Chapter 1

ERROR ESTIMATIONS FOR INDIRECT MEASUREMENTS: RANDOMIZED VS. DETERMINISTIC ALGORITHMS FOR "BLACK-BOX" PROGRAMS

Vladik Kreinovich
and Raúl Trejo

Abstract In many real-life situations, it is very difficult or even impossible to directly measure the quantity y in which we are interested: e.g., we cannot directly measure a distance to a distant galaxy or the amount of oil in a given well. Since we cannot measure such quantities *directly*, we can measure them *indirectly*: by first measuring some relating quantities $x_1, \ldots, x_n$, and then by using the known relation between x_i and y to reconstruct the value of the desired quantity y.

In practice, it is often very important to estimate the error of the resulting indirect measurement. In this paper, we describe and compare different deterministic and randomized algorithms for solving this problem in the situation when a program for transforming the estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$ for x_i into an estimate for y is only available as a *black box* (with no source code at hand).

We consider this problem in two settings: *statistical*, when measurements errors $\Delta x_i = \widetilde{x}_i - x_i$ are independent Gaussian random variables with 0 average and known standard deviations σ_i, and *interval*, when the only known information about Δx_i is that $\Delta x_i \in [-\Delta_i, \Delta_i]$ for a known bound Δ_i. In statistical setting, we describe the optimal error estimation algorithm; in interval setting, we describe a new algorithm which may be not optimal but which is better than the previously known ones.

1. ERROR ESTIMATION FOR INDIRECT MEASUREMENTS – FORMULATION OF THE PROBLEM

What Are Indirect Measurements. In many real-life situations, it is difficult or even impossible to directly measure the quantity y in which we are interested. For example, it is, at present, practically impossible to directly measure a distance to a distant quasar, or the amount of oil in a given area. Since we cannot measure such quantities *directly*, we have to measure them *indirectly*: Namely, we measure some other quantities $x_1, \ldots, x_n$ which are related to y by a known dependence $y = f(x_1, \ldots, x_n)$, and then apply the known function f to the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of measuring x_i.

For example, to determine the amount of oil y in a given area, we measure the results x_i of sending ultrasound signals between the two parallel wells, and then estimate y by solving the appropriate system of partial differential equations (in this example, $f(x_1, \ldots, x_n)$ is a program for solving this system).

In general, such a two-stage procedure (measurement followed by computations) is called an *indirect measurement*, and the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ resulting from this two-stage procedure is called the *result* of indirect measurement.

Toy Example. To make the exposition clearer, we will illustrate these notions on the following toy example: Suppose that we are interested in the voltage V, but we have no voltmeter at hand. One possibility of measuring V indirectly follows from Ohm's law: we can measure the current I and the resistance R, and compute V as $I \cdot R$. In this case, x_1 is the current, $x_2 = R$, and $f(x_1, x_2) = x_1 \cdot x_2$.

If the measured value $\widetilde{x}_1$ of the current is 1.0, and the measured value of the resistance is $\widetilde{x}_2 = 2.0$, then the result of the corresponding indirect measurement is $\widetilde{y} = 1.0 \cdot 2.0 = 2.0$.

Error Estimation for Indirect Measurements: A Real-Life Problem. Measurements are never 100% accurate; hence, the result $\widetilde{x}_i$ of each direct measurement is, in general, somewhat different from the actual value of the measured quantity. As a result of these measurement errors $\Delta x_i = \widetilde{x}_i - x_i$, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of applying f to the measurement result will be, in general, different from the actual value $y = f(x_1, \ldots, x_n)$ of the desired quantity.

> For example, in our toy problem, the actual value of the current may be $x_1 = 0.9 \neq \widetilde{x}_1 = 1.0$, and the actual value of the resistance is $x_2 = 2.05 \neq \widetilde{x}_2 = 2.0$. In this case, the actual value of the voltage is $y = x_1 \cdot x_2 = 0.9 \cdot 2.05 = 1.845 \neq 2.0$.

Since the result $\widetilde{y}$ of indirect measurement is, in general, different from the actual value y, it is desirable to know the characteristics of the error $\Delta y = \widetilde{y} - y$ of indirect measurement. How can we estimate these characteristics?

Possible Information Available for Estimating the Error of Indirect Measurements. First, we know the function $f(x_1, \ldots, x_n)$. This function may be given as an analytical expression, or, more frequently, as an algorithm. It may be a program written in a high-level programming language (i.e., a *source code*), which can be translated into an executable file ready for computations, or it may be only an executable file, with no source code provided.

Second, we know the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of direct measurements.

Finally, we need some information about the errors of the direct measurements. The errors Δx_i come from the imperfection of the corresponding measuring instruments. For an instrument to be called *measuring*, its manufacturer must supply some (well-defined) information about the measurement errors. Ideally, this information must include the probability distribution of different measurement errors.

The knowledge of these probabilities is desirable but not always required and not always possible. In many practical cases, we only know the upper bounds Δ_i for the possible measurement errors, i.e., we only know that $|\Delta x_i| \leq \Delta_i$. In such cases, after each direct measurement, the only information that have about the actual value x_i of the measured quantity is that this value belongs to the *interval* $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.

> For example, in our toy case, the manufacturer of the measuring instruments may guarantee that the measurement error Δx_1 of measuring current cannot exceed $\Delta_1 = 0.1$, and the measurement error Δx_2 of measuring resistance cannot exceed $\Delta_2 = 0.05$. If no other information about the measurement accuracy is given, then, after we got the measurement results $\widetilde{x}_1 = 1.0$ and $\widetilde{x}_2 = 2.0$, the only information we have about the actual value of the current x_1 is that $x_1 \in [1.0 - 0.1, 1.0 + 0.1] = [0.9, 1.1]$. Similarly, the only information we have about the actual value of the resistance x_2 is that $x_2 \in [2.0 - 0.05, 2.0 + 0.05] = [1.95, 2.05]$. (The actual values $x_1 = 0.9$ and $x_2 = 1.05$, of course, belong to these intervals; if they did not, this would mean that the manufacturer's bounds are incorrect.)

In the situations when we only know the upper bounds on the measurement errors, the problem of estimating the error of indirect measurement is called the problem of *interval computations*; for details and examples of practical applications, see, e.g., [31, 65]. The setting when we only know intervals will be one of the settings considered in this paper.

Another setting which we will consider is a setting described in standard engineering textbooks on measurement (see, e.g., [25, 56]; see also [13, 28]). In this setting, the measurement error Δx_i of each direct measurement is normally distributed with 0 average and known standard deviation σ_i, and measurement errors of different direct measurements are independent random variables. These assumptions can be justified by two related reasons:

- The first reason is more *theoretical*. Usually, the manufacturers of the measuring instruments have made their best effort to eliminate the major sources of measurement error. The resulting measurement error comes from a variety of small independent error sources, and thus, can be described as a sum of a large number of small independent random variables. According to the central limit theorem, such a sum, under reasonable conditions, converges to normal distribution. Thus, if there are sufficiently many small random components, the resulting error distribution is indeed close to normal.

- The second reason for choosing Gaussian distribution is *empirical*: for the majority of measuring instruments, the measurement error is indeed normally distributed (see, e.g., [52, 54]).

In This Paper, We Will Only Consider Situations When the Measurements Are Reasonably Accurate. In this paper, we will only consider situations in which the direct measurements are accurate enough, so that the resulting measurement errors Δx_i are small, and terms which are quadratic (or of higher order) in Δx_i can be safely neglected, and so, the dependence of the desired value $y = f(x_1, \ldots, x_n) = f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n)$ on Δx_i can be safely assumed to be linear.

In our toy example, $f(x_1, x_2) = x_1 \cdot x_2$, so $y = f(\widetilde{x}_1 - \Delta x_1, \widetilde{x}_2 - \Delta x_2) = \widetilde{x}_1 \cdot \widetilde{x}_2 - \widetilde{x}_2 \cdot \Delta x_1 - \widetilde{x}_1 \cdot \Delta x_2 + \Delta x_1 \cdot \Delta x_2$. In our case, $\widetilde{x}_1 = 1.0$, $\widetilde{x}_2 = 2.0$, so $y = 2.0 - 2\Delta x_1 - \Delta x_2 + \Delta x_1 \cdot \Delta x_2$. The only non-linear term in this expansion is the quadratic term $\Delta x_1 \cdot \Delta x_2$.

Here, $\Delta x_1 = \widetilde{x}_1 - x_1 = 1.0 - 0.9 = 0.1$, $\Delta x_2 = 2.0 - 2.05 = -0.05$, and $\Delta y = 2.0 - 1.845 = 0.155$. If we ignore the quadratic term, we get approximate values $y_{\text{approx}} = 2.0 - 2\Delta x_1 - \Delta x_2 = 1.85$ and $\Delta y_{\text{approx}} = \widetilde{y} - y_{\text{approx}} = 0.15$. The error of this linear approximation is $0.005 \ll 0.15$.

Comments.

- To avoid possible confusion, we must emphasize the following. We are *not* talking about functions f which are linear for *all* possible values of input data. In this paper, we are considering data processing functions f

which can be approximated by linear ones in the close vicinity of every measurement result $\vec{\widetilde{x}} = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$. These linear approximations, however, are *different* for different measurement results.

- There are practical situations when the accuracy of the direct measurements is not high enough, and hence, quadratic terms cannot be safely neglected (see, e.g., [31] and references therein). In this case, the problem of error estimation for indirect measurements becomes computationally difficult (NP-hard) even when the function $f(x_1, \ldots, x_n)$ is quadratic [43, 62]. However, in most real-life situations, the possibility to ignore quadratic terms is a reasonable assumption, because, e.g., for an error of 1% its square is a negligible 0.01%.

With the above restriction in place, we can easily deduce the explicit expression for the error Δy of indirect measurement.

Indirect Measurement Error: Derivation and the Resulting Formula. Due to the accuracy requirement, we can simplify the expression for $\Delta y = \widetilde{y} - y = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(x_1, \ldots, x_n)$ if we expand the function f in Taylor series around the point $(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ and restrict ourselves only to linear terms in this expansion. As a result, we get the expression

$$\Delta y = c_1 \cdot \Delta x_1 + \ldots + c_n \cdot \Delta x_n, \tag{1}$$

where by c_i, we denoted the value of the partial derivative $\partial f / \partial x_i$ at the point $(\widetilde{x}_1, \ldots, \widetilde{x}_n)$:

$$c_i = \frac{\partial f}{\partial x_i}\Big|_{(\widetilde{x}_1, \ldots, \widetilde{x}_n)}. \tag{2}$$

Probability Distribution of the Indirect Measurement Error: Derivation and the Resulting Formula. In the statistical setting, the desired measurement error Δy is a linear combination of independent Gaussian variables Δx_i, and hence, Δy is also normally distributed, with 0 average and the standard deviation

$$\sigma = \sqrt{c_1^2 \cdot \sigma_1^2 + \ldots + c_n^2 \cdot \sigma_n^2}. \tag{3}$$

Comment. A similar formula holds if we *do not* assume that Δx_i are normally distributed: it is sufficient to assume that they are independent variables with 0 average and known standard deviations σ_i.

Interval of Possible Values of the Indirect Measurement Error: Derivation and the Resulting Formula. In the interval setting, we do not know the probability of different errors Δx_i; instead, we only know that $|\Delta x_i| \leq \Delta_i$. In

this case, the sum (1) attains its largest possible value if each term $c_i \cdot \Delta x_i$ in this sum attains the largest possible value:

- If $c_i \geq 0$, then this term is a monotonically non-decreasing function of Δx_i, so it attains its largest value at the largest possible value $\Delta x_i = \Delta_i$; the corresponding largest value of this term is $c_i \cdot \Delta_i$.
- If $c_i < 0$, then this term is a decreasing function of Δx_i, so it attains its largest value at the smallest possible value $\Delta x_i = -\Delta_i$; the corresponding largest value of this term is $-c_i \cdot \Delta_i = |c_i| \cdot \Delta_i$.

In both cases, the largest possible value of this term is $|c_i| \cdot \Delta_i$, so, the largest possible value of the sum Δy is

$$\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n. \tag{4}$$

Similarly, the smallest possible value of Δy is $-\Delta$.

Hence, the interval of possible values of Δy is $[-\Delta, \Delta]$, with Δ defined by the formula (4).

Comment. In our toy problem, it is easy to compute the actual interval of possible values of $y = x_1 \cdot x_2$ when $x_1 \in [0.9, 1.1]$ and $x_2 \in [1.95, 2.05]$: Indeed, the function $f(x_1, x_2) = x_1 \cdot x_2$ is monotonically increasing as a function of each of its variables (for $x_1 > 0$ and $x_2 > 0$). Thus, the largest possible value of $y = f(x_1, x_2)$ is attained when both input variables take their largest possible values, i.e., when $x_1 = 1.1$ and $x_2 = 2.05$, and is equal to $1.1 \cdot 2.05 = 2.255$. Similarly, the smallest possible value of $y = f(x_1, x_2)$ is attained when both input variables take their smallest possible values, i.e., when $x_1 = 0.9$ and $x_2 = 1.95$; this smallest value of y is equal to $0.9 \cdot 1.95 = 1.755$. So, the interval of possible values of y is equal to $[1.755, 2.255]$. Hence, the interval of possible values for $\Delta y = \widetilde{y} - y = 2 - y$ is $[-0.255, 0.245]$.

On the other hand, applying formula (4), we get $\Delta = 2.0 \cdot 0.1 + 1.0 \cdot 0.05 = 0.25$ and the interval $[-0.25, 0.25]$. (We can see that this is indeed a good approximation to the actual interval.)

Error Estimation for Indirect Measurement: a Precise Computational Formulation of the Problem. As a result of the above analysis, we get the following explicit formulation of the problem: given a function $f(x_1, \ldots, x_n)$, n numbers $\widetilde{x}_1, \ldots, \widetilde{x}_n$, and n positive numbers $\sigma_1, \ldots, \sigma_n$ (or $\Delta_1, \ldots, \Delta_n$), compute the corresponding expression (3) or (4).

Let us describe how this problem is solved now.

Textbook Case: The Function f is Given by Its Analytical Expression. If the function f is given by its analytical expression, then we can simply explicitly differentiate it, and get an explicit expression for (3) and (4). This is

the case which is typically analyzed in textbooks on measurement theory (see, e.g., [25, 56]).

A More Complicated Case: Analytical Differentiation. In many practical cases, we do not have an explicit analytical expression, we only have an *algorithm* for computing the function $f(x_1, \ldots, x_n)$, an algorithm which is too complicated to be expressed as an analytical expression.

When this algorithm is presented in one of the standard programming languages such as Fortran or C, we can apply one of the existing analytical differentiation tools (see, e.g., [8, 27]), and automatically produce a program which computes the partial derivatives c_i. These tools analyze the code and produce the differentiation code as they go.

In Many Practical Applications, We Must Treat the Function $f(x_1, \ldots, x_n)$ As a Black Box. In many other real-life applications, an algorithm for computing $f(x_1, \ldots, x_n)$ may be written in a language for which an automatic differentiation tool is not available, or a program is only available as an executable file, with no source code at hand. In such situations, when we have no easy way to analyze the code, the only thing we can do is to take this program as a *black box*: i.e., to apply it to different inputs and use the results of this application to compute the desired value σ.

In this paper, we will analyze such black-box situations, and describe the optimal algorithm for computing σ, and a new algorithm for computing Δ. Before we describe these algorithms, we must do two things: formulate the black-box approach in mathematical terms, and describe known black-box-oriented algorithms.

