Journal of Applied Mathematics & Bioinformatics, vol.5, no.2, 2015, 1-19
ISSN: 1792-6602 (print), 1792-6939 (online)
Scienpress Ltd, 2015

Smoothing approximation to the k-th power nonlinear penalty function for constrained optimization problems

Nguyen Thanh Binh[1] **and Wenli Yan**[2]

Abstract

In this paper, a new smoothing approximation to the k-th power nonlinear penalty function for constrained optimization problems is presented. We prove that this type of the smoothing penalty functions has good properties in solving constrained optimization problems. Furthermore, based on the smoothed penalty problem, an algorithm is presented to solve the constrained optimization problems, with its convergence under some conditions proved. Some numerical examples are given to illustrate the applicability of the present smoothing method, which show that the algorithm seems efficient.

Mathematics Subject Classification: 90C30; 65K05

[1] Department of Mathematics, Shanghai University, Shanghai, 200444, China.
E-mail: bingbongyb@gmail.com
[2] Department of Mathematics, Shanghai University, Shanghai, 200444, China.
E-mail: wenliyy@163.com

This work is supported by National Natural Science Foundation of China (Grant No.11371242).

Article Info: *Received* : February 17, 2015. *Revised* : April 14, 2015.
Published online : June 20, 2015.

Keywords: Constrained optimization; Nonlinear penalty function; Smoothing method; Optimal solution; ϵ-feasible solution

1 Introduction

We consider the following constrained optimization problem:

$$\text{(P)} \qquad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_i(x) \leq 0, \quad i = 1, 2, \ldots, m, \\ & x \in R^n, \end{array}$$

where $f, g_i : R^n \to R, \quad i \in I = \{1, 2, \ldots, m\}$ are continuously differentiable functions and $F_0 = \{x \in R^n \mid g_i(x) \leq 0, \ i = 1, 2, \ldots, m\}$ is the feasible set to (P). The penalty function methods based on various penalty functions have been proposed to solve problem (P) in literatures. The classic l_1 exact penalty function [18] for problem (P) is first proposed by Zangwill as follows:

$$F_1(x, \rho) = f(x) + \rho \sum_{i=1}^{m} \max\{g_i(x), 0\}, \tag{1}$$

where $\rho > 0$ is a penalty parameter.

In many studies, one of the popular penalty functions is the twice penalty function, which has the following form:

$$F_2(x, \rho) = f(x) + \rho \sum_{i=1}^{m} \max\{g_i(x), 0\}^2, \tag{2}$$

where $\rho > 0$ is a penalty parameter. It is called an l_2 penalty function. This penalty function is smooth, it is not necessarily exact penalty function. Recently, Meng et al. [8] and Wu et al. [13] discussed a lower-order penalty function of the following form:

$$F^k(x, \rho) = f(x) + \rho \sum_{i=1}^{m} [\max\{g_i(x), 0\}]^k, \tag{3}$$

where $k \in (0, 1)$, which is not smooth either. Huang and Yang et al. [15, 16, 17] and Rubinov et al. [11] discussed a nonlinear Lagrangian penalty function,

$$F_k(x, \rho) = \left[f(x)^k + \rho \sum_{i=1}^{m} \left(\max\{g_i(x), 0\}\right)^k \right]^{\frac{1}{k}}, \tag{4}$$

for some $k \in (0, +\infty)$, which is called a k-th power penalty function in [15, 17]. A promising feature of the k-th power nonlinear penalty function is that a smaller exact penalty parameter than that of the classical penalty function (i.e., $k = 1$) can be guaranted when k is sufficiently small. Obviously, when $k = 1$, the k-th power penalty function is reduced to the classical l_1 exact penalty function. This penalty function is smooth for $k > 1$ while it is not smooth for $0 < k \leq 1$. Thus the minimization of the k-th power nonlinear penalty function is not an easy job. However, smoothing methods have been investigated for minimizing nonsmooth penalty function in e.g., [6, 7, 8, 9, 13, 14, 15, 17]. Lian [6] and Wu et al. [14] proposed a smoothing approximation to the classical l_1 exact penalty function and an ϵ-optimal minimum can be obtained by solving the smoothed penalty problem. Meng et al. [8] introduced a smoothing method of lower order penalty function and gave a robust SQP method for nonlinear programming problem by integrating the smoothed penalty function with the SQP method. Wu et al. [13] considered the ϵ-smoothing of lower order penalty function and got a modified exact penalty function under some mild assumptions. Yang et al. [15, 17] developed some smoothing approximations to k-th power penalty function. Pinar et al. [9] proposed a smoothing method of penalty functions for solving convex network optimization problems. Error estimates of the optimal value of the original penalty function and that of the smoothed penalty function are obtained.

In this paper, we first construct a new smoothing function $p_\epsilon^k(t)$ as follows:

$$p_\epsilon^k(t) = \begin{cases} 0 & \text{if} \quad t \leq 0, \\ \left(1 + \dfrac{1}{k\epsilon^{k-1}}\right) \dfrac{t^{2k}}{2\epsilon^k} & \text{if} \quad 0 \leq t \leq \epsilon, \\ t^k + \epsilon \ln t - \dfrac{1}{2}\left(1 - \dfrac{1}{k\epsilon^{k-1}} + \dfrac{2\ln\epsilon}{\epsilon^{k-1}}\right)\epsilon^k & \text{if} \quad t \geq \epsilon, \end{cases}$$

where $0 < k < +\infty$ and $\epsilon > 0$. It is easy to prove that $p_\epsilon^k(t)$ is C^1 at any $t \in R^1$ for $k > \dfrac{1}{2}$ and $\epsilon > 0$. Using $p_\epsilon^k(t)$ as the smoothing function, a new nonlinear penalty function is obtained, based on which an algorithm for solving (P) is proposed herein.

