Interval Computations
No 4, 1993

A Linear-Time Algorithm That Locates Local Extrema of a Function of One Variable From Interval Measurement Results

Karen Villaverde, Vladik Kreinovich*

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.

In mathematical terms, we know n numbers $x_1 < \cdots < x_n$, n values $y_1, \ldots, y_n$, value $\varepsilon > 0$, and we know that the values $\bar{f}(x_i)$ of the unknown function $f(x)$ at the points x_i belong to the intervals $I_i = [y_i^-, y_i^+], i = 1, \ldots, n$, where $y_i^- = y_i - \varepsilon$ and $y_i^+ = y_i + \varepsilon$. The set $\mathcal{F}$ of all the functions $f(x)$ that satisfy this property can be considered as a *function interval* (this definition was, in essence, first proposed by R. Moore). We say that an interval I *locates a local maximum* if all functions $f \in \mathcal{F}$ attain a local maximum at some point from I. So, the problem is to generate intervals $I_1, \ldots, I_k$ that locate local maxima.

Evidently, if I locates a local maximum, then any bigger interval $J \supset I$ also locates this maximum. We want to find the *smallest* possible location I. We propose an algorithm that finds the smallest possible locations in linear time (i.e., in time that is $\leq Cn$ for some C).

*This work was partially supported by NSF grants No. CDA-9015006 and IRI-92-11-662, NASA Research Grant No. 9-482, and a grant from the Materials Research Institute. This work was started under the Grant from the Soviet Science Foundation; part of this work was done at Stanford, where one of the authors (V. K.) was supported by McCarthy Foundation. The authors wish also to express their gratitude to Andrew Bernat (El Paso, TX), Michael Gelfond (El Paso, TX), Yuriy Gurevich (Ann Arbor, MI), Peter Fishburn (AT&T Bell Labs), Vladimir Lifschitz (Austin, TX), Yuriy Matiyasevich (St. Petersburg, Russia), and Patrick Suppes (Stanford, CA) for valuable discussions. The authors are also thankful to Andrey Finkelstein (St. Petersburg, Russia), Leonid Ivanov (St. Petersburg, Russia), and Frederick Schwab (Charlottesville, VA) for discussing possible astronomical applications, and to anonymous referees for important improvements.

Алгоритм с линейным временем для нахождения локальных экстремумов функции одной переменной по даннымм интервальных измерений

К. Виллаверде, В. Крейнович

Задача локализации максимумов и минимумов некоторой функции исходя из приближенных результатов измерения чрезвычайно важна для многих физических приложений. В спектральном анализе осколки соединений идентифицируются путем локализации локальных максимумов в спектрах. В радиоастрономии источники космического излучения, а также их субкомпоненты могут быть идентифицированы с помощью локализации локальных максимумов измеряемой яркости фонового излучения. Элементарные частицы идентифицируются посредством локализации локальных максимумов экспериментальных кривых.

На математическом языке, известны n чисел $x_1 < \cdots < x_n$, n величин $y_1, \ldots, y_n$, величина $\varepsilon > 0$, а также известно, что значения $\bar{f}(x_i)$ неизвестной функции $\bar{f}(x)$ в точках x_i принадлежат интервалам $I_i = [y_i^-, y_i^+], i = 1, \ldots, n$, где $y_i^- = y_i - \varepsilon$ и $y_i^+ = y_i + \varepsilon$. Множество $\mathcal{F}$ всех функций $f(x)$, удовлетворяющих этому свойству, можно рассматривать как функциональный интервал (это определение по существу впервые появилось у Р. Мура). Мы говорим, что интервал I заключает в себе локальный максимум, если все функции $f \in \mathcal{F}$ достигают максимума в некоторой точке из интервала I. Таким образом, задача состоит в генерировании интервалов $I_1, \ldots, I_k$, в которых заключаются локальные максимумы.

Очевидно, что если в I содержится локальный максимум, то больший интервал $J \supset I$ также содержит этот максимум. Мы хотим найти *наименьший* возможный интервал I, содержащий максимум. Предложен алгоритм нахождения наименьших возможных локализаций за линейное время (т. е. за время, меньшее или равное Cn для некоторого C).

1 Why is this problem important for applications?

The problem is really important. In many applications, the following problem arises: we know that a physical quantity y is a function of some other physical quantity x, $y = f(x)$, and we want to know the local maxima of this function f.

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. In other words, we are interested in the local maxima of the *brightness distribution*, i.e., of 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.

Local maxima and minima are also used in the methods that accelerate the convergence of the measurement result to the real value of a physical variable, and thus allow the user to estimate this value without waiting for the oscillations to stop [12].

In all these cases, we have some basic information. In all these cases, we know the results $y_1, \ldots, y_n$ of measuring the (unknown) function $\bar{f}(x)$ for some values $x_1, \ldots, x_n$, and we know the accuracy ε of the measuring instrument (i.e., we know that $|\bar{f}(x_i) - y_i| \leq \varepsilon$ for all i).

In some real-life cases, we may have additional information about $\bar{f}$:

- We may know a *formula* that describes the unknown function $\bar{f}(x)$; e.g., in optical spectroscopy, we often know that the spectrum is a linear combination of several Gaussian functions: $\bar{f}(x) = \sum c_i \exp\left((x - a_i)^2/\sigma_i\right)$ (where the parameters c_i, a_i, and σ_i are apriori unknown). In this case, we can find the parameters of this function $\bar{f}$ from the measurement results, and then use the explicit formula for $\bar{f}$ to describe the local extrema.

