\mathbf{k}}^{\mathrm{T}}\nabla^{\mathbf{2}}\mathbf{f}(\mathbf{x_k})\mathbf{A_k}\mathbf{z}. \tag{2.3}$$

Now the main problem is how to get an approximation to $\mathbf{A}_{\mathbf{k}}^{\mathrm{T}}\nabla^{\mathbf{2}}\mathbf{f}(\mathbf{x_k})\mathbf{A_k}$ in the subspace $\mathbf{L}(\mathbf{A_k})$ by the most recently computed m pairs $(s_i, y_i), i = k-m, \cdots, k-1$. We have two ways to do this by L-BFGS updating formula. The first way is to get the approximation B_k to the Hessian $\nabla^2 f(x_k)$ by formula (2.17) in [8]

$$B_k = \gamma_k I - [\gamma_k S_k \quad Y_k] \begin{bmatrix} \gamma_k S_k^{\mathrm{T}} S_k & L_k \\ L_k^{\mathrm{T}} & -D_k \end{bmatrix}^{-1} \begin{bmatrix} \gamma_k S_k^{\mathrm{T}} \\ Y_k^{\mathrm{T}} \end{bmatrix}, \tag{2.4}$$

where

$$S_k = \begin{cases} [s_0, \ldots, s_{k-1}] & \text{if } k \le m \\ [s_{k-m}, \ldots, s_{k-1}] & \text{otherwise} \end{cases}, \quad Y_k = \begin{cases} [y_0, \ldots, y_{k-1}] & \text{if } k \le m \\ [y_{k-m}, \ldots, y_{k-1}] & \text{otherwise} \end{cases}, \tag{2.5}$$

$$(L_k)_{i,j} = \begin{cases} \tilde{s}_{i-1}^T \tilde{y}_{j-1} & \text{if } i > j \\ 0 & \text{otherwise} \end{cases}, \quad i, j = \begin{cases} 1, 2, \ldots, k & \text{if } k \le m \\ 1, \ldots, m & \text{otherwise} \end{cases}, \tag{2.6}$$

$$\tilde{s}_i = \begin{cases} s_i & \text{if } k \le m \\ s_{k-m+i} & \text{otherwise} \end{cases}, \quad \tilde{y}_i = \begin{cases} y_i & \text{if } k \le m \\ y_{k-m+i} & \text{otherwise} \end{cases}. \tag{2.7}$$

and

$$\gamma_k = \frac{y_k^{\mathrm{T}} s_k}{\|s_k\|^2}, \quad D_k = \begin{cases} \mathrm{diag}[s_0^T y_0, \ldots, s_{k-1}^T y_{k-1}] & \text{if } k \le m \\ \mathrm{diag}[s_{k-m}^T y_{k-m}, \ldots, s_{k-1}^T y_{k-1}] & \text{otherwise} \end{cases}. \tag{2.8}$$

Then we compute $\mathbf{A}_{\mathbf{k}}^{\mathrm{T}}\mathbf{B_k}\mathbf{A_k}$ and $\mathbf{A}_{\mathbf{k}}^{\mathrm{T}}\mathbf{g_k}$ as follows to form the subproblem in formula (2.3).

In order to compute the inverse matrix in (2.4), we can compute the Cholesky decomposition of $\gamma_k S_k^T S_k + L_k D_k^{-1} L_k^T$ to obtain $J_k J_k^{\mathrm{T}}$, where J_k is a lower triangular matrix, then we could get

$$\begin{bmatrix} \gamma_k S_k^{\mathrm{T}} S_k & L_k \\ L_k^{\mathrm{T}} & -D_k \end{bmatrix}^{-1} = M_1^{\mathrm{T}} \begin{bmatrix} I_m & 0 \\ & -D^{-1} \end{bmatrix} M_1, \tag{2.9}$$

where

$$M_1 = \begin{bmatrix} J_k^{-1} & J_k^{-1} L_k D_k^{-1} \\ 0 & I_m \end{bmatrix}.$$

