ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 9, No. 4, 2014, *pp.* 252-261

A steepest descent method with a Wolf type line search

Ju Jing-jie, Pang De-yan, Du Shou-qiang

College of Mathematics, Qingdao University, Qingdao 266071

(*Received February 12, 2014, accepted September 11, 2014*)

Abstract. The steepest decent method is proposed by French mathematician Cauchy and it is one of the simplest and oldest methods for solving unconstrained optimization problems. The steepest descent method, negative gradient direction is chosen as the search direction, also known as the gradient method. Usually, the step length α_k of the steepest decent method can be computed by some inexact line search. In this paper, we will use a Wolfe type line search to evaluate the step length and its convergence property will be given under mild assumptions. From the numerical results, we can see that the steepest descent method with this Wolfe type line search is very promising. Finally, we give an application of the method to solve the nonlinear complementarity problem.

Keywords: the steepest decent method, line search, global convergence

1. Introduction

We consider the following unconstrained optimization problem

$$\min_{x\in R^n} f(x), \tag{1}$$

where $f: R^n \to R$ is continuously differentiable. In the past four decades, many theories and algorithms in globally optimal problems have been developed ([1-15]). Among these methods, the steepest decent method proposed by French mathematician Cauchy is one of the well-known and practical methods for global optimization problems. Because it has the simple structure, the less amount of calculation and the fast convergence speed when it away from the minimum value of the problem. So the steepest descent method has become one of the most famous methods for solving large-scale optimization problem. The steepest descent method has been studied deeply by many scholars, such as [1-4]. In practice, it often used in communication, computer and information engineering and other fields. Therefore, it has an important meaning for the study of the steepest descent method.

The classical steepest descent method for the continuously differentiable function $f: R^n \to R$ is defined by the iteration

$$x_{k+1} = x_k + \alpha_k d_k,$$

where d_k is the steepest descent direction at x_k and α_k is the step length at x_k .Since we use a negative gradient direction as the search direction, so the steepest decent method is also sometimes called the gradient method. In practice, the step length is computed by an inexact line search, this ensures an appropriate reduction in the objective function. Usually, we use Armijo, Wolfe and other inexact line search rule to compute the step length α_k (such as [2]). In this paper, we use a Wolfe type line search to computing the step length. The numerical results in Section 4 indicate that, for some problems, the steepest descent method with Wolfe type line search can obtain the global minimum solutions most close to the function, as well as the steepest descent with Armijo line search.

The Wolfe type line search (see [5]) as follows:

Choose $\alpha_k > 0$ such as

$$f(x_k) - f(x_k + \alpha_k d_k) \ge \rho \alpha_k^2 \|d_k\|^2 \tag{2}$$

$$g(x_k + \alpha_k d_k)^T d_k \ge -2\sigma \alpha_k \|d_k\|^2 \tag{3}$$

where $0 < \rho < \sigma < 1$.

This paper is organized as follows. In Section 2, we will give the algorithm of the steepest descent method with the Wolfe type line search and the steepest descent method with Armijo line search, respectively. The global convergence of the methods will be given in Section 3. In Section 4, some numerical results are reported for both the steepest descent method with the Wolfe line search, the steepest descent method with Armijo line search and some conjugate gradient methods. Finally, in Section 5, we give the conclusion and an application of the steepest descent method for solving nonlinear complementarity problem.

In this paper, we denote $g_k = g(x_k) = \nabla f(x_k)$. The norm $\|\cdot\|$ is the Euclidean norm.

2. Algorithm

In this section, we will give the steepest descent method with the Wolfe type line search and the Armijo line search, respectively. The steepest descent method with Armijo line search has been detail introduced in [2], so we will only give the algorithm.

Algorithm 2.1

Step 0. Given $x_0 \in R^n$, $\beta \in (0,1)$, $\sigma \in (0,0.5)$, $0 < \varepsilon << 1$, $k = 1$.