- In addition of the accuracy, we may also know *statistical characteristics* of the measurement errors $y_i - \bar{f}(x_i)$. In this case, we can use statistical methods to describe the possible locations of local extrema.

In many important cases we do not have any additional information (see, e.g., [1, 18]): we do not know the shape of $\bar{f}(x)$, and we do not know the probabilities of different errors. In these cases, the only information that we have is the values x_i in which $\bar{f}(x)$ was measured, the measured values y_i, and the accuracy ε. The only information we thus have about the value $\bar{f}(x_i)$ of the (unknown) function $\bar{f}(x)$ is that it belongs to an interval $[y_i - \varepsilon, y_i + \varepsilon]$. We can express this fact by saying that we have *interval measurement results.*

For interval measurement results, there are methods that locate local extrema, but these methods do not give a guaranteed location of the extrema. For the cases when no additional information is available, there exist several numerical methods of locating extrema (see, e.g., [4, 8, 16, 17]). These methods turn out to be very successful in many applications in the sense that the locations that they produce are in good accordance with the results of more accurate measurements. However, the existing methods do not provide the user with any *guaranteed* estimates of the *accuracy* of their results.

In other words, the algorithm computes an approximate location $\tilde{x}(e)$ of the extremum. Since we started with the approximate measurement results, the actual location $x(e)$ of this extremum may be different from $\tilde{x}(e)$. What are the possible values of $x(e)$?

What we are going to do. In this paper, we produce an algorithm that not only computes the approximate locations, but also computes the interval of possible values of $x(e)$.

2 Formulation of the problem in mathematical terms

Motivation. In all the above applications (spectral analysis, radioastronomy, elementary particles), the precise form of the actual function $\bar{f}(x)$ is unknown; the only information that we have consists of the results y_i of measuring $\bar{f}(x)$ for finitely many values $x_1, x_2, \ldots, x_n$. If we denote the accuracy

of these measurements by ε, then this information means that the unknown function must satisfy the inequalities $|\bar{f}(x_i) - y_i| \leq \varepsilon$ for all $i = 1, \ldots, n$. In other words, the only information that we have about the function $\bar{f}(x)$ is that for every $i = 1, \ldots, n$, $\bar{f}(x_i) \in I_i$, where $I_i = [y_i^-, y_i^+]$, $y_i^- = y_i - \varepsilon$, and $y_i^+ = y_i + \varepsilon$.

Definition 1. *Suppose that for some integer n, we are given n real numbers $x_1 < \ldots < x_n$, n real numbers $y_1, \ldots, y_n$, and $\varepsilon > 0$. By a function interval we mean the set $\mathcal{F}$ of all continuous functions $f(x)$ such that for all $i = 1, \ldots, n$, $f(x_i) \in I_i = [y_i^-, y_i^+]$, where $y_i^- = y_i - \varepsilon$ and $y_i^+ = y_i + \varepsilon$.*

Remark. This definition is a slight modification of the one originally proposed by R. E. Moore (see, e.g., [9], Section 5.1; [10], Section 2.5).

Definition 2. *We say that a function $f(x)$ attains a local maximum at a point x_0 if there exists an interval $[x^-, x^+]$ such that $x^- < x_0 < x^+$, and $f(x_0) \geq f(x)$ for all $x \in [x^-, x^+]$. Likewise, we say that a function $f(x)$ attains a local minimum at a point x_0 if there exists an interval $[x^-, x^+]$ such $x^- < x_0 < x^+$, and $f(x_0) \leq f(x)$ for all $x \in [x^-, x^+]$.*

Definition 3. *Suppose that a function interval $\mathcal{F}$ is given. We say that an interval I locates a local maximum if all functions $f \in \mathcal{F}$ attain a local maximum at some point from I. We say that an interval I locates a local minimum if all functions $f \in \mathcal{F}$ attain a local minimum at some point from I.*

Main problem: *to generate intervals $I_1, \ldots, I_k$ that locate local maxima and local minima.*

Remarks.

1) There exist various algorithms that locate the *global maxima* of an intervally defined function (see, e.g., [3, 10, 11, 15]). However, the *input* for these methods *is* very *different*: namely, an expression for the function. Besides, for these algorithms, *local maxima are the* main *obstacle* that has to be overcome (and *not the* desired *result*). For these two reasons, we cannot apply these algorithms to locate all *local* maxima.

2) Evidently, if I locates a local maximum, then any bigger interval $J \supset I$ also locates it. We want to find the *smallest* possible locations I. In other words, we want an *optimal* interval estimate in the sense of [13] and [14] (see also [5]). Let's describe this demand in mathematical terms.

