Fast Quantum Algorithms for Handling Probabilistic and Interval Uncertainty

Vladik Kreinovich* and **Luc Longpré**

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

Received 1 December 2003

Key words Data processing, interval computations, quantum computing, partial information about probabilities
MSC (2000) 65G20, 65G30, 65G40, 68Q17, 68Q25

Dedicated to Klaus Weihrauch's 60th Birthday.

In many real-life situations, we are interested in the value of a physical quantity y that is difficult or impossible to measure directly. To estimate y, we find some easier-to-measure quantities $x_1, \ldots, x_n$ which are related to y by a known relation $y = f(x_1, \ldots, x_n)$. Measurements are never 100% accurate; hence, the measured values $\widetilde{x}_i$ are different from x_i, and the resulting estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is different from the desired value $y = f(x_1, \ldots, x_n)$. How different can it be?

Traditional engineering approach to error estimation in data processing assumes that we know the probabilities of different measurement errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$. In many practical situations, we only know the upper bound Δ_i for this error; hence, after the measurement, the only information that we have about x_i is that it belongs to the interval $\mathbf{x}_i \stackrel{\text{def}}{=} [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In this case, it is important to find the range $\mathbf{y}$ of all possible values of $y = f(x_1, \ldots, x_n)$ when $x_i \in \mathbf{x}_i$. We start the paper with a brief overview of the computational complexity of the corresponding *interval computation* problems.

Most of the related problems turn out to be, in general, at least NP-hard. In this paper, we show how the use of quantum computing can speed up some computations related to interval and probabilistic uncertainty. We end the paper with speculations on whether (and how) "hypothetic" physical devices can compute NP-hard problems faster than in exponential time.

Most of the paper's results were first presented at NAFIPS'2003 [30].

1 Introduction: Data Processing – From Computing via Probabilities to Intervals

Why data processing? In many real-life situations, we are interested in the value of a physical quantity y that is difficult or impossible to measure directly. Examples of such quantities are the distance to a star and the amount of oil in a given well. Since we cannot measure y directly, a natural idea is to measure y *indirectly*. Specifically, we find some easier-to-measure quantities $x_1, \ldots, x_n$ which are related to y by a known relation $y = f(x_1, \ldots, x_n)$; this relation may be a simple functional transformation, or complex algorithm (e.g., for the amount of oil, numerical solution to an inverse problem). Then, to estimate y, we first measure the values of the quantities $x_1, \ldots, x_n$, and then we use the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of these measurements to compute an estimate $\widetilde{y}$ for y as $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

For example, to find the resistance R, we measure current I and voltage V, and then use the known relation $R = V/I$ to estimate resistance as $\widetilde{R} = \widetilde{V}/\widetilde{I}$.

Computing an estimate for y based on the results of direct measurements is called *data processing*; data processing is the main reason why computers were invented in the first place, and data processing is still one of the main uses of computers as number crunching devices.

Comment. In this paper, for simplicity, we consider the case when the relation between x_i and y is known exactly; in some practical situations, we only know an approximate relation between x_i and y.

* Corresponding author: e-mail: vladik@cs.utep.edu, Phone: 915 747 6951, Fax: 915 747 5030

Why interval computations? From computing via probabilities to intervals. Measurement are never 100% accurate, so in reality, the actual value x_i of i-th measured quantity can differ from the measurement result $\widetilde{x}_i$. Because of these *measurement errors* $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of data processing is, in general, different from the actual value $y = f(x_1, \ldots, x_n)$ of the desired quantity y [41].

It is desirable to describe the error $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$ of the result of data processing. To do that, we must have some information about the errors of direct measurements.

What do we know about the errors Δx_i of direct measurements? First, the manufacturer of the measuring instrument must supply us with an upper bound Δ_i on the measurement error. (If no such upper bound is supplied, this means that no accuracy is guaranteed, and the corresponding "measuring instrument" is practically useless.) Thus, once we performed a measurement and got a measurement result $\widetilde{x}_i$, we know that the actual (unknown) value x_i of the measured quantity belongs to the interval $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$, where $\underline{x}_i = \widetilde{x}_i - \Delta_i$ and $\overline{x}_i = \widetilde{x}_i + \Delta_i$.

In many practical situations, we not only know the interval $[-\Delta_i, \Delta_i]$ of possible values of the measurement error; we also know the probability of different values Δx_i within this interval. This knowledge underlies the traditional engineering approach to estimating the error of indirect measurement, in which we assume that we know the probability distributions for measurement errors Δx_i.

In practice, we can determine the desired probabilities of different values of Δx_i by comparing the results of measuring with this instrument with the results of measuring the same quantity by a standard (much more accurate) measuring instrument. Since the standard measuring instrument is much more accurate than the one use, the difference between these two measurement results is practically equal to the measurement error; thus, the empirical distribution of this difference is close to the desired probability distribution for measurement error. There are two cases, however, when this determination is not done:

- First is the case of cutting-edge measurements, e.g., measurements in fundamental science. When a Hubble telescope detects the light from a distant galaxy, there is no "standard" (much more accurate) telescope floating nearby that we can use to calibrate the Hubble: the Hubble telescope is the best we have.

- The second case is the case of measurements on the shop floor. In this case, in principle, every sensor can be thoroughly calibrated, but sensor calibration is so costly – usually costing ten times more than the sensor itself – that manufacturers rarely do it.

In both cases, we have no information about the probabilities of Δx_i; the only information we have is the upper bound on the measurement error.

In this case, after we performed a measurement and got a measurement result $\widetilde{x}_i$, the only information that we have about the actual value x_i of the measured quantity is that it belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In such situations, the only information that we have about the (unknown) actual value of $y = f(x_1, \ldots, x_n)$ is that y belongs to the range $\mathbf{y} = [\underline{y}, \overline{y}] = \{f(x_1, \ldots, x_n) \mid x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$ of the function f over the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$. The process of computing this interval range based on the input intervals $\mathbf{x}_i$ is called *interval computations*; see, e.g., [18, 19, 20, 32].