The rest of this article is organized as follows. In Section 2, we introduce a smoothing approximation to the k-th power nonlinear penalty function and the smoothing technique. In Section 3, the algorithm based on the smoothed nonlinear penalty problem is proposed and the convergence of the algorithm

is proved. In Section 4, we give some numerical examples and compare the efficiency of the proposed method with other methods. Finally, conclusions are given in Section 5.

2 Smoothing nonlinear penalty function

Let $p^k(t) : R^1 \to R^1$:

$$p^k(t) = \begin{cases} 0 & \text{if} \quad t \leq 0, \\ t^k & \text{if} \quad t \geq 0, \end{cases} \tag{5}$$

where $0 < k < +\infty$. Obviously, $p^k(t)$ is not C^1 on R^1 for $0 < k \leq 1$, but it is C^1 for $k > 1$. The function $p^k(t)$ is useful in defining exact penalty function for nonlinear programming, see, [1, 9, 17]. In order to smooth the function $p^k(t)$, we define function $p_\epsilon^k(t) : R^1 \to R^1$ as

$$p_\epsilon^k(t) = \begin{cases} 0 & \text{if} \quad t \leq 0, \\ \left(1 + \dfrac{1}{k\epsilon^{k-1}}\right) \dfrac{t^{2k}}{2\epsilon^k} & \text{if} \quad 0 \leq t \leq \epsilon, \\ t^k + \epsilon \ln t - \dfrac{1}{2}\left(1 - \dfrac{1}{k\epsilon^{k-1}} + \dfrac{2\ln\epsilon}{\epsilon^{k-1}}\right)\epsilon^k & \text{if} \quad t \geq \epsilon, \end{cases} \tag{6}$$

where $0 < k < +\infty$ and $\epsilon > 0$.

Lemma 2.1. *For any $\epsilon > 0,\ 0 < k < +\infty$, we have $\lim\limits_{\epsilon \to 0} p_\epsilon^k(t) = p^k(t)$.*

Proof. For any $\epsilon > 0,\ 0 < k < +\infty$, by the definition of $p^k(t)$ and $p_\epsilon^k(t)$, we have

$$p^k(t) - p_\epsilon^k(t) = \begin{cases} 0 & \text{if} \quad t \leq 0, \\ t^k - \left(1 + \dfrac{1}{k\epsilon^{k-1}}\right) \dfrac{t^{2k}}{2\epsilon^k} & \text{if} \quad 0 \leq t \leq \epsilon, \\ \dfrac{1}{2}\left(1 - \dfrac{1}{k\epsilon^{k-1}} + \dfrac{2\ln\epsilon}{\epsilon^{k-1}}\right)\epsilon^k - \epsilon \ln t & \text{if} \quad t \geq \epsilon. \end{cases}$$

When $0 \leq t \leq \epsilon$, we obtain

$$t^k - \left(1 + \frac{1}{k\epsilon^{k-1}}\right) \frac{t^{2k}}{2\epsilon^k} \leq t^k \leq \epsilon^k.$$

On the other hand, when $t \geq \epsilon$, we have

$$\begin{aligned}
&\frac{1}{2}\left(1 - \frac{1}{k\epsilon^{k-1}} + \frac{2\ln\epsilon}{\epsilon^{k-1}}\right)\epsilon^k - \epsilon \ln t \\
&\leq \frac{1}{2}\left(1 - \frac{1}{k\epsilon^{k-1}} + \frac{2\ln\epsilon}{\epsilon^{k-1}}\right)\epsilon^k - \epsilon \ln \epsilon \\
&= \frac{1}{2}\epsilon^k - \frac{1}{2k}\epsilon \leq \frac{1}{2}\epsilon^k.
\end{aligned}$$

So, we have

$$0 \leq p^k(t) - p_\epsilon^k(t) \leq \frac{1}{2}\epsilon^k.$$

That is

$$\lim_{\epsilon \to 0} p_\epsilon^k(t) = p^k(t).$$

This completes the proof. □

Lemma 2.2. *Let* $\frac{1}{2} < k < +\infty$ *and* $\epsilon > 0$. *Then* $p_\epsilon^k(t)$ *is* C^1.

Proof. Let $p_1(t) = 0$ if $t \leq 0$, $p_2(t) = \left(1 + \frac{1}{k\epsilon^{k-1}}\right)\frac{t^{2k}}{2\epsilon^k}$ if $0 \leq t \leq \epsilon$ and $p_3(t) = t^k + \epsilon \ln t - \frac{1}{2}\left(1 - \frac{1}{k\epsilon^{k-1}} + \frac{2\ln\epsilon}{\epsilon^{k-1}}\right)\epsilon^k$ if $t \geq \epsilon$.
We have

$$p_\epsilon^k(t) = \begin{cases} p_1(t) & \text{if} \quad t \leq 0, \\ p_2(t) & \text{if} \quad 0 \leq t \leq \epsilon, \\ p_3(t) & \text{if} \quad t \geq \epsilon. \end{cases}$$

Then, for $\frac{1}{2} < k < +\infty$ we have

$$\nabla p_\epsilon^k(t) = \begin{cases} \nabla p_1(t) = 0 & \text{if} \quad t \leq 0, \\ \nabla p_2(t) = \dfrac{k}{\epsilon^k}\left(1 + \dfrac{1}{k\epsilon^{k-1}}\right)t^{2k-1} & \text{if} \quad 0 \leq t \leq \epsilon, \\ \nabla p_3(t) = kt^{k-1} + \dfrac{\epsilon}{t} & \text{if} \quad t \geq \epsilon. \end{cases} \tag{7}$$

In particular, $\nabla p_1(0) = 0 = \nabla p_2(0)$ and $\nabla p_2(\epsilon) = k\epsilon^{k-1} + 1 = \nabla p_3(\epsilon)$. Therefore, $p_\epsilon^k(t)$ is C^1 at any $t \in R^1$ by (7). This completes the proof. □

Remark 1. If $0 < k < \frac{1}{2}$, $p_\epsilon^k(t)$ is differentiable when $t \neq 0$.