Definition 4. *Suppose that a function interval $\mathcal{F}$ is given. We say that intervals I and J locate the same local maximum if there exists an interval K that locates a local maximum and such that $K \subseteq I$ and $K \subseteq J$. We say that a list $I_1, \ldots, I_k$ locates all local maxima if any other interval I that locates a local maximum, locates the same interval as one of these I_i.*

Definition 5. *Suppose that a function interval $\mathcal{F}$ is given. We say that an interval I locates the local maximum precisely, if I locates a local maximum, and no proper subinterval $I' \subset I$ locates it.*

Similar definitions can be repeated for *local minima*. Taking this into consideration, we can reformulate the main problem as follows:

Main problem: *We must locate all local maxima and local minima precisely.*

Example (see Fig. 1)

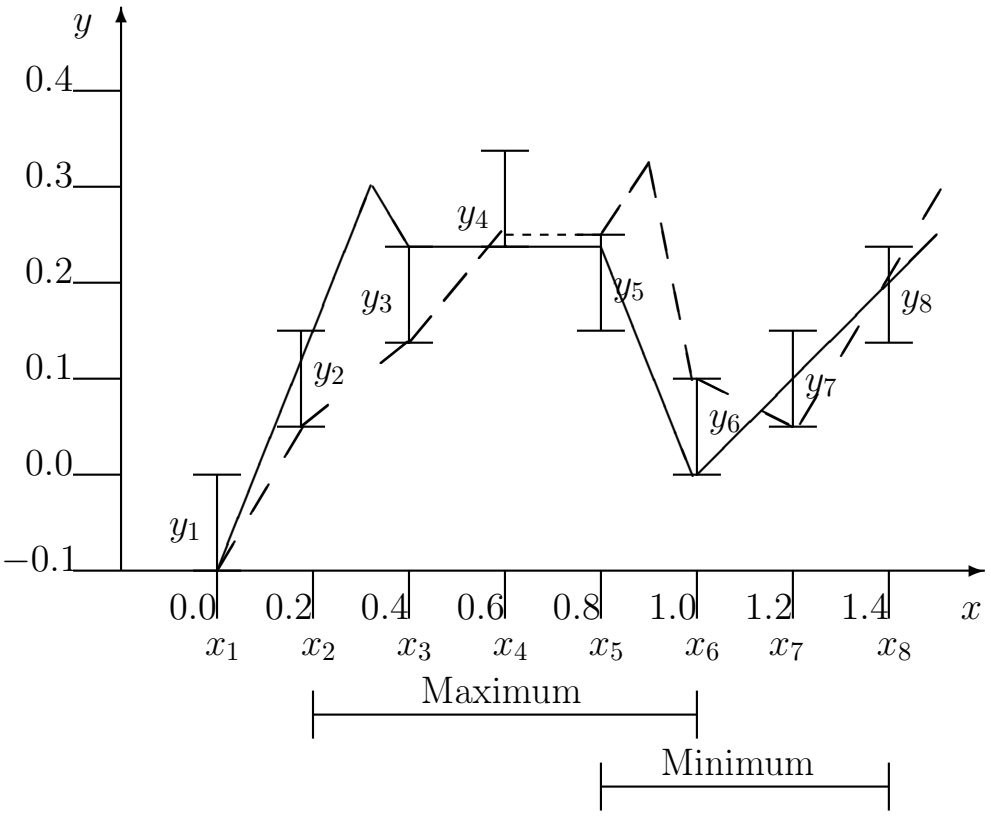

Figure 1: Precise location of local maxima and local minima

Take $n = 8$, $x_i = 0.2(i-1)$ (i.e., $x_1 = 0$, $x_2 = 0.2$, ..., $x_8 = 1.4$), $\varepsilon = 0.05$, $y_1 = -0.05$, $y_2 = 0.11$, $y_3 = 0.19$, $y_4 = 0.29$, $y_5 = 0.21$, $y_6 = 0.05$, $y_7 = 0.11$, and $y_8 = 0.19$. In this case, $I_1 = [-0.1, 0]$, $I_2 = [0.06, 0.16]$, $I_3 =$

$[0.14, 0.24]$, $I_4 = [0.24, 0.34]$, $I_5 = [0.16, 0.26]$, $I_6 = [0, 0.1]$, $I_7 = [0.06, 0.16]$, and $I_8 = [0.14, 0.24]$. The first 6 measurements are consistent with the function $\bar{f}(x) = x(1-x)$ that attains a local maximum at $x = 0.5$. Our algorithm will show that for this $\mathcal{F}$, the interval $I = (0.2, 1.0)$ locates the local maximum precisely. This means that an arbitrary number from the interval $(0.2, 1.0)$ can be the location of some function from $\mathcal{F}$. As an example, in Fig. 1, we show two functions that are consistent with these measurement results and that attain local maxima correspondingly at $x = 0.3$ and $x = 0.9$.

An interval $J = (0.1, 1.0)$ locates the same local maximum, because $K \subseteq I$ and $K \subseteq J$ for $K = I$.

By applying our algorithm, we will also see that an interval $I_{\min} = (0.8, 1.4)$ locates the local *minimum* precisely.

Remark. We are also interested in locating local maxima as fast as possible. So, we must give a formal definition of what "fast" means. The actual running time of an algorithm depends on what computer we use, but we can make a good estimate by computing the total number of computational steps (this number is called a *computational complexity* of an algorithm). In the majority of computers, arithmetic operations $(+, -, *, /, \min, \max,$ taking an absolute value) and comparison of two numbers are hardware implemented, and the running time of all these operations is approximately of the same order. Therefore, we can estimate the total running time by just counting the total number of these arithmetic steps in our algorithm. So, we come to the following definition [2]:

Definition 6. *By a computational step of an algorithm we mean an arithmetic operation $(+, -, *, /, \min, \max,$ taking an absolute value) or a comparison of two numbers. By a running time of an algorithm on some data D, we mean the total number of computational steps that are undertaken, when we apply the algorithm to this data. By a length of the data D, or input length, we mean the number of real numbers that constitute the input data. By a complexity of an algorithm for a given input length L, we mean its largest running time on all the inputs of length L. We say that an algorithm is linear-time if its complexity is $O(L)$ (i.e, for some $C > 0$, its complexity is smaller than CL).*

Remark. In our case, the input data include $x_1, \ldots, x_n$, $y_1, \ldots, y_n$, ε, so the input length L is $2n + 1$. Hence, an algorithm is linear-time if and only if its computational complexity is $O(n)$ (i.e., is $\leq Cn$ for some n).

