COMMUN. MATH. SCI.
Vol. 18, No. 5, pp. 1179–1190

A THREE-TERM CONJUGATE GRADIENT ALGORITHM USING SUBSPACE FOR LARGE-SCALE UNCONSTRAINED OPTIMIZATION*

YUTING CHEN† AND YUETING YANG‡

Abstract. It is well known that conjugate gradient methods are suitable for large-scale nonlinear optimization problems, due to their simple calculation and low storage. In this paper, we present a three-term conjugate gradient method using subspace technique for large-scale unconstrained optimization, in which the search direction is determined by minimizing the quadratic approximation of the objective function in a subspace which is discussed in two cases. We show the search direction can both satisfy the descent condition and Dai-Liao conjugacy condition. Under proper assumptions, global convergence result of the proposed method is established. Numerical experiments show the proposed method is efficient and robust.

Keywords. subspace; three-term conjugate gradient method; global convergence; large-scale; unconstrained optimization.

AMS subject classifications. 65K05; 90C06; 90C30.

1. Introduction

Consider the following large-scale unconstrained optimization

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

where $f:\ \mathbb{R}^n\to\mathbb{R}$ is continuously differentiable and bounded from below. Starting from an initial point x_0, the usual nonlinear conjugate gradient methods generate a sequence $\{x_k\}$ as

$$x_{k+1}=x_k+\alpha_k d_k,\ k\geq 0, \tag{1.2}$$

where the step-length α_k is determined by some line search, and the search direction d_k is generated as

$$d_k=\begin{cases}-g_0, & \text{if } k=0,\\ -g_k+\beta_{k-1}d_{k-1}, & \text{if } k\geq 1,\end{cases} \tag{1.3}$$

where $g_k=\nabla f(x_k)$ is the gradient of $f(x)$ at x_k and β_k is a scalar called a conjugate gradient parameter.

The line search in conjugate gradient methods is usually based on the Wolfe conditions [24, 25]

$$f(x_k+\alpha_k d_k)-f(x_k)\leq\rho\alpha_k g_k^{\mathrm{T}}d_k, \tag{1.4}$$

$$g_{k+1}^{\mathrm{T}}d_k\geq\sigma g_k^{\mathrm{T}}d_k, \tag{1.5}$$

where d_k is a descent direction and the constants ρ, σ satisfy $0<\rho\leq\sigma\leq 1$. However, in order to establish the convergence and enhance the stability, the strong Wolfe conditions given by (1.4) and

$$|g_{k+1}^{\mathrm{T}}d_k|\leq\sigma|g_k^{\mathrm{T}}d_k| \tag{1.6}$$

*Received: June 18, 2019; Accepted (in revised form): January 26, 2020. Communicated by Wotao Yin.

†School of Mathematics and Statistics, Beihua University, Jilin 132013, China; and School of Mathematics, Jilin University, Changchun 130012, China (chenyuting1012@126.com).

‡School of Mathematics and Statistics, Beihua University, Jilin 132013, China (yyt2858@163.com).

are needed.

Different conjugate gradient methods correspond to different choices of β_k. The well-known conjugate gradient methods are Polak and Ribiére (PRP) method [20,21], Hestenes and Steifel (HS) method [14], Fletcher and Reeves (FR) method [11], Dai and Yuan (DY) method [6], Liu and Storey (LS) method [18], and the conjugate descent (CD) method [10], in which β_k are specified by

$$\begin{aligned} \beta_k^{\mathrm{PRP}} &= \frac{g_{k+1}^{\mathrm{T}} y_k}{\|g_k\|^2}, \quad \beta_k^{\mathrm{HS}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{d_k^{\mathrm{T}} y_k}, \quad \beta_k^{\mathrm{FR}} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, \\ \beta_k^{\mathrm{DY}} &= \frac{\|g_{k+1}\|^2}{d_k^{\mathrm{T}} y_k}, \quad \beta_k^{\mathrm{LS}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{-d_k^{\mathrm{T}} g_k}, \quad \beta_k^{\mathrm{CD}} = \frac{\|g_{k+1}\|^2}{-d_k^{\mathrm{T}} g_k}, \end{aligned} \tag{1.7}$$

respectively, where $y_k = g_{k+1} - g_k$ and $\|\cdot\|$ stands for the Euclidean norm.

Recently, in order to obtain a local solution economically, many researchers paid a great deal of attention to the three-term conjugate gradient methods for large-scale unconstrained optimization problems, and gained valuable achievements [4,8,17,19,23]. To have nice numerical results and to have a search direction which satisfies the descent condition and Dai-Liao conjugacy condition, Deng and Wan [8] proposed a three-term conjugate gradient method (MTHREECG) whose search direction is close to the Newton direction. However, the method does not need to compute or store any approximate Hessian matrix of the objective function. Various numerical methods combining with subspace technique are presented in succession [2,3,7,12,13,15,16,22,26,27].