Now compute $\mathbf{A}_{\mathbf{k}}^{\mathrm{T}}\mathbf{B_k}\mathbf{A_k}$. First compute

$$M_2 = M_1 \begin{bmatrix} \gamma_k S_k^{\mathrm{T}} \\ Y_k^{\mathrm{T}} \end{bmatrix} \mathbf{A_k}. \tag{2.10}$$

Then by (2.4) we could obtain

$$\bar{B}_k = \mathbf{A_k^T B_k A_k} = \gamma \mathbf{A_k^T A_k} - \mathbf{M_2^T} \begin{bmatrix} I_m & 0 \\ & -D^{-1} \end{bmatrix} \mathbf{M_2}, \tag{2.11}$$

$$\bar{g}_k = \mathbf{A_k^T g_k}. \tag{2.12}$$

The trust region subproblem is

$$\begin{aligned} \min \ & \phi_k(\mathbf{z}) \equiv \mathbf{\bar{g}_k^T z} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{z^T \bar{B}_k z} \\ s.t. \ & \|\mathbf{z}\| \le \mathbf{\Delta_k}. \end{aligned} \tag{2.13}$$

At each iteration, we need to compute only one row and one column to update $S_k^T S_k$, $S_k^T Y_k$ and $Y_k^T Y_k$, which are needed in (2.9), (2.10) and (2.11).

The second way is to compute the m pairs $(\bar{s}_i, \bar{y}_i), i = k-m, \cdots, k-1$ in the subspace $\mathbf{L(A_k)}$ directly, which is expanded by the columns of the matrix $\mathbf{A_k}$, and $\mathbf{A_k}$ is defined in (2.1). Then we could compute the approximation $\bar{B}_k$ to the Hessian in the subspace directly by the limited memory BFGS updating formula (2.4). In the subspace $\mathbf{L(A_k)}$, the m pairs $(\bar{s}_i, \bar{y}_i)$ are equal to $\mathbf{z_{i+1}} - \mathbf{z_i}$ and $\nabla F(\mathbf{z_{i+1}}) - \nabla \mathbf{F}(\mathbf{z_i})$ respectively, where

$$x_k + \mathbf{A_k z_i} = \mathbf{x_i}, \mathbf{i} = \mathbf{k} - \mathbf{m}, \cdots, \mathbf{k} - \mathbf{1}, \tag{2.14}$$

according to formula (2.2). Suppose $k \ge m$, by (2.14) we could get

$$\begin{aligned} \mathbf{z_k} &= [\mathbf{0}, \ldots, \mathbf{0}]^{\mathrm{T}}, \\ \mathbf{z_{k-1}} &= [\overbrace{0, \ldots, 0}^{\mathbf{m}}, -\|\mathbf{s_{k-1}}\|, \overbrace{0, \ldots, 0}^{\mathbf{m}}]^{\mathrm{T}}, \\ &\cdots\cdots \\ \mathbf{z_{k-i}} &= [\overbrace{0, \ldots, 0}^{\mathbf{m+1-i}}, -\|\mathbf{s_{k-i}}\|, \ldots, -\|\mathbf{s_{k-1}}\|, \overbrace{0, \ldots, 0}^{\mathbf{m}}]^{\mathrm{T}}, \\ &\cdots\cdots \\ \mathbf{z_{k-m}} &= [\mathbf{0}, -\|\mathbf{s_{k-m}}\|, \ldots, -\|\mathbf{s_{k-1}}\|, \mathbf{0}, \ldots, \mathbf{0}]^{\mathrm{T}}, \end{aligned}$$

and the gradient $\bar{g}_i$ of $F(\mathbf{z})$(defined in (2.2)) at point $\mathbf{z_i}$ is $\mathbf{A_k^T g_i}$, where $i = k, \cdots, k-m$. So we obtain

$$\begin{aligned} \bar{s}_i &= \mathbf{z_{i+1}} - \mathbf{z_i} = \|\mathbf{s_i}\| \mathbf{e_{i+m+2-k}}, \\ \bar{y}_i &= \bar{g}_{i+1} - \bar{g}_i = \mathbf{A_k^T g_{i+1}} - \mathbf{A_k^T g_i} = \mathbf{A_k^T y_i}, \end{aligned} \tag{2.15}$$

