RECURSIVE APPROXIMATION OF THE HIGH DIMENSIONAL $\max$ FUNCTION

Ş. İ. Birbil[†], S.-C. Fang[‡], J. B. G. Frenk[♮] and S. Zhang[§*]

[†]*Faculty of Engineering and Natural Sciences*

Sabancı University, Orhanli-Tuzla, 34956, Istanbul, Turkey.

[‡]*Industrial Engineering and Operations Research*

North Carolina State University, Raleigh, NC 26695-7906, USA.

[♮]*Econometrics Institute, Erasmus University Rotterdam*

Postbus 1738, 3000 DR Rotterdam, The Netherlands.

[§]*Systems Engineering and Engineering Management*

The Chinese University of Hong Kong, Shatin, N.T., Hong Kong.

ABSTRACT. This paper proposes a smoothing method for the general n-dimensional $\max$ function, based on a recursive extension of smoothing functions for the 2-dimensional $\max$ function. A theoretical framework is introduced, and some applications are discussed. Finally, a numerical comparison with a well-known smoothing method is presented.

Keywords. smoothing methods, n-dimensional $\max$ function, recursive approximation

1 Introduction

In various areas of mathematical programming the $\max$ function plays an important role. This function appears either naturally during a modelling process or is called for in reformulating an existing model. A large class of problems can be formulated as a $\min - \max$ optimization problem. This class includes generalized fractional programs, Lagrangean duals, two-person zero sum games, robust optimization, variational inequalities and complementarity problems. For these problems the $\max$ function is taken either over a finite set or, in most cases, over an infinite set. To solve such problems a family of algorithms are proposed in [3], where the original problem is approximated by a sequence of $\min - \max$ problems with the $\max$ function ranging over a finite set. Therefore, in this general procedure one needs to solve iteratively a $\min - \max$ problem with the $\max$ function ranging over a finite set. If the original set is a polyhedron, it can be shown that one has to solve only a finite number of these problems. As mentioned above, the $\max$ function can also be applied to reformulate some well-known models. It can be shown that the set of stationary points of a mathematical program with equilibrium constraints can be equivalently formulated by a system of nonlinear equations, which includes several $\max$ functions with two arguments [6]. Although the $\max$ function has nice properties, it lacks differentiability and so,

*Corresponding author. Email: zhang@se.cuhk.edu.hk. Research supported by Hong Kong RGC Earmarked Grants CUHK4233/01E and CUHK4174/03E.

one cannot apply directly to this system of nonlinear equations the powerful computational techniques, such as Newton's method. To overcome the nondifferentiability of this system, different researchers have proposed several smoothing approximation schemes for the $\max$ function with two arguments [5, 6, 14]. An overview of these approaches is given by Qi and Sun [13], where all these approaches are shown to be a special case of the so-called smoothing method. In the particular case of the $\max$ function with two arguments, several computationally stable approximations can be found in the literature [13]. However, if a high dimensional $\max$ function appears in the formulation, one might also need a smooth approximation. Although, in theory, the smoothing method can be applied to approximate all kinds of nonsmooth functions on $\mathbb{R}^n$, in practice, one cannot use it for a smooth approximation of the n-dimensional $\max$ function. This is due to the fact that the method involves the evaluation of n-dimensional integrals, which is computationally intractable. In addition, there exists another analytic approximation for the high dimensional $\max$ function known as the entropy type approximation [1, 2, 7, 10].

Most of the approximation approaches depend on a parameter, which determines the precision of the approximation. It can be shown that, the approximating function converges to the $\max$ function as the approximation parameter goes to a limit (usually 0 or ∞). Since an approximation function is used to replace the $\max$ function in the original problem, two important issues arise: the convergence of the approximating problems to the original problem, and the numerical stability of the approximation scheme. Vazquez *et. al.* [17] focus on the convergence of using entropy type approximation function to the higher dimensional $\max$ function. Under some conditions, the authors establish the relation between the stationary points of the approximating function and the stationary points of the $\max$ function itself. By applying the same entropy type smoothing approximation, Xu [18] proposes an algorithm to solve the finite $\min - \max$ problems and proves that the proposed algorithm is globally convergent under certain assumptions. Xu also supports his analysis by some preliminary results showing that the entropy type approximation is promising for solving finite $\min - \max$ problems. Unfortunately, entropy type approximation may lead to overflow problems and hence, may suffer from numerical instability and slow convergence. In a recent paper, Polak *et. al.* [12] study the numerical difficulties of using the entropy type approximation in solving finite $\min - \max$ problems. The authors also discuss the tradeoff between the accuracy and ill-conditioning. To deal with this tradeoff, they propose an adaptive algorithm that uses a control mechanism for adjusting the approximation parameter during the solution process. This adaptive algorithm circumvents the numerical difficulties, and consequently, leads to a stable solution method for finite $\min - \max$ problems.