Comment. Algorithms designed for the black-box situation can be also used when we do have a code of a program f, and we can, in principle, use automatic differentiation tools; we can always ignore this knowledge and treat the program as a black box. In some practical situations, this turned out to be useful, because both the automatic differentiation algorithms and the black-box-oriented algorithms come with a substantial computational overhead, and the overhead for black-box-oriented algorithms is sometimes smaller.

Error Estimation for Indirect Measurements: Towards a Mathematical Reformulation of the Black-Box Approach. The general idea of a black-box approach is that, in addition to computing the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of the indirect measurement, we also apply the function f to other inputs which are different from $(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

Our ultimate goal is to analyze how errors in x_i induce errors in the value of the function f. We have already assumed that the errors in x_i are small, so the vector $(x_1, \ldots, x_n)$ of the actual values is close to the measured values

$(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ (so close that the squares of measurement errors $\Delta x_i = \widetilde{x}_i - x_i$ can be neglected). From the viewpoint of this goal, we are only interested in the values of the function f in the small vicinity of the measurement point, i.e., for the values $x_i = \widetilde{x}_i + \delta_i$ for some small δ_i (so small that their squares are negligible). Since quadratic terms are negligible, we can expand f in Taylor series and only keep linear terms in this expansion. As a result, when we apply the function f to the values $x_i + \delta_i$, we get the result

$$f(\widetilde{x}_1 + \delta_1, \ldots, \widetilde{x}_n + \delta_n) = \widetilde{y} + c_1 \cdot \delta_1 + \ldots + c_n \cdot \delta_n. \qquad (5)$$

So, in effect, for sufficiently small vectors $\vec{\delta}$ (i.e., for vectors for which $\left\|\vec{\delta}\right\| \le \delta_0$ for some real number $\delta_0 > 0$), we get the *dot* (*scalar*) product $\vec{c} \cdot \vec{\delta}$ between the input vector $\vec{\delta} = (\delta_1, \ldots, \delta_n)$ and the (unknown) vector $\vec{c} = (c_1, \ldots, c_n)$.

Thus, for an arbitrary *small* vector $\vec{\delta}$, we can find its dot product $\vec{c} \cdot \vec{\delta}$ with the (unknown) gradient vector $\vec{c}$ by a single call to the black-box computing f: namely, as $f\left(\vec{\widetilde{x}} + \vec{\delta}\right) - \widetilde{y}$. We can use a slightly more complex idea to compute the product $\vec{c} \cdot \vec{\delta}$ for an *arbitrary* (not necessarily small) vector $\vec{\delta}$ as follows:

- First, we compute the length $\delta = \left\|\vec{\delta}\right\|$ of the given vector $\vec{\delta}$. Then:
- If $\delta \le \delta_0$, then we compute $\vec{c} \cdot \vec{\delta}$ as $f\left(\vec{\widetilde{x}} + \vec{\delta}\right) - \widetilde{y}$.
- If $\delta > \delta_0$, then we compute $\vec{c} \cdot \vec{\delta}$ as follows:
  - first, we compute the normalization coefficient $K_{\rm norm} = \delta / \delta_0$;
  - then, we compute the auxiliary vector $\vec{\beta} = \vec{\delta} / K_{\rm norm}$ with components $\beta_i = \delta_i / K_{\rm norm}$; due to normalization, for this auxiliary vector, $\left\|\vec{\beta}\right\| = \delta_0$;
  - third, we compute $\vec{c} \cdot \vec{\beta} = f\left(\vec{\widetilde{x}} + \vec{\beta}\right) - \widetilde{y}$;
  - finally, we compute $\vec{c} \cdot \vec{\delta}$ as $K_{\rm norm} \cdot \left(\vec{c} \cdot \vec{\beta}\right)$.

Let us explicitly repeat the resulting mathematical reformulation.

Error Estimation for Indirect Measurements: A Mathematical Reformulation of the Black-Box Approach. We know the values $\widetilde{y}, \sigma_1, \ldots, \sigma_n$ (or $\widetilde{y}$ and $\Delta_1, \ldots, \Delta_n$); we know that there is a vector $\vec{c} = (c_1, \ldots, c_n)$ but we do not know the values of its components. For any given vector $\vec{\delta} = (\delta_1, \ldots, \delta_n)$, we can compute the dot product $\vec{c} \cdot \vec{\delta} = c_1 \cdot \delta_1 + \ldots + c_n \cdot \delta_n$. Based on the results of these computations, we must compute, correspondingly, the value (3) or (4).

A Straightforward Method of Solving This Problem: Numerical Differentiation. The most straightforward algorithm for solving this problem is to compute the derivatives c_i one-by-one, and then use the corresponding formula (3) or (4) to compute the desired σ. To compute the i-th partial derivative, we change the i-th input x_i to $\widetilde{x}_i + h_i$ for some h_i, and leave other inputs unchanged, i.e., we take $\delta_i = h_i$ for this i and $\delta_j = 0$ for all $j \neq i$. Then, we estimate c_i as

$$c_i = \frac{1}{h_i} \cdot \left(f\left(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n\right) - \widetilde{y}\right).$$

This algorithm is called *numerical differentiation.*

We want the change h_i to be small (so that quadratic terms can be neglected); we already know that changes of the order σ_i are small. So, it is natural to take $h_i = \sigma_i$ (or, correspondingly, $h_i = \Delta_i$). In other words, to compute c_i, we use the following values: $\delta_1 = \ldots = \delta_{i-1} = 0$, $\delta_i = \sigma_i$ (or $\delta_i = \Delta_i$), $\delta_{i+1} = \ldots = \delta_n = 0$.

This choice of δ_i has an additional advantage: it decreases the computation time. Namely, no matter which values h_i we use, we need the same number of calls to f (i.e., n calls), but by using the above values h_i, we decrease the total number of additional computations. Indeed, e.g., in interval setting:

- For *general* h_i, we need three arithmetic operations to compute each value c_i: one addition to compute $\widetilde{x}_i + h_i$, one subtraction to compute the difference $f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}$, and one division to divide this difference by h_i. So, to compute all n partial derivatives, we need $3n$ arithmetic operations. Then, we need n takings of absolute value, n multiplications and $n-1$ additions to compute the sum (4), which brings the total to $3n + n + n + (n-1) = 6n - 1$ arithmetic operations.

- For the above values of h_i, the difference $f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}$ is equal to $c_i \cdot h_i = c_i \cdot \Delta_i$, so there is no need to first divide it by h_i and then multiply it by Δ_i: we can directly estimate $|c_i| \cdot \Delta_i$ as $|f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}|$. Thus, to compute Δ, we now need n additions $\widetilde{x}_i + h_i$, n subtractions $f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}$, n absolute values to compute the terms $|c_i| \cdot \Delta_i$, and $n-1$ additions to add all these terms, to the total of $4n - 1 (< 6n - 1)$ arithmetic operations.

For statistical setting, we get a similar decrease in computation time.

Comment. The above numerical differentiation algorithm is the simplest possible. There are other algorithms that give better results for non-linear functions

(see, e.g., [26]); e.g., we can estimate c_i as

$$f_{,i} = \frac{1}{h_i} \cdot \left(f\left(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_i + \frac{h_i}{2}, \tilde{x}_{i+1}, \ldots, \tilde{x}_n\right) - f\left(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_i - \frac{h_i}{2}, \tilde{x}_{i+1}, \ldots, \tilde{x}_n\right)\right).$$

Such algorithms take longer to compute (e.g., the above algorithm requires $2n$ calls to f instead of n). Spending this extra time make sense if terms quadratic in h_i cannot be neglected. However, since we assumed that we can neglect these quadratic terms, even our simplest numerical differentiation algorithm leads to exact value of c_i and therefore, there is no need to spend time on more sophisticated differentiation techniques.

A natural next question is: can we use an even simpler numerical differentiation algorithm which would allow us to compute the quantity (3) or (4) with fewer than n calls to f? As we will soon see, if we restrict ourselves to *deterministic* algorithms, the answer is: No, n is the smallest we can get.

Sometimes, Numerical Differentiation Takes Too Long. If a function $f(x_1, \ldots, x_n)$ is simple and fast-to-compute (e.g., if it is given by an explicit analytical expression), then we do not need the black-box-oriented algorithms at all. We only need these algorithms when the program f is itself time-consuming (e.g., computing f may involve solving an inverse problem). In this case, applying the function f is the most time-consuming part of this algorithm. So, the total time T that it takes us to compute σ is (approximately) equal to the running time T_f for the program f multiplied by the number of times N_f that we call the program f.

For numerical differentiation, $N_f = n$ (we call f n times to compute n partial derivatives). Hence, if the program f takes a long time to compute, and n is huge, then the resulting time T may be too long. For example, if we are determining some parameters of an oil well from the geophysical measurements, we may get n in the thousands, and T_f in minutes. In this case, $T = T_f \cdot n$ may take several weeks. This may be OK for a single measurement, but too long if we want more on-line results.

If We Do Not Have Enough Time for Numerical Differentiation, We May Use Randomized Algorithms. The only way to save on computation time is to limit the number of times we call f, i.e., to limit N_f.

If we limit this number of calls to $N_f < n$, we cannot get an answer which is always correct: Indeed, if applied $N_f < n$ different vectors $\vec{\delta} = (\delta_1, \ldots, \delta_n)$ and computed the values $\vec{\delta} \cdot \vec{c}$, then we get $\leq n - 1$ equations for n variables $c_1, \ldots, c_n$; therefore, this system does not have a single solution; it has at least a 1-D line of possible solutions $\vec{c}$, and for points of this line which go to ∞, both

expressions (3) and (4) tend to ∞ as well. Thus, we can have arbitrarily large values of σ and Δ which are consistent with the results of black-box testing, and we cannot, therefore, deduce the exact value of σ or Δ from these testing results.

Since we cannot use *deterministic* algorithms, i.e., algorithms which always guarantee an answer, we can try *randomized* algorithms, which provide us with an answer with a certain probability. In real-life applications, the fact that these algorithms have a (small) probability of erring is OK, because the measuring instruments themselves have a small probability of failing anyway.

Randomized algorithms can indeed speed up computations: e.g., in statistical setting, a straightforward simulation (Monte-Carlo type) saves time drastically:

Monte-Carlo Simulation for Statistical Setting. In this algorithm, we use a computer-based random number generator to simulate the normally distributed error. A standard normal random number generator usually produces a normal distribution with 0 average and standard deviation 1. So, to simulate a distribution with a standard deviation σ_i, we multiply the result α_i of the standard random number generator by σ_i. In other words, we take $\delta_i = \sigma_i \cdot \alpha_i$.

As a result of N Monte-Carlo simulations, we get N values $c^{(1)} = \vec{c} \cdot \vec{\delta}^{(1)}, \ldots, c^{(N)} = \vec{c} \cdot \vec{\delta}^{(N)}$ which are normally distributed with the desired standard deviation σ. So, we can determine σ by using the standard statistical estimate

$$\sigma = \sqrt{\frac{1}{N-1} \cdot \sum_{k=1}^{N} \left(c^{(k)}\right)^2}.$$

The relative error of this estimate depends only on N (as $\approx 1/\sqrt{N}$), and not on the number of variables n. Therefore, the number of steps N_f needed to achieve a given accuracy does not depend on the number of variables at all.

The error of the above algorithm is asymptotically normally distributed, with a standard deviation $\sigma_e \sim \sigma/\sqrt{2N}$. Thus, if we use a "two sigma" bound, we conclude that with probability 95%, this algorithm leads to an estimate for σ which differs from the actual value of σ by $\leq 2\sigma_e = 2\sigma/\sqrt{2N}$.

This is an error with which we estimate the error of indirect measurement; we do not need too much accuracy in this estimation, because, e.g., in real life, we say that an error is $\pm 10\%$ or $\pm 20\%$, but *not* that the error is, say, $\pm 11.8\%$. Therefore, in estimating the error of indirect measurements, it is sufficient to estimate the characteristics of this error with a relative accuracy of, say, 20%.

For the above "two sigma" estimate, this means that we need to select the smallest N for which $2\sigma_e = 2\sigma/\sqrt{2N} \leq 0.2 \cdot \sigma$, i.e., to select $N_f = N = 50$.

In many practical situations, it is sufficient to have a standard deviation of 20% (i.e., to have a "two sigma" guarantee of 40%). In this case, we need only $N = 13$ calls to f.

On the other hand, if we want to guarantee 20% accuracy in 99.9% cases, which correspond to "three sigma", we must use N for which $3\sigma_e = 3 \cdot \sigma/\sqrt{2N} \leq 0.2 \cdot \sigma$, i.e., we must select $N_f = N = 113$, etc.

For $n \approx 10^3$, all these values of N_f are much smaller than $N_f = n$ required for numerical differentiation.

So, if we have to choose between the (deterministic) numerical differentiation and the randomized Monte-Carlo algorithm, we must select:

- a deterministic algorithm when the number of variables n satisfies the inequality $n \leq N_0$ (where $N_0 \approx 50$), and
- a randomized method if $n \geq N_0$.

These two algorithms are the most widely used. A natural question is: are they optimal? Our answer is: no, for many different values of n, it is possible to find an algorithm which is better than both.

What We Are Planning To Do. In this paper, we will describe the *best* randomized algorithm for estimating σ (which is better than the Monte-Carlo simulation), and a new algorithm for estimating Δ.

Our main ideas and the preliminary (restricted) versions of our results were first announced in [33, 37, 40, 41, 42, 44, 45].

2. ERROR ESTIMATION FOR INDIRECT MEASUREMENT: STATISTICAL SETTING

2.1 ERROR ESTIMATION FOR INDIRECT MEASUREMENT REFORMULATED AS A TOMOGRAPHY PROBLEM

When $\sigma_1 = \ldots = \sigma_n = 1$, The Error Estimation Problem Has a Natural Geometric (Tomographic) Interpretation. When $\sigma_1 = \ldots = \sigma_n = 1$, then the formula (3) becomes a formula for the length of a vector $\vec{c} = (c_1, \ldots, c_n)$. Therefore, the above error estimation problem for indirect measurements can be reformulated in the following geometric terms: We have an (unknown) vector $\vec{c}$. We want to know its length $c = \|\vec{c}\|$. In order to compute this length, we can pick an arbitrary vector $\vec{\delta}$ and compute a scalar product $\vec{c} \cdot \vec{\delta}$. The question is: how to estimate the desired length by picking the smallest possible number of vectors. Alternatively, if the number N of picked vectors is fixed, the question is: How to select these vectors $\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}$ in such a way that from the scalar products $c^{(1)}, \ldots, c^{(N)}$, we will able to get the best possible estimate $\widetilde{c}$ for the desired length c.

From the mathematical viewpoint, knowing the scalar product $\vec{c} \cdot \vec{\delta}$ of the unknown vector $\vec{c}$ with a known vector $\vec{\delta}$ is equivalent to knowing the *projection*

$\pi_{\vec{\delta}}(\vec{c}) = (\vec{c} \cdot \vec{\delta}) / \left\|\vec{\delta}\right\|$. Therefore, the above problem means that we need to reconstruct the length of a vector from its projections to different directions.

In general, problems in which we want to reconstruct a certain property of an unknown object based on the characteristics of its projections are called *tomography problems*. Therefore, the above problem can be viewed as a (simple) case of tomography problems.