Suppose that f and g_i, $i \in I$ are C^1, by Lemma 2.2, $p_\epsilon^k(t)$ is C^1 for $\frac{1}{2} < k < +\infty$. In this paper, we always assume that $f(x) \geq 0$ $(x \in R^n)$.

Consider the following optimization problem:

$$(\mathrm{P}): \quad \min \ f(x) \qquad \text{s.t. } x \in R^n,$$

and the nonlinear penalty functions for (P):

$$F(x,\rho) = f^k(x) + \rho \sum_{i=1}^{m} p^k(g_i(x)),$$
$$F_\epsilon(x,\rho) = f^k(x) + \rho \sum_{i=1}^{m} p_\epsilon^k(g_i(x)),$$

where $\rho > 0$ and $0 < k < +\infty$. Hence, the following two penalty problems can be denoted as:

$$(P_\rho): \quad \min \ F(x,\rho) \qquad \text{s.t. } x \in R^n,$$
$$(NP_{\rho,\epsilon}): \quad \min \ F_\epsilon(x,\rho) \qquad \text{s.t. } x \in R^n.$$

Now, the relationship between (P_ρ) and $(NP_{\rho,\epsilon})$ is studied.

Lemma 2.3. *For any given $x \in R^n$, $\epsilon > 0$ and $\rho > 0$, we have*

$$\lim_{\epsilon \to 0} F_\epsilon(x,\rho) = F(x,\rho).$$

Proof. For any $x \in R^n$, by the definition of $p^k(t)$ and $p_\epsilon^k(t)$, we have

$$p^k(g_i(x)) - p_\epsilon^k(g_i(x)) = \begin{cases} 0 & \text{if } g_i(x) \le 0, \\ g_i(x)^k - \left(1 + \dfrac{1}{k\epsilon^{k-1}}\right) \dfrac{g_i(x)^{2k}}{2\epsilon^k} & \text{if } 0 \le g_i(x) \le \epsilon, \\ \dfrac{1}{2}\left(1 - \dfrac{1}{k\epsilon^{k-1}} + \dfrac{2\ln\epsilon}{\epsilon^{k-1}}\right)\epsilon^k - \epsilon \ln(g_i(x)) & \text{if } g_i(x) \ge \epsilon. \end{cases}$$

That is

$$0 \le p^k(g_i(x)) - p_\epsilon^k(g_i(x)) \le \frac{1}{2}\epsilon^k, \qquad i = 1, 2, \ldots, m.$$

Adding up for all i, we obtain

$$0 \le \sum_{i=1}^{m} p^k(g_i(x)) - \sum_{i=1}^{m} p_\epsilon^k(g_i(x)) \le \frac{1}{2} m\epsilon^k.$$

Therefore,

$$0 \le F(x,\rho) - F_\epsilon(x,\rho) \le \frac{1}{2} m\rho\epsilon^k,$$

which implies

$$F(x,\rho) - \frac{1}{2}m\rho\epsilon^k \leq F_\epsilon(x,\rho) \leq F(x,\rho).$$

That is

$$\lim_{\epsilon\to 0} F_\epsilon(x,\rho) = F(x,\rho).$$

This completes the proof. □

A direct result of Lemma 2.3 is given as follows.

Corollary 2.4. *Let $\{\epsilon_j\} \to 0$ be a sequence of positive numbers and assume x^j is a solution to $(NP_{\rho,\epsilon})$ for some given $\rho > 0$. Let $\overline{x}$ be an accumulation point of the sequence $\{x^j\}$. Then $\overline{x}$ is an optimal solution to (P_ρ).*

Theorem 2.5. *Let x^* be an optimal solution of (P_ρ) and $\overline{x} \in R^n$ an optimal solution of $(NP_{\rho,\epsilon})$ for some $\rho > 0$ and $\epsilon > 0$. Then,*

$$\lim_{\epsilon\to 0} F_\epsilon(\overline{x},\rho) = F(x^*,\rho).$$

Proof. From Lemma 2.3, for $\rho > 0$, we have that

$$0 \leq F(x^*,\rho) - F_\epsilon(x^*,\rho) \leq \frac{1}{2}m\rho\epsilon^k,$$
$$0 \leq F(\overline{x},\rho) - F_\epsilon(\overline{x},\rho) \leq \frac{1}{2}m\rho\epsilon^k, \quad 0 < k < +\infty.$$

From the assumption that x^* and $\overline{x}$ are optimal solution of (P_ρ) and $(NP_{\rho,\epsilon})$, respectively, we get

$$F_\epsilon(\overline{x},\rho) \leq F_\epsilon(x^*,\rho),$$
$$F(x^*,\rho) \leq F(\overline{x},\rho).$$

Therefore, we obtain that

$$\begin{aligned} 0 \leq F(x^*,\rho) - F_\epsilon(x^*,\rho) &\leq F(x^*,\rho) - F_\epsilon(\overline{x},\rho) \\ &\leq F(\overline{x},\rho) - F_\epsilon(\overline{x},\rho) \leq \frac{1}{2}m\rho\epsilon^k. \end{aligned}$$

It follows

$$F(x^*,\rho) - \frac{1}{2}m\rho\epsilon^k \leq F_\epsilon(\overline{x},\rho) \leq F(x^*,\rho).$$

That is

$$\lim_{\epsilon\to 0} F_\epsilon(\overline{x},\rho) = F(x^*,\rho).$$

This completes the proof. □

Theorem 2.5 show that an approximate solution to $(NP_{\rho,\epsilon})$ is also an approximate solution to (P_ρ) when the error ϵ is sufficiently small.

