Computational Complexity of Interval Algebraic Problems: Some Are Feasible And Some Are Computationally Intractable: A Survey

Vladik Kreinovich, Anatoly Lakeyev, and Jiří Rohn

0 Introduction

One of the Basic Problems of Interval Computations

One of the basic problems of interval computations is as follows: *given* a function $f(x_1, ..., x_n)$ of n real variables, and n intervals $\mathbf{x}_i$, *compute* the range

$$\mathbf{y} = f(\mathbf{x}_1, ..., \mathbf{x}_n) = \{f(x_1, ..., x_n) \,|\, x_1 \in \mathbf{x}_1, ..., x_n \in \mathbf{x}_n\}.$$

Usually, these intervals $\mathbf{x}_i$ have rational endpoints.

A typical application of this problem is as follows: from the measurements, we know the approximate values $\tilde{x}_i$ of physical quantities x_i, and we know the guaranteed accuracy Δ_i of each measurement. As a result, we know that x_i belongs to the interval $\mathbf{x}_i = [\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i]$. We also know the algorithm f that transforms the values x_i into the value of the desired quantity y. We want to know the set of possible values of y. For a continuous function f, this set is an interval (we will denote it by $\mathbf{y}$). So, the question is: *can we compute endpoints $y^{\pm}$ of this interval $\mathbf{y}$ in reasonable time?*

Let Us Describe This Problem in Precise Terms

To describe this problem in precise terms, first, we must *explain* what we mean by "*computing the endpoints*". If the endpoints are rational numbers, then we want to compute the explicit expressions for these points; if they are not rational numbers, then we may be interested, e.g., in computing $\varepsilon-$close rational approximations to the desired endpoints (for a given $\varepsilon > 0$).

Second, we must *fix the class of functions f*. If the function f itself is difficult to compute, then it is difficult to compute the endpoints of the interval $\mathbf{y}$ even for degenerate input intervals $\mathbf{x}_i = [x_i, x_i]$. To avoid this situation, in this paper, we will restrict

ourselves to the simplest possible functions: functions that can be obtained by finitely many applications of arithmetic operations $+$, $-$, $*$, and $/$, i.e., to *rational* functions.

For Rational Functions, There is a General Algorithm, but This Algorithm Is Not Practical

For rational functions, the problem of computing the endpoints is, in principle, algorithmically solvable: namely, we can apply the so-called Tarski's algorithm [20]. However, this algorithm takes too long [1]: it sometimes take time $\approx 2^{2^n}$ for an input of size n. As a result, even for small n, it may take billions of years. This is not a practical solution.

So, the question is: *is there a practically useful (feasible) algorithm to solve our basic problem?*

Let Us Define "Feasible"

To describe this question in precise terms, we must formalize what "feasible" means. This problem has been studied in theoretical computer science; no completely satisfactory definition has yet been proposed. The best known formalization is: an algorithm $\mathcal{U}$ is feasible iff it is *polynomial time*, i.e., iff there exists a polynomial P such that for every input x, the running time $t_{\mathcal{U}}(x)$ of the algorithm $\mathcal{U}$ on the input x is bounded by $P(|x|)$ (here, $|x|$ denotes the length of the input x). This definition is not perfect, because there are algorithms that are polynomial time but that require billions of years to compute (we will see an example below), and there are algorithms that require in a few cases exponential time but that are, in general, very practical. However, this is the best definition we have so far.

For many mathematical problems, it is not yet known (1995) whether these problems can be solved in polynomial time or not. However, it is known that some combinatorial problems are as tough as possible, in the sense that if we can solve this problem in polynomial time, then, crudely speaking, we can solve all combinatorial problems in polynomial time. Such problems are called *NP-hard*, and the majority of computer scientists believe that these problems are not feasible. For that reason, NP-hard problems are also called *intractable*. For formal definitions and detailed descriptions, see, e.g., [4].