Interval computations techniques. Historically the first method for computing the enclosure for the range is the method which is sometimes called "straightforward" interval computations. This method is based on the fact that inside the computer, every algorithm consists of elementary operations (arithmetic operations, $\min$, $\max$, etc.). For each elementary operation $f(a, b)$, if we know the intervals $\mathbf{a}$ and $\mathbf{b}$ for a and b, we can compute the exact range $f(\mathbf{a}, \mathbf{b})$. The corresponding formulas form the so-called *interval arithmetic*. For example,

$$[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]; \quad [\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\underline{a} - \overline{b}, \overline{a} - \underline{b}];$$

$$[\underline{a}, \overline{a}] \cdot [\underline{b}, \overline{b}] = [\min(\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b}), \max(\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b})].$$

In straightforward interval computations, we repeat the computations forming the program f step-by-step, replacing each operation with real numbers by the corresponding operation of interval arithmetic. It is known that, as a result, we get an enclosure $\mathbf{Y} \supseteq \mathbf{y}$ for the desired range.

In some cases, this enclosure is exact. In more complex cases, the enclosure has excess width.

There exist more sophisticated techniques for producing a narrower enclosure, e.g., a centered form method. However, for each of these techniques, there are cases when we get an excess width. The reason for this is that, as shown in [24, 44], the problem of computing the exact range is known to be NP-hard even for polynomial functions $f(x_1, \ldots, x_n)$ (actually, even for quadratic functions f).

2 Fast Quantum Algorithms for Handling Probabilistic and Interval Uncertainty

As computers become faster, quantum effects must be more and more taken into consideration. According to Moore's law, computer speed doubles every 18 months. One of the main limitations to further speedup is the computer size: every communication is limited by the speed of light c, so, e.g., a computer of a 1 ft size is bounded to have a computation speed 1 ft/c – which corresponds to 1 GHz. To make faster computers, we must thus decrease the size of computer elements. As this size reaches molecular size, we must take into consideration quantum effects.

Quantum effects add to noise, but they can also help. Quantum effects, with their inevitably probabilistic behavior, add to noise. However, it turns out that some quantum effects can be used to drastically speed up computations (in spite of quantum noise).

For example, without using quantum effects, we need – in the worst case – at least N computational steps to search for a desired element in an unsorted list of size N. A quantum computing algorithm proposed by Grover (see, e.g., [12, 13, 36]) can find this element much faster – in $O(\sqrt{N})$ time.

Several other quantum algorithms have been proposed.

What we are planning to do. How can this be of use to interval data processing community? In many application areas ranging from geosciences to bioinformatics to large-scale simulations of complex systems, data processing algorithms require a lot of time to run even with the exact input data. As a result, very little is currently done to analyze the effect of inevitable uncertainty of input data on the results of data processing.

It is desirable to analyze how different types of uncertainty – probabilistic, interval – influence the results of data processing. In this paper, we discuss how quantum algorithms such as Grover's quantum search can be used to speed up this analysis – and thus, make it possible.

We also explain that there is no need to wait until a full-blown quantum computer appears, with all necessary quantum bits ("qubits"): even without all necessary qubits, we can still get some speedup, a speedup that gets better and better as we add more qubits to the quantum computer.

Grover's algorithm for quantum search. We have already mentioned Grover's algorithm that, given a database $a_1, \ldots, a_N$ with N entries, a property P of database entries (i.e., an algorithm that checks whether P is true), and an allowable error probability δ, returns, with probability $\geq 1 - \delta$, either the element a_i that satisfies the property P or the message that there is no such element in the database.

This algorithm requires $C \cdot \sqrt{N}$ steps (= calls to P), where the factor C depends on δ (the smaller δ we want, the larger C we must take).

General comment about quantum algorithms. For our applications, it is important to know that for Grover's algorithm (and for all the other quantum algorithms that we will describe and use), the entries a_i do not need to be all physically given, it is sufficient to have a procedure that, given i, produces a_i.

If all the entries are physically given, then this procedure simply consists of fetching the i-th entry from the database. However, it is quite possible that the entries are given implicitly, e.g., a_i can be given as the value of a known function at i-th grid point; we have this function given as a program, so, when we need a_i, we apply this function to i-th grid point.

Algorithm for quantum counting. Brassard et al. used the ideas behind Grover's algorithm to produce a new quantum algorithm for *quantum counting*; see, e.g., [3, 36]. Their algorithm, given a database $a_1, \ldots, a_N$ with N entries, a property P of database entries (i.e., an algorithm that checks whether P is true), and an allowable error probability δ, returns an approximation $\widetilde{t}$ to the total number t of entries a_i that satisfy the property P.

This algorithm contains a parameter M that determines how accurate the estimates are. The accuracy of this estimate is characterized by the inequality $\left|\widetilde{t} - t\right| \leq \frac{2\pi}{M} \cdot \sqrt{t} + \frac{\pi^2}{M^2}$ that is true with probability $\geq 1 - \delta$.

This algorithm requires $C \cdot M \cdot \sqrt{N}$ steps (= calls to P), where the factor C depends on δ (the smaller δ we want, the larger C we must take).

In particular, to get the exact value t, we must attain accuracy $\left|\widetilde{t} - t\right| \le 1$, for which we need $M \approx \sqrt{t}$. In this case, the algorithm requires $O(\sqrt{t \cdot N})$ steps.

Quantum algorithms for finding the minimum. Dürr et al. used Grover's algorithm to produce a new quantum algorithm for *minimization*; see, e.g., [4, 36]. Their algorithm applied to the database whose entries belong to the set with a defined order (e.g., are numbers). This algorithm, given a database $a_1, \ldots, a_N$ with N entries, and an allowable error probability δ, returns the index i of the smallest entry a_i, with probability of error $\le \delta$.

This algorithm requires $C \cdot \sqrt{N}$ steps (= calls to P), where the factor C depends on δ (the smaller δ we want, the larger C we must take).

Main idea behind quantum computing of the minimum. The main idea behind the above algorithm can be illustrated on the example when all the entries a_i are integers. The algorithm requires that we know, e.g., a number M such that all the entries belong to the interval $[-M, M]$. For every value m between $-M$ and M, we can use Grover's algorithm to check whether there is an entry a_i for which $a_i < m$.

If such an entry exists, then $m_0 \stackrel{\text{def}}{=} \min(a_i) < m$; otherwise $m_0 \ge m$. Thus, for every m, we can check, in $O(\sqrt{N})$ steps, whether $m_0 < m$.