3 Testing whether an approximately given function can be monotonic

Before locating local extrema, it is necessary to know whether the unknown function has any local extrema at all or it is monotonic. In other words, it is necessary to solve the following problem: *given a function interval $\mathcal{F}$, is it possible that a function $f \in \mathcal{F}$ is monotonic?* This problem is sometimes of a separate interest for physical applications. Its solution is also used as a method to accelerate the global optimization algorithms (see, e.g., [15], Section 3.11).

Theorem 1. *There exists a linear-time algorithm that given a function interval $\mathcal{F}$, returns "yes" if and only if this interval contains a monotone non-decreasing function.*

For reader's convenience, all the proofs (that this and other algorithms are really doing what they are supposed to do) are placed in the last section.

Algorithm. *Set $M := y_1$. Then, for $i = 2, \ldots, n$, do the following: check whether $M - 2\varepsilon \leq y_i$, and compute the new value $M := \max(y_i, M)$.*

If for all i, the checked inequality is true, return "yes", else return"no".

Remark. The fact that we cannot check monotonicity faster than in linear time is proved by the following Theorem:

Theorem 2. *Every algorithm that checks whether a given function interval contains a monotone non-decreasing function, requires at least $O(n)$ computational steps.*

Theorem 3. *There exists a linear-time algorithm that given a function interval $\mathcal{F}$, returns "yes" if and only if this interval contains a monotone non-increasing function.*

Algorithm. *Set $m := y_1$. Then, for $i = 2, \ldots, n$, do the following: check whether $m + 2\varepsilon \geq y_i$, and compute the new value $m := \min(y_i, m)$.*

If for all i, the checked inequality is true, return "yes", else return"no".

Theorem 4. *Every algorithm that checks whether a given function interval contains a monotone non-increasing function, requires at least $O(n)$ computational steps.*

4 Main result

Theorem 5. *There exists a linear-time algorithm that locates all local maxima and all local minima precisely.*

Remark. This theorem first appeared in Technical Reports [6] and [7].

There will be a special variable s with possible values -1, 0, and 1. These values have the following meaning:

$s = 0$ means that the data that we have already processed is still consistent with the hypothesis that f is constant; in other words, that some $f \in \mathcal{F}$ is constant. Once this is no longer the case, s only takes the values ± 1.

$s = 1$ means that we are now in an interval on which some $f \in \mathcal{F}$ is monotone non-decreasing;

$s = -1$ means that we are now in an interval on which some $f \in \mathcal{F}$ is monotone non-increasing.

The algorithm itself is as follows:

Algorithm. *First, set $s := 0$, read the first value y_1 and set $m := y_1$ and $M := y_1$. Then, read all the other values y_i, $i = 2, 3, \ldots, n$ one by one and depending on the value of s do the following:*

- *If $s = 0$, then do the following:*
  - *i) Check whether $M - 2\varepsilon \le y_i$ and whether $y_i \le m + 2\varepsilon$.*
  - *ii) If both checked inequalities are true, then leave s unchanged. Else, if the first inequality is false (i.e., $M - 2\varepsilon > y_i$), set $s := -1$. If the second inequality is false (i.e., $y_i > m + 2\varepsilon$), set $s := 1$.*
  - *iii) Update the values M and m: $M := \max(M, y_i)$ and $m := \min(m, y_i)$.*
- *If $s = 1$, then check the inequality $M - 2\varepsilon \le y_i$. If it is true, compute $M := \max(M, y_i)$ and leave s unchanged. If it is false, then do the following:*

i) *for* $j = i-1, i-2, \ldots$ *compare* y_j *with* $M - 2\varepsilon$ *until we find an index* j *for which* $y_j < M - 2\varepsilon$ *(note that* $y_{j+1} \geq M - 2\varepsilon$*);*

ii) *for this* j*, output the interval* (x_j, x_i) *as an interval that locates a local maximum and* $M - \varepsilon$ *as a lower bound on the maximum value;*

iii) *set* $s := -1,\ m := y_i$.

- *If* $s = -1$*, then check the inequality* $m + 2\varepsilon \geq y_i$*. If it is true, compute* $m := \min(m, y_i)$ *and leave* s *unchanged. If this inequality is false, then do the following:*

  i) *for* $j = i-1, i-2, \ldots$ *compare* y_j *with* $m + 2\varepsilon$ *until we find the index* j *for which* $y_j > m + 2\varepsilon$ *(note that* $y_{j+1} \leq m + 2\varepsilon$*);*

  ii) *for this* j*, output the interval* (x_j, x_i) *as an interval that locates a local minimum and* $m + \varepsilon$ *as an upper bound on a minimum value;*

  iii) *set* $s := 1,\ M := y_{i-1}$.