In these terms, we can reformulate our question as follows: *is the basic problem of interval computations* (described above) *feasible or intractable?*

1 Polynomials: First Negative Result, and Where We Go From There

First Negative Result

In 1981, Gaganov proved that our problem is intractable even if we consider polynomials only:

Theorem. (Gaganov) [2, 3] *The basic problem of interval computations is NP-hard even if we restrict ourselves to polynomial functions f, and to such inputs for which the output interval $\mathbf{y}$ has rational endpoints.*

Three Restrictions Behind This Result

This result means that there is no hope to find a feasible algorithm that computes the interval $\mathbf{y}$ (1) *exactly*, (2) *for all polynomials* f, and (3) *for all input intervals* $[x_i^-, x_i^+]$. What if we relax some of these restrictions?

First Try

Let us first relax the first restriction, and ask for an algorithm that computes the endpoint with a given accuracy ε (i.e., that computes the values $\tilde{y}^{\pm}$ for which $|y^+ - \tilde{y}^+| \le \varepsilon$ and $|y^- - \tilde{y}^-| \le \varepsilon$). Alas, the resulting problem is still NP-hard:

Theorem 1. *For every $\varepsilon > 0$, the problem of computing $\varepsilon-$approximations $\tilde{y}^{\pm}$ to the endpoints $y^{\pm}$ of the range $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ for a given polynomial f and for given intervals $[x_i^-, x_i^+]$ is NP-hard.*

Comment. In this survey, we only give the formulations of the results. The proofs of these and related results will appear in our forthcoming book [12]. This book will also give a more comprehensive bibliography of the subject.

Second Try

Computing the range for narrow intervals only is still NP-hard:

Theorem 2. *For every $\delta > 0$, the problem of computing the endpoints $y^{\pm}$ of the range $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ for a given polynomial f and for given intervals $[x_i^-, x_i^+]$ of width $\le \delta$ is NP-hard.*

Third Try

Relaxing the third restriction means that we compute the range not for all possible polynomials, but only for some ("simplest") ones. What does "simple" mean? A polynomial is simple if, first, it has few variables and, second, if its degree is small. Let us try both restrictions.

Polynomials With a Fixed Number of Variables: Good News

Theorem 3. *For every n, there exists a polynomial time algorithm that computes the endpoints of the range $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ for any polynomial f with n variables.*

This result follows from an algorithm proposed by Grigor'ev et al [5]. Here, each endpoint can be proved to be an algebraic number (i.e., a root r of a polynomial $P(y)$ with integer coefficients), so this algorithm actually computes the coefficients of the corresponding polynomial $P(y)$. ¿From these coefficients, for a given $\varepsilon > 0$, we can compute the $\varepsilon-$approximation to the root very fast. This result is true not only for rational functions, but also for *algebraic* functions (i.e., roots of polynomials equations with polynomial coefficients; e.g., $f(x) = \sqrt{x^2+1}$ is a solution of the equation $y^2 - (x^2+1) = 0$).

Polynomials With a Fixed Number of Variables: Bad News

As shown in [6], Grigor'ev algorithm is an example of an algorithm that is polynomial time but that is not in any way practically feasible: for polynomials of only four variables, it can take millions of years to compute, even on the fastest computers.

So, restricting the number of variables does not help much. Let us try our luck with restricting the degree.

Polynomials of Fixed Degree

When degree is 1, we get linear functions. For linear functions f, the problem is clearly feasible: if $f(x_1, ..., x_n) = a_0 + \sum a_i x_i$, then $y^{\pm} = \tilde{y} \pm \Delta$, where $\tilde{y} = f(\tilde{x}_1, ..., \tilde{x}_n)$ and $\Delta = \sum |a_i| \Delta_i$.

The next step is quadratic functions. Alas, for quadratic functions, the problem is already NP-hard (Rohn, 1994):

Theorem 4. *The problem of computing the endpoints $y^{\pm}$ of the range $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ for a given quadratic polynomial $f(x_1, ..., x_n) = a_0 + \sum a_i x_i + \sum a_{ij} x_i x_j$ and for given intervals $[x_i^-, x_i^+]$ is NP-hard.*

We cannot have a feasible algorithm for all quadratic polynomials, but maybe, we can have an algorithm for some of them? There are two cases when this problem is feasible

for quadratic functions f: one case is when f is actually linear, and another is when the matrix $\|a_{ij}\|$ is diagonal:

Theorem 5. *There exists a polynomial time algorithm that computes the endpoints $y^{\pm}$ of the range $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ for a given quadratic polynomial $f(x_1, ..., x_n) = a_0 + \sum a_i x_i + \sum a_{ii} x_i^2$ and for given intervals $[x_i^-, x_i^+]$.*

Both cases are very specific, but maybe, if we take close cases, we will still get a class for which feasible algorithms are known? Alas, the answer is negative: for all non-trivial generalizations of the above degenerate classes that we tried, the resulting problem is NP-hard:

1) A diagonal matrix is a matrix a_{ij} for which $a_{ij} = 0$ for $i \neq j$, i.e., for $|i - j| \geq 1$. The first natural generalization of a diagonal matrix is a $w-$*band* matrix, for which $a_{ij} = 0$ for $|i - j| \geq w$ for some $w > 1$.

Theorem 6. *For $w \geq 3$, the problem of computing the endpoints $y^{\pm}$ of the range $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ for a given quadratic polynomial $f(x_1, ..., x_n) = a_0 + \sum a_i x_i + \sum a_{ij} x_i x_j$ with a $w-$band matrix a_{ij} and for given intervals $[x_i^-, x_i^+]$ is NP-hard.*

2) Another possibility is to generalize one particular case of a diagonal matrix: a *scalar* matrix, for which $a_{ii} = \text{const}$. These matrices can be characterized by the condition that all their eigenvalues λ_i are equal: $\lambda_1 = ... = \lambda_n = \lambda$ for some number λ. A natural generalization is the notion of an *almost scalar* matrix, i.e., a matrix for which all but one eigenvalues coincide.

Theorem 7. *The problem of computing the endpoints $y^{\pm}$ of the range $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ for a given quadratic polynomial $f(x_1, ..., x_n) = a_0 + \sum a_i x_i + \sum a_{ij} x_i x_j$ with an almost scalar matrix a_{ij} and for given intervals $[x_i^-, x_i^+]$ is NP-hard.*

3) The third generalization is a generalization of linear functions. A natural generalization is the notion of a *bilinear function*, i.e., a quadratic function for which $a_{ii} = 0$ for all i.

Theorem 8. *The problem of computing the endpoints $y^{\pm}$ of the range $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ for a given bilinear polynomial $f(x_1, ..., x_n) = a_0 + \sum a_i x_i + \sum a_{ij} x_i x_j$ and for given intervals $[x_i^-, x_i^+]$ is NP-hard.*

Comment. Basically, the problem is feasible only for linear and slightly non-linear functions. As soon as we add one more multiplication and consider quadratic functions, the problem becomes NP-hard. At first glance, it may seem like we are doomed to NP-hardness results: only linear functions lead to feasible problems. However, it turns out that if instead of multiplication we add division (seemingly more complicated operation), and consider *fractionally linear* functions, we get a feasible algorithm!

2 Fractionally Linear Functions: The Problem is Feasible

Theorem 9. [14] *There exists a polynomial time algorithm that computes the endpoints* $y^{\pm}$ *of the range* $f([x_1^-, x_1^+], ..., [x_n^-, x_n^+])$ *for a given fractionally linear function*

$$f(x_1, ..., x_n) = \frac{a_0 + \sum a_i x_i}{b_0 + \sum b_i x_i}$$

and for given intervals $[x_i^-, x_i^+]$.

Comment. The corresponding algorithm is very practical; it is actually used in control applications. It is therefore natural to try to generalize it. Alas, all non-trivial generalizations that we have tried so far lead to NP-hard problems:

3 Linear Interval Systems: A Natural Generalization of Fractionally Linear Functions

Motivations of the Following Definition

A fractionally linear function $f(x_1, ..., x_n)$ can be defined as a solution of a linear equation $(b_0 + \sum b_i x_i)f = a_0 + \sum a_i x_i$. The problem of finding the range of f is thus equivalent to finding the set of all possible solutions of this equation when x_i take the values in their respective intervals.