For example, Stoer and Yuan [22] introduced a line search method, in which the search direction was computed by minimizing the approximate quadratic model in the two-dimension subspace $[-g_{k+1}, s_k]$, i.e.,

$$d_{k+1} = \mu_{k+1} g_{k+1} + \nu_{k+1} s_k, \tag{1.8}$$

where $s_k = x_{k+1} - x_k$. If the objective function is quadratic and line search is exact, this method reduces to the classical conjugate gradient method.

In addition, Andrei [2] suggested a three-term conjugate gradient method (TTS), in which the search direction was determined by minimizing the approximate quadratic model in the three-dimension subspace $[-g_{k+1}, s_k, y_k]$. The search direction in this method can satisfy not only the descent condition but also Dai-Liao conjugacy condition. Furthermore, Yang et al. [26] developed a subspace conjugate gradient method (STT). The search direction in the method is generated by minimizing a quadratic approximation of the objective function in a subspace spanned by the current negative gradient and the latest two search directions. That is, the search direction d_{k+1} is of form $d_{k+1} = -g_{k+1} + a_k s_k + b_k s_{k-1}$, for which both the descent condition and Dai-Liao conjugacy condition are guaranteed.

Inspired by the above, we present a new subspace three-term conjugate gradient method based on the subspace $[-g_{k+1}, s_k, y_k]$ for solving large-scale unconstrained optimization. The search direction of the proposed method is determined by minimizing the quadratic approximation of the objective function in the subspace. For the choices of parameters, new estimations of parameters ensure the global convergence of the proposed method under proper conditions.

This paper is organized as follows. In the next section, we describe the new method and show that the search direction satisfies the descent condition and Dai-Liao conjugacy condition. Section 3 establishes the global convergence result of the proposed method under appropriate assumptions. The numerical results and comparisons with other methods are reported in Section 4 and the conclusion is given in Section 5.

2. The subspace three-term conjugate gradient algorithm

In this section, by minimizing the quadratic approximation of the objective function in the subspace $A_k \triangleq \text{span}\{-g_{k+1}, s_k, y_k\}$, we derive the search direction and propose a new subspace three-term conjugate gradient method (STCG).

Denote

$$\Phi_{k+1}(d) = g_{k+1}^{\mathrm{T}} d + \frac{1}{2} d^{\mathrm{T}} B_{k+1} d \tag{2.1}$$

as the quadratic approximation of function f in the subspace A_k, where $d \in A_k$, B_{k+1} is an approximation of the Hessian $\nabla^2 f(x_{k+1})$.

Let

$$d_{k+1} = -g_{k+1} + a_k s_k + b_k y_k \tag{2.2}$$

be the minimizer of $\Phi_{k+1}(d)$ in the subspace A_k. We describe the STCG algorithm as follows, in which the acceleration scheme given in [1] is employed.

Algorithm 2.1 (STCG)
Step 0. Choose an initial point $x_0 \in \mathbb{R}^n$, $\varepsilon > 0$, and compute $f_0 = f(x_0)$, $g_0 = \nabla f(x_0)$. Set $d_0 := -g_0$ and $k := 0$.
Step 1. If $\|g_k\| < \varepsilon$, stop, else go to Step 2.
Step 2. Compute a step-length α_k by the Wolfe line search (1.4) and (1.5).
Step 3. Compute x_{k+1} by the acceleration scheme,
3.1. Compute $z = x_k + \alpha_k d_k$, $g_z = \nabla f(z)$ and $y_z = g_k - g_z$;
3.2. Compute $\bar{a}_k = \alpha_k g_k^{\mathrm{T}} d_k$ and $\bar{b}_k = -\alpha_k y_k^{\mathrm{T}} d_k$;
3.3. Acceleration scheme. If $\bar{b}_k > 0$, then compute $\xi_k = -\bar{a}_k / \bar{b}_k$ and update the variables as $x_{k+1} = x_k + \xi_k \alpha_k d_k$, otherwise update the variables as $x_{k+1} = x_k + \alpha_k d_k$.
Step 4. Compute $f_{k+1} = f(x_{k+1})$, $g_{k+1} = g(x_{k+1})$, $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$.
Step 5. Compute a_k and b_k by Subalgorithm 2.2.
Step 6. Compute the search direction d_{k+1} by (2.2). Set $k := k+1$ and go to Step 1.

REMARK 2.1. In Step 2, the Wolfe line search employs the quadratic and cubic interpolation conditions in the program, which greatly reduces the number of inner loops as shown in Section 4.

In the following, we discuss the computations of a_k and b_k. Obviously, (a_k, b_k) is the solution of the following minimizing problem:

$$\min_{a,b \in \mathbb{R}} \Phi_{k+1}(-g_{k+1} + a s_k + b y_k). \tag{2.3}$$

Considering that $\nabla^2 f(x_{k+1}) s_k \approx y_k$, we would like to suppose that B_{k+1} is positive definite, and choose B_{k+1} such that the quasi-Newton equation $B_{k+1} s_k = y_k$. In what follows, two cases should be considered.

