Reliable Computing 2 (3) (1996), pp. 213–218

Robust algorithms that locate local extrema of a function of one variable from interval measurement results: A remark

Christoph Eick and Karen Villaverde

The problem of locating local maxima and minima of a function from approximate measurement results is vital for many physical applications: in *spectral analysis*, chemical species are identified by locating local maxima of the spectra; in *radioastronomy*, sources of celestial radio emission, and their subcomponents, are identified by locating local maxima of the measured brightness of the radio sky; *elementary particles* are identified by locating local maxima of the experimental curves. Since measurements are never absolutely precise, as a result of the measurements, we have a *class* of possible functions. If we measure $f(x_i)$ with interval uncertainty, this class consists of all functions f for which $f(x_i) \in [y_i - \varepsilon, y_i + \varepsilon]$, where y_i are the results of measuring $f(x_i)$, and ε is the measurement accuracy. For this class, in [2], a linear-time algorithm was described.

In real life, a measuring instrument can sometimes malfunction, leading to the so-called *outliers*, i.e., measurements y_i that can be way off $f(x_i)$ (and thus do not restrict the actual values $f(x_i)$ at all).

In this paper, we describe *robust* algorithms, i.e., algorithms that find the number of local extrema in the presence of possible outliers. These algorithms solve an important practical problem, but they are not based on any new mathematical results: they simply use algorithms from [2] and [3].

Робастные алгоритмы для нахождения локальных экстремумов функции одной переменной на основе интервальных результатов измерений: замечание

К. Эйк, К. Виллаверде

Задача нахождения локальных максимумов и минимумов функции на основе результатов приближенных измерений важна для многих физических приложений: в *спектральном анализе* химические компоненты идентифицируются путем поиска локальных максимумов в спектре; в *радиоастрономии* источники космического радиоизлучения и их компоненты идентифицируются через отыскание локальных максимумов измеренной яркости неба в радиодиапазоне; *элементарные частицы* обнаруживаются через нахождение локальных максимумов экспериментальных кривых. Поскольку измерения никогда не бывают абсолютно точными, в результате этих измерений мы имеем некий *класс* возможных функций. Если $f(x_i)$ измеряется с интервальной неопределенностью, этот класс включает в себя все функции f, для которых $f(x_i) \in [y_i - \varepsilon, y_i + \varepsilon]$, где y_i есть результаты измерений $f(x_i)$, а ε есть их погрешность. Для такого класса функций в работе [2] описан алгоритм с линейным временем.

На практике измерительные инструменты могут иногда выходить из строя, что приводит к появлению в результатах *всплесков*, т. е. таких значений y_i, которые очень далеко отстоят от $f(x_i)$ и, таким образом, вообще не ограничивают реальные значения $f(x_i)$.

В данной работе описываются *робастные* алгоритмы, т. е. такие алгоритмы, которые находят число локальных экстремумов при возможном присутствии в результатах всплесков. Эти алгоритмы позволяют решить важную практическую задачу, но основываются не на новых математических результатах, а на известных алгоритмах из [2] и [3].

1. Locating local extrema: importance and known results

In many applications, it is important to find the local maxima and/or minima of a function $f(x)$ of one variable x:

- In *spectral analysis*, chemical species are identified by locating local maxima of the spectra.
- In *radioastronomy*, sources of celestial radio emission, and their subcomponents, are identified by locating local maxima of the measured *brightness distribution* of the radio sky, i.e., the function $y(x)$ that describes how the intensity y of the signal depends on the position x of the point from which we receive this signal.
- *Elementary particles* are identified by locating local maxima of the experimental curves that describe (crudely speaking) the scattering intensity y as a function of energy x.

In all these cases, we would like to know the number and location of the local extrema.

Measurements are never absolutely precise; as a result, after the measurements, we never get a unique function $f(x)$; we get a *class* $\mathcal{F}$ of possible functions. In [2, 3], we considered the case when the only information about f comes from measuring the values of $f(x)$ for several values $x = x_1 < \cdots < x_n$ with interval uncertainty. If y_i are the results of measuring $f(x_i)$, and the manufacturer guarantees that the absolute value $|\Delta y_i|$ of the measurement error $\Delta y_i = y_i - f(x_i)$ cannot exceed a given number $\varepsilon > 0$, then, as a result of each measurement of $f(x_i)$, we get an *interval* $\mathbf{y}_i = [y_i^-, y_i^+] = [y_i - \varepsilon, y_i + \varepsilon]$ of possible values of $f(x_i)$. The resulting class of functions has the form

$$\mathcal{F} = \{f \mid f(x_i) \in \mathbf{y}_i \quad \text{for all } i = 1, \ldots, n\}. \tag{1}$$

For this case, in [2], we have described a linear-time algorithm that computes the finite list of *location intervals*, i.e., pairwise disjoint open intervals $I_1, \ldots, I_k$, each of which I_j has the following two properties:

a) Every function $f \in \mathcal{F}$ attains a local maximum on I_j.

b) No proper subinterval $J \subseteq I_j$ has property a).