where $k-m, \cdots, i = k-1$, and $\mathbf{e_j}$ is the j-th unit vector in $\mathbf{R^{2m+1}}$. By formula (2.4), we obtain B_k which is an approximation to the Hessian $\nabla^2 F(0)$ in the subspace $\mathbf{L(A_k)}$ as

$$\bar{B}_k = \gamma_k I - [\gamma_k \bar{S}_k \quad \bar{Y}_k] \begin{bmatrix} \gamma_k \bar{S}_k^T \bar{S}_k & \bar{L}_k \\ \bar{L}_k^T & -\bar{D}_k \end{bmatrix}^{-1} \begin{bmatrix} \gamma_k \bar{S}_k^T \\ \bar{Y}_k^T \end{bmatrix}, \tag{2.16}$$

where $\bar{S}_k, \bar{L}_k$ and $\bar{D}_k$ are obtained by substituting s_i, y_i with $\bar{s}_i$, $\bar{y}_i$ respectively in formula (2.5)–(2.8), γ_k is the same as in (2.8), and we have

$$\bar{S}_k = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \|s_{k-m}\| & 0 & \cdots & 0 \\ 0 & \|s_{k-m+1}\| & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \|s_{k-1}\| \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{2.17}$$

It is easy to see that

$$\bar{S}_k^T \bar{Y}_k = S_k^T Y_k \quad \bar{Y}_k = \mathbf{A_k^T Y_k} \quad \mathbf{S_k^T S_k} = \mathbf{Diag}(\|\mathbf{s_{k-m}}\|^2, \cdots, \|\mathbf{s_{k-1}}\|^2).$$

We can compute the inverse matrix by formula in (2.16) by (2.9), and get the trust region subproblem (2.13), where $\bar{B}_k$ is a $(2m+1) \times (2m+1)$ matrix, $\mathbf{z} \in \mathbf{R^{2m+1}}$, $\mathbf{m \ll n}$.

Next problem is how to evaluate the approximation to the object f by the subspace and the subspace trust region subproblem. We can do this in the same way as in other trust region methods (see ([10; 11; 12]), and perhaps we have to solve the subspace trust region subproblem several times to get a proper trust region radius. We can also solve the problem min $F_k(\mathbf{z}) \equiv \mathbf{f(x_k + A_k z_k)}$ in the subspace by iterating the trust region method and solve the subproblem (2.13) several times(about 2–5) to get a better local solution .

Algorithm 2.1. Subspace Trust Region Algorithm for Unconstrained Optimization

Step 0 : *Given* $x_0 \in R^n, \Delta_0 > 0$, $\varepsilon \geq 0$, $B_0 = \gamma_0 I, \gamma_0 > 0$, $S_0 = Y_0 = \mathbf{0}$, $0 < \tau_1 < \tau_3 < \tau_2 < \tau_4 < 1, c_1 > c_2 \geq c_3 \geq c_4 > 1$. *Compute* $f(x_0)$, g_0, $\mathbf{A_0} = \left[-\frac{\mathbf{g_0}}{\|\mathbf{g_0}\|}\right]$, *Let* $k = 0$.

Step 1 : *If* $\|g_k\|_2 \leq \varepsilon$ *then stop.*

Step 2 : *Solve the subproblem (2.13) to get* $\mathbf{z_k}$ *and* $\phi(\mathbf{z_k})$. *Compute*

$$\hat{s}_k = \mathbf{A_k z_k}$$
$$\rho_k = \frac{f(x_k) - f(x_k + \hat{s}_k)}{(-\phi(\mathbf{z_k}))} \tag{2.18}$$

Step 3 : *If* $\rho_k < \tau_1$, *let* $\Delta_k \leftarrow \Delta_k / c_1$ *(we could also use the interpolation and backtracking techniques proposed by [10; 12] here) and go to step 2; Otherwise, do next step.*