Case I. $\dim(A_k) = 2$, and $-g_{k+1}$ and s_k are linearly independent, then let $b_k = 0$, (2.3) can be rewritten as

$$\min_{a \in \mathbb{R}} \Phi_{k+1}(-g_{k+1} + a s_k), \tag{2.4}$$

the solution of (2.4) can be expressed as

$$a_k = \frac{g_{k+1}^{\mathrm{T}} B_{k+1} s_k - g_{k+1}^{\mathrm{T}} s_k}{s_k^{\mathrm{T}} B_{k+1} s_k}. \tag{2.5}$$

If the exact line search is used, that is $g_{k+1}^{\mathrm{T}} s_k = 0$, then (2.5) reduces to

$$a_k = \frac{g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} s_k}, \tag{2.6}$$

which just implies HS conjugate gradient method.

Case II. $\dim(A_k) = 3$. It is obvious that a_k and b_k can be expressed as solutions of the following linear algebraic system:

$$\begin{cases} a(y_k^{\mathrm{T}} s_k) + b(y_k^{\mathrm{T}} y_k) = g_{k+1}^{\mathrm{T}} y_k - g_{k+1}^{\mathrm{T}} s_k, \\ a(y_k^{\mathrm{T}} y_k) + b(y_k^{\mathrm{T}} B_{k+1} y_k) = g_{k+1}^{\mathrm{T}} B_{k+1} y_k - g_{k+1}^{\mathrm{T}} y_k. \end{cases} \tag{2.7}$$

By using the formula $B_{k+1} s_k = y_k$ and Cauchy inequality, the coefficient determinant of the system (2.7) satisfies

$$\Delta_k = \|B_{k+1}^{\frac{1}{2}} s_k\|^2 \|B_{k+1}^{\frac{1}{2}} y_k\|^2 - (B_{k+1}^{\frac{1}{2}} s_k)^{\mathrm{T}} (B_{k+1}^{\frac{1}{2}} y_k) > 0. \tag{2.8}$$

From $g_{k+1}^{\mathrm{T}} s_k = 0$, a_k and b_k are computed by

$$\begin{aligned} a_k &= \frac{1}{\Delta_k}[(y_k^{\mathrm{T}} B_{k+1} y_k)(g_{k+1}^{\mathrm{T}} y_k) - (y_k^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} B_{k+1} y_k - g_{k+1}^{\mathrm{T}} y_k)], \\ b_k &= \frac{1}{\Delta_k}[(y_k^{\mathrm{T}} s_k)(g_{k+1}^{\mathrm{T}} B_{k+1} y_k - g_{k+1}^{\mathrm{T}} y_k) - (y_k^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} y_k)]. \end{aligned} \tag{2.9}$$

In (2.9), the quantities $\eta_k \equiv y_k^{\mathrm{T}} B_{k+1} y_k$ and $\omega_k \equiv g_{k+1}^{\mathrm{T}} B_{k+1} y_k$ should be evaluated. Using the quasi-Newton equation $B_{k+1} s_k = y_k$, we have

$$\begin{aligned} \eta_k &= \frac{y_k^{\mathrm{T}} B_{k+1} y_k s_k^{\mathrm{T}} B_{k+1} s_k}{(y_k^{\mathrm{T}} B_{k+1} s_k)^2} \cdot \frac{(y_k^{\mathrm{T}} B_{k+1} s_k)^2}{s_k^{\mathrm{T}} B_{k+1} s_k} \\ &= \frac{1}{\cos^2\langle B_{k+1}^{\frac{1}{2}} y_k, B_{k+1}^{\frac{1}{2}} s_k\rangle} \cdot \frac{(y_k^{\mathrm{T}} y_k)^2}{y_k^{\mathrm{T}} s_k}, \end{aligned} \tag{2.10}$$

where $\cos^2\langle B_{k+1}^{\frac{1}{2}} y_k, B_{k+1}^{\frac{1}{2}} s_k\rangle$ is unknown. By taking advantage of the mean value $\frac{1}{2}$ of $\cos^2\xi$, it seems reasonable to replace $\cos^2\langle B_{k+1}^{\frac{1}{2}} y_k, B_{k+1}^{\frac{1}{2}} s_k\rangle$ by $1/2$. Therefore, η_k can be computed as

$$\eta_k = 2\frac{(y_k^{\mathrm{T}} y_k)^2}{y_k^{\mathrm{T}} s_k}. \tag{2.11}$$

Then

$$\Delta_k = \eta_k (y_k^{\mathrm{T}} s_k) - (y_k^{\mathrm{T}} y_k)^2 = (y_k^{\mathrm{T}} y_k)^2. \tag{2.12}$$