The contribution of this paper is to provide a generic approximation scheme for the high dimensional $\max$ function based on any smooth approximation of the two-dimensional $\max$ function. We propose a method which utilizes the approximation functions of the two-dimensional $\max$ function and extends them to higher dimensions via a recursive mechanism. It will become clear in the numerical results section that the proposed scheme is stable and accurate. For an illustrative application of this method in optimization we refer to [4].

We start with preliminary results related to approximation functions. This is followed by a section on the proposed recursive smoothing method. To show the value of the proposed method, we provide illustrative applications and compare the performance of the proposed method with the well-known entropy type approximation function.

2 Preliminary Results on Smoothing Methods

We first derive some result for a smoothing technique already discussed in [1]. Although this result is known we include for convenience a different short proof based on the elementary Lemma 2.1. To start with our exposition of the smoothing technique, let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and introduce the function, $F : \mathbb{R}^n \to [-\infty, \infty]$ given by

$$F(x) := \lim_{t\downarrow 0} tf(t^{-1}x). \tag{1}$$

As will be shown in Theorem 2.1, this limit always exists. In applications generally, the function F is nonsmooth, while the convex function f is chosen to be differentiable. Notice that by relation (1), it is immediately clear that the function F is also positively homogeneous and convex. The next elementary lemma (see for example [9]) is now crucial for the derivation of the main inequality.

Lemma 2.1 *The function $f : \mathbb{R}^n \to \mathbb{R}$ is convex on $\mathbb{R}^n$ if and only if the function $s : (0, \infty) \to \mathbb{R}$ given by*

$$s(t) := t^{-1}(f(y + tx) - f(y))$$

is non-decreasing on $(0, \infty)$ for every $x, y \in \mathbb{R}^n$.

Using Lemma 2.1, it is easy to show the next inequality. This result is already verified in [1] by means of a more difficult and longer proof.

Theorem 2.1 *For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, the function F, given by relation (1), is well defined. Moreover, for every $x \in \mathbb{R}^n$ and $t > 0$, the inequality*

$$tf(t^{-1}x) - F(x) \leq tf(0)$$

holds.

Proof. Since f is a convex function, it follows by Lemma 2.1 that for every $x \in \mathbb{R}^n$ the function $t \longmapsto t(f(t^{-1}x) - f(0))$ is non-increasing on $(0, \infty)$. Hence, we obtain

$$\sup_{t>0} t(f(t^{-1}x) - f(0)) = \lim_{t\downarrow 0} t(f(t^{-1}x) - f(0)) = F(x), \tag{2}$$

and this shows the result. □

Introduce now for the convex function $f : \mathbb{R}^n \to \mathbb{R}$, the set of functions $f(\cdot; t) : \mathbb{R}^n \to \mathbb{R}$, $t > 0$, given by

$$f(x; t) := tf(t^{-1}x). \tag{3}$$

It is shown in the next result under some additional assumptions that the pointwise convergence in relation (1) can be replaced by $supnorm$ convergence. In the sequel, for every compact set $A \subseteq \mathbb{R}^n$ we denote $\sup\{|h(x)| : x \in A\}$ by $\|h\|_A$.

Lemma 2.2 *For a given convex function $f : \mathbb{R}^n \to \mathbb{R}$, the following conditions are equivalent:*

1. *The function F, given by relation (1), is finite everywhere.*
2. *For every compact set $A \subseteq \mathbb{R}^n$, we have $\lim_{t\downarrow 0} \|F - f(\cdot; t)\|_A = 0$.*

Proof. The implication of $(2) \Rightarrow (1)$ is obvious. To show $(1) \Rightarrow (2)$, we observe the following: Since F and f are finite convex functions, we obtain by [Corollary 10.1.1, 15] that they are both continuous. Moreover, if we introduce the functions $\overline{f}(\cdot;t) : \mathbb{R}^n \to \mathbb{R}, t > 0$ given by

$$\overline{f}(x;t) := t(f(t^{-1}x) - f(0)),$$

then by Lemma 2.1 we have $\overline{f}(\cdot;s) \geq \overline{f}(\cdot;t)$ for every $0 < s < t$. Also, by relation (2), we know that $\lim_{t\downarrow 0} \overline{f}(x;t) = F(x)$. Hence the conditions of Dini's theorem [Theorem 7.13, 16] hold and so for every compact set A, we obtain $\lim_{t\downarrow 0} \|F - \overline{f}(\cdot;t)\|_A = 0$. Using now

$$\|F - f(\cdot;t)\|_A \leq \|F - \overline{f}(\cdot;t)\|_A + t|f(0)|,$$

the desired result follows. □

It is obvious that the first condition of Lemma 2.2 is satisfied if the function f is Lipschitz continuous on its domain. Let us now discuss two relations, which will be useful in the next section on recursive smoothing. For any convex function f and function F given by relation (1), we know for finite F that F is continuous. It is now an easy consequence of Lemma 2.2 that