Definition 2.6. *For $\epsilon > 0$, a point $x_\epsilon \in R^n$ is an ϵ-feasible solution or an ϵ-solution of problem (P), if*

$$g_i(x_\epsilon) \leq \epsilon, \qquad i = 1, 2, \ldots, m.$$

Under this definition, we get the following result.

Theorem 2.7. *Let x^* be an optimal solution of (P_ρ) and $\overline{x} \in R^n$ an optimal solution of $(NP_{\rho,\epsilon})$ for some $\rho > 0$ and $\epsilon > 0$. Furthermore, let x^* be feasible to (P) and $\overline{x}$ be ϵ-feasible to (P). Then,*

$$0 \leq f(x^*)^k - f(\overline{x})^k \leq \left(1 + \frac{1}{2k\epsilon^{k-1}}\right) m\rho\epsilon^k, \quad 0 < k < +\infty. \tag{8}$$

Proof. Since $\overline{x}$ is ϵ-feasible to (P), hence

$$\sum_{i\in I} p_\epsilon^k(g_i(\overline{x})) \leq \left(\frac{1}{2} + \frac{1}{2k\epsilon^{k-1}}\right) m\epsilon^k, \quad 0 < k < +\infty.$$

Because x^* is an optimal solution to (P), we have

$$\sum_{i\in I} p^k(g_i(x^*)) = 0.$$

Then, by Theorem 2.5, we have

$$\begin{aligned} 0 \leq F(x^*, \rho) - F_\epsilon(\overline{x}, \rho) = &\left\{ f(x^*)^k + \rho \sum_{i\in I} p^k(g_i(x^*)) \right\} \\ &- \left\{ f(\overline{x})^k + \rho \sum_{i\in I} p_\epsilon^k(g_i(\overline{x})) \right\} \leq \frac{1}{2} m\rho\epsilon^k. \end{aligned}$$

Thus,

$$\rho \sum_{i\in I} p_\epsilon^k(g_i(\overline{x})) \leq f(x^*)^k - f(\overline{x})^k \leq \rho \sum_{i\in I} p_\epsilon^k(g_i(\overline{x})) + \frac{1}{2} m\rho\epsilon^k.$$

Therefore,

$$0 \leq f(x^*)^k - f(\overline{x})^k \leq \left(1 + \frac{1}{2k\epsilon^{k-1}}\right) m\rho\epsilon^k.$$

This completes the proof. □

By Theorem 2.7, an approximate optimal solution to $(NP_{\rho,\epsilon})$ becomes an approximate optimal solution to (P) if the solution to $(NP_{\rho,\epsilon})$ is ϵ-feasible to (P). Therefore, we can obtain an approximate optimal solution to (P) by solving $(NP_{\rho,\epsilon})$ under some mild conditions.

3 A smoothing nonlinear penalty function algorithm

In this section, we give a nonlinear penalty function algorithm for the problem (P). In order to solve (P), we attempt to solve its smoothed penalty problem given by $\min\limits_{x\in R^n} \ F_\epsilon(x,\rho)$.

Algorithm 3.1

Step 1: Given $x^0,\ \epsilon>0,\ \epsilon_0>0,\ \rho_0>0,\ 0<\eta<1,\ k>0$ and $N>1$, let $j=0$ and go to Step 2.

Step 2: Use x^j as the starting point to solve

$$(NP_{\rho_j,\epsilon_j}) \qquad \min_{x\in R^n} \ F_{\epsilon_j}(x,\rho_j)=f(x)^k+\rho_j\sum_{i=1}^{m}p_{\epsilon_j}^k(g_i(x)).$$

Let x^{j+1} be the optimal solution obtained (x^{j+1} is obtained by a quasi -Newton method).

Step 3: If x^{j+1} is ϵ-feasible to (P), then stop and we have obtained an approximate solution x^{j+1} of (P). Otherwise, let $\rho_{j+1}=N\rho_j$, $\epsilon_{j+1}=\eta\epsilon_j$ and $j=j+1$, then go to Step 2.

Remark 2. By Theorem 2.7 and Step 3 of Algorithm 3.1, x^{j+1} is an approximate optimal solution to (P).

In practice, it is difficult to compute $x^{j+1}\in arg\min\limits_{x\in R^n}F_{\epsilon_j}(x,\rho_j)$. We generally look for the local minimizer or stationary point of $F_{\epsilon_j}(x,\rho_j)$ by computing x^{j+1} such that $\nabla F_{\epsilon_j}(x,\rho_j)=0$.

For $x \in R^n, \epsilon > 0$, let us define

$$\begin{aligned} I^0(x) &= \{i \mid g_i(x) = 0, \quad i \in I\}, \\ I^-(x) &= \{i \mid g_i(x) < 0, \quad i \in I\}, \\ I^-_\epsilon(x) &= \{i \mid 0 \le g_i(x) < \epsilon, \quad i \in I\}, \\ I^+_\epsilon(x) &= \{i \mid g_i(x) \ge \epsilon, \quad i \in I\}. \end{aligned}$$

The convergence of the Algorithm 3.1 is proved in the following theorem.