In order to compute ω_k, we use the BFGS update with the scalar matrix $\gamma_0 I$, in which $\gamma_0 = \frac{y_k^{\mathrm{T}} y_k}{s_k^{\mathrm{T}} y_k}$, to obtain B_{k+1}, and then

$$\begin{aligned} \omega_k &= g_{k+1}^{\mathrm{T}}[\gamma_0 I + \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k} - \gamma_0 \frac{s_k s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} s_k}] y_k \\ &= \gamma_0 g_{k+1}^{\mathrm{T}} y_k + \frac{y_k^{\mathrm{T}} y_k \cdot g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} s_k}. \end{aligned} \tag{2.13}$$

By substituting (2.11)-(2.13) into (2.9), we have

$$\begin{aligned} a_k &= \frac{g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} s_k} - (\gamma_0 - 1)\frac{g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} y_k}, \\ b_k &= (\gamma_0 - 1)\frac{y_k^{\mathrm{T}} s_k \cdot g_{k+1}^{\mathrm{T}} y_k}{(y_k^{\mathrm{T}} y_k)^2}. \end{aligned} \tag{2.14}$$

From the above discussion, we use the following subalgorithm to describe the computations of the scalars a_k and b_k.

Subalgorithm 2.2 (The computations of a_k and b_k)
Step 1. Compute $g_k^{\mathrm{T}} s_k$, $g_{k+1}^{\mathrm{T}} y_k$, $y_k^{\mathrm{T}} y_k$, $y_k^{\mathrm{T}} s_k$ and γ_0, respectively.
Step 2. Compute a_k and b_k by (2.6) and (2.14), respectively. Namely,
If $\dim(A_k) = 2$, and $-g_{k+1}$ and s_k are linearly independent,

$$a_k = \frac{g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} s_k},\ b_k = 0, \tag{2.15}$$

else, $\dim(A_k) = 3$,

$$\begin{aligned} a_k &= \frac{g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} s_k} - (\gamma_0 - 1)\frac{g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} y_k}, \\ b_k &= (\gamma_0 - 1)\frac{y_k^{\mathrm{T}} s_k g_{k+1}^{\mathrm{T}} y_k}{(y_k^{\mathrm{T}} y_k)^2}. \end{aligned} \tag{2.16}$$

REMARK 2.2. In Subalgorithm 2.2, a_k and b_k are determined by (2.15) and (2.16) corresponding to Case I and Case II, respectively. In virtue of the discussion of two cases, it could be expected to improve the stability of numerical calculation.

In the following, we show that the search direction satisfies both the descent condition and the Dai-Liao conjugacy condition.

LEMMA 2.1. *Suppose that B_{k+1} is positive definite. Then d_{k+1} generated by Algorithm 2.1 is a descent direction.*

Proof. From (2.1) observe that $\Phi_{k+1}(0) = 0$. Since B_{k+1} is positive definite and d_{k+1} is generated by Algorithm 2.1, it follows that $\Phi_{k+1}(d_{k+1}) \le 0$. Therefore,

$$g_{k+1}^{\mathrm{T}} d_{k+1} \le -\frac{1}{2} d_{k+1}^{\mathrm{T}} B_{k+1} d_{k+1} < 0, \tag{2.17}$$

i.e., d_{k+1} is a descent direction. ∎

LEMMA 2.2. *Let the search direction d_{k+1} be generated by Algorithm 2.1. Then d_{k+1} satisfies the Dai-Liao conjugacy condition $y_k^{\mathrm{T}} d_{k+1} = -g_{k+1}^{\mathrm{T}} s_k$.*

Proof. From (2.2) and the first equation in (2.7), it follows that d_{k+1} satisfies $y_k^{\mathrm{T}} d_{k+1} = -g_{k+1}^{\mathrm{T}} s_k$, which is exactly the Dai-Liao conjugacy condition. ∎

In the next section, we give some suitable assumptions and establish the global convergence of the STCG.

3. Convergence analysis

In this section, we show the proposed method is globally convergent with the Wolfe line search under appropriate assumptions. Without loss of generality, we use the following assumptions throughout the paper.

ASSUMPTION 3.1. *The level set* $\Omega=\{x\in\mathbb{R}^n:\ f(x)\le f(x_0)\}$ *is bounded.*

ASSUMPTION 3.2. *The function* $f:\mathbb{R}^n\to\mathbb{R}$ *is continuously differentiable and its gradient is Lipschitz continuous in a neighborhood* $\mathbb{N}$ *of* Ω*, i.e., there exists a constant* $L>0$ *such that*

$$\|g(x)-g(y)\|\le L\|x-y\|,\quad \forall\ x,y\in\mathbb{N}. \tag{3.1}$$

ASSUMPTION 3.3. *For the function* f *on* Ω*, there exists a constant* $\mu>0$ *such that*

$$(\nabla f(x)-\nabla f(y))^{\mathrm{T}}(x-y)\ge\mu\|x-y\|^2,\quad \forall\ x,y\in\mathbb{N}. \tag{3.2}$$

Under these assumptions on f*, there exists a constant* $\Gamma>0$ *such that*

$$\|g(x)\|\le\Gamma,\quad \forall\ x\in\Omega. \tag{3.3}$$

In order to establish the convergence of the STCG, firstly we need to derive a lower bound on the step length α_k.

