Journal of Computational Science & Engineering 55 (2021) 1266-1270

www.asocse.org Article

A new smoothing BFGS algorithm for constrained optimization problem

Yanxue Yang

School of Mathematics and Statistics, Qingdao University, Qingdao 266071, China

Abstract: **In this paper, the general constrained optimization problem is considered, the process of transforming it into unconstrained optimization problem by penalty function is given. And the smoothing function is used to smooth it, a smoothing BFGS algorithm is given to solve it. Under general conditions, the global convergence analysis of the algorithm is given, and the related numerical examples also show the effectiveness of the algorithm.**

Keywords: Constrained optimization problem; penalty function; BFGS algorithm; global convergence

Received 15 May 2021, Revised 20 May 2021, Accepted 22 May 2021

* Corresponding Author: Yanxue Yang, 2019020127@qdu.edu.cn

1. Introduction

In this paper, we consider the following general constrained optimization problem[1]

$$\begin{aligned} &\min \quad f(x) \\ &s.t. \quad h_i(x)=0,\ i\in M=\{1,2,\cdots,l\}, \\ &\qquad \mathrm{g}_j(x)\ge 0,\ j\in N=\{1,2,\cdots,m\}. \end{aligned} \tag{1}$$

The feasible field of problem (1) is $B=\{x\in\mathbb{R}^n \mid h_i(x)=0(i\in M), g_j(x)\ge 0\ (j\in N)\}$. The research on the theory and algorithm of constrained optimization problem has been widely valued by scholars [1-10]. This paper mainly uses penalty function to transform problem (1), and uses smoothing function to smooth the transformed problem. The BFGS algorithm has the characteristic that the direction generated is the descending direction of the objective function when the second-order Hesse matrix is not positive definite, so we propose a new smoothing BFGS algorithm to solve the transformed problem.

The structure of this paper is as follows: in Section 2, the penalty function derivation and smoothing process of problem (1) are given; in Section 3, the smoothing BFGS algorithm is given to solve the transformed problem; in Section 4, some numerical examples and results are given; the relevant conclusions in Section 5.

2. Preliminaries

For problem (1), we construct the penalty function as[5]

$$H(x)=\sum_{i=1}^{l}h_i^2(x)+\sum_{j=1}^{m}Q^2(g_i(x)),$$

where $Q(t)$ satisfied

$$\begin{cases} Q(t)=0, & t\ge 0, \\ Q(t)<0, & t<0, \\ Q(t_2)>Q(t_1) & 0>t_2>t_1, \end{cases}$$

so the augmented objective function is obtained

$$P(x,\sigma)=f(x)+\sigma H(x),$$

and σ is a penalty factor, the corresponding optimization problem is as follows

$$\min \ P(x,\sigma_k),\ x\in\mathbb{R}^n, \tag{2}$$

where $\{\sigma_k\}\to+\infty$ is a positive sequence.

There are many types of $Q(t)$ satisfying the condition, we use $Q(t)=\min\{0,t\}$, then (4) can be transformed into

$$\min \ P(x,\sigma_k)=f(x)+\sigma_k\left(\sum_{i=1}^{l}h_i^2(x)+\sum_{j=1}^{m}\left[\min\{0,g_j(x)\}\right]^2\right),$$

i.e.,

$$\begin{aligned} \min \ P(x,\sigma_k)&=f(x)+\sigma_k\sum_{i=1}^{l}h_i^2(x) \\ &+\sigma_k\sum_{j=1}^{m}\left(\frac{g_j(x)-|g_j(x)|}{2}\right)^2, \end{aligned} \tag{3}$$

Next, we give the process of smooth (3). According to reference [11], we select

$$\psi_i(\mu,x_i)=\sqrt{4\mu^2+x_i^2},$$

to smooth $|g_j(x)|$ in (3), $0<\mu<1$ is the smooth factor, we can get

2021 © The American Computational Science Society. All rights reserved.

$$\min\ P_\mu(x,\sigma_k)=f(x)+\sigma_k\sum_{i=1}^{l}h_i^2(x)+\sigma_k\sum_{j=1}^{m}\left(\frac{g_j(x)-\sqrt{4\mu^2+g_j^2(x)}}{2}\right)^2. \quad (4)$$

Next, a new BFGS algorithm for solving problem (4) is given.