Thus, every function $f \in \mathcal{F}$ has at least k local maxima. It has also been proven in [2] that there exist functions $f \in \mathcal{F}$ that have exactly k local maxima.

So, e.g., in radioastronomy, from the measurement results, we can guarantee that the observed source has at least k components (maybe more).

For the case when we have several processors working in parallel, in [3], two *parallel* algorithms were developed to compute the intervals I_j. These algorithms require, correspondingly:

1. $O(\log^2(n))$ time on $O(n)$ processors; and
2. $O(\log(n))$ time on $O(n^2)$ processors.

2. Problem: possibility of outliers

The formula (1) was based on the assumption that the measuring instrument always works well, and thus, whenever the measured value of $f(x_i)$ is y_i, we can guarantee that $f(x_i) \in [y_i - \varepsilon, y_i + \varepsilon]$. In real life, we often encounter the situation when a measuring instrument may malfunction and thus, the result y_i of one of the measurements may be "way off". Such measurement results are called *outliers*. The problem of data processing in the presence of outliers is often considered in mathematical statistics (see, e.g., [1; 4, Chapter 16]). The corresponding data processing methods are called *robust* methods.

If it is possible that one of the measurement results is actually an outlier, then the actual function f may not belong to the class (1). Instead, it belongs to the following class:

$$\mathcal{F}_1 = \{f \mid f(x_i) \in \mathbf{y}_i \quad \text{for all } i = 1, \ldots, n, \text{ except, maybe, one } i\}. \tag{2}$$

It is also possible that not one, but *several* values are outliers. If we do not know how many of them are outliers, then, of course, it can happen that all measurements are outliers, and, therefore, we cannot say anything about $f(x)$. Meaningful results can happen only if we know the upper bound b ($b < n$) on the number of outliers. In this case, the class of functions is described by the formula

$$\mathcal{F}_b = \{f \mid f(x_i) \in \mathbf{y}_i \quad \text{for at least } n - b \text{ different } i = 1, \ldots, n\}. \tag{3}$$

The problem is: *What can we say about the number of local extrema and about their locations in these robust situations?*

In this short paper, we will show that the algorithms from [2, 3] can solve this problem as well.

3. Solution

3.1. Computing the possible number of local extrema

Definition 1. *We say that an algorithm $\mathcal{U}$ finds the number of local extrema in the case of $\leq b$ outliers, if, given $x_1, \ldots, x_n, y_1 \ldots, y_n$, and ε as input, this algorithm returns the following two integers:*

- *the smallest possible number of local maxima of a function $f \in \mathcal{F}_b$;*
- *the smallest possible number of local minima of a function $f \in \mathcal{F}_b$.*

Proposition 1. *For every $b \geq 1$, there exist algorithms $\mathcal{U}_0, \ldots, \mathcal{U}_{b+2}$ that find the number of local extrema in the case of $\leq b$ outliers, and that require the following computation time:*

- *$O(n^{b+1})$ on a single processor ($\mathcal{U}_0$);*
- *$O(n^b)$ time on $O(n)$ processors ($\mathcal{U}_1$);*
- *$O(n^{b-1})$ time on $O(n^2)$ processors ($\mathcal{U}_2$);*
- ...

- $O(n^{b+1-k})$ *time on* $O(n^k)$ *processors* $(\mathcal{U}_k)$;
- ...
- $O(n)$ *time on* $O(n^b)$ *processors* $(\mathcal{U}_b)$;
- $O(\log^2(n))$ *time on* $O(n^{b+1})$ *processors* $(\mathcal{U}_{b+1})$;
- $O(\log(n))$ *time on* $O(n^{b+2})$ *processors* $(\mathcal{U}_{b+2})$.

In particular, for $b = 1$ and for $b = 2$, we get the following results:

Corollary 1. *There exist algorithms* $\mathcal{U}_0, \ldots, \mathcal{U}_3$ *that find the number of local extrema in the case of* ≤ 1 *outliers, and that require the following computation time:*

- $O(n^2)$ *on a single processor* $(\mathcal{U}_0)$;
- $O(n)$ *time on* $O(n)$ *processors* $(\mathcal{U}_1)$;
- $O(\log^2(n))$ *time on* $O(n^2)$ *processors* $(\mathcal{U}_2)$;
- $O(\log(n))$ *time on* $O(n^3)$ *processors* $(\mathcal{U}_3)$.

Corollary 2. *There exist algorithms* $\mathcal{U}_0, \ldots, \mathcal{U}_4$ *that find the number of local extrema in the case of* ≤ 2 *outliers, and that require the following computation time:*

- $O(n^3)$ *on a single processor* $(\mathcal{U}_0)$;
- $O(n^2)$ *time on* $O(n)$ *processors* $(\mathcal{U}_1)$;
- $O(n)$ *time on* $O(n^2)$ *processors* $(\mathcal{U}_2)$;
- $O(\log^2(n))$ *time on* $O(n^3)$ *processors* $(\mathcal{U}_3)$;
- $O(\log(n))$ *time on* $O(n^4)$ *processors* $(\mathcal{U}_4)$.