Theorem 3.1. *Let $k > \dfrac{1}{2}$. Assume that $\lim\limits_{\|x\| \to +\infty,\ x \in R^n} f(x) = +\infty$. Let $\{x^j\}$ be the sequence generated by Algorithm 3.1. Suppose that the sequence $\{F_{\epsilon_j}(x^j, \rho_j)\}$ is bounded. Then, $\{x^j\}$ is bounded and any limit point x^* of $\{x^j\}$ is feasible to (P), and there exists $\lambda \ge 0$ and $\mu_i \ge 0, \quad i = 1, 2, \ldots, m$, such that*

$$\lambda \nabla f(x^*) + \sum_{i \in I^0(x^*)} \mu_i \nabla g_i(x^*) = 0. \tag{9}$$

Proof. First, we will prove that $\{x^j\}$ is bounded. Note that

$$F_{\epsilon_j}(x^j, \rho_j) = f(x^j)^k + \rho_j \sum_{i=1}^{m} p^k_{\epsilon_j}\left(g_i(x^j)\right), \qquad j = 0, 1, 2, \ldots, \tag{10}$$

From (6), we have

$$\sum_{i=1}^{m} p^k_{\epsilon_j}\left(g_i(x^j)\right) \ge 0. \tag{11}$$

Suppose to the contrary that $\{x^j\}$ is unbounded. Without loss of generality, we assume that $\|x^j\| \to +\infty$ as $j \to +\infty$. Then $\lim\limits_{j \to +\infty} f^k(x^j) = +\infty$ for $k > \dfrac{1}{2}$ and from (10) and (11), we have

$$F_{\epsilon_j}(x^j, \rho_j) \ge f^k(x^j) \to +\infty, \quad \rho_j > 0,\ j = 0, 1, 2, \ldots,$$

which results in a contradiction since the sequence $\{F_{\epsilon_j}(x^j, \rho_j)\}$ is bounded. Thus $\{x^j\}$ is bounded.

We now show that any limit point of $\{x^j\}$ belongs to F_0. Without loss of generality, we assume $\lim\limits_{j \to +\infty} x^j = x^*$. Suppose to the contrary that $x^* \notin F_0$, then there exits some $i \in I$ such that $g_i(x^*) > \alpha > 0$. As g_i $(i \in I)$ are

continuous, so $F_{\epsilon_j}(x^j, \rho_j)$ $(j = 1, 2, \ldots)$ are continuous. Note that

$$F_{\epsilon_j}(x^j, \rho_j) = f(x^j)^k + \rho_j \sum_{i \in I^-_{\epsilon_j}(x^j)} \left(1 + \frac{1}{k\epsilon_j^{k-1}}\right) \frac{g_i(x^j)^{2k}}{2\epsilon_j^k} \tag{12}$$
$$+ \rho_j \sum_{i \in I^+_{\epsilon_j}(x^j)} \left(g_i(x^j)^k + \epsilon_j \ln(g_i(x^j)) - \frac{1}{2}\left(1 - \frac{1}{k\epsilon_j^{k-1}} + \frac{2\ln\epsilon_j}{\epsilon_j^{k-1}}\right)\epsilon_j^k\right).$$

Then, $\rho_j \to +\infty$ and $\epsilon_j \to 0$ as $j \to +\infty$, $F_{\epsilon_j}(x^j, \rho_j) \to +\infty$, which contradicts the assumption that $\{F_{\epsilon_j}(x^j, \rho_j)\}$ is bounded. Therefore, x^* is feasible to (P).

Finally, we show that (9) holds. By Lemma 2.3 and Step 2 in Algorithm 3.1, we have $\nabla F_{\epsilon_j}(x^j, \rho_j) = 0$, that is

$$kf(x^j)^{k-1}\nabla f(x^j) + \rho_j \sum_{i \in I^+_{\epsilon_j}(x^j)} \left(kg_i(x^j)^{k-1} + \frac{\epsilon_j}{g_i(x^j)}\right) \nabla g_i(x^j)$$
$$+ \rho_j \sum_{i \in I^-_{\epsilon_j}(x^j)} \frac{k}{\epsilon_j^k}\left(1 + \frac{1}{k\epsilon_j^{k-1}}\right) g_i(x^j)^{2k-1}\nabla g_i(x^j) = 0. \tag{13}$$

For $j = 1, 2, \ldots,$ let

$$\gamma_j = kf(x^j)^{k-1} + \sum_{i \in I^+_{\epsilon_j}(x^j)} \rho_j \left(kg_i(x^j)^{k-1} + \frac{\epsilon_j}{g_i(x^j)}\right)$$
$$+ \sum_{i \in I^-_{\epsilon_j}(x^j)} \frac{\rho_j k}{\epsilon_j^k}\left(1 + \frac{1}{k\epsilon_j^{k-1}}\right) g_i(x^j)^{2k-1}.$$

Then $\gamma_j > 0,\ j = 1, 2, \ldots.$ From (13), we have

$$\frac{kf(x^j)^{k-1}}{\gamma_j}\nabla f(x^j) + \sum_{i \in I^+_{\epsilon_j}(x^j)} \frac{\rho_j\left(kg_i(x^j)^{k-1} + \frac{\epsilon_j}{g_i(x^j)}\right)}{\gamma_j}\nabla g_i(x^j)$$
$$+ \sum_{i \in I^-_{\epsilon_j}(x^j)} \frac{\frac{\rho_j k}{\epsilon_j^k}\left(1 + \frac{1}{k\epsilon_j^{k-1}}\right) g_i(x^j)^{2k-1}}{\gamma_j}\nabla g_i(x^j) = 0. \tag{14}$$