Step 1. Compute g_k. If $\|g_k\| \le \varepsilon$, stop. Output x_k as the approximate optimal solution.

Step 2. Choose $d_k = -g_k$.

Step 3. Compute α_k by

$$f(x_k + \beta^m d_k) \le f(x_k) + \sigma\beta^m g_k^T d_k .$$

Step 4. Set $x_{k+1} = x_k + \alpha_k d_k$, $k = k+1$, go to Step 1.

In the following, we present the steepest descent method with the Wolfe type line search.

Algorithm 2.2

Step 0. Given $x_0 \in R^n$, $\beta \in (0,1)$, $\sigma \in (0,0.5)$, $0 < \varepsilon << 1$, $k = 1$.

Step 1. Compute g_k. If $\|g_k\| \le \varepsilon$, stop. Output x_k as the approximate optimal solution.

Step 2. Choose $d_k = -g_k$.

Step 3. Compute α_k by (2) and (3).

Step 4. Set $x_{k+1} = x_k + \alpha_k d_k$, $k = k+1$, go to Step 1.

As a comparison, in Section 4, we will give the numerical results of the above two algorithms, respectively.

3. Convergence analysis of the algorithms

The convergence result of Algorithm 2.1 has been given in [2] in detail. In the following, we only give the global convergence result of Algorithm2.2.

In order to establish the global convergence of Algorithm 2.2, firstly, we give the following Assumption **3.1 and Lemma 3.1.**

Assumption 3.1. $f(x)$ is bounded below on the level set

$$L_0 = \left\{ x \in R^n \,\middle|\, f(x) \le f(x_0) \right\}.$$

Lemma 3.1. Suppose that Assumption 3.1 holds, then the Wolfe type line search (2) and (3) is feasible (see [5]).

Now, we give the following global convergence theorem for the steepest descent method with the Wolfe type line search under mild assumptions.

Theorem 3.1. Suppose that Assumption 3.1 hold, $g(x)$ is Lipschitz continuous on the level set L_0. If α_k satisfies (2) and (3) and $\{x_k\}$ is generated by the Algorithm 2.2, then

$$\lim_{k\to\infty}\|g(x_k)\|=0. \tag{4}$$

Proof. From (2) and Assumption 3.1, we know that

$$\lim_{k\to\infty}\alpha_k\|d_k\|=0 \ . \tag{5}$$

By (3), we have

$$g(x_k+\alpha_k d_k)^T\frac{d_k}{\|d_k\|}\geq -2\sigma\alpha_k\|d_k\|,$$

i.e.,

$$\|g(x_k+\alpha_k d_k)-g(x_k)\|\geq (g(x_k+\alpha_k d_k)-g(x_k))^T\frac{d_k}{\|d_k\|}$$

$$\geq -\frac{g(x_k)^T d_k}{\|d_k\|}-2\sigma\alpha_k\|d_k\|.$$

Because $g(x)$ is Lipschitz continuous on the level set L_0, we know

$$\|g(x_k+\alpha_k d_k)-g(x_k)\|\leq L\|x_k+\alpha_k d_k-x_k\|=L\|\alpha_k d_k\|,$$

where L is the Lipschitz constant. So

$$L\|\alpha_k d_k\|\geq -\frac{g(x_k)^T d_k}{\|d_k\|}-2\sigma\alpha_k\|d_k\| \ .$$

Since $d_k=-g(x_k)$, we have

$$0\leq -\frac{g(x_k)^T d_k}{\|d_k\|}\leq (L+2\sigma)\|\alpha_k d_k\|.$$

Let $k\to\infty$, by (5), we have

$$\lim_{k\to\infty}-\frac{g(x_k)^T d_k}{\|d_k\|}=0 \ .$$

Then, we know that

$$\lim_{k\to\infty}-\frac{g(x_k)^T d_k}{\|d_k\|}=\lim_{k\to\infty}\frac{\|g(x_k)\|^2}{\|g(x_k)\|}=\lim_{k\to\infty}\|g(x_k)\|=0 \ .$$

Therefore (4) holds.