3. A new smoothing BFGS algorithm

Newton method can not guarantee that the direction generated is the descending direction of the objective function when the second-order Hesse matrix is not positive definite. Quasi Newton method overcomes this shortcoming, so it is widely used. Based on references [5,12], this part presents a new smoothing BFGS algorithm for solving problem (4). Denote d_k as the search direction and α_k denotes step length, $\tilde{g}(x)$ denotes the gradient of $P_\mu(x,\sigma_k)$. Denote

$$s_k=x_{k+1}-x_k, \quad (5)$$

$$\delta_k=\tilde{g}_{k+1}-\tilde{g}_k, \quad (6)$$

$$y_k=\delta_k+r_k s_k, \quad (7)$$

$$B_{k+1}=\begin{cases}B_k & y_k^T s_k\le 0,\\ B_k-\dfrac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}+\dfrac{y_k y_k^T}{y_k^T s_k} & y_k^T s_k>0.\end{cases} \quad (8)$$

Next we will give the smoothing BFGS algorithm.

Algorithm 3.1

Step 0 Given starting point $x_0\in\mathbb{R}^n$ and $0<\delta<1$, $\beta>0,\tau>0$ and take the initial symmetric positive definite matrix as unit matrix I_n, set $k:=0$.

Step 1 Compute $\tilde{g}_k=\nabla P_\mu(x_k,\sigma_k)$, if $\|\tilde{g}_k\|<\varepsilon$, stop, output $x^*=x_k$.

Step 2 Solve linear equations $B_k d=-\tilde{g}_k$, and get d_k.

Step 3 Determine step size $\alpha_k=\delta^m$, where m is the smallest non-negative integer satisfying the following inequality:

$$P_\mu(x_k+\delta^m d_k,\sigma_k)\le P_\mu(x_k,\sigma_k)+\beta\alpha_k\tilde{g}_k^T d_k. \quad (9)$$

Step 4 Set $x_{k+1}:=x_k+\alpha_k d_k$, $\eta_k=\dfrac{\delta_k^T s_k}{\|s_k\|^2}$, and $r_k\ge 0$ satisfying $\wp\le\eta_k+r_k\le r$, $\wp>0$, $r>0$, $\mu_{k+1}=\tau\mu_k$. B_{k+1} is determined by formula (8).

Step 5 Set $k=:k+1$, and return to Step 1.

Next, we prove the convergence of Algorithm 3.1. Firstly, we give the following general assumptions and lemmas.

Assumption

(i) For $P_\mu(x,\sigma_k)$, the level set

$$\mathrm{L}=\{x\in\mathbb{R}^n: P_\mu(x,\sigma_k)\le P_\mu(x_0,\sigma_k)\}$$

is bounded.

(ii) In the neighborhood U of L, $\tilde{g}(x)$ is Lipschitz continuous, i.e., there exists a $L>0$, s.t., $\|\tilde{g}(x)-\tilde{g}(y)\|\le L\|x-y\|,\ \forall x,y\in U.$

Lemma 3.1 Suppose that the sequence $\{x_k\}$ is generated by Algorithm 3.1, we get

$$\sum_{k=0}^{\infty}\left(-\tilde{g}_k^T s_k\right)<\infty, \quad (10)$$

$$\wp\|s_k\|^2\le y_k^T s_k\le r\|s_k\|^2,\ k=1,2,\cdots, \quad (11)$$

$$\|y_k\|\le(L+r)\|s_k\|,\ k=1,2,\cdots. \quad (12)$$

Proof From (9) and Assumption (i), we can get (10). And from (7), we have

$$y_k^T s_k=\delta_k^T s_k+r_k\|s_k\|^2, \quad (13)$$

and from Algorithm 3.1, we know that

$$\eta_k\|s_k\|^2=\delta_k^T s_k, \quad (14)$$

and from (13), (14), we can have the conclusion (11). From (6), (7) and Step 4 in Algorithm 3.1, we have $\|y_k\|=\|\delta_k+r_k s_k\|\le(L+r)\|s_k\|$ i.e., (12) is obtained.