Proof. For simplicity, let us start with the case $b = 1$, i.e., with the case when there exists at most one outlier. Then, depending on the existence and the location of an outlier, we can divide the class $\mathcal{F}_1$ into the following $n + 1$ subclasses:

- The class $\mathcal{F}_1^0$ of all functions f for which none of the measurements are outliers, i.e., for which $f(x_i) \in \mathrm{y}_i$ for all i.
- n classes $\mathcal{F}_1^k$, $1 \leq k \leq n$. Each of these classes consists of all functions f for which the $k-$th measurement is an outlier, i.e., for which $f(x_i) \in \mathrm{y}_i$ for $i = 1, \ldots, k-1, k+1, \ldots, n$ (i.e., for $n-1$ different values of i).

Because of this division, in order to find the smallest possible number of local maxima of functions $f \in \mathcal{F}_1$, it is sufficient:

- to find the smallest possible number of local maxima for $f \in \mathcal{F}_1^k$, $k = 0, 1, \ldots, n$, and then
- to compute the smallest of these numbers.

Each of these classes $\mathcal{F}_1^k$ is of type (1) (with n or $n-1$ sets of inputs); therefore, for each of these classes, we can use either the algorithm described in [2], or one of the algorithms 1., 2. described in [3].

For $b > 1$, we can also divide the class $\mathcal{F}_b$ into the classes that correspond to different possible subsets of outlier points. Each of these classes is of type (1), and the total number of these classes is equal to the total number of subsets of size $\leq b$ in the set $\{1, \ldots, n\}$ of n elements. This number is known to be $O(n^b)$.

If we *sequentially* perform computations, corresponding to the algorithm from [2], for these $O(n^b)$ classes, then the computation time gets multiplied by $O(n^b)$; the resulting computation time is $O(n^{b+1})$. This is algorithm $\mathcal{U}_0$.

If we have n^k processors, $k \leq b$, then we can subdivide $O(n^b)$ classes into n^k groups assigned to different processors; each group consists of $O(n^{b-k})$ classes. Inside each group, we sequentially perform computations, corresponding to the algorithm from [2], for all $O(n^{b-k})$ classes. The resulting time is $O(n^{b+1-k})$. This is algorithm $\mathcal{U}_k$.

Finally, if we perform instead the computations that correspond to the algorithms from [3] *in parallel* for each of the $O(n^b)$ classes, we must multiply the number of processors by $O(n^b)$; this idea leads to algorithms $\mathcal{U}_{b+1}$ and $\mathcal{U}_{b+2}$. □

3.2. Locating local extrema

Location-wise, the best we can do is to present to the user all possible sets of interval locations for the local extrema that correspond to different possible outliers.

To illustrate the possible complexity of the situation, let is give a simple example: $n = 100$, $b = 1$ (i.e., at most one outlier is possible), $y_2 = y_{99} = 3\varepsilon$, and $y_i = 0$ for all other i. In this case, according to the algorithm from [2], all function $f \in \mathcal{F}_1$ must have at least one local maximum:

- If y_2 is an outlier, then the maximum is located on the interval (x_{98}, x_{100}).
- If y_{99} is an outlier, then the maximum is located on the interval (x_1, x_3).
- If none of y_i are outliers, or if y_i is an outlier for some $i \neq 2$ and $i \neq 99$, then there are *two* intervals, each of which must contain a local maximum.

As a result, we can conclude that there can be only one local maximum, but the location of this local maximum depends on which of the measurement results is an outlier.

Acknowledgments

The authors are very thankful to V. Kreinovich for his help and encouragement.

References

[1] Huber, P. J. *Robust statistics.* Wiley, N.Y., 1981.

[2] Villaverde, K. and Kreinovich, V. *A linear-time algorithm that locates local extrema of a function of one variable from interval measurement results.* Interval Computations 4 (1993), pp. 176–194.

[3] Villaverde, K. and Kreinovich, V. *Parallel algorithms that locate local extrema of a function of one variable from interval measurement results.* In: "Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995", Reliable Computing (1995), Supplement, pp. 212–219.

[4] Wadsworth, Jr, H. M. (ed.) *Handbook of statistical methods for engineers and scientists.* McGraw-Hill Publishing Co., N.Y., 1990.

Received: December 10, 1995
Revised version: January 30, 1996

CH. EICK
Computer Science Department
University of Houston
Houston, TX
USA
E-mail: ceick@cs.uh.edu

KAREN VILLAVERDE
Systems Engineering
Northern Telecom/Bell Northern Research
M/S D0–207
P.O. Box 833871
Richardson, TX 75083–3871
USA
E-mail: villa@bnr.ca