4. Numerical results

In this section, we will give some numerical results. They are respectively from the Algorithm 2.1, the Algorithm 2.2, the FR conjugate gradient method, the PRP conjugate gradient method and the spectral conjugate gradient method.

In this section, the testing functions we will use are well-known functions that are often used [2, 6].

The test functions are the follow functions,

(i) 6-hump camel back function

$$f(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6-x_1x_2-4x_2^2+4x_2^4,\qquad -3\le x_1,x_2\le 3$$

The global minimum solutions $x^*=(0.0898,0.7127)$ or $x^*=(-0.0898,-0.7127)$ and $f^*=-1.0316$.

(ii) function

$$f(x)=100(x_1^2-x_2)^2+(x_1-1)^2,$$

The global minimum solution $x^*=(1.0000,1.0000)$ and $f^*=0.0000$.

(iii) Rastrigin function

$$f(x)=x_1^2+x_2^2\text{ - }\cos(18x_1)-\cos(18x_2),\qquad -1\le x_1,x_2\le 1$$

The global minimum solutions $x^*=(0.0000,0.0000)$ or $x^*=(-2.0000,0.0000)$ and $f^*=-2.0000$.

First presents the definition of several notations in the table,

x_0 indicates the initial point,

x^* indicates the calculated global minimizer,

$f(x^*)$ indicates the calculated function value of the global minimizer.

In algorithms we set $\beta=0.5$, $\rho=0.2$, $\sigma=0.4$ and $\varepsilon=10^{-5}$.

4.1. Part Ⅰ

The calculation results are given in the tables below. The data in the table (a) and (b) is computed by the Algorithm 2.1 and Algorithm 2.2, respectively.

Function (i)

Table 4.1.1. (a)

x_0	x^*	$f(x^*)$
(0.1,0.8)	(0.089842089632932,0.712656303523469)	-1.031628453489766
(0.424,0.286)	(0.089841909056407,0.712656424904444)	-1.031628453489829
(-0.089,-0.633)	(-0.089841963420045,-0.712656330523904)	-1.031628453489828

Table 4.1.1. (b)

x_0	x^*	$f(x^*)$
(0.1,0.8)	(0.089842074004650,0.712656538129374)	-1.031628453489722
(0.424,0.286)	(0.089841798439412,0.712656422581195)	-1.031628453489690
(-0.089,-0.633)	(-0.089841433791874,-0.712656440898500)	-1.031628453488535

Function (ii)

Table 4.1.2. (a)

x_0	x^*	$f(x^*)$
(0,0)	(0.999989222566424,0.999978407179369)	1.162980041764064e-010
(2,1)	(1.000010676532803,1.000021395023532)	1.141634442014956e-010
(1,-1)	(0.999988939604407,0.999977837179067)	1.225100304529775e-010

Table 4.1.2. (b)

x_0	x^*	$f(x^*)$
(0，0)	(0.999992175224127,0.999984335009511)	6.125114236575247e-011
(2，1)	(1.000007857200306,1.000015730032360)	6.175983917201136e-011
(1，-1)	(1.000007832119298,1.000015679790382)	6.136608808308800e-011

Function (iii)

Table 4.1.3. (a)

x_0	x^*	$f(x^*)$
(0.9,0.9)	(1.040758689281631, 1.040758689281631)	0.179774966368992
(0.087,0.05)	1.0e-007 * (0.114742873638724, -0.038252512615108)	-1.999999999999976
(0.898,0.91)	(1.040758715794927, 1.040758699645090)	0.179774966368884

Table 4.1.3. (b)

x_0	x^*	$f(x^*)$
(0.9,0.9)	1.0e-007 *(-0.188828543001469, -0.188828543001469)	-1.999999999999884
(0.087,0.05)	1.0e-008 *(0.980814674548121, 0.490561612735529)	-1.999999999999981
(0.898,0.91)	1.0e-007 *(0.141998555082016, 0.011710431701756)	-1.999999999999967

4.2. Part Ⅱ

The algorithm of the FR conjugate gradient method is as following,

Algorithm 4.1

Step 0. Given $x_0 \in R^n, 0 < \varepsilon << 1$. Compute $g_0 = \nabla f(x_0)$, let $k = 1$.