Step 4 : *If* $\rho_k > \tau_4$, *let* $\Delta_k \leftarrow c_4 \Delta_k$ *and solve the subproblem (2.13) again to get* $\tilde{\mathbf{z}}_\mathbf{k}$ *and* $\tilde{\rho}_k$. *If* $\tilde{\rho}_k \geq \tau_2$, *set* $\mathbf{z_k} = \tilde{\mathbf{z}}_\mathbf{k}$ *and* $\rho_k = \tilde{\rho}_k$; *Otherwise* *do not change* $\mathbf{z_k}$ *and* ρ_k. *Let*

$$\begin{aligned} s_k &= \hat{s}_k \equiv \mathbf{A_k z_k} \\ x_{k+1} &= x_k + s_k \\ \Delta_{k+1} &= \begin{cases} \Delta_k / c_3, & \text{if } \rho_k < \tau_2 \\ c_4 \Delta_k, & \text{if } \rho_k > \tau_3 \text{ and } \|\mathbf{z_k}\| = \mathbf{\Delta_k} \\ \Delta_k, & \text{otherwise} \end{cases} \end{aligned} \tag{2.19}$$

Step 5 : *Compute* $g_{k+1} = \nabla f(x_{k+1})$, $y_k = g_{k+1} - g_k$. *Let* $k \leftarrow k+1$ *and update* $\mathbf{A_k}$ *in (2.1), update* S_k, Y_k *in (2.5) or* $\bar{S}_k$, $\bar{Y}_k$ *in (2.17) to obtain* $\bar{B}_k$ *according to (2.11) or (2.16),* $\bar{g}_k$ *according to (2.12), then go to step 2.*

In the above algorithm, the parameter could be chosen as $\tau_1 = 0.001$, $\tau_2 = 0.2$, $\tau_3 = 0.7$, $\tau_4 = 0.9$, and $c_1 = 4, c_2 = c_3 = c_4 = 2$. The parameter m should be chosen between 3 and 8. We could also loop between step 2 and step 4 several times(2–5) in order to solve the problem $\min F_k(\mathbf{z}) \equiv \mathbf{f}(\mathbf{x_k} + \mathbf{A_k z})$ by trust region method in the subspace $\mathbf{L}(\mathbf{A_k})$ to get a better local solution. Similar ideas can be found in [13; 14]. The details would be a little tedious, but they do not influence the following convergence analysis.

3 Convergence

The convergence analysis is in the same way as in [15; 6; 7]. By the definition of $\mathbf{A_k}$ in (2.1), we get $\|\mathbf{A_k}\|_{\mathrm{F}} = \sqrt{\tilde{\mathbf{m}}}$, where $\tilde{m} = 2m+1$. So

$$\sqrt{\tilde{m}} = \|\mathbf{A_k}\|_{\mathrm{F}} \geq \|\mathbf{A_k}\| \geq \|\mathbf{A_k}\|_{\mathrm{F}} / \sqrt{\tilde{\mathbf{m}}} = \mathbf{1} \tag{3.1}$$

where $\| \ \|$ denotes the Euclidean norm.

Theorem 3.1. *Let* $f : R^n \to R$ *be convex, bounded below and twice continuous differentiable. There exists a positive constant* $M > 0$ *such that the Hessian satisfy* $\|\nabla^2 f(x)\| \leq M$ *for all* x *in the level set* $\{x : f(x) \leq f(x_1)\}$. *Then* $\lim_{k\to\infty} g_k = 0$ *and the loops between step 2 and step 3 must be finite if* $\varepsilon > 0$.

Proof. By Theorem 4 in [15], let $\mathbf{z_k}$ be the solution of (2.13), then

$$-\phi(\mathbf{z_k}) \geq \frac{\mathbf{1}}{\mathbf{2}} \|\bar{\mathbf{g}}_\mathbf{k}\| \min\left(\mathbf{\Delta_k}, \frac{\|\bar{\mathbf{g}}_\mathbf{k}\|}{\|\bar{\mathbf{B}}_\mathbf{k}\|}\right). \tag{3.2}$$

First let us prove the boundedness of $\|\bar{B}_k\|$. If $\bar{B}_k$ is defined by (2.11), by (3.1)

$$\|\bar{B}_k\| = \|\mathbf{A_k^T B_k A_k}\| \leq \tilde{\mathbf{m}} \|\mathbf{B_k}\|. \tag{3.3}$$