Lemma 3.2 Suppose that the sequence $\{x_k\}$ is generated by Algorithm 3.1, then there exists a constant $M>0$,

$$trB_{k+1}\le M(k+1),$$

$$\sum_{i=0}^{k}\frac{\|B_i s_i\|^2}{s_i^T B_i s_i}\le M(k+1).$$

Proof

$$trB_{k+1}=tr\left(B_k-\frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}+\frac{y_k y_k^T}{y_k^T s_k}\right)$$

$$=trB_k-\frac{\|B_k s_k\|^2}{s_k^T B_k s_k}+\frac{\|y_k\|^2}{y_k^T s_k}$$

$$\le \quad \vdots$$

$$\le trB_0-\sum_{i=0}^{k}\frac{\|B_i s_i\|^2}{s_i^T B_i s_i}+\frac{(L+r)}{\wp}(k+1),$$

then we have

2021 © The American Computational Science Society. All rights reserved.

$$trB_{k+1}+\sum_{i=0}^{k}\frac{\|B_i s_i\|^2}{s_i^T B_i s_i}$$
$$\le trB_0+\frac{(L+r)}{\wp}(k+1)$$
$$=\left(\frac{n}{k+1}+\frac{(L+r)^2}{\wp}\right)(k+1),$$

denote $M=\frac{n}{k+1}+\frac{(L+r)^2}{\wp}$, we get this lemma.

Lemma 3.3 Suppose that the sequence $\{x_k\}$ is generated by Algorithm 3.1, then there exists a constant $c>0$, for sufficiently large k, there exists

$$\prod_{i=0}^{k}\alpha_k \ge c.$$

Proof From [12] and Lemma 7.6 in reference [13], we have

$$\det B_{k+1}=\det\left(B_k-\frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}+\frac{y_k y_k^T}{y_k^T s_k}\right)$$
$$=\det[B_k\left(I-\frac{s_k\left(B_k^T s_k\right)^T}{s_k^T B_k s_k}+B_k^{-1}y_k\frac{y_k^T}{y_k^T s_k}\right)]$$
$$=\det B_k\det\left(I-\frac{s_k\left(B_k^T s_k\right)^T}{s_k^T B_k s_k}+B_k^{-1}y_k\frac{y_k^T}{y_k^T s_k}\right)$$
$$=\det B_k[\left(1-s_k^T\frac{B_k^T s_k}{s_k^T B_k s_k}\right)\left(1+\left(B_k^{-1}y_k\right)^T\frac{y_k}{y_k^T s_k}\right)$$
$$+\left(s_k^T\frac{y_k}{y_k^T s_k}\right)\left(\frac{\left(B_k^T s_k\right)^T}{s_k^T B_k s_k}B_k^{-1}y_k\right)]$$
$$=\det B_k\frac{y_k^T s_k}{s_k^T B_k s_k}$$
$$\ge\det B_k\frac{\delta_k^T s_k}{s_k^T B_k s_k}$$
$$\ge\det B_k\frac{\eta_k\|s_k\|^2}{-\lambda g_k^T s_k},$$

and from Lemma 3.1, $\|s_k\|^2<\infty$, we know that for $k_1,k_2>0$, there exists

$$\det B_{k+1}\ge\det B_k\frac{Lk_1}{\lambda k_2}$$
$$\ge\ \vdots$$
$$\ge\prod_{i=0}^{k}\frac{1}{\alpha_k}\cdot\left(\frac{Lk_1}{k_2}\right)^{k+1}\det B_0,$$

from the mean value theorem, we have

$$\det B_{k+1}\le\left(\frac{trB_{k+1}}{n}\right)^n\le\left(\frac{M(k+1)}{n}\right)^n,$$

then

$$\prod_{i=0}^{k}\alpha_k\ge\frac{\left(\frac{Lk_1}{k_2}\right)^{k+1}\det B_0}{\left(\frac{M(k+1)}{n}\right)^n}=c,$$

the proof is complete. Next, the convergence of Algorithm 3.1 is proved.