$$\lim_{x_t\downarrow x_0, t\downarrow 0} f(x_t;t) = F(x_0). \tag{4}$$

Moreover, since F is continuous, we have

$$\beta := \max\{F(x) : \|x\|_\infty = 1\} < \infty,$$

where $\|.\|_\infty$ denotes the Chebyshev norm on $\mathbb{R}^n$ and this implies by Theorem 2.1 that

$$f(x) - f(0) \leq F(x) = \|x\|_\infty F(\|x\|_\infty^{-1}x) \leq \beta\|x\|_\infty \tag{5}$$

for every $x \neq 0$. When we consider the $\max$ function in the next section, it is important to note that β in the above relation is equal to 1.

Up to now, we only focused on deriving an upper bound. To derive a lower bound, we observe that in many applications one can choose the differentiable convex function f satisfying $f \geq F$. Since for every $t > 0$ and $x \in \mathbb{R}^n$

$$0 \leq tf(t^{-1}x) - tF(t^{-1}x) = f(x;t) - F(x),$$

we obtain in this case a trivial lower bound.

At this point a natural question, related to real life applications, arises: Given F, can one find a function f satisfying equation (1)? Clearly, the answer to this question depends on the function F. Moreover, depending on F, there may exist different alternatives for the function f. Nevertheless, to our belief finding a generic procedure to provide an answer to this question is quite difficult and it is a kind of art rather than a mathematical skill.

3 Recursive Smoothing

There exist different approximation functions with stable convergence properties, particularly for the two dimensional $\max$ function. However, an analytic form may not be easily extended from two dimensions to n dimensions. Before providing a solution to this problem, first notice that the n dimensional $\max$ function can be written recursively as follows

$$\max\{x_1, \cdots, x_n\} = \max\{\max\{x_1, \cdots, x_m\}, \max\{x_{m+1}, \cdots, x_n\}\},$$

where $1 < m < n$. Following this simple observation, we propose a recursive procedure to construct a high dimensional approximation for the $\max$ function with more than two variables. In the sequel, $f : \mathbb{R}^2 \to \mathbb{R}$ refers to the function that is used for approximating the two dimensional $\max$ function, i.e.,

$$\lim_{t\downarrow 0} tf(t^{-1}x) = \max\{x_1, x_2\}.$$

As an example for the two dimensional approximation [5], we might consider the following nondecreasing, convex function f given by

$$f(x) = \frac{\sqrt{(x_1 - x_2)^2 + 1} + x_1 + x_2}{2}.$$

This function is also used in the numerical results section. We start now with defining the recursive functions $f_{i,j} : \mathbb{R}^{j-i+1} \to \mathbb{R}$, $1 \le i < j \le n$, associated with the function f.

Definition 3.1 *Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a nondecreasing convex function, and define for every $1 \le i \le n-1$ the function $f_{i,i+1} : \mathbb{R}^2 \to \mathbb{R}$ by*

$$f_{i,i+1}(x) := f(x).$$

Moreover, for $1 \le i < j \le n$ and $k = j - i + 1 > 2$, let the function $f_{i,j} : \mathbb{R}^k \to \mathbb{R}$ be given by

$$f_{i,j}(x_i, ..., x_j) = f(f_{i,u_k}(x_i, ..., x_{u_k}), f_{l_k,j}(x_{l_k}, ..., x_j)), \tag{6}$$

where

$$u_k := i + \lceil \frac{k}{2} \rceil - 1 \text{ and } l_k := \begin{cases} u_k & \text{if } k \text{ is odd} \\ u_k + 1 & \text{if } k \text{ is even.} \end{cases}$$

To simplify the notation for the above recursive procedure, we introduce for every $1 \le i < j \le n$, the vectors

$$x^{(1)} := (x_i, ..., x_{u_k}) \text{ and } x^{(2)} := (x_{l_k}, ...x_j),$$

and the associated vector

$$x = \begin{cases} (x_i, ..., x_h, ..., x_j) & \text{if } k \text{ is odd} \\ (x_i, ..., x_{u_k}, x_{l_k}, ..., x_j) & \text{if } k \text{ is even,} \end{cases}$$

where $x_h = x_{u_k} = x_{l_k}$ whenever k is odd. In other words, the vector x is the proper concatenation of the vectors $x^{(1)}$ and $x^{(2)}$. The recursive procedure, defined in relation (6), can now be rewritten as

$$f_{i,j}(x) = f(f_{i,u_k}(x^{(1)}), f_{l_k,j}(x^{(2)})), 1 \le j < i \le n. \tag{7}$$

Before considering the computational scheme of evaluating the above recursive function, we first focus on its theoretical properties. As shown by the next lemma, it follows by induction that for every $i < j$, the function $f_{i,j}$ is an increasing convex function.