A natural generalization is, therefore, the solution of a *system* of linear equations $\sum a_{ij} f_j = b_i$, where the coefficients a_{ij} and b_i are linear functions of the variables that are defined with interval uncertainty. It turns out that this problem is NP-hard even in the simplest case when each of the coefficients a_{ij}, b_i depends on only one interval-valued variable, and therefore, each of these coefficients can take any value from its corresponding interval irrespective of the values of the other coefficients.

As a result, we arrive at the following problem:

Formulation of the Problem

Given the intervals $\mathbf{a}_{ij}$ and $\mathbf{b}_i$ and an integer j, *compute* the endpoints of the range of possible values of f_j (i.e., values for which for some values $f_1, ..., f_{j-1}, f_{j+1}, ..., f_n$, we have $\sum a_{ij} f_j = b_i$ for some $a_{ij} \in \mathbf{a}_{ij}$ and $b_i \in \mathbf{b}_i$).

Alas, this problem is NP-hard.

A Little Bit of History

- There exist many algorithms for solving interval linear equations; the algorithms that compute the precise range sometimes take an exponentially long time. This fact lead to the suspicion that the problem of computing the exact range is not feasible.

- The first NP-hardness result for this problem was proved by Lakeyev et al in 1993 [10]: the problem of computing the bounds *exactly* for arbitrary *rectangular* (not necessarily square) matrices is NP-hard.

- Later in 1993, for arbitrary *rectangular* matrices, it was shown that the problem of estimating the range with a given *accuracy* $\varepsilon > 0$ is also NP-hard [11].

- In 1994, Rohn et al. have proved [19] that the problem of computing the bounds *exactly* is NP-hard even for *square regular* matrices (regular means that every matrix $a_{ij} \in \mathbf{a}_{ij}$ is regular).

- On hearing about these two results, A. Neumaier suggested that the problem of computing the bounds with a given *accuracy* is NP-hard even for *square regular* matrices. This hypothesis was proven correct in 1995 [18, 9].

- Finally, it has been recently proved that for every $\varepsilon > 0$ and $\delta > 0$, the problem of computing the bounds with an *accuracy* ε for *square regular* interval matrices made of intervals of width $\leq \delta$ is also NP-hard (Kahl, 1995, unpublished; the proof will appear in [12]).

Comment. The situation is not so gloomy as it may seem:

- For a *different* notion of a solution to a linear interval equation, there is a polynomial time algorithm that solves it: namely, if we consider f_j a *solution* iff $\sum a_{ij} f_j \in \mathbf{b}_i$ for *all* $a_{ij} \in \mathbf{a}_{ij}$. This type of a solution has a direct application to *economics*: if $\mathbf{b}_i$ is the desired consumption of $i-$th item, and a_{ij} describes how much item of $j-$th type is used to produce $i-$th product, then f_j is the production that guarantees the given consumption levels (for details, see, e.g., Rohn [15, 16, 17]).

- And even for the cases when the problem is NP-hard, this NP-hardness has a bright side in it:

4 Bright Side of NP-Hardness

NP-Hard Means That Good Interval Heuristics Can Solve Other Hard Problems

By definition, the fact that a problem $\mathcal{P}$ is NP-hard means, as we have mentioned, that if we can solve the problem $\mathcal{P}$ is polynomial time, then we will be able to solve many other hard problems in polynomial time. Based on this reduction, the majority of computer scientists believe that there is algorithm that solves *all* instances of the problem $\mathcal{P}$. But this does not prevent us from having good heuristics that solve *many* instances of $\mathcal{P}$. The reduction mentioned above means that we can then solve many cases of other hard problems.

For interval computations, many good heuristics are indeed known. In Traylor et al. [21], it is shown that these heuristics lead to good heuristic algorithms for solving another NP-hard problem: propositional satisfiability problem.

Freedom of Will

According to traditional physical formalisms, if we know the initial state of the world, then we can uniquely determine the state of the world at any future moment of time. This is not an abstract possibility: physical equations usually lead to reasonably efficient predictions.

This *determinism* flies in the face of *freedom of will*, i.e., of the fact that we humans feel ourselves capable of making free decisions that are not uniquely pre-determined.