Because B_k is symmetric and positive definite, by $\|\nabla^2 f(x)\| \leq M$ we have

$$\begin{aligned} y_k &= \nabla f(x_{k+1}) - \nabla f(x_k) = \int_0^1 \nabla^2 f(x_k + \theta s_k) s_k \, \mathrm{d}\theta = \bar{\mathrm{G}}_\mathrm{k} \mathrm{s}_\mathrm{k}, \\ \gamma_k &= \frac{y_k^{\mathrm{T}} s_k}{\|s_k\|^2} \leq M, \\ \frac{\|y_i\|^2}{y_i^{\mathrm{T}} s_i} &= \frac{s_i^{\mathrm{T}} \bar{G}_k^2 s_i}{s_i^{\mathrm{T}} \bar{G}_k s_i} \leq M \text{ for all } i \geq 1, \end{aligned} \tag{3.4}$$

where $\bar{G}_k = \int_0^1 \nabla^2 f(x_k + \theta s_k)\,\mathrm{d}\theta$ is positive definite or positive semi-definite by the convexity of f. So in the same way as [16] it follows that

$$\|B_k\| \le \mathrm{tr}(B_k) = \mathrm{tr}(\gamma_k I) - \sum_{i=k-m}^{k-1} \frac{\|B_k^{(i)}\|^2}{s_i^{\mathrm{T}} B_k^{(i)} s_i} + \sum_{i=k-m}^{k-1} \frac{\|y_i\|^2}{s_i^{\mathrm{T}} y_i} \le (n+m)M, \tag{3.5}$$

where $B_k^{(i)}$ is the i-th updated matrix by BFGS formula from the basic matrix $\gamma_k I$, γ_k is defined in (2.8). By (3.3) and (3.5), we know that $\|\bar{B}_k\|$ is bounded. If we obtain $\bar{B}_k$ by (2.15)-(2.17), we have

$$\begin{aligned} &\mathbf{A_k \bar{s}_i = s_i}, \quad \mathbf{\bar{y}_i = A_k^T y_i} \text{ for } \mathbf{i = k - m}, \cdots, \mathbf{k-1}, \\ &\|\bar{s}_i\| = \|s_i\|, \bar{s}_i^{\mathrm{T}} \bar{y}_i = \bar{y}_i^{\mathrm{T}} \bar{s}_i = y_i^{\mathrm{T}} s_i = s_i^{\mathrm{T}} y_i. \end{aligned} \tag{3.6}$$

Then by (3.1), (3.4) and (3.6)

$$\begin{aligned} \bar{\gamma}_k &= \frac{\bar{y}_k^{\mathrm{T}} \bar{s}_k}{\|\bar{s}_k\|^2} = \frac{y_k^{\mathrm{T}} s_k}{\|s_k\|^2} \le M, \\ \frac{\|\bar{y}_i\|^2}{\bar{y}_i^{\mathrm{T}} \bar{s}_i} &= \frac{y_i^{\mathrm{T}} \mathbf{A_k A_k^T y_i}}{y_i^{\mathrm{T}} s_i} \le \tilde{m} \frac{\|y_i\|^2}{y_i^{\mathrm{T}} s_i} \le \tilde{m} M. \end{aligned} \tag{3.7}$$

So in the same way, we know that $\|\bar{B}_k\|$ is bounded by (3.5). Suppose

$$\|\bar{B}_k\| \le M_2 \quad \text{for all} \quad k \ge 1. \tag{3.8}$$

By (2.1), we have

$$\|\bar{g}_k\| = \|\mathbf{A_k^T g_k}\| \ge \|\mathbf{g_k}\|. \tag{3.9}$$

If $\|g_k\|$ is bounded away from 0, there exists a constant $\varepsilon_1 > 0$, such that $\|g_k\| \ge \varepsilon_1$ for all $k \ge 1$. By (3.2), (3.8) and (3.9), we get

$$|\phi(\mathbf{z_k})| = -\phi(\mathbf{z_k}) \ge \frac{\mathbf{1}}{\mathbf{2}} \varepsilon_1 \min\left(\mathbf{\Delta_k}, \frac{\varepsilon_1}{\mathbf{M_2}}\right). \tag{3.10}$$