Lemma 3.1 *If $f : \mathbb{R}^2 \to \mathbb{R}$ is an increasing convex function then the function $f_{i,j} : \mathbb{R}^{j-i+1} \to \mathbb{R}$ with $1 \le i < j$ is also an increasing convex function.*

Proof. Clearly for $k := j - i + 1 = 2$ the function $f_{i,j} = f$ is an increasing convex function. Suppose now the induction hypothesis holds for the functions $f_{i,j}$ satisfying $k \le p$ and consider a function $f_{i,j}$ satisfying $k = p+1$. Since $u_{p+1} \le j-1$ and $l_{p+1} \ge i+1$, we obtain $u_{p+1} - i + 1 \le p$ and $j - l_{p+1} + 1 \le p$.

This shows by our induction hypothesis that the functions $f_{i,u_{p+1}}$ and $f_{l_{p+1},j}$ are increasing and convex. Hence by relation (7), the function

$$f_{i,j}(x) = f(f_{i,u_{p+1}}(x^{(1)}), f_{l_{p+1},j}(x^{(2)}))$$

is increasing and convex. □

By Theorem 2.1 and Lemma 3.1, we immediately obtain for the increasing convex function f the following results:

1. For every $1 \leq i < j \leq n$, the function $F_{i,j} : \mathbb{R}^k \to (-\infty, \infty]$ given by

   $$F_{i,j}(x) := \lim_{t \downarrow 0} t f_{i,j}(t^{-1}x)$$

   exists, and it is positively homogeneous and convex. Observe that the functions $F_{i,i+1}$, $1 < i < n-1$ refer to the same function, which is also denoted by F.

2. For every $1 \leq i < j \leq n$, $t > 0$, and $x \in \mathbb{R}^{j-i+1}$

   $$t f_{i,j}(t^{-1}x) - F_{i,j}(x) \leq t f_{i,j}(0). \tag{8}$$

3. If additionally $f \geq F$, then as a result of the monotonicity and by induction on every $1 \leq i < j \leq n$ and $x \in \mathbb{R}^k$ we have

   $$t f_{i,j}(t^{-1}x) \geq F_{i,j}(x).$$

Lemma 3.2 *If $f : \mathbb{R}^2 \to \mathbb{R}$ is an increasing convex function and the function F, given by relation (1), is finite everywhere, then for every $1 \leq i < j$, the function $F_{i,j}$ is finite everywhere. Moreover, the recursion*

$$F_{i,j}(x) = F(F_{i,u_k}(x^{(1)}), F_{l_k,j}(x^{(2)}))$$

holds.

Proof. Clearly for $k := j - i + 1 = 2$ the function $F_{i,i+1} = F$ is finite everywhere. Suppose now that the result holds for the functions $F_{i,j}$ satisfying $k \leq p$ and consider some function $F_{i,j}$ with $k = p + 1$. By relations (1) and (7), it follows with $x_t := (t f_{i,u_{p+1}}(t^{-1}x^{(1)}), t f_{l_{k+1}}(t^{-1}x^{(2)}))$ that

$$F_{i,j}(x) = \lim_{t \downarrow 0} t f(t^{-1}x_t). \tag{9}$$

Also by relation (7), we obtain $\lim_{t \downarrow 0} x_t = x_0$ where

$$x_0 = (F_{i,u_{p+1}}(x^{(1)}), F_{l_{p+1},j}(x^{(2)})).$$

This shows by our induction hypothesis that the vector x_0 belongs to $\mathbb{R}^2$ and, since the function F is finite everywhere, this implies by relations (4) and (9) that

$$F_{i,j}(x) = F(F_{i,u_{p+1}}(x^{(1)}), F_{l_{p+1},j}(x^{(2)})).$$

□

We are ready to give an upper bound for the relation (8). This upper bound can be constructed by using relation (5) and induction.

Lemma 3.3 *Let $f : \mathbb{R}^2 \to \mathbb{R}$ be an increasing convex function and for every $1 \leq i < j \leq n$, $F_{i,j}$ be finite. Then,*

$$tf_{i,j}(t^{-1}x) - F_{i,j}(x) \leq t(\log_2(k-1)+1)f(0)$$

where $k = j - i + 1$.

Proof. Clearly, for $k := j - i + 1 = 2$ the function $f_{i,j}(0) = f(0)$. So the result is true for $k = 2$. Suppose by induction that the result is true for $k \leq p$ and consider some $f_{i,j}(0)$ with $k = p + 1$. By relation (5), we obtain

$$f_{i,j}(0) = f(f_{i,u_{p+1}}(0), f_{l_{p+1},j}(0)) \leq \max(f_{i,u_{p+1}}(0), f_{l_{p+1},j}(0)) + f(0).$$