Let

$$\begin{aligned}
\lambda^j &= \frac{kf(x^j)^{k-1}}{\gamma_j},\\
\mu_i^j &= \frac{\rho_j\left(kg_i(x^j)^{k-1}+\dfrac{\epsilon_j}{g_i(x^j)}\right)}{\gamma_j}, \quad i\in I_{\epsilon_j}^+(x^j),\\
\mu_i^j &= \frac{\dfrac{\rho_j k}{\epsilon_j^k}\left(1+\dfrac{1}{k\epsilon_j^{k-1}}\right)g_i(x^j)^{2k-1}}{\gamma_j}, \quad i\in I_{\epsilon_j}^-(x^j),\\
\mu_i^j &= 0, \quad i\in I\setminus\left(I_{\epsilon_j}^+(x^j)\cup I_{\epsilon_j}^-(x^j)\right).
\end{aligned}$$

Then we have

$$\begin{aligned}
\lambda^j+\sum_{i\in I}\mu_i^j &= 1, \quad \forall j, \qquad (15)\\
\mu_i^j &\geq 0, \quad i\in I, \quad \forall j.
\end{aligned}$$

Obviously, we can assume without loss of generality that $\lambda^j\to\lambda\geq 0$, $\mu_i^j\to\mu_i\geq 0$, $\forall i\in I$. By (14) and (15), as $j\to+\infty$, we have

$$\begin{aligned}
\lambda\nabla f(x^*)+\sum_{i\in I}\mu_i\nabla g_i(x^*) &= 0,\\
\lambda+\sum_{i\in I}\mu_i &= 1.
\end{aligned}$$

For $i\in I^-(x^*)$, as $j\to\infty$, we get $\mu_i^j\to 0$. Therefore, $\mu_i=0$, $\forall i\in I^-(x^*)$, so, (9) holds, and this completes the proof. □

The speed of convergence of Algorithm 3.1 depends on the speed of convergence of the algorithm employed in Step 2 to solve the unconstrained optimization problem $\min\limits_{x\in R^n} F_{\epsilon_j}(x,\rho_j)$.

4 Numerical examples

In this section, we solve some constrained optimization problems with Algorithm 3.1 on MATLAB. In each of the following examples, the MATLAB 7.12 subroutine *fmincon* is used to obtain the local minima of problem (NP_{ρ_j,ϵ_j}).

Table 1: Results for Example 4.1 with $k = 2/3,\ x^0 = (-1, 1),\ \rho_0 = 1,\ N = 3$

j	ρ_j	ϵ_j	$f(x^j)$	$g_1(x^j)$	$g_2(x^j)$	x^j
1	1	0.01	3.517837	8.117813	-2.498812	(-1.078642,1.095343)
2	3	0.0005	2.859830	1.084860	-3.381246	(0.417153,1.067453)
3	9	0.000025	1.837548	-0.775885	-0.000000	(0.725360,0.399259)

The numerical results of each example are presented in the following tables. It is shown that Algorithm 3.1 yield some approximate solutions that have a better objective function value in comparison with some other algorithms.

Example 4.1. *Consider the example in [6],*

$$\begin{aligned} \textit{(P4.1)} \quad \min \quad & f(x) = x_1^2 + x_2^2 - \cos(17x_1) - \cos(17x_2) + 3 \\ \textit{s.t.} \quad & g_1(x) = (x_1 - 2)^2 + x_2^2 - 1.6^2 \le 0, \\ & g_2(x) = x_1^2 + (x_2 - 3)^2 - 2.7^2 \le 0, \\ & 0 \le x_1 \le 2,\ 0 \le x_2 \le 2. \end{aligned}$$

Let $k = 2/3,\ x^0 = (-1, 1),\ \epsilon_0 = 0.01,\ \eta = 0.05,\ \rho_0 = 1,\ N = 3$ and $\epsilon = 10^{-6}$. Numerical results of Algorithm 3.1 for solving (P4.1) are given in Table 1.

Therefore, we get an approximate solution

$$x^3 = (0.725360, 0.399259)$$

at the 3'th iteration. The objective function value is given by $f(x^3) = 1.837548$. One can easily check that x^3 is a feasible solution since the constraints of (P4.1) at x^3 are as follows:

$$\begin{aligned} g_1(x^3) &= (0.725360 - 2)^2 + 0.399259^2 - 1.6^2 = -0.775885121319001, \\ g_2(x^3) &= 0.725360^2 + (0.399259 - 3)^2 - 2.7^2 = -0.000000878680999, \\ 0 &\le x_1 = 0.725360 \le 2,\ 0 \le x_2 = 0.399259 \le 2. \end{aligned}$$

The solution we obtained is slightly better than the solution obtained in the 3'th iteration by method in [6] (the objective function value $f(x^*) = 1.837623$) for this example.

Note: j is the number of iteration in the Algorithm 3.1.

ρ_j is constrain penalty parameter at the j'th iteration.

x^j is a solution at the j'th iteration in the Algorithm 3.1.
$f(x^j)$ is an objective value at x^j.
$g_i(x^j)$ $(i = 1, \ldots, m)$ is a constrain value at x^j.