We can therefore apply bisection to narrow down the interval containing the desired until it narrows down to a single integer.

- We start with an interval $[\underline{M}, \overline{M}] = [-M, M]$.
- At each iteration, we pick a midpoint $M_0 = \dfrac{\underline{M} + \overline{M}}{2}$, and check whether $m_0 < M_0$.

  If $m_0 < M_0$, this means that $m_0 \in [\underline{M}, M_0]$; otherwise, $m_0 \in [M_0, \overline{M}]$. In both cases, we get a half-size interval containing m_0.
- After $\log_2(2M)$ iterations, this interval becomes so narrow that it can only contain one integer – which is m_0.

Thus, in $\log_2(M) \cdot O(\sqrt{N})$ steps, we can compute the desired minimum.

Quantum algorithm for computing the mean. The above algorithms can be used to compute the average of several numbers, and, in general, the mean of a given random variable. The first such algorithm was proposed by Grover in [14]; for further developments, see, e.g., [16, 35, 37].

The traditional Monte-Carlo method for computing the mean consists of picking M random values and averaging them. It is a well known fact [41, 45], that the accuracy of this method is $\sim 1/\sqrt{M}$, so, to achieve the given accuracy ε, we need $M \approx \varepsilon^{-2}$ iterations. Another way to compute the average of n given numbers is to add them up and divide by n, which requires n steps.

Thus, when $n < \varepsilon^{-2}$, it is faster to add all the values; otherwise, it is better to use the Monte-Carlo method.

Grover's quantum analog of the Monte-Carlo method attains accuracy $\sim 1/M$ after M iterations; thus, for a given accuracy ε, we only need $M \approx \varepsilon^{-1}$ steps.

Similarly to the traditional Monte-Carlo methods, this quantum algorithm can compute multi-dimensional integrals $\int \ldots \int f(x_1, \ldots, x_n)\, dx_1 \ldots dx_n$: indeed, if we assume that the vector $(x_1, \ldots, x_n)$ is uniformly distributed over the corresponding domain, then this integral is proportional to the average value of $f(x_1, \ldots, x_n)$. The advantage of this method is that it is faster: namely, it requires only $M \approx \varepsilon^{-1}$ steps in contrast to the traditional Monte-Carlo methods that require $M \approx \varepsilon^{-2}$ steps.

Quantum algorithms for probabilistic analysis. In the probabilistic case, the problem of describing the influence of the input uncertainty on the result of data processing takes the following form (see, e.g., [41, 45]). Given:

- the data processing algorithm $f(x_1, \ldots, x_n)$ that transforms any n input values $x_1, \ldots, x_n$ into the result of $y = f(x_1, \ldots, x_n)$ of data processing, and
- the mean values $\widetilde{x}_i$ and standard deviations σ_i of the inputs,

compute the standard deviation σ of the result y of data processing.

This standard deviation can be described as a mean (= mathematical expectation) of the square $(y - \widetilde{y})^2$, where $y \stackrel{\text{def}}{=} f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$, $y \stackrel{\text{def}}{=} f(x_1, \ldots, x_n)$, and each x_i is normally distributed with mean $\widetilde{x}_i$ and standard deviation σ_i. The traditional Monte-Carlo algorithm requires $\sim 1/\varepsilon^2$ iterations to compute this average; thus, for accuracy 20%, we need 25 iterations; see, e.g., [43].

The quantum Monte-Carlo algorithm computes this mean with accuracy ε in $\sim 1/\varepsilon$ iterations (i.e., calls to f); so, for accuracy 20%, we only need 5 iterations. Since computing f may take a long time, this drastic (5 times) speed-up may be essential.

Quantum algorithms for interval computations: problem. In interval computations (see, e.g., [18, 19, 32]), the main objective is as follows. Given:

- intervals $[\underline{x}_i, \overline{x}_i]$ of possible values of the inputs $x_1, \ldots, x_n$, and
- the data processing algorithm $f(x_1, \ldots, x_n)$ that transforms any n input values $x_1, \ldots, x_n$ into the result of $y = f(x_1, \ldots, x_n)$ of data processing,

compute the exact range $[\underline{y}, \overline{y}]$ of possible values of y.

We can describe each interval in a more traditional form $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$, where $\widetilde{x}_i$ is the interval's midpoint, and Δ_i is its half-width. The resulting range can also be described as $[\widetilde{y} - \Delta, \widetilde{y} + \Delta]$, where $\widetilde{y}$ is the result of data processing (described by the above formula), and Δ is the desired largest possible difference $|\widetilde{y} - y|$.

Quantum algorithms for interval computations: case of relatively small errors. When the input errors are relatively small, we can linearize the function f around the midpoints $\widetilde{x}_i$. In this case, Cauchy distributions turn out to be useful, with probability density

$$\rho(x) \sim \frac{1}{1 + \dfrac{(x-a)^2}{\Delta^2}}.$$

It is known [43] that if we take x_i distributed according to Cauchy distribution with a center $a = \widetilde{x}_i$ and the width parameter Δ_i, then the difference $\widetilde{y} - y$ between the corresponding quantities $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ and $y = f(x_1, \ldots, x_n)$ is also Cauchy distributed, with the width parameter equal to the desired value Δ.

The reason for this fact is as follows. We can expand the difference

$$\Delta y = \widetilde{y} - y = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1 - \Delta_1, \ldots, \widetilde{x}_n - \Delta x_n)$$

in Taylor series in Δx_i When the measurement errors $\Delta x_i = \widetilde{x}_i - x_i$ are small, we can ignore quadratic and higher order terms in this expansion, and arrive at the expression $\Delta y = \sum\limits_{i=1}^n f_i \cdot \Delta x_i$, where $f_i \stackrel{\text{def}}{=} \dfrac{\partial f(x_1, \ldots, x_n)}{\partial x_i}_{|x_i = \widetilde{x}_i}$. We want to find the maximum and the minimum of this expression when $\Delta x_i \in [-\Delta_i, \Delta_i]$. The sum $\sum\limits_{i=1}^n f_i \cdot \Delta x_i$ attains its largest possible value when each of n terms attains its largest value. For each i, if $f_i \geq 0$, then the term $f_i \cdot \Delta x_i$ is increasing, hence its maximum is attained when Δx_i takes the largest value Δ_i; then, $f_i \cdot \Delta x_i = f_i \cdot \Delta_i$. If $f_i < 0$, then the term $f_i \cdot \Delta x_i$ is decreasing, hence its maximum is attained when Δx_i takes the smallest value $-\Delta_i$; then, $f_i \cdot \Delta x_i = -f_i \cdot \Delta_i$. In both cases, the largest value of i-th term is $|f_i| \cdot \Delta_i$, and hence the largest possible value of Δy is $\sum\limits_{i=1}^n |f_i| \cdot \Delta_i$. Similarly, the smallest possible value of Δy is $-\sum\limits_{i=1}^n |f_i| \cdot \Delta_i$.