The General Case Can Be Naturally Reduced to the (Geometrically Interpretable) Case When $\sigma_1 = \ldots = \sigma_n = 1$. Let us show that the general case of the error estimation problem for indirect measurements can be naturally reduced to the case of $\sigma_i = 1$. Indeed, in the general case, we must find the expression (2) for unknown values $c_1, \ldots, c_n$. Instead of considering the values c_i as unknowns, we can introduce new unknowns: $c'_i = c_i \cdot \sigma_i$. Then, the desired expression (3) is simply the length $\sigma = \left\|\vec{c'}\right\|$ of the new unknown vector $\vec{c'} = (c'_1, \ldots, c'_n)$.

To complete the reduction to the case of $\sigma_i = 1$, we need to reformulate the expression for the observed quantity $\sum c_i \cdot \delta_i$ in terms of the new variables $c'_i = c_i \cdot \sigma_i$. Such a reformulation is straightforward: $\sum c_i \cdot \delta_i = \sum c'_i \cdot \delta'_i$, where $\delta'_i = \delta_i / \sigma_i$. If we know δ'_i, then we can reconstruct δ_i as $\delta_i = \delta'_i \cdot \sigma_i$.

Therefore, if we know how to solve the geometrizable problem (with $\sigma_i = 1$), we can then solve the problem with arbitrary σ_i as follows:

- first, we generate a vector $\vec{\delta'}$ which is appropriate for the geometrized problem, and
- then use the values $\delta_i = \delta'_i \cdot \sigma_i$ to solve the given problem.

This reduction is very natural, because in both above algorithms – numerical differentiation and Monte-Carlo – the values δ_i were actually of the form $\delta_i = \sigma_i \cdot \alpha_i$ for some values α_i which did not depend on σ_i.

Precise Formulation of the Corresponding Geometric Problem. In view of the above reduction, it is sufficient to solve the geometrizable particular case of the error estimation problem. Let us formulate this problem in precise mathematical terms.

Definition 1. *Let positive integers n and N be given. By an* algorithm which solves the n-variable error estimation problem in N calls *(or simply an (n, N)-*algorithm*, for short), we mean a tuple $\mathcal{U} = (p_1, \ldots, p_N, \widetilde{e})$, where:*

- *for every k from 1 to N, p_k is a function which assigns, to every element from $R^{n \times (k-1)} \times R^{k-1}$, a probability measure on R^n, and*
- *$\widetilde{e}$ is a function which assigns to every element of $R^{n \times N} \times R^N$ a probability measure on R.*

Definition 2. *Let $n, N > 0$. By a* (n, N)-computational trajectory, *we mean a sequence $\vec{\delta}^{(1)}, c^{(1)}, \vec{\delta}^{(2)}, c^{(2)}, \ldots, \vec{\delta}^{(N)}, c^{(N)}, \widetilde{c}$, where $\widetilde{c}$ is a real number, and for each k from 1 to N, $\vec{\delta}^{(k)} \in R^n$ is an n-dimensional vector and $c^{(k)}$ is a real number.*

Definition 3. *Let $\mathcal{U} = (p_1, \ldots, p_N, \widetilde{e})$ be an (n, N)-algorithm. Then, for each n-dimensional vector $\vec{c} \in R^n$, the algorithm $\mathcal{U}$ generates the following probability distribution on the set of all possible (n, N)-computational trajectories:*

- *for $k = 1$, a random vector $\vec{\delta}^{(1)}$ is generated according to the probability distribution p_1, and we compute $c^{(1)} = \vec{c} \cdot \vec{\delta}^{(1)}$;*
- *next, for $k = 2, 3, \ldots, N$, a random vector $\vec{\delta}^{(k)}$ is generated according to the probability distribution $p_k\left(\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(k-1)}, c^{(1)}, \ldots, c^{(k-1)}\right)$; then, we compute $c^{(k)} = \vec{c} \cdot \vec{\delta}^{(k)}$;*
- *finally, a random value $\widetilde{c}$ is generated according to the probability measure $\widetilde{e}\left(\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}, c^{(1)}, \ldots, c^{(N)}\right)$.*

Let $\Phi(\widetilde{c}, c)$ be a function which, for each c, attains its minimum for $\widetilde{c} = c$ and is convex in c.

- *This function will be called the* inaccuracy *of approximating c by $\widetilde{c}$.*
- *For each vector $\vec{c} \in R^n$ of length $c = \|\vec{c}\| \neq 0$, the* inaccuracy $I(\mathcal{U}, \vec{c})$ of an algorithm $\mathcal{U}$ on this vector *is defined as $I(\mathcal{U}, \vec{c}) = E(\Phi(\widetilde{c}, c))$.*
- *By an* inaccuracy $I(\mathcal{U})$ *of an algorithm $\mathcal{U}$, we mean the largest possible value of its inaccuracy on any vector $\vec{c} \neq 0$.*

Example. For example, we can take

$$\Phi(\widetilde{c}, c) = \left(\frac{\widetilde{c} - c}{c}\right)^2.$$

Comment. In the above definitions, we define a randomized algorithm for solving our problem. In particular, if all probability distributions are degenerate (i.e., concentrate on a single vector with probability 1), we get a deterministic algorithm.

Definition 4. *Let $n, N > 0$. We say that an (n, N)-algorithm is* optimal *if it has the smallest possible inaccuracy.*

Now, we are ready to formulate and to prove the main result of this section:

2.2 THE MAIN RESULT FOR STATISTICAL SETTING: OPTIMAL ERROR ESTIMATION ALGORITHM FOR INDIRECT MEASUREMENTS

In this subsection, we will explicitly describe, for each n and N and for each inaccuracy criterion, the optimal (n, N)-algorithm. For readers' convenience, all the proofs are located in a special (last) section.

Theorem 1. *When $N \geq n$, numerical differentiation is the optimal (n, N)-algorithm.*

When $N < n$, the following (n, N)-algorithm $\mathcal{U} = (p_1, \ldots, p_k, \widetilde{e})$ is optimal:

- *p_1 is a uniform distribution on a unit sphere in R^n;*
- *for each $k > 1$, $p_k\left(\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(k-1)}, c^{(1)}, \ldots, c^{(k-1)}\right)$ is a uniform distribution on the set of all unit vectors of R^n which are orthogonal to $(k-1)$ vectors $\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(k-1)}$ (this set is an $(n-k)$-dimensional sphere);*
- *finally, the probability distribution $\widetilde{e}\left(\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}, c^{(1)}, \ldots, c^{(N)}\right)$ is concentrated, with probability 1, on a real number*

$$\widetilde{c} = g\left(\sum_{k=1}^{N}\left(c^{(k)}\right)^2\right), \tag{6}$$

where $g : R \to R$ is a function of one variable which depends on the criterion Φ.

Comments.

- So, if we allow at least n calls to a black-box program f, then the deterministic numerical differentiation algorithm is optimal for error estimation; when we allow fewer than n calls, we have to use a randomized algorithm.
- As we will see from the proof, a similar result holds if instead of estimating the length c of the vector $\vec{c}$, we estimate an arbitrary *function* of this length. For example, if we want to estimate $\sigma^2 = c^2$, then it is natural to choose a criterion

$$\Phi(\widetilde{c}, c) = \left(\frac{(\widetilde{c})^2 - c^2}{c^2}\right)^2.$$

For this criterion, the optimal estimate comes from

$$g(z) = \sqrt{\frac{n+2}{N+2}} \cdot z :$$

that $g(z) = k \cdot z$ follows from the fact that the problem is invariant under scaling $z \to \lambda \cdot z$ for an arbitrary $\lambda > 0$, and the specific optimal choice of k comes from the explicit computations.

- A similar theorem holds if instead of *arbitrary* (n, N)-algorithms, we will only consider algorithm which produce *un-biased* estimates for c or for an appropriate function of c. For example, if we are estimating $c^2 = \sigma^2$, and we want to choose an unbiased estimate which is optimal under the above criterion, then we get the estimate

$$(\widetilde{c})^2 = \frac{n}{N} \cdot \sum_{k=1}^{N} \left(c^{(k)}\right)^2.$$

  Here, $g(z) = (n/N) \cdot z$.

- When we choose the desired accuracy of an error estimation algorithm, then we select the algorithm which requires the smallest number of calls to the black-box program $f(x_1, \ldots, x_n)$. For the three algorithms that we have so far described – numerical differentiation, Monte-Carlo simulation, and the new (optimal) algorithm – a typical dependence of the corresponding number of calls on the number of variables n is depicted in Figure 1.1.

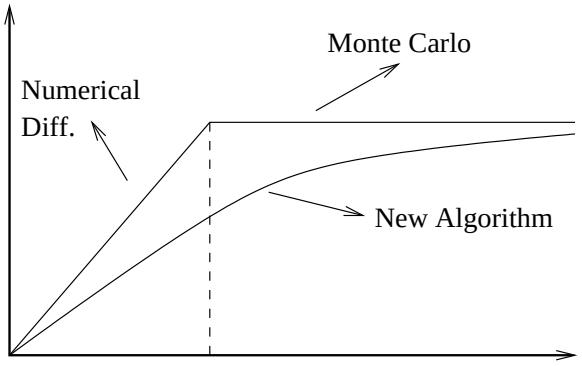

Figure 1.1 Comparison of different algorithms

2.3 HOW TO ACTUALLY PROGRAM THIS OPTIMAL ALGORITHM

The mathematical description of the above optimal algorithm may look somewhat complex, but in reality, this algorithm is rather easy to program. In this programming, we must know the value δ_0 such that for vectors $\vec{\delta} =$

$(\delta_1, \ldots, \delta_n)$ with $\left\|\vec{\delta}\right\| \leq \delta_0$, we can neglect quadratic terms in the expansion of the (black-box) function $f\left(\vec{\widetilde{x}} + \vec{\delta}\right)$ in δ_i.

First Auxiliary Algorithm: Gram-Schmidt Orthogonalization. In the first auxiliary algorithm, we use the well-known Gram-Schmidt orthogonalization procedure: If we have N vectors $\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}$, then the following formulas produce orthogonal unit vectors $\vec{e}^{(1)}, \vec{e}^{(2)}, \ldots$ which span the same linear space as the vectors $\vec{\delta}^{(k)}$:

$$\vec{e}^{(1)} = \frac{\vec{\delta}^{(1)}}{\left\|\vec{\delta}^{(1)}\right\|}, \tag{7}$$

and then for each $k > 1$,

$$\vec{e}^{(k)} = \frac{\vec{\varphi}^{(k)}}{\left\|\vec{\varphi}^{(k)}\right\|},$$

where

$$\vec{\varphi}^{(k)} = \vec{\delta}^{(k)} - \sum_{i=1}^{k-1} \left(\vec{\delta}^{(k)} \cdot \vec{e}^{(i)}\right) \cdot \vec{e}^{(i)}. \tag{8}$$

Second Auxiliary Algorithm: Gaussian Random Number Generator. We must use a random number generator for generating a normally distributed random variable with 0 average and unit standard deviation, so that consequent calls to this program produce independent and equally distributed random variables (some programming languages have explicit commands for such a generator, others have known algorithms for it).

The Algorithm Itself.

- First, we apply the above random number generator n times, and get n components of the first vector $\vec{a}^{\,(1)}$. Then, we apply this same random number generator n more times, and get n components of the second vector $\vec{a}^{\,(2)}$, etc.
- After we have generated N such vectors, we apply Gram-Schmidt orthogonalization to generate the vectors $\vec{e}^{(1)}, \ldots, \vec{e}^{(N)}$.
- Then, for each k from 1 to N, we compute $c^{(k)} = \vec{c} \cdot \vec{e}^{(k)}$ by applying the function f to the values $\vec{\widetilde{x}} + \delta_0 \cdot \vec{e}^{(k)}$, and then computing $c^{(k)}$ as

  $$c^{(k)} = \frac{1}{\delta_0} \cdot \left(f\left(\vec{\widetilde{x}} + \delta_0 \cdot \vec{e}^{(k)}\right) - \widetilde{y}\right).$$

  (This computation is slightly easier than in the general case, because we know that $\vec{\delta} = \vec{e}^{(k)}$ is a unit vector, so its length $\delta = \left\|\vec{e}^{(k)}\right\|$ equals 1.)

- Finally, we compute the estimate $\widetilde{c}$ by using the formula (6).

3. ERROR ESTIMATION FOR INDIRECT MEASUREMENT: INTERVAL SETTING

3.1 MONTE-CARLO SIMULATION FOR INTERVAL SETTING: A KNOWN ALGORITHM

Main Idea Behind the Known Randomized Algorithm. In statistical setting, Monte-Carlo simulation is a known error estimation algorithm. It is somewhat less well known that a similar simulation algorithm can be designed for interval setting. Let us explain how such an algorithm can be designed. In this explanation, we will try to show that the resulting algorithm is not an unnatural mathematical trick, it does appear naturally. (A reader who is only interested in the algorithm itself, not in its motivations, is welcome to go directly to the next subsubsection.)

In Monte-Carlo algorithm for statistical setting, we start with n independent variables $\alpha_1, \ldots, \alpha_n$ which are normally distributed with 0 average and standard deviation 1. The variable α_i simulates the error Δx_i when the standard deviation σ_i of Δx_i equals 1. To simulate the error Δx_i for a general σ_i, we use $\delta_i = \sigma_i \cdot \alpha_i$. The simulation algorithm is based on the fact that for all possible sequences of real numbers $c_1, \ldots, c_n$, and $\sigma_1 \geq 0, \ldots, \sigma_n \geq 0$, the distribution of the linear combination

$$c = c_1 \cdot \delta_1 + \ldots + c_n \cdot \delta_n = c_1 \cdot \sigma_1 \cdot \alpha_1 + \ldots + c_n \cdot \sigma_n \cdot \alpha_n$$

is the same as for $\sigma \cdot \alpha$, where σ is determined by the formula (3).

To design a similar algorithm for the interval setting, it is therefore desirable to find a new probability distribution which will play the same role for this setting as a standard normal distribution plays for the statistical setting. In other words, we would like to be able to start with n independent variables $\alpha_1, \ldots, \alpha_n$ which are all distributed according to this new distribution. The variable α_i simulates the error Δx_i when the bound Δ_i corresponding to Δx_i equals 1. To simulate the error Δx_i for a general Δ_i, we use $\delta_i = \Delta_i \cdot \alpha_i$.

For this idea to work, we must select the distribution in such a way that if $\alpha_1, \ldots, \alpha_n$ are n independent random variables distributed according to this distribution, then for all possible sequences of real numbers $c_1, \ldots, c_n$, and $\Delta_1 \geq 0, \ldots, \Delta_n \geq 0$, the distribution of the linear combination

$$c = c_1 \cdot \delta_1 + \ldots + c_n \cdot \delta_n = c_1 \cdot \Delta_1 \cdot \alpha_1 + \ldots + c_n \cdot \Delta_n \cdot \alpha_n$$

is the same as for $\Delta \cdot \alpha$, where Δ is determined by the formula (4). Let us find distributions which satisfy this condition.

A natural way to describe a probability distribution is to describe its probability density function $\rho(z)$. However, this may not be the best way to represent

the (unknown) desired probability distribution, because the probability density for the sum of several independent random variables is described by a convolution, and hence, in terms of $\rho(z)$, the above condition becomes a complex integral equation. We therefore need to look for alternative descriptions of a probability distribution which will enable us to handle the sum of independent random variables easier.