Step 1. If $\|g_k\| \le \varepsilon$, stop. Output x_k as the approximate optimal solution.

Step 2. Compute d_k by

$$d_k = \begin{cases} -g_k, & k = 0, \\ -g_k + \beta_{k-1} d_{k-1}, & k \ge 1, \end{cases}$$

where $\beta_{k-1} = \dfrac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$ $(k \ge 1)$.

Step 3. Compute α_k by

$$f(x_k + \beta^m d_k) \le f(x_k) + \sigma \beta^m g_k^T d_k.$$

Step 4. Set $x_{k+1} = x_k + \alpha_k d_k$, compute $g_{k+1} = \nabla f(x_{k+1})$.

Step 5. Set $k = k + 1$ and go to Step 1.

The calculation results are given in the tables below. The data in the table (a) and (b) is computed by the Algorithm 4.1 and Algorithm 2.2, respectively.

Function (i)

Table 4.2.1. (a)

x_0	x^*	$f(x^*)$
(0.1,0.8)	(0.089842697705424, 0.712656168820365)	-1.031628453487441
(0.424,0.286)	(0.089842587116499, 0.712656053536417)	-1.031628453487392
(-0.089,-0.633)	(-0.089841879036056, -0.712656181058019)	-1.031628453489433

Table 4.2.1.(b)

x_0	x^*	$f(x^*)$
(0.1,0.8)	(0.089842074004650,0.712656538129374)	-1.031628453489722
(0.424,0.286)	(0.089841798439412,0.712656422581195)	-1.031628453489690
(-0.089,-0.633)	(-0.089841433791874,-0.712656440898500)	-1.031628453488535

Function (ii)

Table 4.2.2. (a)

x_0	x^*	$f(x^*)$
(0,0)	(1.000008546928186, 1.000017142529611)	7.328617925712770e-011
(2,1)	(1.000008148420339, 1.000016314203475)	6.642667056318378e-011
(1,-1)	(1.000005740918587, 1.000011515455341)	3.307094284929168e-011

Table 4.2.2. (b)

x_0	x^*	$f(x^*)$
(0，0)	(0.999992175224127,0.999984335009511)	6.125114236575247e-011
(2，1)	(1.000007857200306,1.000015730032360)	6.175983917201136e-011
(1，-1)	(1.000007832119298,1.000015679790382)	6.136608808308800e-011

Function (iii)

Table 4.2.3. (a)

x_0	x^*	$f(x^*)$
(0.9,0.9)	(1.040758711091145, 1.040758711091145)	0.179774966368863
(0.087,0.05)	1.0e-008 * (0.099902941034896, -0.967539795907809)	-1.999999999999985
(0.898,0.91)	(1.040758710534585, 1.040758715778078)	0.179774966368869

Table 4.2.3. (b)

x_0	x^*	$f(x^*)$
(0.9,0.9)	1.0e-007 *(-0.188828543001469, -0.188828543001469)	-1.999999999999884
(0.087,0.05)	1.0e-008 *(0.980814674548121, 0.490561612735529)	-1.999999999999981
(0.898,0.91)	1.0e-007 *(0.141998555082016, 0.011710431701756)	-1.999999999999967

4.3. PartⅢ

The algorithm of the PRP conjugate gradient method is as following,

Algorithm 4.2

Step 0. Given $x_0 \in R^n, 0 < \varepsilon << 1$. Compute $g_0 = \nabla f(x_0)$, let $k = 1$.

Step 1. If $\|g_k\| \le \varepsilon$, stop. Output x_k as the approximate optimal solution.