Because f is bounded below, $\{f(x_k)\}$ decrease monotonically, there must exist a f^* such that $f(x_k) \to f^*$. By step 2–step 4 of the algorithm,

$$f(x_k) - f(x_{k+1}) \ge \tau_1 |\phi(\mathbf{z_k})| \ge \frac{\tau_1 \varepsilon_1}{\mathbf{2}} \min\left(\mathbf{\Delta_k}, \frac{\varepsilon_1}{\mathbf{M_2}}\right).$$

So we get $\Delta_k \to 0$, and there exists a $k_1 \ge 1$, such that

$$|\phi(\mathbf{z_k})| = -\phi(\mathbf{z_k}) \ge \frac{\varepsilon_1}{\mathbf{2}} \mathbf{\Delta_k} \text{ for all } \mathbf{k \ge 1}, \tag{3.11}$$

and

$$\begin{aligned} |\rho_k - 1| &= \left| \frac{f(x_k) - f(x_k + \mathbf{A_k z_k}) + \mathbf{\bar{g}_k^T z_k} + \frac{1}{2} \mathbf{z_k^T \bar{B}_k z_k}}{-\phi(\mathbf{z_k})} \right| \\ &= \frac{|-\int_0^1 (1-t) \mathbf{z_k^T A_k^T \nabla^2 f(x_k + t A_k z_k) A_k z_k}\,\mathrm{dt} + \frac{1}{2} \mathbf{z_k^T \bar{B}_k z_k}|}{|\phi(\mathbf{z_k})|} \\ &\le \frac{1}{\varepsilon_1} \cdot \frac{(\tilde{m}M + M_2)\|\mathbf{z_k}\|^2}{\Delta_k} \le \frac{\tilde{m}M + M_2}{\varepsilon_1} \Delta_k \to 0. \end{aligned} \tag{3.12}$$

Table 4.1 Numerical Results

Prob.	Dimension	TRSub	L-BFGS
1	500	19/57/36	37/47/47
1	5000	24/60/46	33/48/48
1	10000	27/67/53	33/48/48
2	1000	51/144/67	53/58/58
3	100	28/124/48	106/111/111
3	1000	30/109/48	—
3	5000	35/142/64	52/61/61
3	10000	39/137/70	52/61/61

So there exist $k_2 \geq k_1$, such that $\rho_k > \tau_2$ for all $k \geq k_2$. By step 4 of the algorithm, there exists a $\bar{\Delta} > 0$ such that $\Delta_k \geq \bar{\Delta}$ for all $k \geq k_2$. This contradicts with $\Delta_k \to 0$. So $\|g_k\|$ can not be bounded away from 0. Now in the same way as [6] or [7], we could get $\lim\limits_{k\to\infty} g_k = 0$. (3.11) and (3.12) also tell us that the loops between step 2 and step 3 must be finite if $\|g_k\| \geq \varepsilon > 0$. $\square$

By the proof of Theorem 3.1, we know that there are many ways for choosing the basis(i.e. the columns of $\mathbf{A}_\mathrm{k}$) of subspace which will not influence the convergence result.

4 Numerical Results

We have test the new algorithm on some typical large scale problem on a microcomputer which has an Intel-based 32bit CPU and the main frequency is 1.2GHz. The method for constructing the subproblem is formula (2.11) and (2.12), and the trust region radius is set to $\Delta_{\max} = 5$ at each iteration and m is set to 6. In our implementation, the algorithm loops between step 2 and step 4 twice in order to get a better local solution, and it was counted as one iteration. We do not use interpolation or backtracking techniques in this version of the new code. The first problem is the "extended separable Rosenbrock function" with n=500, 5000, 10000, where n is the number of variables. The second problem is the "trigonometric" function with n=1000. The 3rd problem is the "extended Powell singular function with n=100, 1000, 5000, 10000. All these problems are listed in [17]. The results are reported in the form:

number of iterations/number of function evaluations/number of gradient evaluations