- *When we have processed all the* y_i *and no max or min intervals have been printed, then, depending on the final value of* s*, it means the following:*

  if $s = 0$*, then the function interval* $\mathcal{F}$ *contains a constant function;*

  if $s = 1$*, then the function interval* $\mathcal{F}$ *contains a monotone non-decreasing function* f*;*

  if $s = -1$*, then the function interval* $\mathcal{F}$ *contains a monotone non-increasing function* f.

The following Mathematica program (written by the referee) implements this algorithm:

```
alg[x_,y_]:=
Module[{n,s,m,M,maxlist,minlist,i,j},
   n=Length[y]; s=0;m=y[[1]]; M=y[[1]];
   maxlist={}; minlist={};
   i=2;
```

```
   While[i<=n,
      If[s==0,
         If[M-2 eps<=y[[i]] && y[[i]]<=m+2 eps,{},
            Block[{},If[M-2 eps>y[[i]],s=-1];
         If[m+2 eps<y[[i]],s=1]];
            M=Max[M,y[[i]]];m=Min[m,y[[i]]]],
      If[s==1,
         Block[{},If[M-2 eps<=y[[i]],
            M=Max[M,y[[i]]],
         Block[{},j=i-1;While[y[[j]]>=M-2 eps,j--];
            maxlist=Append[maxlist,{x[[j]],x[[i]]}];
            s=-1;m=y[[i]]]]],
      If[s==-1,
         Block[{},If[m+2 eps>=y[[i]],m=Min[m,y[[i]]],
         Block[{},j=i-1;While[y[[j]]<=m+2 eps,j--];
            minlist=Append[minlist,{x[[j]],x[[i]]}];
            s=1;M=y[[i]]]]]]];
   i++];
Print[maxlist]; Print[minlist];
If[Length[maxlist]==0 && Length[minlist]==0,Print[s]]]
```

Example. For the above example, we start with $s = 0$ and $m = M = y_1 = -0.05$. For $i = 2$, we have $y_2 > m + 2\varepsilon$, therefore, we change the value of s to 1 and update M to $\max(-0.05, y_2) = 0.11$. For $i = 3, 4, 5$, the inequality $y_i \geq M - 2\varepsilon$ is true, so we leave s unchanged ($= 1$) and update M. After we process the value y_5, we will have $M = 0.29$.

For $i = 6$, we have $y_6 = 0.05 < M - 2\varepsilon = 0.19$. Therefore, we compare y_j for $j = 5, 4, \ldots$ with $M - 2\varepsilon = 0.19$ until we find an index $j = 2$ for which $y_2 < 0.19$. So, we return an interval $(x_2, x_6) = (0.2, 1.0)$ as an interval that locates a local maximum.

After that, we set s to -1, and take $m = y_6 = 0.05$. For $i = 7$, we have $y_7 = 0.11 \leq m + 2\varepsilon$, so we leave s unchanged and update m to $\min(0.05, 0.11) = 0.05$. For $i = 8$, we have $y_8 = 0.19 > m + 2\varepsilon = 0.15$. Therefore, for $j = 7, 6, \ldots$, we compare y_j with $m + 2\varepsilon = 0.15$ until we find an index $j = 5$ for which $y_5 > 0.15$. So, we return an interval $(x_5, x_8) = (0.8, 1.4)$ as an interval that locates a local minimum.

Theorem 6. *Every algorithm that locates all local maxima and all local minima precisely, requires at least $O(n)$ computational steps.*

5 Remaining problems

- In some cases, different measurements have *different accuracy*. In these cases, we have different ε_i for different i. How to locate local extrema based on such data?

- In some important real-life applications, we are interested in the local extrema of *functions of several variables*. In these cases, we have multidimensional data, sampled, e.g., over a (hyper-)rectangular lattice. How to use this data to locate local extrema?

6 Proofs

We will first prove Theorems 1–4, and then Theorems 6 and 5.

Proof of Theorem 1. The algorithm described after Theorem 1 takes $1 + 3(n-1) = 3n - 2$ computational steps: 1 multiplication to compute 2ε, $n-1$ subtractions to compute $M - 2\varepsilon$, $n-1$ comparisons between $M - 2\varepsilon$ and y_i, and $n-1$ comparisons between M and y_i. So, this algorithm is linear-time.

Let us prove that this algorithm produces the correct result. In this algorithm, the variable M changes. Let us denote the value of M after we have processed i values $y_1, \ldots, y_i$, by M_i. According to the algorithm, $M_i = \max(y_1, y_2, \ldots, y_i)$. In the algorithm, we compare $M_{i-1} - 2\varepsilon$ with y_i for all i.

Let us first prove that if a function interval $\mathcal{F}$ contains a monotone non-decreasing function f, then this algorithm returns "yes". Indeed, since f is non-decreasing, for $j < i$, we have $f(x_j) \le f(x_i)$. Since $f \in \mathcal{F}$, we have $|f(x_j) - y_j| \le \varepsilon$ (hence $y_j \le f(x_j) + \varepsilon$) and $|f(x_i) - y_i| \le \varepsilon$ (hence $f(x_i) \le y_i + \varepsilon$). Combining these three inequalities, we conclude that

$$y_j \le f(x_j) + \varepsilon \le f(x_i) + \varepsilon \le (y_i + \varepsilon) + \varepsilon = y_i + 2\varepsilon.$$

Therefore, for $j = 1, 2, \ldots, i-1$, we have $y_j \le y_i + 2\varepsilon$. Hence, the biggest of these values y_j (that is equal to M_{i-1}) is also $\le y_i + 2\varepsilon$: $M_{i-1} \le y_i + 2\varepsilon$. Subtracting 2ε from both sides, we conclude that $M_{i-1} - 2\varepsilon \le y_i$, and so our algorithm will return "yes".