Step 2. Compute d_k by

$$d_k = \begin{cases} -g_k, & k = 0, \\ -g_k + \beta_{k-1} d_{k-1}, & k \ge 1, \end{cases}$$

where $\beta_{k-1} = \dfrac{g_k^T (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}}$ $(k \ge 1)$.

Step 3. Compute α_k by

$$f(x_k + \beta^m d_k) \le f(x_k) + \sigma \beta^m g_k^T d_k.$$

Step 4. Set $x_{k+1} = x_k + \alpha_k d_k$, compute $g_{k+1} = \nabla f(x_{k+1})$.

Step 5. Set $k = k+1$ and go to Step 1.

The calculation results are given in the tables below. The data in the table (a) and (b) is computed by the Algorithm 4.2 and Algorithm 2.2, respectively.

Function (i)

Table 4.3.1. (a)

x_0	x^*	$f(x^*)$
(0.1,0.8)	(0.089841762699973, 0.712656219576425)	-1.031628453489404
(0.424,0.286)	(0.089842201183537, 0.712656571349010)	-1.031628453489539
(-0.089,-0.633)	(-0.089841794962414, -0.712655906515560)	-1.031628453487782

Table 4.3.1.(b)

x_0	x^*	$f(x^*)$
(0.1,0.8)	(0.089842074004650,0.712656538129374)	-1.031628453489722
(0.424,0.286)	(0.089841798439412,0.712656422581195)	-1.031628453489690
(-0.089,-0.633)	(-0.089841433791874,-0.712656440898500)	-1.031628453488535

Function (ii)

Table 4.3.2. (a)

x_0	x^*	$f(x^*)$
(0,0)	(0.999997438981979, 0.999994875421996)	6.559462797894382e-012
(2,1)	(0.999997325751484, 0.999994648964635)	7.152253072945237e-012
(1,-1)	(1.000002057771020, 1.000004121363929)	4.237806081218173e-012

Table 4.3.2. (b)

x_0	x^*	$f(x^*)$
(0，0)	(0.999992175224127,0.999984335009511)	6.125114236575247e-011
(2，1)	(1.000007857200306,1.000015730032360)	6.175983917201136e-011
(1，-1)	(1.000007832119298,1.000015679790382)	6.136608808308800e-011

Function (iii)

Table 4.3.3. (a)

x_0	x^*	$f(x^*)$
(0.9,0.9)	(1.040758730815448, 1.040758730815448)	0.179774966369012
(0.087,0.05)	1.0e-007 * (0.011140898081130, 0.115493523260776)	-1.999999999999978
(0.898,0.91)	(1.040758692213578, 1.040758730557294)	0.179774966368982

Table 4.3.3. (b)

x_0	x^*	$f(x^*)$
(0.9,0.9)	1.0e-007 *(-0.188828543001469, -0.188828543001469)	-1.999999999999884
(0.087,0.05)	1.0e-008 *(0.980814674548121, 0.490561612735529)	-1.999999999999981
(0.898,0.91)	1.0e-007 *(0.141998555082016, 0.011710431701756)	-1.999999999999967

4.4. PartⅣ

The algorithm of the spectral conjugate gradient method is as following (see [16]),

Algorithm 4.3

Step 0. Given $x_0 \in R^n, 0 < \varepsilon << 1$. Compute $g_0 = \nabla f(x_0)$, let $d_0 = -g_0$ and $k = 1$.

Step 1. If $\|g_k\| \le \varepsilon$, stop. Output x_k as the approximate optimal solution.

Step 2. Compute α_k by

$$f(x_k+\beta^m d_k)\le f(x_k)+\sigma\beta^m g_k^T d_k .$$

Set $x_{k+1}=x_k+\alpha_k d_k$.