Independence of two random variables x and x' means, in particular, that the mathematical expectation $E[x \cdot x']$ of their product is equal to the product of their mathematical expectations: $E[x \cdot x'] = E[x] \cdot E[x']$. Independence of x and x' also means that for any two functions $f(z)$ and $f'(z)$, the variables $f(x)$ and $f'(x')$ are also independent and therefore,

$$E[f(x) \cdot f'(x')] = E[f(x)] \cdot E[f'(x')].$$

Therefore, to use the condition that c is a sum of independent random variables $x_n = c_n \cdot \Delta_n \cdot \alpha_n$, we can use the fact that for an exponential function $f(z) = \exp(a \cdot z)$, we have $f(c) = f(x_1 + \ldots + x_n) = f(x_1) \cdot \ldots \cdot f(x_n)$, and thus, $E[f(c)] = E[f(x_1)] \cdot \ldots \cdot E[f(x_n)]$, i.e.,

$$E[\exp(a \cdot c)] = E[\exp(a \cdot c_1 \cdot \Delta_1 \cdot \alpha_1)] \cdot \ldots \cdot E[\exp(a \cdot c_n \cdot \Delta_n \cdot \alpha_n)].$$

The requirement that c has the same distribution as $\Delta \cdot \alpha$ implies that

$$E[\exp(a \cdot \Delta \cdot \alpha)] = E[\exp(a \cdot c_1 \cdot \Delta_1 \cdot \alpha_1)] \cdot \ldots \cdot E[\exp(a \cdot c_n \cdot \Delta_n \cdot \alpha_n)].$$

When the real part $\mathrm{Re}(a)$ of the coefficient a describing the function $f(z) = \exp(a \cdot z)$ is different from 0, then $|f(z)|$ attains arbitrarily large values either for $z \to \infty$ (if $\mathrm{Re}(a) > 0$), or for $z \to -\infty$ (if $\mathrm{Re}(a) > 0$). In both cases, the mathematical expectations may become infinite, thus making the above formula meaningless. So, to make this formula always meaningful, we should only consider coefficients a with $\mathrm{Re}(a) = 0$, i.e., purely imaginary coefficients $a = \mathrm{i} \cdot \omega$ for real values ω. For such a, all the above expectations are of the form $E[\exp(\mathrm{i} \cdot z)]$ for some real number z. This expression is called a *characteristic function* of the probability distribution and denoted by $\chi(z)$. If we use this notation, then the above formula becomes

$$\chi(\omega \cdot \Delta) = \chi(\omega \cdot c_1 \cdot \Delta_1) \cdot \ldots \cdot \chi(\omega \cdot c_n \cdot \Delta_n).$$

For $n = 1$, $\Delta_1 = 1$, and $c_1 = -1$, the formula (4) leads to $\Delta = 1$, so we can conclude that $\chi(\omega) = \chi(-\omega)$, i.e., that the value $\chi(\omega)$ depends only on the absolute value of ω and not on it sign: $\chi(\omega) = \chi(|\omega|)$.

For $n = 2$, $\Delta_1 = \Delta_2 = 1$, $c_1 > 0$, and $c_2 > 0$, we have $\Delta = c_1 + c_2$, so we can conclude that $\chi(c_1 + c_2) = \chi(c_1) \cdot \chi(c_2)$. It is known that the only measurable function which satisfies this functional equation is $\chi(\omega) = \exp(b \cdot \omega)$ for some real value b [1]. (The proof is that χ's logarithm is an

additive function, and it has been, basically, known since Cauchy that the only measurable additive functions are linear ones.) Hence, for $\omega > 0$, we have $\chi(\omega) = \exp(b \cdot \omega)$, and for general ω, we have $\chi(\omega) = \exp(b \cdot |\omega|)$. Since $\chi(\omega)$ is defined as a mathematical expectation of the expression $\exp(\mathrm{i} \cdot \omega \cdot \alpha)$ whose absolute value is 1, the absolute value of $\chi(\omega)$ cannot exceed 1, thus $b < 0$. For simplicity, we will take $b = -1$ (but any other value will also work).

By definition, the characteristic function is a Fourier transform of a probability density function $\rho(z)$; thus, we can reconstruct the probability density by taking the inverse Fourier transform of $\chi(\omega) = \exp(-|\omega|)$. As a result, we get the probability density $\rho(z) = 1/(\pi \cdot (z^2 + 1))$ called *Cauchy distribution*.

Towards Implementation of the Main Idea. In our simulations, we multiply variables α distributed according to this distribution by positive real numbers Δ. If α is distributed according to this distribution, then the probability density for $\delta = \Delta \cdot \alpha$ is equal to

$$\rho(z) = \frac{\Delta}{\pi \cdot (z^2 + \Delta^2)};$$

This more general formula is also called *Cauchy distribution*. The value Δ will be called the *scale parameter* of this distribution, or simply a *parameter*, for short.

The property that we aimed for (and that we will use) is that if $z_1, \ldots, z_n$ are independent random variables, and each of z_i is distributed according to the Cauchy law with parameter Δ_i, then their linear combination $z = c_1 \cdot z_1 + \ldots + c_n \cdot z_n$ is also distributed according to a Cauchy law, with a scale parameter $\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n$.

Therefore, if we take random variables δ_i which are Cauchy distributed with parameters Δ_i, then the value

$$c = f(\widetilde{x}_1 + \delta_1, \ldots, \widetilde{x}_n + \delta_n) - f(\widetilde{x}_1, \ldots, \widetilde{x}_n) = c_1 \cdot \delta_1 + \ldots + c_n \cdot \delta_n$$

is Cauchy distributed with the desired parameter (4). So, repeating this experiment N times, we get N values $c^{(1)}, \ldots, c^{(N)}$ which are Cauchy distributed with the unknown parameter, and from them we can estimate Δ.

The bigger N, the better estimates we get.

There are two questions to be solved:

- how to simulate the Cauchy distribution;
- how to estimate the parameter Δ of this distribution from a finite sample.

Simulation can be based on the functional transformation of uniformly distributed sample values: $\delta_i = \Delta_i \cdot \tan(\pi \cdot (r_i - 0.5))$, where r_i is uniformly distributed on the interval $[0, 1]$.

In order to estimate σ we can apply the Maximum Likelihood Method $\rho(d^1) \cdot \rho(d^2) \cdot \ldots \cdot \rho(d^n) \to \max$, where $\rho(z)$ is a Cauchy distribution density with the unknown Δ. When we substitute the above-given formula for $\rho(z)$ and equate the derivative of the product with respect to Δ to 0 (since it is a maximum), we get an equation

$$\frac{1}{1+\left(\frac{c^{(1)}}{\Delta}\right)^2} + \ldots + \frac{1}{1+\left(\frac{c^{(N)}}{\Delta}\right)^2} = \frac{N}{2}. \tag{9}$$

The left-hand side of (9) is an increasing function that is equal to $0 (< N/2)$ for $\Delta = 0$ and $> N/2$ for $\Delta = \max \left|c^{(k)}\right|$; therefore the solution to the equation (9) can be found by applying a bisection method to the interval $\left[0, \max \left|c^{(k)}\right|\right]$.

So, we arrive at the following algorithm:

Algorithm. For $k = 1, 2, \ldots, N$, repeat the following:

- use the random number generator to compute n numbers $r_i^{(k)}$, $i = 1, 2, \ldots, n$, that are uniformly distributed on the interval $[0, 1]$;
- compute $\delta_i^{(k)} = \Delta_i \cdot \tan(\pi \cdot (r_i^{(k)} - 0.5))$;
- compute the length $\delta^{(k)} = \left\|\vec{\delta}^{(k)}\right\|$ of the vector $\vec{\delta}^{(k)} = (\delta_1^{(k)}, \ldots, \delta_n^{(k)})$;
- compute the normalized coefficient $K_{\mathrm{norm}}^{(k)} = \delta^{(k)}/\delta_0$;
- compute the auxiliary vector $\vec{\beta}^{(k)} = \vec{\delta}^{(k)}/K_{\mathrm{norm}}^{(k)}$ with components $\beta_i^{(k)} = \delta_i^{(k)}/K_{\mathrm{norm}}^{(k)}$;
- substitute $\widetilde{x}_i + \beta_i^{(k)}$ into the program f and compute

$$c^{(k)} = \frac{\delta^{(k)}}{\delta_0} \cdot \left(f\left(\widetilde{x}_1 + \beta_1^{(k)}, \ldots, \widetilde{x}_n + \beta_n^{(k)}\right) - \widetilde{y}\right);$$

- compute Δ by applying the bisection method to solve the equation (9).

Philosophical Comment: Sometimes, Distortion of Simulated Phenomenon Makes Simulation More Efficient. The use of Cauchy distribution in the above algorithm may seem somewhat counter-intuitive (see, e.g., [36, 49]). Indeed, in the interval setting, we do not know the exact probability distribution of each error Δ_i, but we do know that each error Δ_i belongs to the corresponding interval $[-\Delta_i, \Delta_i]$, so the actual (unknown) probability distribution for Δ_i must be located on this interval with probability 1. So, at first

glance, if we want to design a simulation-type technique for computing Δ, we should use one of such possible distributions in our simulations. Instead, we use a Cauchy distribution for which the probability to be outside the interval $[-\Delta_i, \Delta_i]$ is non-zero. In other words, in order to make the simulations work, in these simulations, we *distort* the actual distributions.

At first glance, it may therefore seem natural to use, in our simulations, instead of n independent variables distributed according to Cauchy distribution, n independent variables δ_i distributed according to some distributions which are actually located on the interval $[-\Delta_i, \Delta_i]$. It is sufficient to select a distribution corresponding to $\Delta_i = 1$; let a and σ denote the average and standard deviation of this variable. Then, by scaling (namely, by multiplying by Δ_i), we can get a distribution corresponding to an arbitrary Δ_i. In this case, for each variable δ_i, its average is equal to $\Delta_i \cdot a$, and its standard deviation is equal to $\Delta_i \cdot \sigma$.

As a result of each simulation, we get the value $c_1 \cdot \delta_1 + \ldots + c_n \cdot \delta_n$. For large n, we can apply the limit theorem to this sum and conclude that this value is approximately normally distributed, with an average $\sum c_i \cdot \Delta_i$ and the standard deviation $\sqrt{\sum c_i^2 \cdot \Delta_i^2}$. The larger n, the closer this resulting distribution to normal. It is known that a normal distribution is uniquely determined by its first two moments; hence, for large n, the only information that we will be able to extract from the simulation results are the average and the standard deviation of the resulting distribution. From these two sums $\sum c_i \cdot \Delta_i$ and $\sum c_i^2 \cdot \Delta_i^2$, we cannot uniquely determine the desired value $\sum |c_i| \cdot \Delta_i$. Thus, we cannot use un-distorted simulation, and *distortion is inevitable*.

A general conclusion is: *In simulation, sometimes distorting the simulated process leads to a faster simulation-based algorithm.*

At a more general level, the advantages of simulations with distortions over accurate simulations may explain:

- why an artistic (somewhat geometrically distorted) portrait often captures our impression of a person much better than a (geometrically correct) photo, and
- why, in spite of humans' high optical abilities, we sometimes (as in optical illusions) distort the image that we are trying to reproduce.

When Is This Randomized Algorithm Better Than Deterministic Numerical Differentiation. To determine the parameter Δ, we use the maximum likelihood method. It is known that the error of this method is asymptotically normally distributed, with 0 average and standard deviation $1/\sqrt{N \cdot I}$, where I is Fisher's information:

$$I = \int_{-\infty}^{\infty} \frac{1}{\rho} \cdot \left(\frac{\partial \rho}{\partial \Delta}\right)^2 \, \mathrm{d}z.$$

For Cauchy probability density $\rho(z)$, we have $I = 1/(2\Delta^2)$, so the error of the above randomized algorithm is asymptotically normally distributed, with a standard deviation $\sigma_e \sim \Delta \cdot \sqrt{2/N}$. Thus, if we use a "two sigma" bound, we conclude that with probability 95%, this algorithm leads to an estimate for Δ which differs from the actual value of Δ by $\leq 2\sigma_e = 2\Delta \cdot \sqrt{2/N}$. So, if we want to achieve a 20% accuracy in the error estimation, we must use the smallest N for which $2\sigma_e = 2\Delta \cdot \sqrt{2/N} \leq 0.2 \cdot \Delta$, i.e., to select $N_f = N = 200$.

When it is sufficient to have a standard deviation of 20% (i.e., to have a "two sigma" guarantee of 40%), we need only $N = 50$ calls to f. For $n \approx 10^3$, both values N_f are much smaller than $N_f = n$ required for numerical differentiation.

So, if we have to choose between the (deterministic) numerical differentiation and the randomized Monte-Carlo algorithm, we must select:

- a deterministic algorithm when the number of variables n satisfies the inequality $n \leq N_0$ (where $N_0 \approx 200$), and
- a randomized algorithm if $n \geq N_0$.

A natural question is: is this the optimal choice? In the interval setting, in contrast to the statistical setting, we do not know the optimal algorithm. However, we will still be able to conclude that the answer to the above question is: No, for many different values of n, it is possible to find a new algorithm which is better than both above algorithms. This algorithm was first announced in [45].

3.2 MONTE-CARLO SIMULATION FOR INTERVAL SETTING: A NEW ALGORITHM

Main Idea Behind the New Algorithm. In the interval setting, we have the following error estimation algorithms for indirect measurements:

- we have a (deterministic) *numerical differentiation* algorithm A_n which requires $N_f = n$ calls to a black-box program f and returns the exact result Δ (with 0 standard deviation);
- for every positive integer $N < n$, we have a (randomized) Cauchy-based algorithm A_N which requires N calls to a black-box program f, and which returns an approximate estimate whose standard deviation is $\sim \sqrt{2/N} \cdot \Delta$.

The larger the number N_f of calls to f, the better the estimate of Δ, until we reach $N_f = n$ for which we can get the exact estimate. In many real-life situations, as we have mentioned earlier, we cannot afford n calls to f, so we must use approximate approximate estimates.

Let $\bar{N}$ be the average number of calls which we can afford. Then, instead of simply using the randomized algorithm corresponding to this number of calls, we can use the following *combined* algorithm:

- select probabilities $p_1, \ldots, p_n$ for which $\sum p_i = 1$; and
- with probability p_i, use the algorithm A_i.

The probabilities p_i serve as parameters for the new algorithm. Which values of these parameters should we choose to get the best estimate?

Optimal Choice of Parameters of the New Algorithm. We must select the parameters p_i of the new algorithm so as to minimize the standard deviation (or, equivalently, the variance) of the resulting algorithm under the condition that the average number of calls is equal to a given number $\bar{N}$.