It is known that if n independent random variables ξ_i are Cauchy distributed with parameters Δ_i, then their linear combination $\sum\limits_{i=1}^n f_i \cdot \xi_i$ is also Cauchy distributed, with parameter $\Delta = \sum\limits_{i=1}^n |f_i| \cdot \Delta_i$. So, the above simulation technique indeed leads to the desired estimate for Δ.

The use of Cauchy distribution speeds up computations – from n calls to f (that are needed for a deterministic method) to a constant number of calls to f (that are needed to compute Δ with given accuracy when using Cauchy distribution technique). For example, to guarantee the standard deviation of 20%, we need ≈ 50 calls to f, which for $n = 10^3$ is 20 times faster than the deterministic method.

For Cauchy distribution, the standard deviation is infinite, so we cannot literally apply the quantum computing idea that worked in the case of probabilistic analysis. However, if we apply a function $g(x)$ (e.g., $\arctan$) that reduces the entire real line to an interval, then the expected value of $g\left((\widetilde{y}-y)^2\right)$ – that depends only on Δ – can be computed by the quantum Monte-Carlo algorithm; from this value, we can reconstruct Δ.

In this case, quantum techniques also speed up computations. Indeed, the quantum Monte-Carlo algorithm computes Δ with accuracy ε in $\sim 1/\varepsilon$ iterations (i.e., calls to f), while the non-quantum Cauchy technique requires a much larger number of $\sim 1/\varepsilon^2$ iterations. In particular, for accuracy 20%, we get a 5 times speed-up. Since computing f may take a long time, this drastic speed-up may be essential.

Quantum algorithms for interval computations: general case. Known results about the computational complexity of interval computations (see, e.g., [24]) state that in the general case, when the input errors are not necessarily small and the function f may be complex, this problem is NP-hard. This, crudely speaking, means that in the worst case, we cannot find the exact range for y faster than by using some version of exhaustive search of all appropriate grid points. The problem is not in exactness: it is also known that the problem of computing the range with a given approximation accuracy ε is also NP-hard.

How can we actually compute this range? We can find, e.g., $\underline{y}$ with a given accuracy δ as follows. The function f is continuous; hence, for a given ε, there exists an δ such that the $\le \delta$-difference in x_i leads to $\le \varepsilon$ change in y. Thus, within a given accuracy ε, it is sufficient to consider a grid with step δ, and take the smallest of all the values of f on this grid as $\underline{y}$.

If the linear size of the domain is D, then, in this grid, we have D/δ values for each of the variables, hence, the total of $(D/\delta)^n$ points.

In non-quantum computations, to compute the minimum, we need to check every point from this grid, so we must use $N = (D/\delta)^n$ calls to f. The quantum algorithm for computing minimum enables us to use only $\sqrt{N} = (D/\delta)^{n/2}$ calls.

Thus, quantum algorithms can double the dimension of the problem for which we are able to compute the desired uncertainty.

Quantum algorithms for the case when we have several different types of uncertainty. How can we extend the above results to the case when we have several different types of uncertainty? In this section, we present preliminary results about the case when we have both probabilistic and interval uncertainty.

When we have n measurement results $x_1, \ldots, x_n$, traditional statistical approach usually starts with computing their population average $E = \dfrac{x_1 + \ldots + x_n}{n}$ and their population variance

$$V = \frac{(x_1 - E)^2 + \ldots + (x_n - E)^2}{n}$$

(or, equivalently, the population standard deviation $\sigma = \sqrt{V}$); see, e.g., [41]. If we know the exact values of x_i, then these formulas require linear computation time $O(n)$.

As we have mentioned, in many practical situations, we only have intervals $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ of possible values of x_i. As a result, the sets of possible values of E and V are also intervals.

The function E is monotonic in each x_i, so the range $[\underline{E}, \overline{E}]$ for E can be easily computed:

$$\underline{E} = \frac{\underline{x}_1 + \ldots + \underline{x}_n}{n}; \ \ \overline{E} = \frac{\overline{x}_1 + \ldots + \overline{x}_n}{n}.$$

In [5, 6], we have shown that the problem of computing the range $[\underline{V}, \overline{V}]$ is, in general, NP-hard (even when we are interested in computing this range with a given accuracy); we have also described a quadratic-time $O(n^2)$ algorithm $\underline{A}$ for computing $\underline{V}$ and a a quadratic-time algorithm $\overline{A}$ that computes $\overline{V}$ for all the cases in which, for some integer C, no more than C "narrowed" intervals $[\widetilde{x}_i - \Delta_i/n, \widetilde{x}_i + \Delta_i/n]$ can have a common intersection.

Let us first show that by using Monte-Carlo simulations, we can compute $\underline{V}$ with given accuracy in time $O(n \cdot \log_2(n)) \ll O(n^2)$; to be more precise, we need time $O(n \cdot \log_2(n))$ time to sort $2n$ values and then $O(n)$ steps to complete the computations.

Indeed, the algorithm $\underline{A}$ from [5, 6] is as follows:

- First, we sort all $2n$ values $\underline{x}_i, \overline{x}_i$ into a sequence $x_{(1)} \le x_{(2)} \le \ldots \le x_{(2n)}$.