Step 3. Compute $\theta_k=\frac{s_k^T s_k}{s_k^T y_k}$ and $\beta_k=\frac{(\theta_k y_k-s_k)^T g_{k+1}}{s_k^T y_k}$. Define

$$d=-\theta_k g_{k+1}+\beta_k s_k .$$

Then compute d_{k+1} by

$$d_{k+1}=\begin{cases} d, & \text{if } \mathrm{d}^{\mathrm{T}} g_{k+1}\le -10^{-3}\|d\|_2\|g_{k+1}\|_2 \\ -\theta_k g_{k+1}, & \text{others} \end{cases} .$$

Step 4. Set $k=k+1$ and go to Step 1.

The calculation results are given in the tables below. The data in the table (a) and (b) is computed by the Algorithm 4.3 and Algorithm 2.2, respectively.

Function (i)

Table 4.4.1. (a)

x_0	x^*	$f(x^*)$
(0.1,0.8)	(0.089842583251361, 0.712656607352788)	-1.031628453488385
(0.424,0.286)	(0.089841833241950, 0.712656270695127)	-1.031628453489632
(-0.089,-0.633)	(-0.089841750284188, -0.712656542030075)	-1.031628453489414

Table 4.4.1.(b)

x_0	x^*	$f(x^*)$
(0.1,0.8)	(0.089842074004650,0.712656538129374)	-1.031628453489722
(0.424,0.286)	(0.089841798439412,0.712656422581195)	-1.031628453489690
(-0.089,-0.633)	(-0.089841433791874,-0.712656440898500)	-1.031628453488535

Function (ii)

Table 4.4.2. (a)

x_0	x^*	$f(x^*)$
(0,0)	(0.999999785507294, 0.999999573073940)	4.643119494828477e-014
(2,1)	(1.000003050819062, 1.000006113709804)	9.322047031107793e-012
(1,-1)	(0.999999268834521, 0.999998529052728)	5.420279651816089e-013

Table 4.4.2. (b)

x_0	x^*	$f(x^*)$
(0，0)	(0.999992175224127,0.999984335009511)	6.125114236575247e-011
(2，1)	(1.000007857200306,1.000015730032360)	6.175983917201136e-011
(1，-1)	(1.000007832119298,1.000015679790382)	6.136608808308800e-011

Function (iii)

Table 4.4.3. (a)

x_0	x^*	$f(x^*)$
(0.9,0.9)	(1.040758709325329, 1.040758709325329)	0.179774966368862
(0.087,0.05)	1.0e-007 * (0.002044808565636, -0.111757681749955)	-1.999999999999980
(0.898,0.91)	(1.040758709306231, 1.040758709326703)	0.179774966368862

Table 4.4.3. (b)

x_0	x^*	$f(x^*)$
(0.9,0.9)	1.0e-007 *(-0.188828543001469, -0.188828543001469)	-1.999999999999884
(0.087,0.05)	1.0e-008 *(0.980814674548121, 0.490561612735529)	-1.999999999999981
(0.898,0.91)	1.0e-007 *(0.141998555082016, 0.011710431701756)	-1.999999999999967

5. Conclusions

According to the numerical results of Section 4 we can see that, the numerical performance of the steepest descent method with the Wolfe type line search is excellent, as well as the steepest descent method with Armijo line search, the conjugate gradient methods and the spectral conjugate gradient method. So, the steepest descent method with the Wolfe type line search is promising. Hence, we can use this method to solve the optimization problems.

Next, we give some discussions about the application of the steepest descent method with the Wolfe type line search.

As we known, the nonlinear complementarity problem, $NCP(f)$ for short, can be transformed into unconstrained optimization problem [10]. The nonlinear complementarity problem is to find an $x \in R^n$, such that

$$x \geq 0, \quad f(x) \geq 0, \quad x^T f(x) = 0,$$

where $f(x)$ is continuously differentiable. We can use the Fischer-Burmeister function to transform the nonlinear complementarity problem to unconstrained optimization problem. The Fischer-Burmeister function, ϕ: $R^2 \to R$,