The variance V of an algorithm A_i is equal to $2\Delta^2/i$ for $i < n$ and to 0 for $i = n$; the number of calls for A_i is equal to i. Thus, the above optimization problem takes the following form:

$$V = \sum_{i=1}^{n-1} p_i \cdot \frac{2\Delta^2}{i} \to \min_{p_1,\ldots,p_n} \tag{10}$$

under the conditions that

$$p_i \ge 0; \quad \sum_{i=1}^{n} p_i = 1; \quad \text{and} \quad \sum_{i=1}^{n} p_i \cdot i = \bar{N}. \tag{11}$$

The solution to this problem is given by the following theorem:

Theorem 2. *Let $\bar{n} = \lfloor \bar{N} \rfloor$ denote the integer part of $\bar{N}$, and $\theta = \bar{N} - \bar{n}$ its fractional part. Then, the solution to the problem* (10), (11) *is as follows:*

- *When $\bar{N} \le n/2$, we have $p_{\bar{n}} = 1 - \theta$, $p_{\bar{n}+1} = \theta$, and $p_i = 0$ for all other i. In this case,*
$$V = 2\Delta^2 \cdot \left(\frac{1-\theta}{\bar{n}} + \frac{\theta}{\bar{n}+1} \right).$$
(In particular, when $\bar{N}$ is an integer, we have $p_{\bar{N}} = 1$ and $p_i = 0$ for all other i.)
- *When $\bar{N} > n/2$, we have*
$$p_{\lfloor n/2 \rfloor} = \frac{n - \bar{N}}{\lceil n/2 \rceil}, \quad p_n = \frac{\bar{N} - \lfloor n/2 \rfloor}{\lceil n/2 \rceil},$$
and $p_i = 0$ for all other i. In this case,
$$V = \frac{2\Delta^2 \cdot (n - \bar{N})}{\lfloor n/2 \rfloor \cdot \lceil n/2 \rceil}.$$

Comment. In slightly less formal terms, we can reformulate this theorem as follows:

- when $\bar{N} \leq n/2$, combination does not help much, but
- when $\bar{N} > n/2$, it is better, instead of using a randomized algorithm $A_{\bar{N}}$ with $\bar{N}$ calls to f, to "flip a coin" and use either a randomized algorithm $A_{\lceil n/2 \rceil}$ with $\approx n/2$ calls, or numerical differentiation A_n.

For $\bar{N} > n/2$, the combined algorithm is indeed better: e.g., when n is even and $\bar{N} = n - 1$, the combined algorithm has a variance $8\Delta^2/n^2$, while the randomized algorithm A_{n-1} has a variance $2\Delta^2/(n-1) \sim \Delta^2/n$ which is $\approx n/4$ times larger. This algorithm may be not optimal; finding an optimal algorithm for interval setting is an important *open problem*.

4. NECESSITY AND POSSIBILITY OF PARALLELIZATION

Parallelization Is Necessary. The most time-consuming part of the above algorithms is calling the program f.

Even for the optimal algorithm corresponding to statistical setting, when the number of variables n is large, we need at least 50 calls to f. If each call requires a minute, the resulting time takes about an hour, which may be too long for on-line results.

Since this algorithm is optimal, we cannot decrease the number of calls, so the only way to decrease the total computation time is to run these calls *in parallel*.

Parallelization Is Possible. Not every algorithm can be easily parallelized. In principle, we could have algorithms for which the next vector $\vec{\delta}$ depends on the results of the previous call (and our general definitions allow such algorithms).

Luckily to us, however, for the above optimal algorithm (and for all other algorithms described above), the vectors $\vec{\delta}$ do not depend on the results of the previous calls. Thus, we can make all the calls in parallel (see, e.g., [6, 7, 9, 14, 15, 16, 39, 40, 41, 42, 47, 63]). If we do not have the actual parallel machine, we can use several processors connected into a net [7, 9, 47].

The more processors we have, the less time the resulting computation will take. If we have as many processors as the required number of calls, then the time needed to estimate the error of indirect measurement becomes equal to the time of a single call, i.e., to the time necessary to compute the result $\widetilde{y}$ of this indirect measurement. Thus, if we have enough processors working in parallel, we can compute the result of the indirect measurement *and* estimate its error during the same time that it normally takes just to compute the result.

In particular, if the result $\widetilde{y}$ of indirect measurement can be computed in real time, we can estimate the error of this result in real time as well.

Optimal Choice of Parallel Architecture. The efficiency of parallelization depends on the exact computer architecture. For the above Monte-Carlo-type algorithms, asymptotically optimal parallel architecture is described in [63].

Practical Applications. The above parallelization techniques have been applied to real-life problem: to expert systems [14, 15], to geophysics and petroleum engineering [9, 16, 47], etc.

5. ERROR ESTIMATION FOR INDIRECT MEASUREMENT: MORE COMPLICATED SETTINGS

In the above text, we considered typical settings, when:

- we either know that the errors Δx_i are independent (statistical setting), or we have no information about probabilities at all (interval setting), and
- we have a program which computes $f(x_1, \ldots, x_n)$ exactly.

In real life, in addition to these typical settings, we sometimes encounter more complicated situations. In this section, we describe how the above algorithms can be modified to cover such settings.

5.1 POSSIBLE CORRELATIONS IN STATISTICAL SETTING

In the above text, we assumed that the errors Δx_i are independent random variables with 0 average and known standard deviations σ_i. In real life applications, some of the measurement errors may be caused by the same physical phenomenon and thus, may be correlated. It is therefore desirable to consider a more general setting in which about some variables are known to be independent from each other, while others may be correlated.

When all the variables Δx_i are independent, the standard deviation σ of the error Δy (defined by the formula (1)) is uniquely determined by the known values σ_i (formula (3)). In a more general setting, the actual value of σ may also depend on the correlation between Δx_i; if we do not know this correlation, we can only determine the *upper bound* $\widetilde{\sigma}$ for the standard deviation σ.

We can describe the general information about independence between n variables by an (undirected) *graph* G if we take n vertices corresponding to n variables and connect two vertices by an edge if and only if the corresponding variables are known to be independent. In Section 2, we considered the setting in which this graph G is *complete*. As another extreme case, we can consider

the case when the graph is *fully disconnected*, i.e., when we have no information about independence. In this case, the largest possible value $\widetilde{\sigma}$ of the standard deviation σ is attained when all the variables are fully correlated, and it is given by a formula

$$\widetilde{\sigma} = |c_1| \cdot \sigma_1 + \ldots + |c_n| \cdot \sigma_n. \tag{4a}$$

This formula is identical to the formula (4) from interval setting, and thus, all the above-described interval-setting algorithms can be also used to estimate $\widetilde{\sigma}$.

In real life, in addition to these two extreme cases, we have many intermediate situations. In practical problems, usually, we can divide the variables Δx_i into several classes of related variables so that within each class, variables are related and thus may be dependent, but variables belonging to different classes are mutually independent. In graph terms, such situations can be described by *hierarchical* graphs, in which the vertices of a graph G can be divided into subgraphs $G_1, \ldots, G_k$ so that within each graph G_i there are no edges, but each two vertices belonging to different sets are connected by an edge. In this case, we can find the desired estimate $\widetilde{\sigma}$ as follows:

$$\widetilde{\sigma} = \sqrt{\left(\sigma^{(1)}\right)^2 + \ldots + \left(\sigma^{(k)}\right)^2},$$

where

$$\sigma^{(j)} = \sum_{i \in G_j} |c_i| \cdot \sigma_i.$$

These formulas describe a two-level hierarchy, in which we have subgraphs $G_1, \ldots, G_k$ (level 1) and the graph itself (level 2). These formulas can be naturally generalized to a *multi-level* hierarchy. Such a hierarchy can be described by a *tree* of clusters (subsets of the graph):

- the root of this tree is the graph G itself; this cluster corresponds to the highest level L of the hierarchy;
- its direct children are the clusters $G_1, \ldots, G_k$ of the next level $L - 1$;
- etc., until we reach the leaves of the tree – vertices – "clusters" of the lowest possible level 0.

The intuition behind this hierarchy is that vertices from a node of level 1 may be dependent; clusters of level 1 belonging to a cluster of level 2 are independent; clusters of level 2 belonging to a single cluster of level 3 may be dependent, etc. In other words, to determine whether two vertices are connected or not, we must find the smallest cluster which contains both;

- if this cluster is on the even level, these vertices are connected;
- if this cluster is on the odd level, the vertices are not connected.

For such a hierarchical graph, we can estimate the desired bound $\widetilde{\sigma}$ corresponding to the largest cluster by first estimating similar bounds $\sigma(H)$ for the standard deviation of the sum $\sum_{i \in G} c_i \cdot \Delta x_i$ corresponding first to clusters of level 0, then to clusters of level 1, etc.:

- For clusters of level 0 (i.e., for clusters $H = \{i\}$), we have $\sigma(\{i\}) = |c_i| \cdot \sigma_i$.
- If we have already computed $\sigma(H)$ for all clusters of levels $0, \ldots, k-1$, then, to compute $\sigma(H)$ for clusters of level k, we must do the following:
  - for odd k, we must use the formula
  $$\sigma(H) = \sum_{H_i \subset H} \sigma(H_i),$$
  where the sum is taken over all subclusters of level $k-1$;
  - for even k, we must use the formula
  $$\sigma(H) = \sqrt{\sum_{H_i \subset H} \sigma^2(H_i)}.$$

This generalized situations covers a lot of practical situations, but sometimes, the graph has a more complicated structure. For two variables, they are either connected or not, so nothing more complicated can happen. But already for three variables, we can have a non-hierarchical situation when Δx_1 and Δx_2 are independent, Δx_2 and Δx_3 are independent, but we have no information about the dependence between Δx_1 and Δx_3. This particular 3-variable situation has an explicit solution: $\widetilde{\sigma} = \sqrt{(\sigma^{(1)} + \sigma^{(3)})^2 + (\sigma^{(2)})^2}$, but in general, we do not know of any efficient algorithm which would work for an arbitrary graph.

It is known that normally distributed random variables can be represented as vectors in multi-D space, so that linear combinations turn into linear combinations, and a length of a vector is equal to its standard deviation. Due to this representation, the above *open problem* can be reformulated in geometric terms:

- we have a graph G with n nodes, and we have n positive real numbers $\sigma^{(1)}, \ldots, \sigma^{(n)}$;
- we want to find the largest possible length $\widetilde{\sigma}$ of a vector $\vec{x} = \vec{x}^{\,(1)} + \ldots + \vec{x}^{\,(n)}$ for which the following two conditions hold:
  - for every i, $\left\| \vec{x}^{\,(i)} \right\| = \sigma^{(i)}$, and
  - if nodes i and j are connected in the graph G, then the vectors $\vec{x}^{\,(i)}$ and $\vec{x}^{\,(j)}$ are orthogonal: $\vec{x}^{\,(i)} \cdot \vec{x}^{\,(j)} = 0$.

5.2 POSSIBLE CORRELATIONS IN INTERVAL SETTING

Formulation of the Problem. In the interval setting, in addition to the upper bounds Δ_i on each of the errors Δx_i, we may have some additional information about the relation between the errors Δx_i: e.g., in addition to knowing that $|\Delta x_1| \leq 0.1$ and $|\Delta x_2| \leq 0.1$, we may also know that the sum $\Delta x_1 + \Delta x_2$ of these two errors should also not exceed 0.1. This additional information can be easily reformulated in geometric terms:

- in the standard interval setting, the possible values of the vector $\vec{\Delta} = (\Delta x_1, \ldots, \Delta x_n)$ form a *box* (parallelepiped) $[-\Delta_1, \Delta_1] \times \ldots \times [-\Delta_n, \Delta_n] \subseteq R^n$;
- when we have an additional information about the errors, not all points from this box are possible; as a result, the set G of possible values of the vector $\vec{\Delta}$ can have a more complicated shape than a box.

In this more complicated setting, we also want to be able, given a black-box program $f(x_1, \ldots, x_n)$ and n values $\widetilde{x}_1, \ldots, \widetilde{x}_n$, to determine the interval of possible values of Δy (determined by the formula (1)).

In more precise terms, we are interesting in computing the endpoints of the interval $[\Delta_-, \Delta_+] = [f_-(\vec{c}), f_+(\vec{c})]$, where we denoted

$$f_-(\vec{c}) = \min_{\vec{x} \in G} \vec{c} \cdot \vec{x}, \quad f_+(\vec{c}) = \max_{\vec{x} \in G} \vec{c} \cdot \vec{x}.$$

Solution: Main Idea. We will show that this problem can be solved by an appropriate modification of the Cauchy-distribution based randomized algorithm described in Section 3.1 (this modification was first described in [67] and announced in [49]). The main difference will be that instead of using n independent random variables δ_i, we will now use an appropriate joint distribution for the vector $\vec{\delta}$ in which components are not necessarily independent.

Specifically, we will be using an n-dimensional distribution for which the probability density $\rho(\vec{x})$ is equal to the inverse Fourier transform of the complex function $\exp(\mathrm{i} \cdot f_a(\vec{\omega}) - f_s(\vec{\omega}))$, where $f_a(\vec{\omega})$ and $f_s(\vec{\omega})$ are, correspondingly, the antisymmetric and symmetric parts of the above functions $f_-(\vec{\omega})$ and $f_+(\vec{\omega})$:

$$f_s(\vec{\omega}) = \frac{1}{2} \cdot (f_+(\vec{\omega}) + f_-(\vec{\omega})); \quad f_a(\vec{\omega}) = \frac{1}{2} \cdot (f_+(\vec{\omega}) - f_-(\vec{\omega})).$$

The resulting algorithm consists of two stages:

- first, a *preliminary stage*: when we know the set G, but before we know anything about the function f, we prepare the random number generator to generate numbers distributed according to the probability density $\rho(\vec{c})$;

- then, the *main stage*: for any given black-box program f, we use the random number generator constructed on the preliminary stage to estimate the desired interval $[f_-(\vec{c}), f_+(\vec{c})]$.

Algorithm: Preliminary Stage. To simulate the desired distribution, we first do some preliminary computations:

- First, we compute the values $f_-(\vec{c})$ and $f_+(\vec{c})$ for several values $\vec{c}$.
- Then, we compute the values $f_s(\vec{c})$ and $f_a(\vec{c})$.
- After that, we compute the inverse Fourier transform $\rho(\vec{x})$ of the function $\exp(\mathrm{i} \cdot f_a(\vec{\omega}) - f_s(\vec{\omega}))$.
- The density $\rho(\vec{x})$ tends to 0 as $\|\vec{x}\| \to \infty$, so there exists a number D such that for $\|\vec{x}\| > D$, the value of $\rho(\vec{x})$ is so small that we can safely assume that it is equal to 0; let us compute such a D.
- Finally, we compute the maximum $s = \sup \rho(\vec{x})$ of the density $\rho(\vec{x})$.

Now, we are ready to design the desired random vector generator. Namely, to get a vector $\vec{\delta}$ with the desired probability density function $\rho(\vec{x})$:

- we run $n+1$ times a standard random number generator which produced random numbers $r_1, \ldots, r_n, r_{n+1}$ uniformly distributed on the interval $[0, 1]$;
- then, we form a vector $\vec{x} = (x_1, \ldots, x_n)$ with components $x_i = (2r_i - 1) \cdot D$, and check whether $r_{n+1} \le \rho(\vec{x})/s$;
- if this inequality holds, we return $\vec{\delta} = \vec{x}$, otherwise, we generate r_i anew, etc.

One can easily see that for this algorithm, the probability of a point $\vec{x} \in [-D, D]^n$ to be produced is proportional to $\rho(\vec{x})$, and thus, we get the desired distribution.