- Second, we compute $\underline{E}$ and $\overline{E}$ and select all zones $[x_{(k)}, x_{(k+1)}]$ that intersect with $[\underline{E}, \overline{E}]$.
- For each of the selected small zones $[x_{(k)}, x_{(k+1)}]$, we compute the ratio $r_k = S_k/N_k$, where

$$S_k \stackrel{\text{def}}{=} \sum_{i:\underline{x}_i \geq x_{(k+1)}} \underline{x}_i + \sum_{j:\overline{x}_j \leq x_{(k)}} \overline{x}_j,$$

and N_k is the total number of such is and js. If $r_k \in [x_{(k)}, x_{(k+1)}]$, then we compute V'_k as

$$\frac{1}{n} \cdot \left(\sum_{i:\underline{x}_i \geq x_{(k+1)}} (\underline{x}_i - r_k)^2 + \sum_{j:\overline{x}_j \leq x_{(k)}} (\overline{x}_j - r_k)^2 \right).$$

If $N_k = 0$, we take $V'_k \stackrel{\text{def}}{=} 0$.

- Finally, we return the smallest of the values V'_k as $\underline{V}$.

For each k, the value r_k is a mean, so, by using Monte-Carlo methods, we can compute it in time that does not depend on n at all; similarly, we can compute V'_k in constant time. The only remaining step is to compute the smallest of $\leq 2n$ values V'_k; this requires $O(n)$ steps.

If quantum computing is available, then we can compute the minimum in $O(\sqrt{n})$ steps; thus, we only need $O(\sqrt{n})$ steps after sorting.

Similarly, the algorithm $\overline{\mathcal{A}}$ is as follows:

- First, we sort all $2n$ endpoints of the narrowed intervals $\widetilde{x}_i - \Delta_i/n$ and $\widetilde{x}_i + \Delta_i/n$ into a sequence $x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(2n)}$. This enables us to divide the real line into $2n+1$ zones $[x_{(k)}, x_{(k+1)}]$, where we denoted $x_{(0)} \stackrel{\text{def}}{=} -\infty$ and $x_{(2n+1)} \stackrel{\text{def}}{=} +\infty$.
- Second, we compute $\underline{E}$ and $\overline{E}$ and select all zones $[x_{(k)}, x_{(k+1)}]$ that intersect with $[\underline{E}, \overline{E}]$.
- For each of remaining zones $[x_{(k)}, x_{(k+1)}]$, for each i from 1 to n, we pick the following value of x_i:
  - if $x_{(k+1)} < \widetilde{x}_i - \Delta_i/n$, then we pick $x_i = \overline{x}_i$;
  - if $x_{(k)} > \widetilde{x}_i + \Delta_i/n$, then we pick $x_i = \underline{x}_i$;
  - for all other i, we consider both possible values $x_i = \overline{x}_i$ and $x_i = \underline{x}_i$.

  As a result, we get one or several sequences of x_i. For each of these sequences, we check whether the average E of the selected values $x_1, \ldots, x_n$ is indeed within the corresponding zone, and if it is, we compute the population variance V by using the definition of V.
- Finally, we return the largest of the computed population variances as $\overline{V}$.

It is shown that we end up with $\leq 2^C \cdot 2n = O(n)$ population variances.

Here also, computing E and V can be done in constant time, and selecting the largest of $O(n)$ variances requires linear time $O(n)$ for non-quantum computations and $O(\sqrt{n})$ time for quantum computing.

Can quantum computers still be useful when there are not yet enough qubits? In view of the great potential for computation speedup, engineers and physicists are actively working on the design of actual quantum computers. There already exist working prototypes: e.g., a several mile long communication system, with simple quantum computers used for encoding and decoding, is at government disposal. Microsoft and IBM actively work on designing quantum computers. However, at present, these computers can only solve trivial instances of the above problems, instances that have already been efficiently solved by non-quantum computers. The main reason why quantum computers are not currently used to solve more complex problems is that the existing quantum computers have only a few qubits, while known quantum algorithms require a lot of qubits. For example, Grover's algorithm requires a register with $q = \log(n)$ qubits for a search in a database of n elements.

Of course, while we only have 2 or 3 or 4 qubits, we cannot do much. However, due to the active research and development in quantum computer hardware, we will (hopefully) have computers with larger number of qubits reasonably soon.

A natural question is: while we are still waiting for the qubit register size that is necessary to implement the existing quantum computing algorithms (and thus, to achieve the theoretically possible speedup), can we somehow utilize the registers of smaller size to achieve a partial speed up?

In this section, we start answering this question by showing the following: for quantum search, even when we do not have enough qubits, we can still get a partial speedup; for details, see [29]. The fact that we do get a partial speedup for quantum search makes us hope that even when we do not have all the qubits, we can still get a partial speedup for other quantum computing algorithms as well.

Let us assume that we are interested in searching in an unsorted database of n elements, and that instead of all $\log(N)$ qubits that are necessary for Grover's algorithm, we only have, say 90% or 50% of them. To be more precise, we only have a register consisting of $r = \alpha \cdot \log(N)$ qubits, where $0 < \alpha < 1$. How can we use this register to speed up the search?

Grover's algorithm enables us to use a register with r qubits to search in a database of $M = 2^r$ elements in time $C \cdot \sqrt{M}$. For our available register, $r = \alpha \cdot \log(N)$, hence $M = 2^r = N^\alpha$, so we can use Grover's algorithm with this qubit register to search in a database of size N^α in time $C \cdot \sqrt{M} = C \cdot N^{\alpha/2}$.

To search in the original database of size N, we can do the following:

- divide this original database into $N^{1-\alpha}$ pieces of size N^α; and then
- consequently apply Grover's algorithm with a given qubit register to look for the desired element in each piece.

Searching each piece requires $C \cdot N^{\alpha/2}$ steps, so the sequential search in all $N^{1-\alpha}$ pieces requires time $N^{1-\alpha} \cdot (C \cdot N^{\alpha/2}) = C \cdot N^{1-\alpha/2}$. Since $\alpha > 0$, we get a speedup.

When α tends to 0, the computation time tends to $C \cdot N$, i.e., to the time of non-quantum search; when α tends to 1, the computation time tends to $C \cdot N^{1/2}$, i.e., to the time of quantum search.

3 Does "NP-Hard" Really Mean "Intractable"?

Introduction. Most of the computational problems related to interval computations are, in general, NP-hard. Most computer scientists believe that NP-hard problem are really computationally intractable. This belief is well justified for traditional computers, but there are non-traditional physical and engineering ideas that may make NP-hard problem easily solvable. Let us briefly overview these ideas.