At first glance, there seems to be a contradiction, and philosophers have been viewing it as a one. However, this contradiction between physics and freedom of will almost disappears if we take into consideration that we never know the initial state of the world precisely: we get this state from measurements, and measurements are never absolutely precise. So, instead of knowing the *exact* values of the parameters of the initial state, we only know *intervals* of possible values of these parameters. For intervals, as we have seen, the problem of computing the exact range is NP-hard, and therefore, it is not possible to predict the future state. This impossibility leaves room for freedom of will.

5 What Does NP-Hardness Mean in Practical Terms? Are Interval Computations Really Intractable?

NP-Hardness Reformulated in Practical Terms

Ideally, in the basic problem considered above, we would like to compute the exact range interval of the function $y = f(x_1, ..., x_n)$. Traditional methods of interval computations do not always give the exact range, but they usually give an *enclosure* to the desired range, i.e., an interval that *contains* the range $\mathbf{y}$. The result that computing the exact range is NP-hard means, crudely speaking, that there is no feasible (polynomial time) algorithm that can always compute the exact range. In other words, *every enclosure-computing feasible algorithm sometimes overestimates.*

The natural question is: how often is that "sometimes"?

Two Possibilities: Pessimistic and Optimistic

There are two possible scenarios here:

- a *pessimistic* one: many particular cases are hard, so, every feasible algorithm overestimates in almost all cases;
- an *optimistic* one: a few cases are hard, but the vast majority are feasible.

Our Result

Th existign algorithms almost always overestimate, so, we may be inclined to conclude that we are in pessimistic situation. However, it turns out that we are in the optimistic situation: for small input intervals, *almost all interval computation problems are feasible.* This result was proven in [13]. To formulate it in precise terms, we need the following definition:

Definition.

- *Let us assume that $\varepsilon > 0$ is a real number, $D \subseteq R^n$ is a compact domain with a positive Lebesque measure $\mu(D) > 0$, and $P(x)$ is a property that is true for some points $x \in D$. We say that P is true for $(D, \varepsilon)-$almost all x if $\mu(\{x \in D \,|\, \neg P(x)\}) \leq \varepsilon \cdot \mu(D)$.*
- *We say that an algorithm $\mathcal{U}$ for solving the basic problem is almost always exact for narrow input intervals if for every compact domain D with $\mu(D) > 0$, for every rational function f with rational coefficients that if finite on an open set*

$N \supset D$, and for every $\varepsilon > 0$, there exists a $\delta > 0$ such that for $(D, \varepsilon)-$almost all $\tilde{x}_1, ..., \tilde{x}_n$, if all n input intervals are $\delta-$narrow ($\Delta_i \leq \delta$), the algorithm $\mathcal{U}$ returns the exact endpoints of the interval $f(\mathbf{x}_1, ..., \mathbf{x}_n)$.

Theorem 10. *There exists a polynomial-time algorithm $\mathcal{U}$ that, given n intervals $\mathbf{x}_i = [\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i]$, and a rational function f with rational coefficients, returns an enclosure for $f(\mathbf{x}_1, ..., \mathbf{x}_n)$, and that is almost always exact for narrow intervals.*

Comment. A similar result shows that solving linear interval equations is almost always easy for narrow interval inputs (for details, see [13]).

6 Other Problems in Which Interval Computations are Used, And Their Computational Complexity

Optimization

The simplest optimization problem has the form $f(x_1, ..., x_n) \to \max$ for $x_i \in [x_i^-, x_i^+]$. This is an example of a range problem, so, from the fact that computing the range is NP-hard, we can easily deduce that the *optimization problem is NP-hard* for quadratic, bilinear, etc. functions f.

Unbounded Optimization

Unbounded optimization often occurs in theoretical physics, where even equations are often formulated in terms of a variational principle of the type $S = \int L\, d^4x \to \min$. The complexity of unbounded optimization problem $f(x_1, ..., x_n) \to \min$ is different from the complexity of bounded optimization:

Theorem 11.

- *Bounded optimization is feasible for linear functions f and NP-hard for quadratic and higher degree polynomials f.*
- *Unbounded optimization is feasible for linear, quadratic, and cubic polynomials, and is NP-hard for polynomials of degree ≥ 4.*

Solving Equations

If we were able to check in polynomial time where a given polynomial equation $f(x_1, ..., x_n) = c$ is solvable, then, we would be able, by using bisection, to approximate the endpoints of the range of f in polynomial time. Since approixmating the

endpoints of the range is NP-hard, *checking whether an equation is solvable is* thus also *NP-hard*.