Since for every $k \geq 3$, $2(u_k - i + 1) \leq k + 1$ and hence $2(u_{p+1} - i) \leq p$, we obtain by our induction hypothesis that

$$\begin{aligned} f_{i,j}(0) &\leq (\log_2(u_{p+1} - i) + 1)f(0) + f(0) \\ &\leq (\log_2(p)f(0) + f(0) \\ &= (\log_2(p) + 1)f(0). \end{aligned}$$

Combining this by relation (8), we have the desired result

$$tf_{i,j}(t^{-1}x) - F_{i,j}(x) \leq tf_{i,j}(0) \leq t(\log_2(k-1)+1)f(0).$$

□

Remark 3.1 *For the $\max$ function with n arguments, Lemma 3.3 implies the following relation*

$$tf_{1,n}(t^{-1}x) - \max\{x_1, \cdots, x_n\} \leq t(\log_2 n + 1)f(0).$$

It is desirable to have a fast computation scheme for the recursive approximation and its higher order information. Before we show the computational complexity of these evaluations, notice that the recursive function $f_{i,j}(x)$ is a composition of several $f(x)$ functions. Therefore, for $t > 0$ if we define the smooth approximation of $F(x)$ by

$$f(x;t) := tf(t^{-1}x),$$

then the approximation function defined as

$$f_{i,j}(x;t) := tf_{i,j}(t^{-1}x)$$

also becomes differentiable. Furthermore, we assume that there exists an oracle, which provides one of the following outputs with an appropriate call:

1. The function value, $f(x;t)$ at point x for $t > 0$.
2. The gradient vector, $\nabla f(x;t)$ at point x for $t > 0$.
3. The Hessian matrix, $\nabla^2 f(x;t)$ at point x for $t > 0$.

Proposition 3.1 *Given $k > 2$ and the vector $x \in \mathbb{R}^k$, for every $i < j$ it requires $O(k)$ calls to the oracle to compute the value of $f_{i,j}(x;t)$.*

Proof. A simple counting reveals that the procedure requires at most $\lceil \log_2(k-1) \rceil$ number of recursive steps to complete the computation. At each step, say $s+1$, twice step s calls are made. Hence, the overall complexity is bounded by a constant times $2^{\lceil \log_2(k-1) \rceil} \leq 2(k-1)$. Therefore, computing the value of $f_{i,j}(x;t)$ requires at most $2(k-1)$ calls to the oracle. □

The next proposition shows that the higher order information of $f_{i,j}(x;t)$ can also be computed in polynomial time.

Proposition 3.2 *Given $k > 2$ and the vector $x \in \mathbb{R}^k$, for every $i < j$ it requires $O(k)$ calls to the oracle to compute the value of the gradient, $\nabla f_{i,j}(x;t)$ and similarly, $O(k)$ calls to compute the value of the Hessian, $\nabla^2 f_{i,j}(x;t)$.*

Proof. Since

$$f_{i,j}(x;t) = f(f_{i,u_k}(x^{(1)};t), f_{l_k,j}(x^{(2)};t);t), \tag{10}$$

by using the chain rule, we obtain

$$\nabla f_{i,j}(x;t) = \frac{\partial f(f_{i,u_k}(x^{(1)};t);t)}{\partial f_{i,u_k}(x^{(1)};t)} \nabla f_{i,u_k}(x^{(1)};t) + \frac{\partial f(f_{l_k,j}(x^{(2)};t);t)}{\partial f_{l_k,j}(x^{(2)};t)} \nabla f_{l_k,j}(x^{(2)};t) \tag{11}$$

and similarly,

$$\begin{aligned}
\nabla^2 f_{i,j}(x;t) = \quad & \tfrac{\partial f(f_{i,u_k}(x^{(1)};t);t)}{\partial f_{i,u_k}(x^{(1)};t)} \nabla^2 f_{i,u_k}(x^{(1)};t) + \tfrac{\partial f(f_{l_k,j}(x^{(2)};t);t)}{\partial f_{l_k,j}(x^{(2)};t)} \nabla^2 f_{l_k,j}(x^{(2)};t) \\
& + \tfrac{\partial^2 f(f_{i,u_k}(x^{(1)};t);t)}{\partial^2 f_{i,u_k}(x^{(1)};t)} \nabla f_{i,u_k}(x^{(1)};t)(\nabla f_{i,u_k}(x^{(1)};t))^T \\
& + \tfrac{\partial^2 f(f_{l_k,j}(x^{(2)};t);t)}{\partial^2 f_{l_k,j}(x^{(2)};t)} \nabla f_{l_k,j}(x^{(2)};t)(\nabla f_{l_k,j}(x^{(2)};t))^T \\
& + \tfrac{\partial^2 f(f_{i,u_k}(x^{(1)};t);t)}{\partial f_{i,u_k}(x^{(1)};t) \partial f_{l_k,j}(x^{(2)};t)} \nabla f_{i,u_k}(x^{(1)};t)(\nabla f_{i,u_k}(x^{(1)};t))^T \\
& + \tfrac{\partial^2 f(f_{l_k,j}(x^{(2)};t);t)}{\partial f_{i,u_k}(x^{(1)};t) \partial f_{l_k,j}(x^{(2)};t)} \nabla f_{l_k,j}(x^{(2)};t)(\nabla f_{l_k,j}(x^{(2)};t))^T
\end{aligned}$$