Theorem 3.1 Suppose that the sequence $\{x_k\}$ is generated by Algorithm 3.1, then

$$\liminf_{k\to\infty}\|\tilde{g}_k\|=0.$$

Proof We will prove this theorem by contradiction. If the conclusion is not true, therefore, suppose $\|\tilde{g}_k\|\ge\gamma>0$, because of $B_k s_k=\alpha_k B_k d_k$ $=-\alpha_k\tilde{g}_k$, then $\alpha_k=\frac{\|B_k s_k\|}{\|\tilde{g}_k\|}$. From Lemma 3.1, we have

$$\sum_{k=0}^{\infty}\left(-\tilde{g}^T s_k\right)=\sum_{k=0}^{\infty}\frac{1}{\alpha_k}s_k^T B_k s_k$$
$$=\sum_{k=0}^{\infty}\|g_k\|^2\alpha_k\frac{s_k^T B_k s_k}{\|B_k s_k\|^2}$$
$$\ge\gamma^2\sum_{k=0}^{\infty}\alpha_k\frac{s_k^T B_k s_k}{\|B_k s_k\|},$$

then

$$\gamma^2\sum_{k=0}^{\infty}\alpha_k\frac{s_k^T B_k s_k}{\|B_k s_k\|}\le 0,$$

then from mean value theorem, for any $\xi>0$, there exists $k>0$, for all q, we have

$$q\left(\prod_{k=k_0+1}^{k_0+q}\alpha_k\frac{s_k^T B_k s_k}{\|B_k s_k\|^2}\right)^{\frac{1}{q}}\le\sum_{k=k_0+1}^{k_0+q}\alpha_k\frac{s_k^T B_k s_k}{\|B_k s_k\|^2}\le\xi,$$

then

2021 © The American Computational Science Society. All rights reserved.

$$\left(\prod_{k=k_0+1}^{k_0+q}\alpha_k\right)^{\frac{1}{q}} \le \frac{\xi}{q}\left(\prod_{k=k_0+1}^{k_0+q}\frac{\|B_k s_k\|^2}{s_k^T B_k s_k}\right)^{\frac{1}{q}}$$
$$\le \frac{\xi}{q^2}\sum_{k=k_0+1}^{k_0+q}\frac{\|B_k s_k\|^2}{s_k^T B_k s_k}$$
$$\le \frac{\xi}{q^2}\sum_{k=0}^{k_0+q}\frac{\|B_k s_k\|^2}{s_k^T B_k s_k}$$
$$\le \frac{\xi(k_0+q+1)M}{q^2},$$

which contradicts Lemma 3.3, therefore we get this theorem.

4. Numerical experiment

In order to verify the effectiveness of Algorithm 3.1, numerical experiments are given, the examples are from references [5,14]. All codes are run in Matlab (2016a). The parameters involved are set as follows:

σ_0=2, ε=10^{-3}, ρ=0.65, $\gamma=2$, β=0.45, μ_0=0.5, $\wp$=0.5, $r=10$.

The meanings of the parameters in the table are as follows: x_0 is the randomly selected initial point, x^* is the optimal solution, k is the number of iterations, $P_\mu(x,\sigma_k)$ is the minimum value of the transformed optimization problem.

Example 4.1 Consider constrained optimization problems:

$$\begin{aligned}\min\quad & f(x)=(x_1-2)^2+(x_2-3)^2\\ s.t.\quad & 2x_1-x_2-1=0,\\ & (x_1-2)^2-x_2\ge 0\end{aligned}$$

The initial points are selected randomly and tested by Algorithm 3.1. The results are shown in Table 1.

Example 4.2 Consider constrained optimization problems:

$$\begin{aligned}\min\quad & f(x)=1000-x_1^2-2x_2^2-x_3^2-x_1x_2-x_1x_3\\ s.t.\quad & x_1^2+x_2^2+x_3^2-25=0,\\ & (x_1-5)^2+x_2^2+x_3^2-25=0,\\ & -(x_1-5)^2-(x_2-5)^2-(x_3-5)^2+25\ge 0.\end{aligned}$$

The initial points are selected randomly and tested by Algorithm 3.1. The results are shown in Table 2.