Algorithm: Main Stage. For $k = 1, 2, \ldots, N$, repeat the following:

- use the random number generator designed on the preliminary stage to compute a random vector $\vec{\delta}^{(k)} = (\delta_1^{(k)}, \ldots, \delta_n^{(k)})$;
- compute the length $\delta^{(k)} = \left\|\vec{\delta}^{(k)}\right\|$ of the vector $\vec{\delta}^{(k)}$;
- compute the normalized coefficient $K_{\mathrm{norm}}^{(k)} = \delta^{(k)}/\delta_0$;

- compute the auxiliary vector $\vec{\beta}^{(k)} = \vec{\delta}^{(k)}/K_{\rm norm}^{(k)}$ with components $\beta_i^{(k)} = \delta_i^{(k)}/K_{\rm norm}^{(k)}$;
- substitute $\widetilde{x}_i + \beta_i^{(k)}$ into the program f and compute

$$c^{(k)} = \frac{\delta^{(k)}}{\delta_0} \cdot \left(f\left(\widetilde{x}_1 + \beta_1^{(k)}, \ldots, \widetilde{x}_n + \beta_n^{(k)}\right) - \widetilde{y}\right);$$

- compute the average

$$\Delta_a = \frac{1}{N} \cdot \sum_{k=1}^{N} c^{(k)};$$

- compute Δ_s by applying the bisection method to solve the equation

$$\frac{1}{1 + \left(\dfrac{c^{(1)} - \Delta_a}{\Delta_s}\right)^2} + \ldots + \frac{1}{1 + \left(\dfrac{c^{(N)} - \Delta_a}{\Delta_s}\right)^2} = \frac{N}{2}; \qquad (9a)$$

- compute $\Delta_- = \Delta_s - \Delta_a$ and $\Delta_+ = \Delta_s + \Delta_a$.

Comment. The justification of this algorithm is given in the Proofs section.

Special Case When the Set *G* is an Ellipsoid. An important particular case is when the set G is an ellipsoid with a center at $\vec{0}$. Ellipsoids are often used to describe uncertainty domains; see, e.g., [4, 11, 12, 22, 24, 51, 57, 58, 60, 61]. Moreover, it has been shown, both experimentally [11, 12] and theoretically [23], that for many practical problems, ellipsoids are an *optimal* family for describing domains.

For ellipsoids, we can describe the *optimal* error estimation algorithm (the idea of this algorithm was first partly presented in [35]).

Let us start with the case when G is a *sphere* of a given radius r, i.e., the set of all the vectors $\vec{\Delta} = (\Delta x_1, \ldots, \Delta x_n)$ for which $\left\|\vec{\Delta}\right\| \le r$. We want to find the maximum and minimum of the scalar product $\vec{c} \cdot \vec{\Delta}$ under this inequality. Geometrically, the scalar product is equal to $\|\vec{c}\| \cdot \left\|\vec{\Delta}\right\| \cdot \cos(\theta)$, where θ is an angle between the two vectors, so maximum and minimum are attained, correspondingly, when $\cos(\theta) = 1$ or $\cos(\theta) = -1$, i.e., when the vectors are parallel or anti-parallel. Thus, the maximum is this attained for $\vec{\Delta} = \vec{c} \cdot (r/\|\vec{c}\|)$, and the resulting maximum is equal to $\vec{c} \cdot \vec{\Delta} = \|\vec{c}\| \cdot r$. Similarly, the minimum is equal to $-\|\vec{c}\| \cdot r$.

Thus, to find both maximum and minimum, it is sufficient to estimate the length $\|\vec{c}\|$ of the (unknown) gradient vector $\vec{c}$. This is exactly the problem

that we solved in statistical setting, a problem for which we even produced the optimal algorithm. Thus, we can use that algorithm here as well.

This idea can be naturally generalized to a general *non-spherical* ellipsoid. Namely, for each ellipsoid, one can easily find new coordinates $x_i' = \sum_j a_{ij} \cdot x_j$ in which this ellipsoid becomes a sphere. By inverting the matrix a_{ij}, we can get the expression of the old coordinates x_i in terms of the new ones: $x_i = \sum_j b_{ij} \cdot x_j'$, which enables us to re-describe the function f in terms of the new coordinates x_i': $f(x_1, \ldots, x_n) = g(x_1', \ldots, x_n')$, where

$$g(x_1', \ldots, x_n') = f\left(\sum_{j=1}^{n} b_{1j} \cdot x_j', \ldots, \sum_{j=1}^{n} b_{nj} \cdot x_j'\right).$$

For the new function g, the corresponding region G is a sphere, so, as we have mentioned, we can solve the problem of finding the interval of possible values for Δy by using the smallest possible number of calls to g. According to the above explicit formula for g, every call to g consists of a single call to f (plus a few simple computations); thus, we still get the optimal number of calls.

5.3 A PRACTICALLY USEFUL CASE OF KNOWN CORRELATION

Description of the Case: Determining Parameters of a Known Model. In this paper, we consider situations in which the program $f(x_1, \ldots, x_n)$ implements a complex algorithm which is given as a black box. In many practical applications, we have an explicit parameterized model, with unknown parameters $y_1, \ldots, y_m$, which predicts the values of the measured quantities $x_1, \ldots, x_n$ as $x_i = M_i(y_1, \ldots, y_m)$. In these applications, one of the main goals of data processing is to find the values of these parameters which fit with the known measurement results $\widetilde{x}_i$. Usually, the functions M_i which represent these models are either explicitly given or rather easy to compute.

When the functions M_i are highly non-linear, the program f – which solves these equations and actually finds the values y_i – can be really complicated, and thus, require a lot of computation time (much more than computing $M_i(y_1, \ldots, y_m)$ for known values y_j). If we use the above algorithms to estimate the error of the corresponding indirect measurement of y_i, we thus have to make several calls to f and hence, further increase the computation time.

In This Case, It Is Possible to Estimate the Error of Indirect Measurements Without Any Extra Calls to f. Let us show that in this case, we can estimates the errors of indirect measurements much faster than in the general situation: namely, we can do it without any extra calls to the time-consuming program f.

Indeed, since we assumed that the errors Δx_i and Δy_j are small, we can linearize the above system of equations, and get a linear system

$$\Delta x_i = \sum_j M_{i,j} \cdot \Delta x_j.$$

Here, the derivatives $M_{i,j}$ of the functions M_i can be computed, e.g., by applying numerical differentiation to the functions M_i. Since computing M_i is much faster than computing f (i.e., than solving the system of equations), the computation time necessary for computing these partial derivatives is much smaller than a single call to f.

Inverting the matrix $M_{i,j}$, we can explicitly express Δy_j in terms of Δx_i – as $\Delta y_j = \sum_i c_i^{(j)} \cdot \Delta x_i$ with known coefficients $c_i^{(j)}$. Then, we can use the corresponding formula (3) or (4) to estimate the error of indirect measurement of y_j.

This method was successfully used in *pavement engineering* [10, 21].

5.4 WHAT IF THE BLACK BOX PROGRAM IS ITSELF A RANDOMIZED ALGORITHM

Formulation of the Problem. In the previous text, we assumed that we have a black-box program which computes $f(x_1, \ldots, x_n)$ *exactly*. In many practical cases, the black-box program implements a *randomized* algorithm which, instead of computing f exactly, returns an *approximate* value $\widetilde{f}$ of f. In most such cases, in good accordance with the central limit theorem, the distribution of the corresponding error $\Delta f = \widetilde{f} - f(x_1, \ldots, x_n)$ is almost normal, so for all practical purposes, we can assume it to be normal. We can also assume that the estimate $\widetilde{f}$ is unbiased, i.e., that the average approximation error $E[\Delta f]$ is 0.

To get a more accurate estimate, we can run the original black-box (randomized) program several (K) times, get K results $\widetilde{f}^{(1)}, \ldots, \widetilde{f}^{(K)}$, and compute their arithmetic average $f_{\text{av}} = (\widetilde{f}^{(1)} + \ldots + \widetilde{f}^{(K)})/K \approx f(x_1, \ldots, x_n)$. As a result of this averaging, the random error component of Δf will decrease $\sqrt{K}$ times. For example, if we want to decrease it 10 times, we must use $K = 100$.

We can also use these same values $\widetilde{f}^{(1)}, \ldots, \widetilde{f}^{(K)}$ to estimate the standard deviation σ_f of the original Δf as

$$\sigma_f = \sqrt{\frac{1}{K-1} \cdot \sum_{k=1}^{K} \left(\widetilde{f}^{(k)} - f_{\text{av}}\right)^2}.$$

To estimate the error of indirect measurement, we can, in principle, apply one of the above black-box-oriented algorithms directly: namely, every time the algorithm asks us to apply the function f to a vector $(\widetilde{x}_1 + \delta_1, \ldots, \widetilde{x}_n + \delta_n)$, we apply the black-box procedure K times and do the averaging.

The problem with this direct application is that in the above algorithms, we already had quite a few calls to f (for large n, 50 or 200), and if each call requires quite a few (e.g., $K = 100$) applications of the black-box program $\widetilde{f}$, then we have a lot of applications of the black-box program which take a lot of time.

It turns out that if the function f is smooth enough so that the quadratic terms can be neglected for reasonable δ_i (i.e., that δ_0 is large enough), then we can drastically decrease the number of calls to a black box. We will describe the corresponding modified algorithms for statistical and interval settings.

Algorithm for Statistical Setting: Motivation. In the standard Monte-Carlo simulation for statistical setting, we do the following for $k = 1, \ldots, N$:

- first, we simulate random values $\delta_i^{(k)}$ which are independently normally distributed with 0 average and standard deviation σ_i;
- then, we apply the black box program (in our case, $\widetilde{f}$) to the vector $\vec{\widetilde{x}} + \vec{\delta}$ and compute the value

$$c^{(k)} = \widetilde{f}\left(\widetilde{x}_1 + \delta_1^{(k)}, \ldots, \widetilde{x}_n + \delta_n^{(k)}\right) - \widetilde{y}.$$

We know that

$$\widetilde{f}\left(\widetilde{x}_1 + \delta_1^{(k)}, \ldots, \widetilde{x}_n + \delta_n^{(k)}\right) = f\left(\widetilde{x}_1 + \delta_1^{(k)}, \ldots, \widetilde{x}_n + \delta_n^{(k)}\right) + \Delta f^{(k)},$$

where $\Delta f^{(k)}$ is a normally distributed random variable with 0 average and standard deviation σ_f. We also know that

$$f\left(\widetilde{x}_1 + \delta_1^{(k)}, \ldots, \widetilde{x}_n + \delta_n^{(k)}\right) = \widetilde{y} + c_1 \cdot \delta_1^{(k)} + \ldots + c_n \cdot \delta_n^{(k)}.$$

Therefore, $c^{(k)} = c_1 \cdot \delta_1^{(k)} + \ldots + c_n \cdot \delta_n^{(k)} + \Delta f^{(k)}$ is a linear combination of independent normally distributed random variables. Hence, the standard deviation $\widetilde{\sigma}$ of $c^{(k)}$ is equal to

$$\widetilde{\sigma} = \sqrt{c_1^2 \cdot \sigma_1^2 + \ldots + c_n^2 \cdot \sigma_n^2 + \sigma_f^2},$$

i.e., to $\sqrt{\sigma^2 + \sigma_f^2}$. Thus, we can estimate $\widetilde{\sigma}$ by using standard statistical techniques, and then determine σ as $\sigma = \sqrt{(\widetilde{\sigma})^2 - \sigma_f^2}$. As a result, we arrive at the following algorithm:

Algorithm for Statistical Setting. For $k = 1, \ldots, N$, repeat the following:

- use a random number generator to compute n numbers $\alpha_i^{(k)}$, $i = 1, 2, \ldots, n$, that are normally distributed with 0 average and standard deviation 1;
- compute $\delta_i^{(k)} = \sigma_i \cdot \alpha_i^{(k)}$;
- substitute $\widetilde{x}_i + \delta_i^{(k)}$ into the black box $\widetilde{f}$ and compute

$$c^{(k)} = \widetilde{f}\left(\widetilde{x}_1 + \delta_1^{(k)}, \ldots, \widetilde{x}_n + \delta_n^{(k)}\right) - \widetilde{y};$$

- compute

$$\sigma = \sqrt{\frac{1}{N-1} \cdot \sum_{k=1}^{N} \left(c^{(k)}\right)^2 - \sigma_f^2}.$$

Comment. If $\widetilde{f}$ provides a biased estimate, with non-zero $E[\Delta f]$, then this algorithm still works, because when we compute the differences $c^{(k)}$, the biases cancel each other.

Algorithm for Interval Setting: Motivation. If we apply the interval-setting algorithm based on Cauchy distribution, then we conclude that the random variable $c^{(k)}$ is a sum of two independent random components: a component $\vec{c} \cdot \vec{\delta}$ which is Cauchy distributed with the desired parameter Δ and a normally distributed component Δf with 0 average and standard deviation σ_f.

Since these components are independent, the characteristic function $E(\exp(\mathrm{i} \cdot \omega \cdot c))$ of this sum is equal to the product of the corresponding characteristic functions, i.e., to $\chi(\omega) = \exp(-|\omega| \cdot \Delta - \omega^2 \cdot \sigma_f^2)$. Hence, to determine Δ, we can estimate $\chi(\omega)$, compute its negative logarithm, and then compute Δ (see the formula below).

Since the value $\chi(\omega)$ is real, it is sufficient to consider only the real part $\cos(\ldots)$ of the complex exponent $\exp(\mathrm{i} \cdot \ldots)$. Thus, we arrive at the following algorithm:

Algorithm for Interval Setting. For $k = 1, 2, \ldots, N$, repeat the following:

- use a random number generator to compute n numbers $r_i^{(k)}$, $i = 1, 2, \ldots, n$, that are uniformly distributed on the interval $[0, 1]$;
- compute $\delta_i^{(k)} = \Delta_i \cdot \tan(\pi \cdot (r_i^{(k)} - 0.5))$;
- substitute $\widetilde{x}_i + \delta_i^{(k)}$ into the black box $\widetilde{f}$ and compute

$$c^{(k)} = \widetilde{f}\left(\widetilde{x}_1 + \delta_1^{(k)}, \ldots, \widetilde{x}_n + \delta_n^{(k)}\right) - \widetilde{y};$$

- for a real number $\omega > 0$, compute

$$\chi(\omega) = \frac{1}{N} \cdot \sum_{k=1}^{N} \cos\left(\omega \cdot c^{(k)}\right);$$

- compute $\Delta = -\dfrac{\ln(\chi(\omega))}{\omega} - \sigma_f^2 \cdot \dfrac{\omega}{2}$.

Comment. Instead of a *statistical* information about the error Δf, we sometimes only have an upper bound for Δf, i.e., an *interval* of possible values for f. In this case, the exact error estimation for indirect measurements becomes exponentially hard [38].

6. ERROR ESTIMATION ALGORITHMS CAN BE APPLIED TO OTHER PRACTICAL PROBLEMS

In the previous sections, we mentioned that the above black-box-oriented algorithms have been actually used to estimate the error for actual indirect measurements. Let us give two examples in which these same algorithms can be also used in solving other practical problems.

6.1 TOLERANCE ANALYSIS

Main Problem of Tolerance Analysis. The shape of a manufactured object is provided by several manufacturing operations. Each operation produces a simple geometric form, and each operation is not perfectly accurate: the resulting characteristics x_i of a shape (position of a hole, radius of a circle, size of an line, angle, etc) are maintained only within a certain limit $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. The ideal value of the characteristic $\widetilde{x}_i$ is called its *nominal value*, and Δ_i is called *tolerance.*

When after several operations, we assemble the parts into the final product, we want to know the geometric characteristics y of the resulting shape. For the ideal case when all manufacturing operations are precise, the corresponding value $\widetilde{y}$ of the desired characteristic (it is called a *nominal* value of y) have been computed during design. So, we are only interested in the interval of possible deviations $\Delta y = \widetilde{y} - y$ of this characteristic from the nominal one. Computation of such an interval is called *tolerance analysis* [2, 3, 55, 59, 64].

Tolerances are small, so we can usually safely neglect quadratic terms and consider a linearized problem.