LEMMA 3.1. *Let* $\{d_k\}$ *be generated by Algorithm 2.1. Suppose that Assumption 3.2 holds. Then the step-length* α_k *with the Wolfe line search (1.4) and (1.5) satisfies*

$$\alpha_k\ge\frac{(1-\sigma)|g_k^{\mathrm{T}}d_k|}{L\|d_k\|^2}. \tag{3.4}$$

The following lemma is called Zoutendijk condition, which is essentially suggested by Zoutendijk and Wolfe.

LEMMA 3.2 ([28]). *Consider any iteration of the form (1.2), where* d_k *is a descent direction and the step-length* α_k *is determined by the Wolfe line search (1.4) and (1.5). Suppose that Assumptions 3.1 and 3.2 hold. Then*

$$\sum_{k=0}^{\infty}\frac{(g_k^{\mathrm{T}}d_k)^2}{\|d_k\|^2}<+\infty. \tag{3.5}$$

From Lemma 2.1, we can obtain that the sequence $\{d_k\}$ generated by Algorithm 2.1 satisfies (3.5). Next lemma shows that the sequence of gradient norms $\|g_k\|$ can be bounded away from zero only if $\sum_{k\ge0}1/\|d_k\|<+\infty$ for any conjugate gradient method with strong Wolfe line search.

LEMMA 3.3. *Consider any iteration of the form (1.2), where* d_k *is a descent direction and the step-length* α_k *is determined by strong Wolfe line search (1.4) and (1.6). Suppose that Assumptions 3.1 and 3.2 hold. If*

$$\sum_{k\ge0}\frac{1}{\|d_k\|^2}=+\infty, \tag{3.6}$$

then

$$\liminf_{k\to\infty}\|g_k\|=0. \tag{3.7}$$

In the following, we establish the global convergence theorem of Algorithm 2.1 with the Wolfe line search.

THEOREM 3.1. *Let the sequence $\{x_k\}$ be generated by Algorithm 2.1, where the step-length α_k is determined by the Wolfe line search (1.4) and (1.5). Suppose that Assumptions 3.1, 3.2 and 3.3 hold. Then (3.7) holds.*

Proof. From (3.1) we have

$$\|y_k\|\le L\|s_k\|. \tag{3.8}$$

From (3.2), we have

$$y_k^{\mathrm{T}}s_k\ge\mu\|s_k\|^2. \tag{3.9}$$

From Cauchy inequality and (3.9), it is obvious that

$$\|y_k\|\ge\mu\|s_k\|. \tag{3.10}$$

Using triangle inequality, Cauchy inequality, (3.8) and (3.9), we obtain

$$|\gamma_0-1|\le\frac{|y_k^{\mathrm{T}}s_k|+\|y_k\|^2}{|y_k^{\mathrm{T}}s_k|}\le\frac{L\|s_k\|^2+L^2\|s_k\|^2}{\mu\|s_k\|^2}=\frac{L+L^2}{\mu}\equiv M_0. \tag{3.11}$$

In the following, two cases should be discussed.

Case I. $\dim(A_k)=2$, and $-g_{k+1}$, s_k are linearly independent, then a_k and b_k are determined by (2.15).

Using (3.3), (3.8) and (3.9), we have

$$|a_k|\le\frac{|g_{k+1}^{\mathrm{T}}y_k|}{|y_k^{\mathrm{T}}s_k|}\le\frac{\Gamma L}{\mu}\frac{1}{\|s_k\|}\equiv M_1\frac{1}{\|s_k\|}. \tag{3.12}$$

Therefore,

$$\|d_{k+1}\|\le\|g_{k+1}\|+|a_k|\|s_k\|\le\Gamma+M_1. \tag{3.13}$$

Case II. $\dim(A_k)=3$, then a_k and b_k are determined by (2.16).