If we do not restrict the range of possible solutions, then we can prove not only that this problem is NP-hard, but that it actually requires exponential time: e.g., for a system $x_1 = 2$, $x_2 = x_1^2$, $x_3 = x_2^2$, ..., $x_n = x_{n-1}^2$, the only possible solution is $x_n = x_1^{2^n} = 2^{2^n}$. In binary terms, it means 1 followed by 2^n zeros. We need exponential time just to write this answer down, so *solving systems of polynomials equations is intractable.*

7 Ellipsoid Uncertainty

One Reason Why the Problem is Hard for Interval Computations

For a smooth function $f(x)$ of one variable x, it is usually easy to find a maximum on a given interval $[a, b]$: this maximum is attained either inside the interval, in which case it is a zero of the derivative $f'(x) = 0$, or at one of the endpoints (a or b). A similar result is true if we look for a maximum of a function f of n variables inside a box $\mathbf{x}_1 \times ...\mathbf{x}_n$: this maximum is either inside the box, or on one of its faces. The only problem now is that we have 2^n possible faces, so this approach leads to an exponentially long (thus, non-feasible) algorithm.

Uncertainty is Often Described by an Ellipsoid, and for Ellipsoids, This Argument Does not Seem to Work

In many cases, there is a dependency between the variables x_i. As a result, the set of possible values of $x = (x_1, ..., x_n)$ is described not by a box, but, e.g., by an *ellipsoid* E. An ellipsoid does not have many different faces, so, it may seem at first glance that for ellipsoids, the basic problem is feasible. Alas, no.

Theorem 12. [8] *The problem of computing the endpoints of the range $f(E)$ of a given polynomial function $f(x_1, ..., x_n)$ for a given ellipsoid $E \subseteq R^n$ is NP-hard.*

8 Statistical and Interval Computations

In many real-life cases, in addition to *intervals* of possible values of x_i, we know the *probabilities* of different values of x_i. Most frequently, the "measurement errors" $\Delta x_i = \tilde{x}_i - x_i$ are independent Gaussian random variables with 0 average and known standard deviations σ_i. In this case, the natural problem is: to compute the standard deviation $\sigma[y]$ of $y = f(x_1, ..., x_n) = f(\tilde{x}_1 - \Delta x_1, ..., \tilde{x}_n - \Delta x_n)$. We will call this

problem the *basic problem of statistical computations.* It turns out that this problem is computationally easier than the basic problem of interval computations [7]:

Theorem 13.

- *For linear functions f, both interval and statistical computation problems are feasible.*
- *For polynomial functions f, interval computations are NP-hard, and statistical computations are feasible.*
- *For rational functions f, both interval and statistical computations are NP-hard.*

Corollary. Interval computations are harder than statistical ones.

Acknowledgments. This work was partially supported by NSF Grants No. CDA-9015006 and EEC-9322370, NASA Grant No. NAG 9-757, grant GACR 201/95/1484, and by a grant from the German Science Foundation. The authors are greatly thankful to Gerhard Heindl, Hoon Hong, R. Baker Kearfott, Werner Krandick, Wolfram Luther, Arnold Neumaier, Sergey Shary, Yuri Shokin, Jürgen Wolff von Gudenberg, and to all participants of SCAN'95 for the attention to this work and for valuable discussions.

References

[1] J. H. Davenport, J. Heintz, "Real quantifier elimination is doubly exponential", *Journal of Symbolic Computations*, 1988, Vol. 5, No. 1/2, pp. 29–35.

[2] A. A. Gaganov, *Computational complexity of the range of the polynomial in several variables*, Leningrad University, Math. Department, M.S. Thesis, 1981 (in Russian).

[3] A. A. Gaganov, "Computational complexity of the range of the polynomial in several variables", *Cybernetics*, 1985, pp. 418–421.