Example 4.3 Consider constrained optimization problems:

$$\begin{aligned}\min\quad & f(x)=(x_1^2-2)^2+(x_2^2-1)^2\\ s.t\quad & x_1-2x_2+1=0,\\ & -\frac{1}{4}x_1^2-x_2^2+1\ge 0.\end{aligned}$$

The initial points are selected randomly and tested by Algorithm 3.1. The results are shown in Table 3.

Table 1 Numerical results of Example 4.1.

x_0	x^*	k	$P_\mu(x,\sigma_k)$
$(0.8175,0.7224)^t$	$(2.2132,3.7426)^t$	44	-6.8303
$(0.1499,0.6596)^t$	$(2.21324,3.7432)^t$	13	-6.8302
$(0.6280,0.2920)^t$	$(2.2134,3.7432)^t$	19	-6.8302

Table 2 Numerical results of Example 4.2.

x_0	x^*	k	$P_\mu(x,\sigma_k)$
$(0.6020,0.2630,0.6541)^t$	$(2.6008,4.2742,1.0233)^T$	87	944.2742
$(0.3786,0.8116,0.5328)^T$	$(2.6008,4.2741,1.0238)^t$	97	944.2316
$(0.4899,0.1679,0.9787)^t$	$(2.6008,4.2741,1.0238)^t$	115	944.2316

Table 3 Numerical results of Example 4.3.

x_0	x^*	k	$P_\mu(x,\sigma_k)$
$(0.1981,0.4897)^t$	$(1.1849,0.8645)^t$	5	1.4991
$(0.1062,0.3724)^t$	$(1.1851,0.8645)^t$	8	1.4991
$(0.9841,0.1672)^t$	$(1.1851,0.8645)^t$	7	1.4991

2021 © The American Computational Science Society. All rights reserved.

5. Conclusions

By using penalty function, we transform the general constrained optimization problem into unconstrained optimization problem. A new smoothing BFGS algorithm is proposed to solve the problem. The related numerical results show the effectiveness and efficiency of the given algorithm.

References

[1] J. Nocedal, S.T. Wright. Numerical optimization[M]. Berlin: Springer-Veriag, 2000.
[2] J. Kowalik, M.R. Osborne, D.M. Ryan. A new method for constrained optimization problems[J]. Operations Research, 17(1969): 973-983.
[3] J.W. Bandler, C. Charalambous. Nonlinear programming using minimax techniques[J]. Journal Of Optimization Theory And Applications, 13(1974): 607-619.
[4] Y..X. Yuan, W.Y. Sun. Optimization theory and methods[M]. Beijing: Science Press. 1997.
[5] C.F. Ma. Optimization method and Matlab program design[M]. Beijing: Science Press. 2010.
[6] S.W. Fu. A hybrid algorithm for linearly constrained minimax problems[J]. Annals Of Operations Research, 206(2013): 501-525.
[7] S.T. Yang. Penalty function smoothing methods for constrained optimization problems[D]. Dalian: Dalian University of Technology, 2018.
[8] S.J. Lian, J.H. Tang, A.H. Du. A new class of exact penalty functions for equality constrained smooth optimization[J]. Operations Research Transactions, 22(2018): 108-116.
[9] X.L. Geng, X.J. Tong. A review distributionally robust chance constrained optimization problems[J]. Operations Research Transactions, 24(2020): 115-130.
[10] Q.P. Ruan, Y.Q. Bai. A smoothing exact penalty function algorithm for nonlinear constrained optimization problems[J]. Communications on Applied Mathematics and Computation, 31(2017): 356-369.
[11] C.Y. Wu, J.M. Zhan, Y. Lu, J-S. Chen. Signal reconstruction by conjugate gradient algorithm based on smoothing l_1-norm[J]. Calcolo, 56(2019): 1-26.
[12] D.H. Li, M. A modified BFGS method and its global convergence in nonconvex minimization[J]. Fukushima. Journal Of Computational And Applied Mathematics, 129(2011): 15-35.
[13] J.E. Dennis, A. Moré. Quasi-Newton methods, motivation and theory[J]. SIAM Review, 19(1977): 46-89.
[14] Y. Zheng, Z.Q. Meng, R. Shen. An m-objective penalty function algorithm under big penalty parameters[J]. Journal of systems science and complexity, 29(2016): 455-471.

2021 © The American Computational Science Society. All rights reserved.