Using (3.3) and (3.8)-(3.11), we have

$$\begin{aligned}|a_k|&\le\frac{|g_{k+1}^{\mathrm{T}}y_k|}{|y_k^{\mathrm{T}}s_k|}+|\gamma_0-1|\frac{|g_{k+1}^{\mathrm{T}}y_k|}{\|y_k\|^2}\\&\le\frac{\Gamma L\|s_k\|}{\mu\|s_k\|^2}+M_0\frac{\Gamma L\|s_k\|}{\mu^2\|s_k\|^2}\\&=\frac{\Gamma L}{\mu}(1+\frac{M_0}{\mu})\frac{1}{\|s_k\|}\equiv M_2\frac{1}{\|s_k\|}.\end{aligned} \tag{3.14}$$

On the other hand,

$$|b_k|\le|\gamma_0-1|\frac{|y_k^{\mathrm{T}}s_k||g_{k+1}^{\mathrm{T}}y_k|}{\|y_k\|^4}$$

$$\leq M_0 \frac{L\|s_k\|^2 \Gamma \|y_k\|}{\mu^2 \|s_k\|^2 \|y_k\|^2}$$
$$= \frac{M_0 \Gamma L}{\mu^2} \frac{1}{\|y_k\|} \equiv M_3 \frac{1}{\|y_k\|}. \tag{3.15}$$

Therefore,

$$\|d_{k+1}\| \leq \|g_{k+1}\| + |a_k| \|s_k\| + |b_k| \|y_k\| \leq \Gamma + M_2 + M_3. \tag{3.16}$$

From Lemma 3.3, it follows that (3.7) holds. □

4. Numerical results

In this section, we report some numerical results. The 80 test problems are the unconstrained problems from [5] with the dimensions varying from 2 to 100000. All codes are written in Matlab R2010a and ran on PC with 2.20 GHz CPU processor and 2.00 GB RAM memory. The iteration is terminated by the following conditions

$$\|g_k\| \leq \varepsilon \quad \text{or} \quad |f(x_{k+1}) - f(x_k)| \leq \varepsilon \max\{1.0, |f(x_k)|\}. \tag{4.1}$$

We set the parameters $\varepsilon = 10^{-6}$ in (4.1), and $\rho = 0.35$, $\sigma = 0.5$ in the Wolfe line search, the other parameters are set as default. Table 4.1 lists the test problems and their dimensions.

No.	Prob	dim
1.	*the Freudenstein and Roth function*	2
2.	*Powell badly scaled function*	2
3.	*Brown badly scaled function*	2
4.	*the Beale function*	2
5.	*the Helical valley function*	3
6.	*the Gaussian function*	3
7.	*the Biggs EXP 6 function*	6
8.	*the trigonometric function*	10
9.	*the Variable dimension function*	50,100,500,1000,5000,10000
10.	*the Chebyquad function*	50,100,500,1000,5000,10000
11.	*the Boundary value function*	50,100,500,1000,5000,10000
12.	*Integral equation function*	50,100,500,1000,5000,10000
13.	*Broyden tridiagonal function*	50,100,500,1000,5000,10000
14.	*Separable cubic function*	50,100,500,1000,5000,10000
15.	*Nearly separable function*	50,100,500,1000,5000,10000
16.	*Yang tridiagonal function*	50,100,500,1000,5000,10000
17.	*Allgower function*	50,100,500,1000,5000,10000
18.	*Schittkowski function* 302	50,100,500,1000,5000,10000
19.	*Extended Powell singular function*	500,1000,5000,10000,50000,100000
20.	*the Penalty function I*	500,1000,5000,10000,50000,100000

TABLE 4.1. *The test problems and their dimensions*

We compare the numerical performance of the proposed algorithm STCG with the following three algorithms: the MTHREECG [8], the TTS [2] and the STT [26]. The numerical performance of the MTHREECG is superior to the recently developed methods THREECG [3] and CG_DESCENT [13], which have been acclaimed to be powerful for solving unconstrained optimization problems. Compared with the classical conjugate gradient methods such as PRP [20, 21], HS [14] and DY [6], the numerical

performance of the TTS outperforms. A large number of numerical results show that the STT is competitive among the existing conjugate gradient methods for solving unconstrained optimization.

We use the profiles by Dolan and Moré [9] to compare the performance of the above four algorithms. In a performance profile plot, the top curve is the method that solved the most problems in a time that is within a factor of the best time. The horizontal axis gives the percentage (τ) of the test problems for which a method is the fastest (efficiency), while the vertical side gives the percentage (ψ) of the test problems that are successfully solved by each of the methods (robustness).

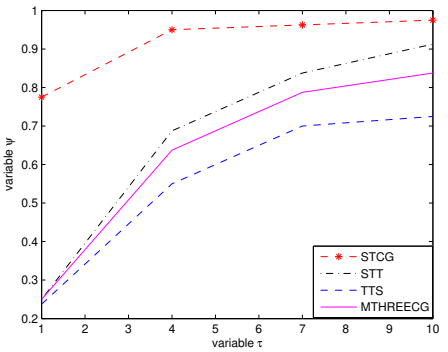

FIG. 4.1. *The number of iterations.*

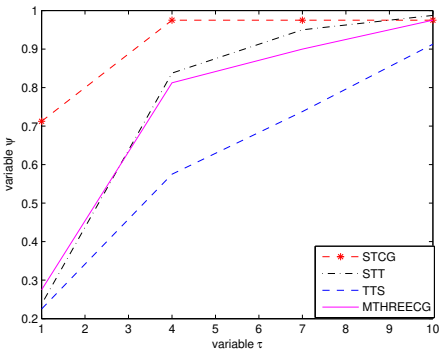

FIG. 4.2. *The CPU time.*

Figures 4.1-4.4 plot the performance profiles for the number of iterations (k), the CPU time (t), the number of function evaluations (nf) and the number of gradient evaluations (ng), respectively, which clearly show that the curve "STCG" is the best.