$$\phi(a,b) = \sqrt{a^2 + b^2} - a - b,$$

Then ϕ is continuously differentiable, except the point $(0,0)$. And

$$\phi(a,b) = 0 \Leftrightarrow a \geq 0, \quad b \geq 0, \quad ab = 0.$$

We denote $\psi : R^2 \to R$,

$$\psi(u) = \frac{1}{2}\phi(u)^2, \quad u \in R^2.$$

where,

$$\phi(u) = \|u\| - e^T u, \quad u = (a,b)^T \in R^2.$$

Denote the merit function

$$\theta(x) = \sum_{i=1}^{n} \theta_i(x), \quad \theta_i(x) = \psi(x_i, f_i(x)) = \frac{1}{2}\phi(x_i, f_i(x))^2, \quad i = 1,2,\cdots,n.$$

where $\theta : R^2 \to R$ is continuously differentiable. Then, we know that x is the solution of $\min_{x \in R^n} \theta(x)$ if and only if x is the solution of $NCP(f)$. So, the $NCP(f)$ is transformed into unconstrained optimization problem. The detailed algorithm and convergence property will be given in the future.

Acknowledgements

Supported by National Science Foundation of China (11101231), A project of Shandong province higher educational science and technology program (J10LA05).

6. References

[1] G. C. Bento, O. P. Ferreira, P. R. Oliveira, *Unconstrained steepest descent method for multicriteria optimization on Riemannian manifolds[J]*, Journal of Optimization Theory and Applications, 2012(154):88-107

[2] C. F. Ma, *Optimization methods and matlab programming[M]*, Beijing: Science press, 2010

[3] Y. X. Yuan, W. Y. Sun, *Optimization theory and method [M]*, Beijing: Science press, 1997

[4] Y. J. Wang, N. H. Xiu, *Nonlinear optimization theory and method [M]*, Beijing: Science press, 2012

[5] C. Y. Wang, Y. Y. Chen, S. Q. Du, *Further insight into the Shamanskii modification of Newton method [J]*, Applied Mathematics and Computation, 2006(180):46-52

[6] W. Wang, Y. J. Yang, L. S. Zhang, *Unification of Filled Function and Tunnelling Function in Global Optimization [J]*, Acta Mathematicae Applicatae Sinica, English Series, 2007(23):59-66

[7] F. Alvarez, A. Cabot, *Steepest descent with curvature dynamical system [J]*, Journal of Optimization Theory and Applications, 2004(120):247-273

[8] K. C. Kiwiel, K. Murty, *Convergence of the Steepest Descent Method for Minimizing Qussiconvex Functions [J]*, Journal of Optimization Theory and Applications, 1996(89):221-226

[9] R. Fletcher, *A limited memory steepest descent method [J]*, Mathematical Programming, 2012(135):413-436

[10] C. Y. Wu, G. Q. Chen, *A conjugate gradient algorithm for large scale nonlinear complementarity problems [J]*, Mathematics in Practice and Theory, 2012(42): 185-193

[11] Y. X. Yuan, Y. H. Dai, J. Y. Yuan, *Modified two-point stepsize gradient methods for unconstrained optimization [J]*, Computational Optimization and Applications, 2002(22): 103—109

[12] Y. X. Yuan, Y. H. Dai, *Alternate minimization gradient method [J]*, IMA J. Num. Anal., 2003(23):377-393

[13] G. H. Yu, L. T. Guan, W. F. Chen, *Spectral conjugate gradient methods with sufficient descent property for large-scale unconstrained optimization [J]*, Optimization Methods and Software, 2008(23):275-293

[14] Y. X. Yuan, *A new stepsize for the steepest descent method [J]*, Journal of Computational Mathematics, 2006(24):149-156

[15] Y. X. Yuan, *A short note on the Q-linear convergence of the steepest descent method [J]*, Mathematical Programming, 2010(123):339-343

[16] E. G. Birgin, J. M. Martinez, *A spectral conjugate method for unconstrained optimization [J]*, Appl Math Optim, 2001(43):117-128