From the physical viewpoint, this situation is slightly different from our main problem:

- we do not actually measure x_i; we just rely on the nominal values that have been supplied to us; and

- we do not actually calculate y; we simply use the value that has been pre-calculated during the design.

However, mathematically, we have exactly the same problem as with indirect measurements.

Inverse Problem of Tolerance Analysis. If after estimating the interval of possible values for y, we find out that all these values lie within the desired tolerance interval, then the analyzed manufacturing process is thus verified. However, sometimes, the interval of possible values of y turns out to be too wide. In this case, we are interested in finding the parameters of the manufacturing process that need to be improved.

In principle, this problem can be solved by computing the contribution of each manufacturing parameter. However, if there are many parameters (as, e.g., in VLSI manufacturing), then this algorithm takes too long. It turns out that the ideas of Cauchy-based simulation can lead to a faster algorithm [5].

6.2 PROCESSING EXPERT ESTIMATES

The Need for Expert Estimates. In the interval setting, the above algorithms enable us to compute the upper bound Δ on the error Δy of indirect measurement. This upper bound is based on the guaranteed upper bounds Δ_i for the errors Δx_i of the corresponding direct measurements. These upper bounds must be guaranteed, therefore, they are based on the worst-case estimations. In many cases, the actual error will be much lower. E.g., if in 99% of all measurements the error was ≤ 0.01 V, and in only one case out of 100 it was equal to 0.05 V, then we can only guarantee the error ≤ 0.05 V.

In some cases, the manufacturer also provides us with the probabilities of different values of error; in these cases, we can also estimate probabilities of different values of Δy. But to obtain these probabilities, we must undertake many experiments. The necessary statistical analysis of a sensor is often very time- and cost-consuming, and is, therefore, often avoided.

In many of such cases, when we do not have statistically justified probabilities (because we did not make enough experiments), we often have an opinion of the experts who designed and tested the sensors about what the errors can be. For example, in the above-described case, the guaranteed estimate for a sensor is 0.05 V, but since an expert who designed this sensor knows that usually (in 99 cases out of 100) the error was ≤ 0.01, he can also say that usually, this error is of order ≤ 0.01 V.

He cannot guarantee that, because it is not a precise statement (e.g., what does "of order" mean?). However, this is an additional information about the errors of Δx_i, and it is desirable to use this information in estimating the error Δy of an indirect measurement.

How to Formalize Expert Estimates. We are interested in formalizing a statement "the error e is (usually) of order $\leq \sigma$". This is not a precise statement, therefore, for given e and σ, we cannot say whether it is exactly true or not. If $e \leq \sigma$, this is true; if $e = 1.5\sigma$, this is probably true. If $e = k\sigma$ for some $k > 0$, then, the bigger k, the smaller is our degree of belief that this particular e satisfies this statement.

This degree of belief is usually expressed by numbers from the interval [0,1] (0 corresponds to no belief, 1 to absolute belief, and values between 0 and 1 correspond to intermediate degrees of belief). This idea was first proposed by L. Zadeh [66] under the name of *fuzzy set theory*. There are many different procedures that enable to obtain these numerical values (see, e.g., [19, 32]). If we apply any of these procedures, then for every e, we get the degree of belief $\mu(e)$ that this e satisfies the above statement. In mathematical terms, we get a function from the set of all real numbers to the interval [0,1]. This function is called a *membership function*. So, for each σ, we get a separate membership function. How are these functions related? First, let's notice that $e \leq \sigma$ if and only if $e/\sigma \leq 1$. Therefore, if we know that e is usually of order $\leq \sigma$, then we can conclude that e/σ is usually of order ≤ 1, and vice versa. In other words, for every e and σ, our degree of belief that e is of order $\leq \sigma$ is equal to the degree of belief that e/σ is of order ≤ 1. Therefore, it is sufficient to describe only one membership function $\mu_0(e)$ that describes degrees of belief for $\sigma = 1$. Then, for an arbitrary σ, the degree of belief that e is of order $\leq \sigma$, is equal to $\mu_0(e/\sigma)$.

What are the properties of the function $\mu_0(e)$?

- The value e is of order ≤ 1 if and only if $-e$ is of order ≤ 1; therefore, $\mu_0(-e) = \mu_0(e)$ (i.e., μ_0 is an even function).
- If we consider only positive values of e, then the bigger e, the smaller is our degree of belief that e is of order ≤ 1. In other words, for positive e, μ_0 is a *non-increasing function.*

Because of these two properties, μ_0 is non-decreasing for $e < 0$, and it is non-increasing for $e > 0$. For such functions μ_0, membership functions $\mu_0(e/\sigma)$ were called *LR-numbers* by D. Dubois and H. Prade who introduced them in [17, 18]. They also described how to apply arithmetic operations to these "fuzzy numbers". For detailed expositions of LR-numbers, see [19], Section II.2.B; [30], Section 1.9; [20], Chapter 2 and Section 3.2.4; [69], Chapter 5.

Let's describe their main formulas in application to our problem.

How to Describe a Resulting Expert Estimate For Δy? Extension Principle For LR-Numbers: General Idea. For each i, we know (from the experts) that Δx_i is of order $\leq \sigma_i$. How to compute the membership function that correspond to Δy?

The value Δy is possible if and only if there exists the real numbers Δx_1, $\ldots$, Δx_n such that Δx_1 is of order $\leq \sigma_1$, $\ldots$, Δx_n is of order $\leq \sigma_n$, and after applying formula (1) to these Δx_i, we get this particular value of Δy. In other words, Δy is possible if and only if:

$$\bigvee_{\Delta x_1,\ldots,\Delta x_n} [(\Delta x_1 \text{ is of order } \leq \sigma_1)\,\&\,\ldots\,\&\,(\Delta x_n \text{ is of order } \leq \sigma_n)\,\&$$

$$(\Delta y = c_1 \cdot \Delta x_1 + \ldots + c_n \cdot \Delta x_n)].$$

Here, $\vee$ means "there exists", which is the same as "or" (i.e., either for one set of values Δx_i, or for some other set, the statement in the square brackets must be true).

We know the degrees of the component statements: for "Δx_i is of order $\leq \sigma_i$", the degree of belief is equal to $\mu_0(\Delta x_i/\sigma_i)$. For "$\Delta y = c_1 \cdot \Delta x_1 + \ldots + c + n \cdot \Delta x_n$", the degree of belief is equal to 1 if and only if this statement is true, else it is equal to 0.

So, to find the membership function for Δy, it is sufficient to find out how to estimate our degree of belief $t(A\&B)$ ($t(A \vee B)$) of a composite statement (obtained by applying "and" and "or" to the component ones A and B) if we know the degrees of beliefs $t(A)$ and $t(B)$ of the component statements.

Which $\&$- and $\vee$-Operations Should We Choose. Experiments with expert estimates showed [29, 53, 68] that, on average, the following estimates describe the human reasoning the best: $\min(t(A), t(B))$ for $t(A\&B)$ and $\max(t(A), t(B))$ for $t(A \vee B)$. Second best fits are $t(A) \cdot t(B)$ for $t(A\&B)$ and $t(A) + t(B) - t(A) \cdot t(B)$ for $t(A \vee B)$.

Thus, we have two possibilities for each operation, and hence we must consider four possible combinations of these operations. Let's show that two of them are meaningless for our problem. Namely, we will show that $t(A) + t(B) - t(A) \cdot t(B)$ is not applicable to it.

Indeed, we are applying "or" to infinitely many statements that correspond to different tuples Δx_i. The $\vee$-operation $a, b \to a + b - a \cdot b$ has the following property: when we apply it to several statements with equal degrees of belief, then the resulting degree of belief increases. If we combine two statements, we go from $a = t(A)$ to $2a - a^2 = 1 - (1 - a)^2$; if we combine three statements, we get $1 - (1 - a)^3$, and when we combine k statements, we get $1 - (1 - a)^k$. When $k \to \infty$, the result of this combination tends to 1. So, if we apply this operation to describe the membership function for Δy, we will get a function whose value is 1 for all Δy, which is meaningless.

Therefore, for our problem, the only possible $\vee$-operation is $\max(t(A), t(B))$. As a result, we are left with two cases:

- $t(A \vee B) \approx \max(t(A), t(B))$ and $t(A\&B) \approx \min(t(A), t(B))$;

- $t(A \vee B) \approx \max(t(A), t(B))$, $t(A\&B) \approx t(A)t(B)$.

Let's consider these two cases.

Extension Principle For the Case When $\min$ is Used as an $\&$-Operation. If in the above definition of the term "Δy is possible", we formalize "and" as $\min$, and "or" as $\max$, then we will obtain the following expression for the desired membership function $\mu(\Delta y)$ (i.e., for the degree of belief $\mu(\Delta y)$ that Δy is possible):

$$\mu(\Delta y) = \max_{\Delta x_1, \ldots, \Delta x_n} \left[\min \left(\mu_0 \left(\frac{\Delta x_1}{\sigma_1} \right), \ldots, \mu_0 \left(\frac{\Delta x_n}{\sigma_n} \right) \right) \right],$$

and maximum is taken over all values Δx_i for which $\Delta y = c_1 \cdot \Delta x_1 + \ldots + c_n \cdot \Delta x_n$ (if $\Delta y \neq c_1 \cdot \Delta x_1 + \ldots + c_n \cdot \Delta x_n$, then the degree of belief in this equality is 0, hence the minimum of $n+1$ terms is 0, so these values need not be considered at all).

Comment. This formula is called an *extension principle*. It was proposed by Zadeh in his pioneer paper [66].

In [18], it was shown that $\mu(\Delta y) = \mu_0(\Delta y/\sigma)$, where $\sigma = |c_1| \cdot \sigma_1 + \ldots + |c_n| \cdot \sigma_n$. To describe this membership function, it is therefore sufficient to compute the value σ. The formula relating σ with σ_i is exactly the formula (4). Thus, *we can use the above algorithms for interval setting to find the expert estimate for Δy.*

Extension Principle for the Case When a Product Is Used as an $\&$-Operation. If in the above definition of the term "Δy is possible", we formalize "and" as a product, and "or" as $\max$, then we will obtain the following expression for the desired membership function $\mu(\Delta y)$ (i.e., for the degree of belief $\mu(\Delta y)$ that Δy is possible):

$$\mu(\Delta y) = \max_{\Delta x_1, \ldots, \Delta x_n} \left[\mu_0 \left(\frac{\Delta x_1}{\sigma_1} \right) \cdot \ldots \cdot \mu_0 \left(\frac{\Delta x_n}{\sigma_n} \right) \right],$$

where maximum is taken over all values Δx_i for which $\Delta y = c_1 \cdot \Delta x_1 + \ldots + c_n \cdot \Delta x_n$.

In this case, the result of applying extension principle essentially depends on what exactly membership function we use. Dubois and Prade considered a Gaussian membership function $\mu_0(e) = \exp(-\beta \cdot e^2)$ for some constant $\beta > 0$ (additional arguments in favor of choosing a Gaussian membership function are presented in [46]). For this function, they showed that the extension principle leads to the membership function $\mu_0(\Delta y/\sigma)$, where σ is determined by the formula (3). This is exactly the same formula as for statistical setting, so *we can use the above algorithms for interval setting to find the expert estimate for Δy.*

7. PROOFS

7.1 PROOF OF THEOREM 1

The Main Idea of the Proof. The original problem is rotation-invariant. Therefore, if we have an (n, N)-algorithm for solving this problem which is not rotation-invariant, we can design a new algorithm if we first apply a random rotation and then apply this algorithm to the rotated problem. The resulting new algorithm is rotation-invariant, and because the function (inaccuracy) that we want to minimize is assumed to be convex, additional randomization can only decrease its expected value, and thus, can only improve the algorithm. This, when we are looking for an optimal algorithm, we only need to consider rotation-invariant algorithms. We will show, step-by-step, that the only rotation-invariant algorithm is the algorithm described in Theorem 1.

Comment. For a general overview of how the general ideas of symmetry and invariance help in solving optimization problems in computer-related areas, see, e.g., [50].

When Looking for an Optimal Algorithm, We Can Always Assume That the Vectors $\vec{\delta}^{(k)}$ Produced by an Algorithm Form an Orthonormal Basis. Indeed, let us show that an arbitrary (n, N)-algorithm can be modified, without changing its inaccuracy, into a new algorithm which only deals with the orthonormal bases.

Ultimately, whenever the original algorithm produces a sequence of vectors $\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}$, the new algorithm will produce the Gram-Schmidt orthogonalization $\vec{e}^{(1)}, \ldots, \vec{e}^{(N)}$ of this sequence of vectors. To be more precise, a new algorithm follows the original one, with the following modifications:

- On the 1st step, when the original algorithm produced a vector $\vec{\delta}^{(1)}$ and wants to compute the value $c^{(1)} = \vec{c} \cdot \vec{\delta}^{(1)}$, the new algorithm first produces the unit vector $\vec{e}^{(1)}$ by applying the first orthogonalization step (7), then computes a new scalar product $c^{(1)}_{\rm new} = \vec{c} \cdot \vec{e}^{(1)}$, and then reconstructs $c^{(1)}$ as $c^{(1)} = c^{(1)}_{\rm new} \cdot \left\| \vec{\delta}^{(1)} \right\|$.
- For each $k \geq 2$, on k-th step, when the original algorithm produced a vector $\vec{\delta}^{(k)}$, and wants to compute the value $c^{(k)} = \vec{c} \cdot \vec{\delta}^{(k)}$, the new algorithm first performs the k-th step (8) of the orthogonalization procedure, i.e., produces a vector $\vec{e}^{(k)}$, then computes $c^{(k)}_{\rm new} = \vec{c} \cdot \vec{e}^{(k)}$, and reconstructs the desired value as

$$c^{(k)} = \left(\vec{\delta}^{(k)} \cdot \vec{e}^{(1)}\right) \cdot c^{(1)}_{\rm new} + \ldots + \left(\vec{\delta}^{(k)} \cdot \vec{e}^{(k)}\right) \cdot c^{(k)}_{\rm new}.$$

  The possibility of this reconstruction follows from the fact that the vectors $\vec{e}^{(i)}$, $1 \leq i \leq k$, form an orthonormal basis in a space which includes

$\vec{\delta}^{(k)}$ and therefore,

$$\vec{\delta}^{(k)} = \sum_{i=1}^{k} \left(\vec{\delta}^{(k)} \cdot \vec{e}^{(i)}\right) \cdot \vec{e}^{(i)};$$

multiplying both sides of this equality by $\vec{c}$ and taking into consideration that $\vec{c} \cdot \vec{e}^{(i)} = c_{\text{new}}^{(i)}$, we get the above formula.

The final result of the new algorithm is exactly the same as for the old one, so the inaccuracy of the new algorithm is the same as for the old one. On each step, we replace each call to f (i.e., each computation of a scalar product) by exactly one call (computation); thus, the total number of calls (N) remains the same. However, in the new algorithm, we only compute the scalar product for a sequence of vectors which forms an orthonormal basis.

In view of this transformation, we can, without losing generality, assume that an (n, N)-algorithm only uses orthonormal bases, i.e., that with probability 1, in each trajectory, the vectors $\vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}$ form an orthonormal basis.

A Known Result: There Is Only One Rotation-Invariant Probability Measure on the Set of All Bases. We want to generate random orthonormal bases. Since the problem is rotation-invariant, it is reasonable to look for rotation-invariant probability measures on the set of all such bases.