Within Newtonian physics, NP-hardness does seem to mean "intractable". The standard definitions of the notions of NP-hardness are based on the abstract models of computation such as Turing machines or RAM machines. Real-life computers are often much more complex than these simplified models. A natural question is: can the additional features present in the real-life computers speed up computations?

Most existing computers are based on the processes well described by the traditional Newtonian physics. For such computers, researchers have tried several different schemes, but they have not been able to come up with schemes that solve NP-hard problem is guaranteed reasonable time. Moreover, it turns out that whatever computation speed we can achieve by using these features, we can achieve approximately the same computation speed by using simplified computers as well.

This empirical observation was later theoretically justified: specifically, it has been proven that is we only use processes from Newtonian physics, then we do not add any extra computational ability to the computational devices; for exact formulations and proofs, see, e.g., Gandy [8, 9].

The only feature of real-life computations that is not present in the standard (simplified) models from theoretical computer science is *randomness*, i.e., the ability to input *truly random* data and use them in computations. This ability has, however, been well analyzed in theoretical computer science. In the language of theory of computation, the outside source of data is called an oracle. As early as 1981, Bennet *et al.* have shown [2] that if we allow a random sequence as an oracle, and correspondingly reformulate the definitions of the classes P and NP, then we can prove that the correspondingly reformulated classes are different [2]. In other words, if we allow randomness but restrict ourselves to processes based on Newtonian physics, then NP-hard problems cannot be solved in feasible (polynomial) time, i.e., NP-hard problems are really intractable.

What if we use non-traditional physical and engineering ideas in computer design? Since we seem not to be able to avoid the unrealistic exponential time with traditional, Newtonian-physics-based computers, a question naturally appears: what if in the future, we will find *non-Newtonian* processes; will then NP-hard problems still be intractable? This question was first formulated by G. Kreisel [26].

Traditional computers use discrete-oriented deterministic processes in normal space and time. In reality, physical processes are (1) continuous, (2) non-deterministic (as described by quantum mechanics), and (3) they occur in non-traditional (curved) space-time. So, to describe how using additional physical processes will help in computations, we must consider how these three factors (adding non-determinism and taking curvature into consideration) change our computational abilities.

Non-Newtonian processes of first type: Use of physical fields. In physical terms, computational devices (Turing machines, computers, etc.) can be described as follows: at some initial moment of time t_0, we set up the initial state of the device; we set up the design of the computational device in such a way the dynamics of this device follows the desired pattern, and then, at some moment of time t, we extract the result of the computations from the state of the computational device at this moment t.

Traditional computational devices consist of discrete cells, and the dynamics describes how the states of these cells change with time. According to modern physics, the world contains not only discrete objects and processes, it also contains continuous processes. Crudely speaking, according to modern physics, the world consists of particles and fields. (Crudely speaking, because, according to quantum mechanics, every particle can also be viewed as a field, and every field can also be viewed as an infinite collection of particles – e.g., an electromagnetic field is a collection of photons.) What can we achieve if we use such continuous processes (fields) in computation?

The state of a field at a given moment of time t_0 is characterized by the values $f(\vec{x}, t_0)$ of this field at different points $\vec{x}$. Setting up an initial state is usually possible: if someone provides us with a computable function $f(\vec{x}, t_0)$ and a given accuracy $\varepsilon > 0$, then we are usually able to implement a state $\widetilde{f}(\vec{x}, t_0)$ that is ε-close to $f(\vec{x}, t_0)$, i.e., e.g., for which $|\widetilde{f}(\vec{x}, t_0) - f(\vec{x}, t_0)| \le \varepsilon$. The dynamics of a physical field $f(\vec{x}, t)$ is usually described by a partial differential equation (PDE). So, the crucial question is: if the original state $f(\vec{x}, t_0)$ is computable, and if it changes according to a physical PDE, will the resulting state $f(\vec{x}, t)$ still be computable?

For many physical PDEs, the value $f(\vec{x}, t)$ of the field f in a future moment of time t can be expressed by an explicit integral formula in which the integral right-hand side depends on the current state of this field $f(\vec{x}, 0)$. The corresponding integral operator is usually computable, in the sense that if the original state $f(\vec{x}, t_0)$ is computable, then the resulting state $f(\vec{x}, t)$ is computable as well; relevant theorems are proved, e.g., in Pour-El *et al.* [40].

For some physical PDEs, however, $f(\vec{x}, t)$ can be described as an integral depending both on value of the function $f(\vec{x}, t_0)$ and on the values of its spatial derivatives. So, if we start with a function $f(\vec{x}, t_0)$ that is computable, but whose (spatial) derivatives are not computable, we may end up with a non-computable value $f(\vec{x}, t)$. This was shown by Pour-El *et al.* in [39] (see also Beeson [1], Ch. 15, and Pour-El *et al.* [40]). This result generalizes a theorem proved by Aberth in 1971 and rediscovered in Pour-El *et al.* [38].

This result is very interesting and encouraging, but this result does not necessarily mean that we have found a way to compute a function that is not computable on a normal computer (and hopefully, that we can solve some NP-hard problems in reasonable time). Indeed, to be able to do that, we would need to find a way to implement the initial conditions with a non-computable derivative and it is not clear how we can do that; for a detailed discussion, see, e.g., [27, 46].

In other words, it is not clear whether the use of continuous physical processes can help in solving NP-hard problems. A more definite possibility of solving NP-hard problem fast comes from the other two aspects of physical processes: non-deterministic processes and processes in non-traditional (curved) space-time.

Non-Newtonian processes of second type: Quantum processes (adding non-determinism). We have already mentioned that quantum computers can solve several useful problems – like factoring integers – faster than non-quantum ones; see, e.g., [36]. It may be possible to use them for solving NP-hard problems. Several such "hypothetic" schemes – using quantum field theory – have been proposed by Freedman, Kitaev, etc.