The complexity result follows from the same arguments as in the proof of Proposition 3.1. □

4 Numerical Examples

As mentioned in the introduction section, the recursive smoothing method can be applied to diverse problems modelled by a high dimensional $\max$ function. In $\min - \max$ optimization the smooth approximation is directly used to replace the $\max$ function, and so that the resulting model becomes a regular nonlinear programming problem [2]. On the other hand, in complementarity problems the system of equalities and nonnegativity constraints is equivalently modelled as a system of equalities composed of high dimensional $\max$ functions. After replacing the $\max$ functions with their approximations, a system of nonlinear inequalities are formed. Then different methods, like Newton's method, are utilized to solve this system [11]. In the remaining part of this section we will give two elementary examples that will focus on the main idea of these applications. Also, we will discuss the solution approach with the recursive approximation method and compare its performance with another approximation function of the high dimensional $\max$ function.

Before considering any example, let us introduce the following entropy type function $g : \mathbb{R}^n \to \mathbb{R}$, which is one of the most frequently used approximations in the literature [2, 7, 10, 12, 18]:

$$g(x) = \log \left(\sum_{i=1}^{n} e^{x_i} \right). \tag{12}$$

This function exhibits the following convergence property

$$\lim_{t\downarrow 0} tg(t^{-1}x) = \lim_{t\downarrow 0} t\log\left(\sum_{i=1}^{n} e^{\frac{x_i}{t}}\right) = \max\{x_1, \cdots, x_n\}.$$

Therefore, (12) is successfully utilized in solving both $\min - \max$ optimization and VLCP problems [2, 10–12, 18].

Notice that an overflow easily occurs when the exponential function in (12) is computed with a very large argument. A well-known technique to handle this potential problem is introducing a constant $z \geq \max\{x_1, \cdots, x_n\}$ and then computing

$$g_z(x;t) := t\log\left(\sum_{i=1}^{n} e^{\frac{x_i - z}{t}}\right) + z. \tag{13}$$

In order to apply the recursive approximation method, we choose the following nondecreasing convex function $f : \mathbb{R}^2 \to \mathbb{R}$

$$f(x) = \frac{\sqrt{(x_1 - x_2)^2 + 1} + x_1 + x_2}{2}.$$

which leads to the following approximation function

$$f(x;t) := tf(t^{-1}x) = \frac{\sqrt{(x_1 - x_2)^2 + t^2} + x_1 + x_2}{2}. \tag{14}$$

The right hand side of the equation (14) is the well-known Chen-Harker-Kanzow-Smale function [5]. This function has been extensively used for approximating the two dimensional $\max$ function, and it is known to be more stable than (12).

Although the technique in (13) effectively overcomes the difficulty of computing (12), another overflow problem occurs whenever the gradient or the Jacobian is computed. However, with the recursive approximation function using (14), the computation of the higher order information does not cause any overflow problems. The following elementary examples demonstrate this observation.

Example 4.1 *Solve the following optimization problem*

$$\min_{x\in\mathbb{R}^3} \max\{2x_1, 3x_2 - 4, 10x_3^2\}.$$

The arguments of the $\max$ function in Example 1 are differentiable. Hence, if we replace the $\max$ function with its smooth approximation, the resulting problem becomes a smooth unconstrained nonlinear programming problem. By using (13), the approximation becomes

$$g_z(x;t) = t\log(e^{t^{-1}(2x_1 - z)} + e^{t^{-1}(3x_2 - 4 - z)} + e^{t^{-1}(10x_3^2 - z)}) + z, \tag{15}$$

where $z \geq \max\{2x_1, 3x_2 - 4, 10x_3^2\}$. On the other hand by using (14), we have the following recursive approximation

$$f_{1,3}(x;t) := f(f(2x_1, 3x_2 - 4; t), f(3x_2 - 4, 10x_3^2; t); t). \tag{16}$$

Then for $t > 0$, we are interested in comparing the computational aspects of solving

$$\min_{x\in\mathbb{R}^3} g_z(x;t)$$

and

$$\min_{x\in\mathbb{R}^3} f_{1,3}(x;t).$$

A straightforward approach to solve these optimization problems is the gradient method [2]. In this approach, for $t > 0$ we need to compute the gradients of both (15) and (16) denoted by

$$\bigtriangledown g_z(x;t) = (\bigtriangledown g_z^{(1)}(x;t), \bigtriangledown g_z^{(2)}(x;t), \bigtriangledown g_z^{(3)}(x;t))^T \tag{17}$$

and

$$\bigtriangledown f_{1,3}(x;t) = (\bigtriangledown f_{1,3}^{(1)}(x;t), \bigtriangledown f_{1,3}^{(2)}(x;t), \bigtriangledown f_{1,3}^{(3)}(x;t))^T, \tag{18}$$

respectively. Consider the computation of the first component of (17)

$$\bigtriangledown g_z^{(1)}(x;t) = \frac{2(e^{t^{-1}(2x_1-z)})}{e^{t^{-1}(2x_1-z)} + e^{t^{-1}(3x_2-4-z)} + e^{t^{-1}(10x_3^2)}}.$$