Let us now prove that if an algorithm returns "yes", then there is a non-decreasing $f \in \mathcal{F}$. Let us take $f_i = M_i - \varepsilon = \max(y_1, \ldots, y_i) - \varepsilon$ for $i < n$ and $f_n = \max(y_1, \ldots, y_n) + \varepsilon$. Evidently, f_i is a non-decreasing sequence.

Let us show that $|y_i - f_i| \le \varepsilon$ for all i. Indeed, from $y_i \le \max(y_1, \ldots, y_i)$, we conclude that $y_i \le f_i + \varepsilon$, i.e., $y_i - f_i \le \varepsilon$.

On the other hand, since the algorithm returned "yes", we have $M_{i-1} - 2\varepsilon \le y_i$, i.e., $\max(y_1, \ldots, y_{i-1}) \le y_i + 2\varepsilon$. Since $y_i \le y_i + 2\varepsilon$, we can conclude that the biggest of the two numbers $\max(y_1, \ldots, y_{i-1})$ and y_i, i.e., the number $M_i = \max(y_1, \ldots, y_i)$, also does not exceed $y_i + 2\varepsilon$. Subtracting ε from both sides of this inequality $M_i \le y_i + 2\varepsilon$, we conclude that $f_i \le y_i + \varepsilon$, i.e., $f_i - y_i \le \varepsilon$.

Combining these two inequalities, we conclude that $|y_i - f_i| \le \varepsilon$.

Now, we can define the desired f: for $x = x_i$, we take $f(x) = f_i$; and for $x \in (x_i, x_{i+1})$, we define $f(x)$ by linear interpolation:

$$f(x) = f(x_{i-1}) + \big[(x - x_i)/(x_{i+1} - x_i)\big]\big(f(x_{i+1} - f(x_i)\big).$$

Thus defined function $f : [x_1, x_n] \to R$ is non-decreasing, and satisfies the inequalities $|y_i - f(x_i)| \le \varepsilon$.

We can extend this f to a non-decreasing function on the set of all real numbers by taking $f(x) = f_1$ for $x < x_1$ and $f(x) = f_n$ for $x > x_n$. This f belongs to $\mathcal{F}$. Q.E.D.

Comment. This function $f : [x_1, x_n] \to R$ attains its biggest value in only one point: x_n. Likewise, if instead of the values f_i that we have taken, we take different values $\bar{f}_1 = y_1 - \varepsilon$ and $\bar{y}_i = M_i + \varepsilon$ for $i = 2, 3, \ldots, n$, we will get a function that attains its smallest value in only one point: x_1.

Proof of Theorem 2. Let us show that if a function interval $\mathcal{F}$ contains a non-decreasing function f, then any algorithm that returns a correct answer "yes" must process all n values $y_1, y_2, \ldots, y_n$. Indeed, assume that some algorithm $\mathcal{U}$ skips a value y_i. Let us prove that it cannot always give a correct answer. Indeed, for the initial set of values $y_1, \ldots, y_n$ (from $\mathcal{F}$) it must give an answer "yes". Now, if we input the values $\bar{y}_j$ such that $\bar{y}_j = y_j$

for $j \neq i$, and $\bar{y}_i = \min(y_1, \ldots, y_n) - 3\varepsilon$, then, since $\mathcal{U}$ pays no attention to i-th value, it will generate the same answer "yes". However, by applying Theorem 1, we can easily see that the answer should be "no": there is no non-decreasing function in this new function interval.

So, any algorithm must process all n values $y_1, \ldots, y_n$. Each computational step $(+, -$, etc) enables us to process at most two numbers. So, we need at least $n/2$ computational steps. Q.E.D.

Proofs of Theorems 3, 4, 6 are similar to the proofs of Theorems 1, 2.

Proof of Theorem 5.

1) Let us first prove that if this algorithm returns an interval (x_j, x_i) as containing a local maximum, then every continuous function f from $\mathcal{F}$ attains a local maximum at some point from that interval.

Indeed, our algorithm returns such an interval if $y_i < M_{i-1} - 2\varepsilon$ (here, we use the denotation M_i from the proof of Theorem 1) and $y_j < M_{i-1} - 2\varepsilon$.

By definition, M_{i-1} is the biggest of several consequent values y_l. Let us prove that $M_{i-1} = y_l$ for some l between j and i.

Indeed, suppose that it is not so. Then, $y_l < M_{i-1}$ for all $l = j + 1, \ldots, i - 1$. For $l = j$ and $l = i$, we even have $y_l < M_{i-1} - 2\varepsilon$, so also $y_l < M_{i-1}$. According to the algorithm, the only way to increase M_i is to encounter the value y_l that is greater than M_{l-1}; in this case, the new value M_l is equal to y_l. So, if M_l was ever increased on one of the steps from j to i, we would have $M_{i-1} = y_l$ for that step l. Since this is not true, it means that for all these l, the value M_l did not increase. Hence, $M_{i-1} = M_{i-2} = \cdots = M_j = M_{j-1}$. So, $M_{i-1} = M_{j-1}$, and from $y_j < M_{i-1} - 2\varepsilon$, we conclude that $y_j < M_{j-1} - 2\varepsilon$.