and the results are compared with the L-BFGS method. In the Table 4.1, TRSub is the new code. The result for L-BFGS in the first line is obtained by executing the L-BFGS code which is downloaded from netlib on the microcomputer with m=6. Other results for L-BFGS are extracted from [4] with m=5. Comparing with L-BFGS, we could see that the number of gradient evaluations is almost the same, but the number of function evaluations is more. We think that the number of function evaluations can be decreased if trust region radius Δ_k is not set to $\Delta_{\max}$ at each iteration and interpolation or backtracking techniques are used. When we implement L-BFGS code on the microcomputer for the "extended separable Rosenbrock function" with n=5000, m=5, the line search method failed with "INFO=4", which means the step is at the lower bound STPMIN, which is 1.D-20 by default. These preliminary numerical results shows the new method is promising, and we could expect better numerical stability compared with line search limited memory methods. We will do some improvements on the code and will report further numerical results in CUTE later.

5 Conclusions

A new subspace trust region algorithm for unconstrained optimization is constructed, which is based on the analysis of the searching direction in limited memory BFGS method. Just as proceeding from quasi-Newton methods with line searching to trust region method, we proceed from limited memory quasi-Newton methods with line searching to subspace trust region method. The size of the subspace trust region subproblems is very small(about 6–16). We can solve these subproblems very quickly and exactly, and the method can be applied to very large scale problems. Convergence properties of the method are analyzed. Preliminary numerical results show that the new algorithm is encouraging, and because of the strong convergence properties of trust region method and the special structure of the subspace trust region subproblem, we could expect to get better numerical stability compared with other line searching limited memory methods . The main problem in the new method is how to construct the subspace. There are many other ways for choosing the basis of the subspace. This paper only present one. If the basis of the subspace is not chosen properly, the method may converge very slowly, although we can still prove its convergence in theory. There are also other matrix updating formula can be chosen, such as the SR1 formula. Further studies will show how to evaluate the effective of the subspace and better ways to update the subspace.

References

[1] A.Buckley and A. LeNir, QN-like variable storage conjugate gradients, Math. Prog. 27(1983), 155-175.

[2] A.Griewank and Ph.L.Toint, Partitioned variable metric updates foe large structured optimization problems, Numerische Mathematik 39(1982), 119-137

[3] J. Nocedal, Updating Quasi-Newton Matrices with Limited Storage, Mathematics of Computation 35(1980), 773-782.

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

[5] J.C.Gilbert and C.Lemaréchal. Some numerical experiments with variable storage quasi-Newton algorithms, Math. Prog. 45(1989),407-436.

[6] D.C. Sorensen . Newton's method with a model trust region modfications, SIAM J. Numerical Analysis 19(1982), 409-426.

[7] 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 Journal on Numerical Analysis 22(1985), 47-67.

[8] R. Byrd, J. Nocedal and R. Schnabel, Representations of Quasi-Newton Matrices and their use in Limited Memory Methods, Mathematical Programming 63(1994), 129-156.

[9] Burdakov and Yaxiang Yuan, Limited Memory Trust Region Methods for Solving Large-Scale Unconstrained Optimization, 2002, preprint paper.

[10] J.E. Dennis, Jr. and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Inc., (Englewood Cliffs, NJ, 1983

[11] R. Fletcher, Practical Methods of Optimization, 2nd edition, John Wiley and Sons, New York, 1987.

[12] J. Nocedal and Y. Yuan, Combining trust region and line search techniques, in: Y. Yuan, ed., Advances in Nonlinear Programming ,Kluwer, 1997, 153-175.

[13] M.A.Wolfe. A quasi-Newton method with memory for unconstrained function minimization, J.Inst .Maths . Applics .15(1975), 85-94.

[14] M.A.Wolfe and C.Viazminsky, Supermemory descent methods for unconstrained minimization, J. of Optimization theory and applications, 18(1976),455-468

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

[16] M.J.D. Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line search, in: R.W. Cottle and C.E. Lemke, eds., Nonlinear Programming, SIAM-AMS Proceedings *IX*, SIAM, Philadelphia, PA, 1976, 54-72.

[17] J.J. Moré, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, ACM Transactions on Mathematical Software 7(1981), 17-41.