[4] M. E. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[5] D. Yu. Grigor'ev and N. N. Vorobjov (Jr.), "Solving systems of polynomial inequalities in subexponential time", *Journal of Symbolic Computation*, 1988, Vol. 5, No. 1/2, pp. 37–64.

[6] J. Heintz, M.-F. Roy, and P. Solerno, "On the theoretical and practical complexity of the existential theory of reals", *The Computer Journal*, 1993, Vol. 36, No. 5.

[7] O. Kosheleva and V. Kreinovich, *Interval computations are harder than statistical computations*, University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-94-19, July 1994; available by ftp from `cs.utep.edu`,

login `anonymous`, file `pub/reports/tr94-19.tex`; submitted to *Reliable Computing*.

[8] V. Kreinovich, *Ellipsoid computations are intractable even for polynomials*, University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-93-53a, August 1993 (updated December 1993); available by ftp from `cs.utep.edu`, login `anonymous`, file `pub/reports/tr93-53a.tex`.

[9] V. Kreinovich and A. V. Lakeyev, *Linear Interval Equations: Computing Enclosures with Bounded Relative Or Absolute Overestimation is NP-Hard*, University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-95-27, July 1995; available by ftp from `cs.utep.edu`, login `anonymous`, file `pub/reports/tr95-27.tex`; submitted to *Reliable Computing*.

[10] V. Kreinovich, A. V. Lakeyev, and S. I. Noskov. "Optimal solution of interval linear systems is intractable (NP-hard)." *Interval Computations,* 1993, No. 1, pp. 6–14.

[11] V. Kreinovich, A. V. Lakeyev, S. I. Noskov, "Approximate linear algebra is intractable", *Lin. Alg. Appls.*, 1995 (to appear).

[12] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Feasible? Intractable? On computational complexity of data processing and interval computations*, Kluwer Academic Press, 1996 (to appear).

[13] A. V. Lakeyev and V. Kreinovich, "If Input Intervals Are Small Enough, Then Interval Computations Are Almost Always Easy", *Reliable Computing*, 1995, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), pp. 134–139.

[14] R. Lea, V. Kreinovich, and R. Trejo, "Optimal interval enclosures for fractionally-linear functions, and their application to intelligent control", *Reliable Computing*, 1996, Vol. 2, No. 2 (to appear).

[15] J. Rohn, *Input-output planning with inexact data*, Freiburg Intervall Berichte, 1978, Vol. 9.

[16] J. Rohn, *Correction of coefficients of the input-output model*, Z. Angew. Math. Mech., 1978, Vol. 58, pp. T494–T495.

[17] J. Rohn, *Input-output model with interval data*, Econometrica, 1980, Vol. 48, No. 3, pp. 767–769.

[18] J. Rohn, "Linear Interval Equations: Computing Enclosures with Bounded Relative Overestimation is NP-Hard", In: R. B. Kearfott and V. Kreinovich (eds.), *Applications of Interval Computations*, Kluwer, Boston, MA, 1996 (to appear).

[19] J. Rohn and V. Kreinovich, "Computing exact componentwise bounds on solutions of linear systems with interval data is NP-hard," *SIAM Journal on Matrix Analysis and Applications (SIMAX),* 1995, Vol. 16, pp. 415–420.

[20] A. Tarski, *A decision method for elementary algebra and geometry*, 2nd ed., Berkeley and Los Angeles, 1951.

[21] B. Traylor and V. Kreinovich, "A bright side of NP-hardness of interval computations: interval heuristics applied to NP-problems", *Reliable Computing,* 1995, Vol. 1, No. 3, pp. 343–360.

Addresses:

V. Kreinovich, Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968, USA, E-mail `vladik@cs.utep.edu`.

A. V. Lakeyev, Irkutsk Computing Center, Russian Academy of Science, Siberian Branch, Lermontov Str. 134, Irkutsk 664033, Russia, E-mail `lakeyev@icc.ccsoan.irkutsk.su`.

J. Rohn, Faculty of Mathematics and Physics, Charles University, Malostranské nám. 25, 118 00 Prague, Czech Republic, and Institute of Computer Science, Academy of Sciences, Pod vodárenskou věží 2, 182 07 Prague, Czech Republic, E-mail `rohn@uivt.cas.cz`.