Example 4.2. *Consider the example in [7],*

$$
\begin{aligned}
\textit{(P4.2)} \quad \min \quad & f(x) = 10x_2 + 2x_3 + x_4 + 3x_5 + 4x_6 \\
\textit{s.t.} \quad & g_1(x) = x_1 + x_2 - 10 = 0, \\
& g_2(x) = -x_1 + x_3 + x_4 + x_5 = 0, \\
& g_3(x) = -x_2 - x_3 + x_5 + x_6 = 0, \\
& g_4(x) = 10x_1 - 2x_3 + 3x_4 - 2x_5 - 16 \leq 0, \\
& g_5(x) = x_1 + 4x_3 + x_5 - 10 \leq 0, \\
& 0 \leq x_1 \leq 12, \\
& 0 \leq x_2 \leq 18, \\
& 0 \leq x_3 \leq 5, \\
& 0 \leq x_4 \leq 12, \\
& 0 \leq x_5 \leq 1, \\
& 0 \leq x_6 \leq 16.
\end{aligned}
$$

Let $k = 2,\ x^0 = (0, 0, 0, 0, 0, 0),\ \epsilon_0 = 0.1,\ \eta = 0.05,\ \rho_0 = 1000,\ N = 10$ and $\epsilon = 10^{-6}$. Numerical results of Algorithm 3.1 for solving (P4.2) are given in Table 2 and Table 3.

From Tables 2 and 3, it is said that an approximate ϵ-feasible solution to (P4.2) is obtained at the 3'th iteration, that is

$$x^3 = (1.665206, 8.318518, 0.138232, 0.527605, 0.994833, 7.456874)$$

with corresponding objective function value $f(x^3) = 116.801239$. It is easy to check that the x^3 is feasible solution to (P3.1). The solution we obtained is slightly better than the solution obtained in the 3'th iteration by method in [7] (the objective function value $f(x^*) = 117.010399$) for this example.

Table 2: Results for Example 4.2 with $k = 2,\ x^0 = (0,0,0,0,0,0),\ \rho_0 = 1000$ and $N = 10$

j	ρ_j	$f(x^j)$	x^j
1	1000	92.073639	(1.916631,6.794338,-0.002734,-0.003214,1.554299,4.869011)
2	10000	114.371962	(1.658223,8.181657,0.080941,0.528827,1.002706,7.214142)
3	100000	116.801239	(1.665206,8.318518,0.138232,0.527605,0.994833,7.456874)

Table 3: Results for Example 4.2 with $k = 2,\ x^0 = (0,0,0,0,0,0),\ \rho_0 = 1000$ and $N = 10$ (Continued)

j	ϵ_j	$g_1(x^j)$	$g_2(x^j)$	$g_3(x^j)$	$g_4(x^j)$	$g_5(x^j)$
1	0.1	-1.289031	-0.368279	-0.368295	0.053538	-6.540004
2	0.005	-0.160121	-0.045749	-0.045749	0.001415	-7.015308
3	0.00025	-0.016276	-0.004536	-0.005043	-0.031255	-6.787033

Example 4.3. *Consider the example in [7],*

$$\begin{aligned} \textit{(P4.3)} \quad \min \quad & f(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \\ s.t. \quad & g_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0, \\ & g_2(x) = (x_1 - 5)^2 + x_2^2 + x_3^2 - 25 = 0, \\ & g_3(x) = (x_1 - 5)^2 + (x_2 - 5)^2 + (x_3 - 5)^2 - 25 \leq 0. \end{aligned}$$

Let $k = 1,\ x^0 = (1,1,1),\ \epsilon_0 = 0.01,\ \eta = 0.02,\ \rho_0 = 10,\ N = 1.4$ and $\epsilon = 10^{-6}$. Numerical results of Algorithm 3.1 for solving (P4.3) are given in Table 4 and Table 5.

By Tables 4 and 5, an approximate optimal solution to (P4.3) is obtained at the 4'th iteration, that is $x^* = (2.500000, 4.221237, 0.964966)$ with corresponding objective function value $f(x^*) = 944.215652$. The solution we obtained is slightly better than the solution obtained in the 4'th iteration by method in [7] (the objective function value $f(x^*) = 944.215662$) for this example.

Now we change the initial parameters. Let $k = 1,\ x^0 = (1,1,1),\ \epsilon_0 = 0.01,\ \eta = 0.01,\ \rho_0 = 10,\ N = 1.5$ and $\epsilon = 10^{-6}$. Numerical results of Algorithm 3.1 for solving (P4.3) are given in Table 6 and Table 7. Further, with the same parameters $k,\ \rho_0,\ N,\ \epsilon_0,$
η as above, we change the starting point to $x^0 = (2,4,1)$. New numerical results are given in Table 8 and Table 9.

Table 4: Results for Example 4.3 with $k = 1,\ x^0 = (1, 1, 1),\ \rho_0 = 10,\ N = 1.4$

j	ϵ_j	$f(x^j)$	x^j
1	0.01	944.213296	(2.500102,4.221422,0.964456)
2	0.0002	944.215618	(2.500001,4.221362,0.964423)
3	0.000004	944.215655	(2.500000,4.220569,0.967885)
4	0.00000008	944.215652	(2.500000,4.221237,0.964966)

Table 5: Results for Example 4.3 with $k = 1,\ x^0 = (1, 1, 1)$ (Continued)

j	ρ_j	$g_1(x^j)$	$g_2(x^j)$	$g_3(x^j)$
1	10	0.001083	0.000065	-1.858705
2	14	0.000015	0.000001	-1.857845
3	19.6	0.000005	0.000005	-1.884536
4	27.44	-0.000000	-0.000000	-1.862025

It is easy to see from Tables 6-9 that the convergence of Algorithm 3.1 is the same and the objective function values are almost the same. That is to say, the efficiency of Algorithm 3.1 does not completely depend on how to choose a starting point in this example.