We propose to use one more phenomenon: namely, some potentially observable dependencies between physical quantities become not only non-smooth but even discontinuous; these cases have been summarized in a recent monograph [11]. From the computational viewpoint, this seems to open the doors to the possibility of checking whether a given real number is equal to 0 or not, something that the halting problem explicitly prohibits us from doing for normal (non-quantum) computing. We are therefore planning to investigate the possibility of using this new opportunity in actual computing. Our preliminary results are presented in [15].

Non-Newtonian processes of third type: Using curved space-time for computations. If we allow heavily curved space (e.g., semi-closed black holes, we can get the results faster if we stay in the area where the curvature is strong and time goes slower, and let the computations be done outside (see, e.g., Morgenstein *et al.* [23, 34]); then, we will even be able to compute NP-hard problems in polynomial time.

A similar speed-up can be obtained if we assume that our space-time is hyperbolic [17, 23, 34].

Previously described non-Newtonian processes relate to well-recognized physics, but there are other physical theories that describe possible but not universally accepted physics. Namely, several physical theories have led to the appearance of closed timelike curves, the possibility to go back in time. Suffice it to say that one of the main ideas which helped R. Feynman to develop a modern version of quantum electrodynamics was the idea of positrons as electrons going back in time [7]. Until the late 1980s, these possibilities were largely dismissed by mainstream physics as mathematical artifacts which cannot lead to actual going back in time. Only when Kip Thorne, the world's leading astrophysicist, published several papers on acausal solutions in *Physical Reviews*, the topic became more mainstream.

Reasonable solutions with causal anomalies were discovered in many physical theories. For example, in general relativity, the curved space-time generated by a sufficiently massive cylinder that rotates sufficiently fast contains a closed timelike curve (causal anomaly). In string theory, a theory that describes elementary particles as non-point objects ("strings"), seemingly interactions between such particles sometime lead to the possibility to influence the past (i.e., to a causal anomaly). It was also shown that modern cosmological theories, in which the current cosmological expansion is preceded by a short period of exponentially fast growth ("inflation"), also lead to the possibility of a causal anomaly. An interested interested can find the detailed description of these causal anomalies, e.g., in book [42] and in the papers referenced in this book.

The main obstacle to accepting acausal phenomena used to be paradoxes associated with time travel. These paradoxes can be illustrated on a commonsense example of a "father paradox": a time traveler can go to the past and kill his father before he himself was conceived, thus creating a paradox. The accepted solution to the acausal paradoxes can also be illustrated on the "father paradox" example: since the time traveler was born, this means that some unexpected event prevented him from killing his father. Maybe a policeman stopped him, maybe his gun malfunctioned. Even if the time traveler takes care of all such probably events, there are always events with small probability – like a meteor falling on the traveler's head – which cannot all be avoided. Thus, all we will achieve if we try to implement a paradox is that some event with a very low probability will occur.

There are several ways to use acausal processes in computing. The trivial way is to let a computer run a long program for whatever it takes and then send the results back in time. A less trivial way of saving time is similar to quantum "computing without computing" – speeding up without actually using time machines. This method was originally proposed in [22]; see also [21, 31, 33]. This method is related to the above solution to the father paradox. Indeed, to solve, e.g., a propositional satisfiability problem with n variables, we generate n random bits and check whether they satisfy a given formula; if not, we launch a time machine that is set up to implement a low-probability event (with some probability $p_0 \ll 1$). Nature has two choices: either it generates n variables which satisfy the given formula (probability of this is 2^{-n}), or the time machine is used which leads to an event with a probability p_0. If $2^{-n} \gg p_0$, then statistically, the first event is much more probable, and so, the solution to the satisfiability problem will actually be generated without the actual use of a time machine.

(On the other hand, maybe the other outcome is that the universe will just unravel and all existence will cease ...)

Yet another possibility: Gell-Mann's approach to complex systems. In several papers and in his book [10], a Nobelist physicist M. Gell-Mann suggests that our difficulties in describing the dynamics of complex systems can be resolved if we assume that the true physical equations actually explicitly contain Kolmogorov complexity of the described system. In [25], we show that a natural physical approach indeed leads to new equations which are equivalent to an explicit incorporation of Kolmogorov complexity. Since Kolmogorov complexity is not computable (see, e.g., [28]), the possibility described by Gell-Mann's hypothesis can be used to speed up computations.

Acknowledgements This work was supported in part by NASA under cooperative agreement NCC5-209, by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-1-0365, by NSF grants EAR-0112968 and EAR-0225670, by the Army Research Laboratories grant DATM-05-02-C-0046, and by IEEE/ACM SC2001 and SC2002 Minority Serving Institutions Participation Grants.

This work was partly supported by a travel grant from the conference organizers.

The authors are thankful to all the participants of the CCA'2003 conference for valuable discussions, and to the anonymous referees for valuable suggestions.