If program runs fail, or the number of iterations can reach more than 500, it is regarded as failed. And we denote the number of iterations, function evaluations, gradient evaluations by 500, respectively, and denote the CPU time by 1000 seconds. In this way, the numerical results indicate that the STCG is promising.

It is worth mentioning that the Wolfe line search in our algorithm adopts quadratic and cubic interpolation conditions in the program which can reduce the number of

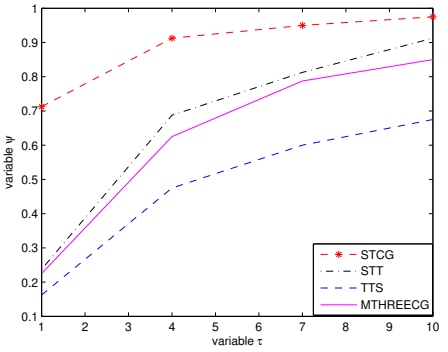

FIG. 4.3. *The number of function evaluations.*

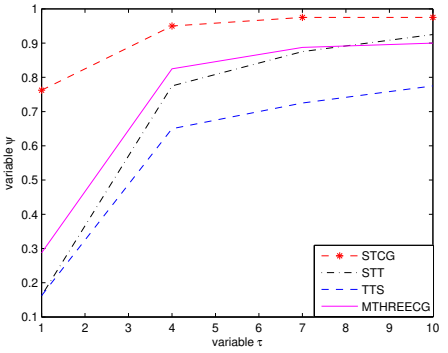

FIG. 4.4. *The number of gradient evaluations.*

No.	Prob	dim	Wolfe $k/nf/ng/t$	Armijo $k/nf/ng/t$
1.	*the Freudenstein and Roth function*	2	10/24/13/0	15/150/17/0.0469
2.	*the Helical valley function*	3	205/341/272/0.0938	492/4101/494/0.2500
3.	*the trigonometric function*	10	9/46/28/0.0313	10/151/12/0.0313
4.	*the Variable dimension function*	50	3/15/5/0.0313	38/610/40/0.0625
5.	*the Variable dimension function*	100	3/24/6/0.0313	2/90/4/0.0313
6.	*the Variable dimension function*	500	3/32/5/0.0313	3/158/5/0.0469
7.	*the Variable dimension function*	1000	3/40/4/0.0313	3/174/5/0.0469
8.	*the Variable dimension function*	5000	3/44/6/0.0469	3/243/5/0.0938
9.	*the Variable dimension function*	10000	3/46/9/0.0625	3/264/5/0.1875
10.	*the Penalty function I*	500	2/52/4/0.0313	2/54/4/0.0469
11.	*the Penalty function I*	1000	2/11/4/0.0469	2/60/4/0.0469
12.	*the Penalty function I*	5000	22/92/53/0.1875	29/419/33/0.1875
13.	*the Penalty function I*	10000	23/176/102/0.0936	27/314/30/0.3906
14.	*the Penalty function I*	50000	17/81/43/1.0156	35/706/38/1.8906
15.	*the Penalty function I*	100000	13/66/32/1.3438	40/830/44/3.8281

TABLE 4.2. *The numerical results*

inner loops in a way. In the Armijo-type line search, the step-length α_k is determined by letting $\alpha_k = \max\{\sigma^j,\ j = 0,1,2,\ldots\}$ satisfy the condition

$$f(x_k + \sigma^j d_k) \leq f(x_k) + \rho\sigma^j g_k^{\mathrm{T}} d_k, \tag{4.2}$$

where ρ, $\sigma \in (0,1)$ are constants. Here, we set $\rho = 0.35$, $\sigma = 0.5$ in (4.2) and other parame-

ters are set as default. For the STCG, we randomly compare the numerical performance of using the Wolfe line search with that of using the Armijo-type line search for 15 test problems with different dimensions and other conditions being the same. The numerical results in Table 4.2 provide evidences that the number of function evaluations (nf) and the number of gradient functions (ng) are both less when the Wolfe line search is used. Specifically, the advantage is greater when the dimension of the test problem is larger.

5. Conclusion

In this paper, a new subspace three-term conjugate gradient method has been presented. In this method, the search direction is determined by minimizing the affine quadratic approximate of the objective function in a subspace which is discussed in two cases. We prove the global convergence of the proposed method with the Wolfe line search under proper conditions. For a set of 80 test problems, the performance profiles show that the proposed method is robust and meaningful.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (11171003), the Innovation Talent Training Program of Science and Technology of Jilin Province of China (20180519011JH) and the Science and Technology Development Project Program of Jilin Province (20190303132SF).