5 Conclusions

This paper has presented a smoothing approximation to the k-th power nonlinear penalty function and an algorithm based on this smoothed nonlinear penalty problem. It is shown that an optimal solution to the $(NP_{\rho,\epsilon})$ is

Table 6: Results for Example 4.3 with $k = 1,\ x^0 = (1, 1, 1),\ \rho_0 = 10,\ N = 1.5$

j	ϵ_j	$f(x^j)$	x^j
1	0.01	944.213296	(2.500102,4.221422,0.964456)
2	0.0001	944.215636	(2.500001,4.221362,0.964422)
3	0.000001	944.215649	(2.500000,4.221301,0.964684)
4	0.00000001	944.215691	(2.500000,4.222596,0.959002)

Table 7: Results for Example 4.3 with $k = 1,\ x^0 = (1, 1, 1)$ (Continued)

j	ρ_j	$g_1(x^j)$	$g_2(x^j)$	$g_3(x^j)$
1	10	0.001083	0.000065	-1.858705
2	15	0.000007	0.000004	-1.857838
3	22.5	0.000001	0.000001	-1.859853
4	33.75	-0.000000	-0.000002	-1.815976

Table 8: Results for Example 4.3 with $k = 1,\ x^0 = (2, 4, 1),\ \rho_0 = 10,\ N = 1.5$

j	ϵ_j	$f(x^j)$	x^j
1	0.01	944.213296	(2.500102,4.221422,0.964456)
2	0.0001	944.215636	(2.500001,4.221362,0.964422)
3	0.000001	944.215650	(2.500000,4.221422,0.964155)
4	0.00000001	944.215692	(2.500000,4.222568,0.959122)

Table 9: Results for Example 4.3 with $k = 1,\ x^0 = (2, 4, 1)$ (Continued)

j	ρ_j	$g_1(x^j)$	$g_2(x^j)$	$g_3(x^j)$
1	10	0.001083	0.000065	-1.858705
2	15	0.000007	0.000004	-1.857838
3	22.5	0.000001	0.000001	-1.855774
4	33.75	-0.000002	-0.000002	-1.816903

an approximate optimal solution to the original optimization problem under some mild conditions. Numerical experiments show that the Algorithm 3.1 has a good convergence for a global approximate solution.

References

[1] D. Bertskas, *Constrained optimization and Lagrange Multiplier Methods*, New York: Academic press, 1982.

[2] G. Di Pillo and L. Grippo, An exact penalty function method with global conergence properties for nonlinear programming problems, *Math. Program.*, **36**, (1986), 1-18.

[3] G. Di Pillo and L. Grippo, On the exactness of a class of nondifferentiable penalty function, *J. Optim. Theory Appl.*, **57**, (1988), 385-406.

[4] S.P. Han and O.L. Mangasrian, Exact penalty function in nonlinear programming, *Math. Program.*, **17**, (1979), 257-269.

[5] J.B. Lasserre, A globally convergent algorithm for exact penalty functions, *Eur. J. Oper. Res.*, **7**, (1981), 389-395.

[6] S.J. Lian, Smoothing approximation to l_1 exact penalty function for inequality constrained optimization, *Appl. Math. Comput.*, **219**, (2012), 3113-3121.

[7] Z.Q. Meng, C.Y. Dang and X.Q. Yang, On the smoothing of the square-root exact penalty function for inequality constrained optimization, *Comput. Optim. Appl.*,**35**, (2006), 375-398.

[8] K.W. Meng, S.J. Li and X.Q. Yang, A robust SQP method based on a smoothing lower order penalty function, *Optimization.*, **58**, (2009), 22-38.

[9] M.C. Pinar and S.A. Zenios, On smoothing exact penalty function for convex constrained optimization, *SIAM J. Optim.*, **4**, (1994), 486-511.

[10] E. Rosenberg, Exact penalty functions and stability in locally Lipschitz programming, *Math. Program.*, **30**, (1984), 340-356.

[11] A.M. Rubinov, B.M. Glover and X.Q. Yang, Extended Lagrange and penalty function in continuous optimization, *Optimization.*, **46**, (1999), 327-351.

[12] X.L. Sun and D. Li, Value-estimation function method for constrained global optimization, *J. Optim. Theory Appl.*, **24**, (1999), 385-409.

[13] Z.Y. Wu, F.S. Bai, X.Q. Yang and L.S. Zhang, An exact lower order penalty function and its smoothing in nonlinear programming, *Optimization.*, **53**, (2004), 57-68.

[14] Z.Y. Wu, H.W.J. Lee, F.S. Bai, L.S. Zhang and X.M. Yang, Quadratic smoothing approximation to l_1 exact penalty function in global optimization, *J. Ind. Manag. Optim.*, **1**, (2005), 533-547.

[15] X.Q. Yang, Smoothing approximations to nonsmooth optimization problems, *J. Aust. Math. Soc. B.*, **36**, (1994), 274-285.

[16] X.Q. Yang and X.X. Huang, A nonlinear Lagrangian approach to constrained optimization problems, *SIAM J. Optim.*, **11**, (2001), 1119-1144.

[17] X.Q. Yang, Z.Q. Meng, X.X. Huang and G.T.Y. Pong, Smoothing nonlinear penalty function for constrained optimization, *Numer. Funct. Anal. Optim.*, **24**, (2003), 357-364.

[18] W.I. Zangwill, Nonlinear programming via penalty function, *Mgmt. Sci.*, **13**, (1967), 334-358.

[19] S.A. Zenios, M.C. Pinar and R.S. Dembo, A smooth penalty function algorithm for network-structured problems, *Eur. J. Oper. Res.*, **64**, (1993), 258-277.