But this inequality, according to our algorithm, would mean that we switched to phase $s = -1$ on the value y_j and would not have waited until y_i (as we did). This contradiction shows that our assumption that $M_{i-1} > y_l$ for all $l = j, j + 1, \ldots, i$ is false. Therefore, there exists an l such that $j < l < i$, and $y_l = M_{i-1}$.

Now, let $f \in \mathcal{F}$. This means, in particular, that $f(x_j) \leq y_j + \varepsilon$, so from $y_j < M_{i-1} - 2\varepsilon$, we conclude that $f(x_j) < M_{j-1} - \varepsilon$. Likewise, $f(x_i) < M_{i-1} - \varepsilon$. From $y_l = M_{i-1}$ and $f(x_l) \geq y_l - \varepsilon$, we likewise conclude that $f(x_l) \geq M_{i-1} - \varepsilon$. Therefore, $f(x_j) < M_{i-1} - \varepsilon \leq f(x_l)$, i.e., $f(x_i) < f(x_l)$. Likewise, $f(x_j) < f(x_l)$.

The function f is a continuous function on an interval $[x_j, x_i]$. Therefore, it must attain its biggest value at some point x from this interval. This x cannot coincide with one of the endpoints, because there is a point inside this interval (namely, x_l) for which $f(x_l)$ is bigger than the value in both endpoints. Therefore, this x is inside the interval, and is, therefore, a local maximum.

A similar proof shows that intervals about which the algorithm claims that they locate a local minimum actually locate it.

2) Let us now prove that this algorithm locates all local maxima, and that it locates them precisely. The idea of this proof is as follows: let us write down the sequence of intervals that this algorithm generates. According to an algorithm, in this sequence, after each interval that locates a local maximum, the next one locates a local minimum, and vice versa. Let us choose a point t_k on each of the intervals that locate a local maximum, and a point s_k on each interval that locates a local minimum. We want to design a continuous function whose only local maxima are t_k.

If we succeed in this construction, we will thus prove that our algorithm locates all local maxima, and locates them precisely. Indeed, since this function f has no local maxima outside the intervals generated by our algorithm, it proves that we enumerated all intervals that necessarily contain local maxima. The fact that an arbitrary point t_k from each interval is (thus) a local maximum for some $f \in \mathcal{F}$, proves that our intervals cannot be diminished, i.e., that we have located them precisely.

The construction of this f consists of two steps. First, for those values t_k and s_k that do not coincide with x_i, we add them to our list of x_i. If this new x_i belongs to an interval that locates a local maximum, then as the corresponding value y_i, we take the value of M at the moment when we have formed this interval. Likewise, if this new x_i belongs to an interval that locates a local minimum, then we take $y_i = m$.

After this, the entire interval $[x_1, x_n]$ is divided into subintervals, that either go from s_k to the nearest t_k, or from t_k to the nearest s_k. One can see that on each subinterval $[x_m, x_n]$, where x_m is one of the points s_k, and x_n is one of the points t_k, the algorithm from Theorem 1 will generate "yes". We can, therefore, use the construction from the proof of Theorem 1 to design a non-decreasing function f that satisfies the conditions $|f(x_i) - y_i| \leq \varepsilon$ for all

i. The values of f in the endpoints are $f(x_m) = y_m - \varepsilon$ and $f(x_n) = y_n + \varepsilon$, and f attains its biggest value $y_n + \varepsilon$ only in one point: x_n.

A similar construction applies to a subinterval that starts with the left endpoint x_1 and ends in a point t_1.

For intervals that start with t_k and end with s_k (or with an endpoint), we can similarly construct a non-increasing function, for which the maximum is attained in only one point.

These functions agree on endpoints. Therefore, we can combine them into a single continuous function whose only local maxima are the points t_k.

3) A similar construction proves that our algorithm enumerates all local minima, and enumerates them precisely.

4) If our algorithm generates no intervals at all, and at the end $s = 1$, then, as one can easily see, the algorithm from Theorem 1 will generate a "yes" answer. Therefore, we can use the proof of Theorem 1 and construct a non-decreasing function $f \in \mathcal{F}$ on $[x_1, x_n]$ that attains its biggest value in only one point: x_n. Hence, this f has no local maxima at all.

Likewise, if our algorithm generated no intervals, and at the end $s = -1$, there exists a function $f \in \mathcal{F}$ with no local minima.

5) Let us now prove that $s = 0$ at the end if and only if the function interval $\mathcal{F}$ contains a constant function.

Indeed, if $f(x) = c$ belongs to $\mathcal{F}$, i.e., $|y_i - c| \le \varepsilon$ for all i, then $y_i - y_j \le 2\varepsilon$ for all i and j. From $y_i \le y_1 + 2\varepsilon, \ldots, y_i \le y_{i-1} + 2\varepsilon$, we conclude that $y_i \le m_{i-1} + 2\varepsilon$, where $m_{i-1} = \min(y_1, \ldots, y_{i-1})$. Likewise, for all i, we have $y_i \ge M_{i-1} - 2\varepsilon$. Therefore, according to our algorithm, no intervals will be generated, and the value s will never change from the initial value 0.