References

[1] Beeson M. J. (1985). "Foundations of constructive mathematics", Springer-Verlag, N.Y.
[2] Bennet G. G., Gill J. (1981). "Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq$co-$\mathrm{NP}^A$ with probability 1", *SIAM Journal of Computer Science, vol. 10*, 96–113.
[3] Brassard G., Hoyer P., Tapp A. (1998). Quantum counting. In: *Proc. 25th ICALP*, Lecture Notes in Computer Science, Vol. 1443, Springer, Berlin, 820–831.
[4] Dürr C., Hoyer P. (1996). "A quantum algorithm for finding the minimum"; LANL arXiv:quant-ph/9607014
[5] Ferson S., Ginzburg L., Kreinovich V., Longpré L., and Aviles M. (2002), "Computing Variance for Interval Data is NP-Hard", *ACM SIGACT News, vol. 33*, 108–118.
[6] Ferson S., Ginzburg L., Kreinovich V., and Lopez J. (2002), "Absolute Bounds on the Mean of Sum, Product, etc.: A Probabilistic Extension of Interval Arithmetic", *Extended Abstracts of the 2002 SIAM Workshop on Validated Computing*, Toronto, Canada, May 23–25, 70–72.
[7] Feynman R. P. (1949). "The theory of positrons", *Physical Review, vol. 76*, 749–759.
[8] Gandy, R. (1980). "Church's thesis and principles for mechanisms", In: J. Barwise, H. J. Keisler, and K. Kunen, *The Kleene Symposium*, North Holland, Amsterdam, 123–148.
[9] Gandy, R. (1988). "The confluence of ideas in 1936", In: R. Herken (ed.) *The universal Turing machine: a half-century survey*, Kammerer & Unverzagt, Hamburg.
[10] Gell-Mann M. (1994). "The Quark and the Jaguar", Freeman, N.Y.
[11] Grib A. A., Rodriguez W. .A. Jr. (1999). "Nonlocality in quantum physics", Plenum, N.Y.
[12] Grover L. (1996). A fast quantum mechanical algorithm for database search, *Proc. 28th ACM Symp. on Theory of Computing*, 212–219.
[13] Grover L. K. (1997). Quantum mechanics helps in searching for a needle in a haystack, *Phys. Rev. Lett., vol. 79*, 325–328.
[14] Grover L. (1998). A framework for fast quantum mechanical algorithms, *Phys. Rev. Lett., vol. 80*, 4329–4332.
[15] Harary F., Kreinovich V., Longpré L. (2001). "A new graph characteristic and its application to numerical computability", *Information Processing Letters, vol. 77*, 277–282.
[16] Heinrich S. (2002). Quantum summation with an application to integration, *J. Complexity, vol. 18(1)*, 1–50.
[17] Herrmann F., Margenstern M. (2003). "A universal cellular automaton in the hyperbolic plane", *Theoretical Computer Science, vol. 296*, No. 2, 327–364.
[18] Jaulin L., Keiffer M., Didrit O., and Walter E. (2001), "Applied Interval Analysis", Springer-Verlag, Berlin.
[19] Kearfott R. B. (1996), "Rigorous Global Search: Continuous Problems", Kluwer, Dordrecht.
[20] Kearfott R. B. and Kreinovich V., eds. (1996), "Applications of Interval Computations" (Pardalos. P. M., Hearn, D., "Applied Optimization", Vol. 3), Kluwer, Dordrecht.
[21] Koshelev M. (1998). "Maximum entropy and acausal processes: astrophysical applications and challenges", In: G. J. Erickson et al. (eds.), *Maximum Entropy and Bayesian Methods*, Kluwer, Dordrecht, 253–262.
[22] Kosheleva O. M., Kreinovich V. (1981). "What can physics give to constructive mathematics", In: *Mathematical Logic and Mathematical Linguistics*, Kalinin, Russia, 117–128 (in Russian).
[23] Kreinovich V. (1989). "On the possibility of using physical processes when solving hard problems", Leningrad Center for New Information Technology "Informatika", Technical Report, Leningrad (in Russian).
[24] Kreinovich V., Lakeyev A., Rohn J., and Kahl P. (1997), "Computational Complexity and Feasibility of Data Processing and Interval Computations" (Pardalos. P. M., Hearn, D., "Applied Optimization", Vol. 10), Kluwer, Dordrecht.
[25] Kreinovich V., Longpré L. (1998). "Why Kolmogorov Complexity in Physical Equations", *Int'l J. of Theor. Physics, vol. 37*, 2791–2801.
[26] Kreisel G. (1974). "A notion of mechanistic theory", *Synthese, vol. 29*, 11–26.
[27] Kreisel G. (1982). "A review of [38] and [39]", *Journal of Symbolic Logic, vol. 47*, 900–902.
[28] Li M., Vitányi, P. M. B. (1997). "An Introduction to Kolmogorov Complexity and its Applications", Springer-Verlag, N.Y.
[29] Longpré L., Kreinovich V. (2003). "Can Quantum Computers Be Useful When There Are Not Yet Enough Qubits?", Bull. European Association for Theoretical Computer Science (EATCS), vol. 79, 164–169.

[30] Martinez M., Longpré L., Kreinovich V., Starks S. A., Nguyen H. T. (2003). "Fast Quantum Algorithms for Handling Probabilistic, Interval, and Fuzzy Uncertainty", *Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society NAFIPS'2003*, Chicago, Illinois, July 24–26, 2003.
[31] Maslov S. Yu. (1987). "Theory Of Deductive Systems And Its Applications", MIT Press, Cambridge, MA.
[32] Moore R. E. (1979), "Methods and Applications of Interval Analysis", SIAM, Philadelphia.
[33] Moravec H. (1991). "Time travel and computing", Carnegie-Mellon Univ., CS Dept. Preprint.
[34] Morgenstein D., Kreinovich V. (1995). "Which algorithms are feasible and which are not depends on the geometry of space-time", *Geombinatorics, vol. 4*, No. 3, 80–97.
[35] Nayak A., Wu, F. (1999). The quantum query complexity of approximating the median and related statistics, *Proc. Symp. on Theory of Computing STOC'99*, 384–393.
[36] Nielsen M. A., Chuang I. L. (2000). *Quantum computation and quantum information*, Cambridge University Press, Cambridge, U.K.
[37] Novak E. (2001). Quantum Complexity of integration, *J. Complexity, vol. 17*, 2–16.
[38] Pour-El M. B., Richards I. (1979). "A computable ordinary differential equation which possesses no computable solutions", *Annals of Mathematical Logic, vol. 17*, pp. 61–90.
[39] Pour-El M. B., Richards I. (1981). "The wave equation with computable initial data such that its unique solution is not computable", *Advances in Mathematics, vol. 39*, pp. 215–139.
[40] Pour-El M. B., Richards I. (1989). "Computability in analysis and physics", Springer-Verlag, N.Y.
[41] Rabinovich S. (1993), "Measurement Errors: Theory and Practice", American Institute of Physics, New York.
[42] Thorne K. S. (1994). "From black holes to time warps", W.W. Norton, N.Y.
[43] Trejo R., Kreinovich V. (2001). Error Estimations for Indirect Measurements: Randomized vs. Deterministic Algorithms. In: S. Rajasekaran et al. (eds.), *Handbook on Randomized Computing*, 673–729.
[44] Vavasis S. A. (1991), "Nonlinear Optimization: Complexity Issues", Oxford University Press, N.Y.
[45] Wadsworth H. M. Jr., ed. (1990). "Handbook of statistical methods for engineers and scientists", McGraw-Hill Publishing Co., N.Y.
[46] Weihrauch, K., Zhong, N. (2002). "Is wave propagation computable or can wave computers beat the Turing machine?", *Proceedings of the London Mathematical Society, vol. 85*, 312–332.