REFERENCES

[1] N. Andrei, *Acceleration of conjugate gradient algorithms for unconstrained optimization*, Appl. Math. Comput., 213:361–369, 2009. 2

[2] N. Andrei, *An accelerated subspace minimization three-term conjugate gradient algorithm for unconstrained optimization*, Numer. Algorithms, 65:859–874, 2014. 1, 1, 4

[3] N. Andrei, *On three-term conjugate gradient algorithms for unconstrained optimization*, Appl. Math. Comput., 219:6316–6327, 2013. 1, 4

[4] R.L. Chen and J.P. Zeng, *A new subspace correction method for nonlinear unconstrained convex optimization problems*, Acta Math. Appl. Sin., 4:745–756, 2012. 1

[5] Y.H. Dai and Y.X. Yuan, *An efficient hybrid conjugate gradient method for unconstrained optimization*, Ann. Oper. Res., 103:33–47, 2001. 4

[6] Y.H. Dai and Y.X. Yuan, *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM J. Optim., 10:177–182, 1999. 1, 4

[7] S.H. Deng, J. Lv, and Z. Wan, *A new Dai-Liao type of conjugate gradient algorithm for unconstrained optimization problems*, Pacific J. Optim., 15(2):237–248, 2019. 1

[8] S.H. Deng and Z. Wan, *A three-term conjugate gradient algorithm for large-scale unconstrained optimization problems*, Appl. Numer. Math., 92:70–81, 2015 . 1, 4

[9] E. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91:201–213, 2002 . 4

[10] R. Flecther, *Practical Methods of Optimization. Vol I: Unconstrained Optimization*, Wiley & Sons, New York, 1987. 1

[11] R. Flether and C.M. Reeves, *Function minimization by conjugate gradient*, Comput. J., 7:149–154, 1964. 1

[12] J. Guo and Z. Wan, *A modified spectral PRP conjugate gradient projection method for solving large-scale monotone equations and its application in compressed sensing*, Math. Probl. Eng., 2019:5261830, 2019. 1

[13] W.W. Hager and H.C. Zhang, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM J. Optim., 16:170–192, 2005. 1, 4

[14] M.R. Hestenes and E.L. Stiefel, *Methods of conjugate gradient for solving linear systems*, J. Res. Natl. Bur. Stand., 6:409–436, 1952. 1, 4

[15] T. Li and Z. Wan, *New adaptive Barzilar-Borwein step size and its application in solving large scale optimization problems*, ANZIAM J., 61:76–98, 2019. 1

[16] H.W. Liu and Z.X. Liu, *An efficient Barzilai-Borwein conjugate gradient method for unconstrained optimization*, J. Optim. Theory Appl., 180:879–906, 2019. 1

[17] J.K. Liu, *Convergence properties of a class of nonlinear conjugate gradient methods*, Comput. Oper. Res., 40:2656–2661, 2013. 1

[18] Y.L. Liu and C.S. Storey, *Efficient generalized conjugate gradient algorithms, Part I: Theory*, J. Optim. Theory Appl., 69:129–137, 1991. 1

[19] L.B. Lv, *A new algorithm for solving large-scale trust region subproblem*, Oper. Res. Manage. Sci., 5:48–52, 2007. 1

[20] E. Polak and G. Ribiére, *Note sur la convergence des méthodes de directions conjuguées*, ESAIM Math. Model. Numer. Anal., 3:35–43, 1969. 1, 4

[21] B.T. Polyak, *The conjugate gradient method in extreme problems*, USSR Comput. Math. Math. Phys., 9:94–112, 1969. 1, 4

[22] J. Stoer and Y.X. Yuan, *A subspace study on conjugate gradient algorithms*, Z. Angew. Math. Mech., 75:69–77, 1995. 1

[23] Z.H. Wang, *A limited memory trust region method for unconstrained optimization and its omplementation*, Math. Numer. Sin., 4:395–404, 2005. 1

[24] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Rev., 11:226–235, 1968 . 1

[25] P. Wolfe, *Convergence conditions for ascent methods, (II)*, SIAM Rev., 13:185–188, 1971 . 1

[26] Y.T. Yang, Y.T. Chen, and Y.L. Lu, *A subspace conjugate gradient algorithm for large-scale unconstrained optimization*, Numer. Algorithms, 76:813–828, 2017. 1, 1, 4

[27] Y.X. Yuan, *Subspace techniques for nonlinear optimization*, in R. Jeltsch, T.-T. Li and I.H. Sloan (eds.), Some Topics in Industrial and Applied Mathematics, World Scientific, 2006. 1

[28] G. Zoutendijk, *Nonlinear programming. Computational method*, in J. Abadie (ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 37–86, 1970. 3.2