Notice that the numerator converges fast to zero when the difference between z and the arguments of the $\max$ function is big. Therefore, if one uses (17), the first order information required for the gradient method diminishes. On the other hand, the computation of the gradient by (18) is more stable as a direct consequence of using (14). In order to support these statements, we have applied the unconstrained optimization procedure fminunc in MATLAB. We have selected the vector $(1, 1, 1)$ as the starting point and used equations (15-18) as the function and the gradient pointers. As we expected, even with parameter t set to $1.0e$-4, the first approximation (15) using (17) has led to an overflow problem, but the recursive approximation (16) using (18) has converged to the point, $(-9, -1, 0)$.

Example 4.2 *Find a solution $x \in \mathbb{R}^3$ to the following system of equations*

$$H(x) := \begin{bmatrix} \max\{2x_1, 3x_2 - 4, 10x_3^2\} \\ \max\{x_1^2, x_2 + 11, x_3 - 1\} \\ \max\{x_1, x_2 - 0.4, 2x_3^2\} \end{bmatrix} = 0.$$

In this example we are interested in solving a system of nonlinear equations. We first replace the $\max$ functions with their smooth approximations. Let $z_1 \geq \max\{2x_1, 3x_2 - 4, 10x_3^2\}$, $z_2 \geq \max\{x_1^2, x_2 + 11, x_3 - 1\}$, and $z_3 \geq \max\{x_1, x_2 - 0.4, 2x_3^2\}$ then by using (13) and (14), we can write the approximation functions

$$\begin{aligned} g_{z_1}^{(1)}(x;t) &: = t\log(e^{t^{-1}(2x_1-z_1)} + e^{t^{-1}(3x_2-4-z_1)} + e^{t^{-1}(10x_3^2-z_1)}) + z_1, \\ g_{z_2}^{(2)}(x;t) &: = t\log(e^{t^{-1}(x_1^2-z_2)} + e^{t^{-1}(x_2+11-z_2)} + e^{t^{-1}(x_3-1-z_2)}) + z_2, \\ g_{z_3}^{(3)}(x;t) &: = t\log(e^{t^{-1}(x_1-z_3)} + e^{t^{-1}(x_2-0.4-z_3)} + e^{t^{-1}(2x_3^2-z_3)}) + z_3, \end{aligned}$$

and

$$\begin{aligned} f_{1,3}^{(1)}(x;t) &: = f(f(2x_1, 3x_2 - 4;t), f(3x_2 - 4, 10x_3^2;t);t), \\ f_{1,3}^{(2)}(x;t) &: = f(f(x_1^2, x_2 + 11;t), f(x_2 + 11, x_3 - 1;t);t), \\ f_{1,3}^{(3)}(x;t) &: = f(f(x_1, x_2 - 0.4;t), f(x_2 - 0.4, 2x_3^2;t);t). \end{aligned}$$

Define

$$G_z(x;t) := (g_{z_1}^{(1)}(x;t), g_{z_2}^{(2)}(x;t), g_{z_3}^{(3)}(x;t))^T,$$

where $z := (z_1, z_2, z_3)$ and

$$F_{1,3}(x;t) := (f_{1,3}^{(1)}(x;t), f_{1,3}^{(2)}(x;t), f_{1,3}^{(3)}(x;t))^T.$$

Then for $t > 0$, we want to compare the solution efforts invested in solving

$$G_z(x;t) = 0, \tag{19}$$

and

$$F_{1,3}(x;t) = 0. \tag{20}$$

Commonly used methods for solving a system of nonlinear equations require the computation of the first order information. Therefore, we need to compute the Jacobians of the functions $G_z(x;t)$ and $F_{1,3}(x;t)$. Similar to the computation of the gradient (17), the computation of the Jacobian of $G_z(x;t)$ creates overflow problems. More importantly, the Jacobian becomes almost singular. In other words the reciprocal condition of the Jacobian approaches to 0 fast [8]. In contrast, the recursive computation of the Jacobian of $F_{1,3}(x;t)$ is more stable and leads to a nonsingular Jacobian. To solve (19) and (20), we have used the MATLAB procedure fsolve, which is designed to solve a system of nonlinear equations. For both test problems, we have provided the function and Jacobian pointers as an input to the fsolve procedure. Moreover, we have selected the vector $(1, 1, 1)$ as the starting point.