Now, let us assume that at the end, $s = 0$. We'll prove that $f(x) = c$, where $c = \max(y_1, \ldots, y_n) - \varepsilon$, belongs to $\mathcal{F}$ (i.e., for every i, $|y_i - c| \le \varepsilon$).

Indeed, from $y_i \le \max(y_1, \ldots, y_n)$, we conclude that $y_i \le c + \varepsilon$. So, it remains to prove that $y_i \ge c - \varepsilon = \max(y_1, \ldots, y_n) - 2\varepsilon$, or that $\max(y_1, \ldots, y_n) \le y_i + 2\varepsilon$. The biggest of n numbers y_j does not exceed $y_i + 2\varepsilon$ if and only if each of them does not exceed it, i.e., $y_j \le y_i + 2\varepsilon$ for all i and j. For $j = i$, it is trivially true.

The fact that $s = 0$ at the end means that both inequalities $y_i \ge M_{i-1} - 2\varepsilon$ and $y_i \le m_{i-1} + 2\varepsilon$ are true for all i. So, for $j < i$, from $y_i \ge M_{i-1} - 2\varepsilon$ we conclude that $y_i \ge \max(y_1, \ldots, y_j, \ldots, y_{i-1}) - 2\varepsilon \ge y_j - 2\varepsilon$, and $y_j \le y_i + 2\varepsilon$.

For $j > i$, from $y_j \leq m_{j-1} + 2\varepsilon$, we conclude that $y_j \leq \min(y_1, \ldots, y_i, \ldots, y_{j-1}) + 2\varepsilon \leq y_i + 2\varepsilon$. So, $y_j \leq y_i + 2\varepsilon$ is true for all i and j. This completes the proof of 5).

6) Our algorithm is linear-time, because it spends finitely many computational steps on each of n values y_i: finitely many when we move forward, and maybe 1 when we move backwards to find an interval that locates a local maximum or minimum. Q.E.D.

References

[1] Bloomfield, P. *Fourier analysis of time series: an introduction*. Wiley, N.Y., 1976.

[2] Blum, L., Shub, M., and Smale, S. *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines*. Bull. Amer. Math. Soc. **21** (1) (1989), pp. 1–46.

[3] Dennis, J. E. and Schnabel, R. B. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, Englewood Cliffs, 1983.

[4] Good, I. J. and Gaskins, R. A. *Density estimation and bump-hunting by the penalized likelihood method exemplified by scattering and meteorite data*. J. Amer. Stat. Soc. **75** (1980), pp. 42–56.

[5] Kolacz, H. *On the optimality of inclusion algorithms*. In: Nickel, K. (ed.) "Interval Mathematics 1985", Lecture Notes in Computer Science, Vol. 212, Springer-Verlag, Berlin, Heidelberg, N.Y., 1986, pp. 67–79.

[6] Kreinovich, V. et al. *Theoretical foundations of estimating precision of software results in intellectual systems for control and measurement*. Soviet National Institute for Electro-Measuring Instruments, Technical Report No. 7550-8170-40, Leningrad, 1988 (in Russian).

[7] Kreinovich, V. and Villaverde, K. *Towards modal interval analysis: how to compute maxima and minima of a function from approximate measurement results*. University of Texas at El Paso, Computer Science Department, Technical Report UTEP–CS–91–9.

[8] Martin, R. D. and Thompson, D. J. *Robust-resistant spectrum estimation*. Proceedings IEEE **70** (1982), pp. 1097–1115.

[9] Moore, R. E. *Mathematical elements of scientific computing*. Holt, Rinehart and Winston, N.Y., 1975.

[10] Moore, R. E. *Methods and applications of interval analysis*. SIAM, Philadelphia, 1979.

[11] Moore, R. E. *Global optimization to prescribed accuracy*. Computers and Mathematical Applications **21** (6/7) (1991), pp. 25–39.

[12] Nickel, K. *Interval acceleration of convergence*. In: Moore, R. E. (ed.) "Reliability in Computing", Academic Press, N.Y., 1988, pp. 151–169.

[13] Rall, L. B. *Optimization of interval computations*. In: Nickel, K. (ed.) "Interval Mathematics 1980", Academic Press, N.Y., 1980, pp. 489–498.

[14] Ratschek, H. and Rokne, J. *Optimality of the centered form*. In: Nickel, K. (ed.) "Interval Mathematics 1980", Academic Press, N.Y., 1980, pp. 499–508.

[15] Ratschek, H. and Rokne, J. *New computer methods for global optimization*. Ellis Horwood, Chicester, 1988.

[16] Silverman, B. W. *On a test for multimodality based on kernel density estimates*. Technical Summary Report No. 21–81, U. S. Army Math. Research Center, Madison, WI, 1981.

[17] Thompson, D. J. *Spectrum estimation techniques for characterization and development of WT4 waveguide*. Bell Syst. Techn. J. **56** (Nov. 1977), p. 1769 (part 1); **57** (Dec. 1977), p. 1983 (part 2).

[18] Thompson, D. J. *Spectrum estimation and harmonic analysis*. Proceedings IEEE **70** (1982), pp. 1055–1096.

Computer Science Department
The University of Texas at El Paso
El Paso, TX 79968
USA