Table 1 shows the results with both approximations. The column 1 shows different t values. For each t value, the approximations (19) and (20) are solved by fsolve, and the respective results, denoted by H^*, are reported in columns 2 and 3. Clearly, decreasing the t value leads to better solutions. However, it may cause the Jacobian of $G_z(x;t)$ converging to a singular matrix rapidly. Therefore, as the figures in the second column suggest, overflow problems occur with the first approximation scheme. In the meanwhile, the third column shows that the recursive approximation behaves more stable than the first approximation, and gives improving solutions with decreasing t.

t	Approximation (19)	Approximation (20)
$1.0e$-1	$H^* = (4.1e\text{-}3, 3.6e\text{-}2, 7.0e\text{-}4)^T$	$H^* = (3.2e\text{ -}6, 1.1e\text{-}2, 6.0e\text{-}6)^T$
$1.0e$-2	No Solution	$H^* = (1.3e\text{-}8, 6.0e\text{-}4, 2.0e\text{-}9)^T$
$1.0e$-4	No Solution	$H^* = (7.0e\text{-}15, 2.1e\text{-}5, 1.3e\text{-}15)^T$
$1.0e$-6	No Solution	$H^* = (4.1e\text{-}14, 3.0e\text{-}8, 8.0e\text{-}16)^T$

Table 1: Comparison of the results of the two approximation schemes for Example 2.

5 Conclusions and Future Research

There exist many applications of the high dimensional $\max$ function. As a continuation of this research, the performance of the proposed method may be studied on these problems. An example of such an application could be in nonlinear programming, where there exists a set of difficult constraints. These constraints can be replaced with a single constraint after aggregating them by the $\max$ function. Again a high dimensional approximation would be required for smoothing the $\max$ function. In all these applications a rigorous analysis is required, since the arguments of the $\max$ function will be nontrivial functions. A natural extension of such a study should discuss the convergence of the stationary points of the approximating problems to the stationary points of the original problem. We believe that further research along these lines will be fruitful and promising.

References

[1] Ben-Tal, A. and M.Teboulle. A smoothing technique for nondifferentiable optimization problems. In Dolecki, editor, *Optimization*, Lectures notes in Mathematics 1405, pages 1–11, New York, 1989. Springer Verlag.

[2] Bertsekas, D.P. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York, 1982.

[3] Birbil, Ş.İ., Frenk, J.B.G. and S. Zhang. A finite approximation approach to $\min - \max$ optimization. Forthcoming, 2004.

[4] Birbil, Ş.İ., Frenk, J.B.G. and S. Zhang. Generalized fractional programming with user interaction. Technical Report SEEM2004-2, The Chinese University of Hong Kong, Department of Systems Engineering and Engineering Management, 2004.

[5] Chen, C. and O.L. Mangasarian. Smoothing methods for convex inequalities and linear complementarity problems. *Mathematical Programming*, 71:51–69, 1995.

[6] Facchinei, F., Jiang, H. and L. Qi. A smoothing method for mathematical programs with equilibrium constraints. *Mathematical Programming*, 85:107–134, 1999.

[7] Fang, S.-C., Rajasekara, J.R. and H.-S. Tsao. *Entropy Optimization and Mathematical Programming*. Kluwer Academic Publishers, Boston, 1997.

[8] Golub, G.H. and C.F. Loan. *Matrix Computations*. Johns Hopkins University, Baltimore, MD, 1996.

[9] Hiriart-Urruty, J.B. and C. Lemarechal. *Convex Analysis and Minimization Algorithms*, volume 1. Springer Verlag, Berlin, 1993.

[10] Li, X. An entropy-based aggregate method for minimax optimization. *Engineering Optimization*, 18:277–285, 1992.

[11] Peng, J.M. and Z. Lin. A non-interior continuation method for generalized linear complementarity problems. *Mathematical Programming*, 86:533–563, 1999.

[12] Polak, E., Royset, J.O. and R.S. Womersley. Algorithms with adaptive smoothing for the finite $\min - \max$ problems. *J. of Optimization Theory and Applications*, 119:457–484, 2003.

[13] Qi, L. and D. Sun. Smoothing functions and smoothing Newton method for complementarity and variational inequality problems. *J. of Optimization Theory and Applications*, 113:121–147, 2002.

[14] Qi, L. and X. Chen. A globally convergent successive approximation methods for nonsmooth equations. *SIAM J. Control Optim.*, 33:402–418, 1995.

[15] Rockafellar, R.T. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1972.

[16] Rudin, W. *Principles of Mathematical Analysis*. McGraw-Hill, New York, 1982.

[17] Vazquez, F.G., Günzel, H and H.T. Jongen. On logarithmic smoothing of the maximum function. *Annals of Operations Research*, 101:209–220, 2001.

[18] Xu, S. Smoothing method for minimax problems. *Computational Optimization and Applications*, 20:267–279, 2001.