In our further proof, we will use the fact that there is only one such measure. This result is known, but, for completeness, we will describe the main idea of the proof. To describe the probability measure on the set of all N-element orthonormal bases, we must describe: the probabilities of different values of the first vector $p\left(\vec{e}^{(1)}\right)$, the conditional probabilities of different values of the second vector given the first vector $p\left(\vec{e}^{(2)} \,\middle|\, \vec{e}^{(1)}\right)$, and, for all $k = 3, \ldots, N$, the conditional probabilities $p\left(\vec{e}^{(k)} \,\middle|\, \vec{e}^{(1)}, \ldots, \vec{e}^{(k-1)}\right)$.

For $k = 1$, all possible vectors $\vec{e}^{(1)}$ form a unit sphere (since the bases are orthonormal). On the unit sphere, there is a only one rotation-invariant distribution: uniform distribution on this sphere, for which the probability measure is equal to the normalized Lebesgue measure on this sphere. Similarly, when the vectors $\vec{e}^{(1)}, \ldots, \vec{e}^{(k-1)})$ are fixed, then, due to orthonormality, $\vec{e}^{(k)}$ must be a unit vector orthogonal to all the previous vectors $\vec{e}^{(i)}$, $1 \le i \le k-1$. Thus, all such vectors form a unit sphere in a linear subspace of R^n, namely, on a space $\left[\text{Lin}\left(\vec{e}^{(1)}, \ldots, \vec{e}^{(k-1)}\right)\right]^{\perp}$ which is formed by all vectors orthogonal to all $k-1$ previous vectors. Again, on a unit sphere, there is only one rotation-invariant probability distribution, for which the probability measure is equal to the normalized Lebesgue measure on the corresponding linear subspace.

The uniqueness is proven.

The Algorithm from Section 2.3 Produces a Basis Which Is Random With Respect to a Rotation-Invariant Distribution. Indeed, according to the construction from Section 2.3, each component a_i of each vector $\vec{a} = (a_1, \ldots, a_n)$ is an independent Gaussian variable with 0 average and standard deviation 1. Thus, the corresponding probability density for each component a_i is equal to $\text{const} \cdot \exp(-a_i^2/2)$, and the resulting probability density for each vector $\vec{a}$ is equal to the product of these densities: $\text{const} \cdot \exp(-a_1^2 - \ldots - a_n^2) = \text{const} \cdot \exp\left(-\|\vec{a}\|^2/2\right)$. This probability density depends only on the length $\|\vec{a}\|$ of the vector $\vec{a}$, and is, therefore, rotation-invariant.

Since the distribution of each of the vectors $\vec{a}^{(k)}$ is rotation-invariant, and these vectors are, by construction, statistically independent, the resulting distribution on the set of all tuples $(\vec{a}^{(1)}, \ldots, \vec{a}^{(N)})$ is also rotation-invariant. Since Gram-Schmidt orthogonalization is also a rotation-invariant procedure, the resulting probability distribution on the set of all the bases is also rotation-invariant. The statement is proven.

When Looking for an Optimal Algorithm, We Can Always Assume That the Vectors $\vec{\delta}^{(k)}$ Form an Orthonormal Basis Which Is Random With Respect to a Rotation-Invariant Distribution. Indeed, let us show that an arbitrary (n, N)-algorithm can be modified, without worsening its inaccuracy, into a new algorithm in which orthonormal bases are random with respect to the rotation-invariant distribution.

To produce this new algorithm, we will use a *random rotation* T, which can be defined as a linear transformation that turns the coordinate unit vectors $\vec{b}^{(1)} = (1, 0, \ldots, 0), \ldots, \vec{b}^{(n)} = (0, \ldots, 0, 1)$ into elements of an n-element random orthonormal basis $\vec{e}^{(1)}, \ldots, \vec{e}^{(n)}$. Due to linearity, this rotation turns every vector $\vec{x} = (x_1, \ldots, x_n)$ into a vector $T(\vec{x}) = x_1 \cdot \vec{e}^{(1)} + \ldots + x_n \cdot \vec{e}^{(n)}$.

Now that a random rotation is defined, we can describe the new (n, N)-algorithm. First, we pick a random rotation T. Then, we follow the old algorithm step-by-step. Every time the old algorithm generates a vector $\vec{\delta}^{(k)}$ for computing $c^{(k)} = \vec{c} \cdot \vec{\delta}^{(k)}$, in the new algorithm, we call f with the *new* vector $\vec{\delta}^{(k)}_{\text{new}} = T\left(\vec{\delta}^{(k)}\right)$, i.e., compute $c^{(k)}_{\text{new}} = \vec{c} \cdot T\left(\vec{\delta}^{(k)}\right)$.

Let us show that for the new algorithm, the inaccuracy can only become smaller. Indeed, since the scalar product is rotation-invariant, the new value of $c^{(k)}_{\text{new}}$, which we obtained by combining the new (rotated) vector $\vec{\delta}^{(k)}_{\text{new}}$ with the old vector $\vec{c}$, is equal to the result $T^{-1}(\vec{c}) \cdot \vec{\delta}^{(k)}$ of combining the old (unrotated) vector $\vec{\delta}^{(k)}$ with the new vector $\vec{c}_{\text{new}} = T^{-1}(\vec{c})$ (which is obtained from $\vec{c}$ by the reverse rotation).

If we fix T, the trajectory of an old algorithm for a vector $\vec{c}$ is a trajectory of the new algorithm for the rotated vector $\vec{c}_{\text{new}} = T^{-1}(\vec{c})$. Thus, when T

is fixed, the new algorithm has the same inaccuracies as the old one, except that the new algorithm has them for different vectors $\vec{c}$. So, if we had fixed T, the maximum over $\vec{c}$ would be exactly the same for the old and for the new algorithms.

In the actual new algorithm, there is a new randomization step: instead of fixing T, we choose T according to the (rotation-invariant) random distribution. As a result, for each vector $\vec{c}$, the actual average for a new algorithm is equal to the inaccuracy averaged over different $\vec{c}$ (to be more precise, averaged over all vectors $T^{-1}(\vec{c})$ for a random rotation $\vec{c}$). The average cannot exceed the maximum, and therefore, for every $\vec{c}$, this average (i.e., the inaccuracy of the new algorithm) cannot exceed the worst-case inaccuracy of the old algorithm. Thus, the inaccuracy of the new algorithm is indeed smaller than or equal to the inaccuracy of the old one.

To complete the proof of this statement, it is sufficient to show that in the new algorithm, the resulting probability distribution of the bases $\vec{\delta}^{(k)}_{\rm new}$ is indeed rotation-invariant. Indeed, if we apply an additional rotation S to all bases, it is equivalent to changing from T to the composition $S \circ T$ of the rotations, i.e., equivalently, to replacing a random orthonormal basis $\vec{e}^{(1)}, \ldots, \vec{e}^{(n)}$ in the definition of the rotation T by a rotated basis $S(\vec{e}^{(1)}), \ldots, S(\vec{e}^{(n)})$. The distribution on the set of all bases is rotation-invariant; thus, applying a fixed rotation S to all elements of all the bases should not change the probabilities. Therefore, the resulting distribution of the bases $\vec{\delta}^{(k)}_{\rm new}$ is also rotation-invariant. The statement is proven.

First Preliminary Conclusion. Thus, when we are looking for an optimal algorithm, we can always assume that the bases are distributed according to the rotation-invariant distribution. The only difference between different (n, N)-algorithms of this type is how they estimate c based on these bases and on the results $c^{(k)}$ of multiplying $\vec{c}$ and the vectors from these bases. Let us therefore analyze which estimations lead to smaller inaccuracy.

When Looking for an Optimal Algorithm, We Can Always Assume that the Final Estimate $\widetilde{e}$ Is Deterministic. In our general definitions, we assumed that after we get all the values $c^{(k)}$, we may still need to use some (Monte-Carlo) randomization to produce the final estimate $\widetilde{c}$. Let us show that if this is the case, then we can replace this randomized estimate $\widetilde{c}$ by its deterministic mathematical expectation $\widetilde{c}_{\rm new} = E(\widetilde{c})$ and not increase the algorithm's inaccuracy. (To be more precise, by a mathematical expectation, we mean a *conditional* mathematical expectation, under the condition that the values $c^{(k)}$ and $\vec{\delta}^{(k)}$ are fixed.)

Indeed, since the inaccuracy function $\Phi(\widetilde{c}, c)$ is convex in $\widetilde{c}$, and for convex functions, $\Phi(E(z)) \le E(\Phi(z))$, we have $\Phi(\widetilde{c}_{\rm new}, c) = \Phi(E(\widetilde{c}_{\rm old}), c) \le$

$E(\Phi(\widetilde{c}_{\rm old}, c))$. Therefore, the inaccuracy of the new algorithm (with the deterministic last step) cannot exceed the inaccuracy of the old algorithm (with possibly stochastic last step). The statement is proven.

When Looking for an Optimal Algorithm, We Can Assume That the Final Estimation Depends Only On $c^{(k)}$ And Not on the Vectors $\vec{\delta}^{(k)}$. Indeed, let us take an arbitrary (n, N)-algorithm which have passed through all our previous modifications. In particular, for this algorithm, the final step is deterministic, i.e., $\widetilde{c} = \widetilde{e}\left(c^{(1)}, \ldots, c^{(N)}, \vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}\right)$ for some function $\widetilde{e}$.

For an arbitrary rotation T, we can consider an alternative function

$$\widetilde{e}_T\left(c^{(1)}, \ldots, c^{(N)}, \vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}\right) =$$

$$\widetilde{e}\left(c^{(1)}, \ldots, c^{(N)}, T\left(\vec{\delta}^{(1)}\right), \ldots, T\left(\vec{\delta}^{(N)}\right)\right).$$

The use of this function is equivalent to using the new (rotated) basis $T\left(\vec{\delta}^{(k)}\right)$ *and* the new (rotated) unknown vector $\vec{c}_{\rm new} = T^{-1}(\vec{c})$.

Since we consider an algorithm which has already passed through our improvements, the bases $\vec{\delta}^{(k)}$ are simply randomly distributed in a rotation-invariant way, so adding a rotation T should not change the distribution of these bases. For the same reason, the quality of the new algorithm should not change is we replace $\vec{c}$ by its rotation. Therefore, the new algorithm should have the same inaccuracy as the old one. This is true for all rotations T, i.e., for every T, we can take $\widetilde{c} = \widetilde{e}_T$ and get the same inaccuracy.

Now, we are ready for the final step in the proof of this statement: we can take an *average* of such estimates for all possible rotations T (average over the rotation-invariant measure on all T):

$$\widetilde{c}_{\rm new} = E_T\left[\widetilde{e}_T\left(c^{(1)}, \ldots, c^{(N)}, \vec{\delta}^{(1)}, \ldots, \vec{\delta}^{(N)}\right)\right] =$$

$$E_T\left[\widetilde{e}\left(c^{(1)}, \ldots, c^{(N)}, T\left(\vec{\delta}^{(1)}\right), \ldots, T\left(\vec{\delta}^{(N)}\right)\right)\right].$$

Due to convexity, this averaging can only decrease the inaccuracy of the original algorithm. Since the basis is randomly distributed according to a rotation-invariant distribution, averaging over all T is equivalent to averaging over all possible bases. Hence, this averaging leads to an estimate which does not depend on the basis vectors at all. The statement is proven.

Second Preliminary Conclusion. Thus, when we are looking for optimal algorithms, it is sufficient to consider final estimates of the type $\widetilde{c} = \widetilde{e}\left(c^{(1)}, \ldots, c^{(N)}\right)$. For the following arguments, it is convenient to combine

the N values $c^{(1)}, \ldots, c^{(N)}$ into a single N-dimensional vector $\mathbf{c}$. (We will denote N-dimensional vectors differently to avoid confusion with n-dimensional ones.) In these terms, $\widetilde{c} = \widetilde{e}(\mathbf{c})$.

When Looking for an Optimal Algorithm, We Can Assume That the Final Estimate Only Depends on $\sum (c^{(k)})^2$. This statement can be proved in a similar way, with the only difference that instead of arbitrary rotations in a n-dimensional space, we can consider only rotations within the N-dimensional space generated by the vectors from the basis $\vec{\delta}^{(k)}$, $1 \le k \le N$.

Indeed, if we rotate the basis by such a rotation $\mathbf{T}$, we get a new algorithm, which is equivalent to the old one (because additional rotation does not change the rotation-invariant distribution), but for which the new vectors $\vec{\delta}^{(k)}_{\text{new}} = \sum_l r_{kl} \cdot \vec{\delta}^{(l)}$, where r_{kl} is an $N \times N$ rotation matrix $\mathbf{T}$ in a N-dimensional space. For this new basis, the new values $c^{(k)}_{\text{new}} = \vec{c} \cdot \vec{\delta}^{(k)}_{\text{new}}$ are obtained by the same rotation from the old value $c^{(k)}$: $c^{(k)}_{\text{new}} = \sum_l r_{kl} \cdot c^{(l)}_{\text{old}}$. In other words, the new N-dimensional vector $\mathbf{c}_{\text{new}}$ is obtained from the old vector by a rotation $\mathbf{T}$: $\mathbf{c}_{\text{new}} = \mathbf{T}(\mathbf{c}_{\text{old}})$. Thus, instead of the old estimate $\widetilde{c} = \widetilde{e}(\mathbf{c})$, we can, for every $(N \times N)$-rotation matrix $\mathbf{T}$, consider a new estimate $\widetilde{c} = \widetilde{e}_{\mathbf{T}}(\mathbf{c}) = \widetilde{e}(\mathbf{T}(\mathbf{c}))$.

Now, we are ready for the final step in the proof of this statement: we can take an *average* of such estimates for all possible N-dimensional rotations $\mathbf{T}$ (average over the rotation-invariant measure on all T): $\widetilde{c}_{\text{new}} = E_{\mathbf{T}}\left[\widetilde{e}_{\mathbf{T}}(\mathbf{c})\right] = E_{\mathbf{T}}\left[\widetilde{e}(\mathbf{T}(\mathbf{c}))\right]$. Due to convexity, this averaging can only decrease the inaccuracy of the original algorithm. Since the rotation $\mathbf{T}$ is randomly distributed according to a rotation-invariant distribution, the vector $\mathbf{T}(\mathbf{c})$ is rotation-invariant distributed; all such vectors have the same length as the original N-dimensional vector $\mathbf{c}$. Therefore, the vector $\mathbf{T}(\mathbf{c})$ is uniformly distributed over a sphere of radius $\|\mathbf{c}\| = \sqrt{\sum (c^{(k)})^2}$. Hence, this averaging leads to an estimate which does not depend on the actual values of the components of the vector $\mathbf{c}$, but only depends on this length.

The statement is proven, and so is the theorem.

7.2 PROOF OF THEOREM 2

Main Idea of the Proof. The problem of minimizing (10) under the condition (11) is equivalent to minimizing

$$J = \sum_{i=1}^{n-1} \frac{p_i}{i} \to \min_{p_1, \ldots, p_n} \tag{10a}$$

under the same condition (11). The optimization problem (10a), (11) is a linear programming problem, and it is known that for such problems, the optimum is always attained at one of the vertices. Therefore, to find the

solution to our problem, it is sufficient to enumerate all such vertices, i.e., in this case, all combinations p_i in which at most two probabilities p_i are different from 0. Without losing generality, we can therefore assume that $p_i = 0$ except for two values p_m and p_M, $m < M$, which may be different from 0. From the conditions (11), we conclude that $p_m + p_M = 1$ and that $p_m \cdot m + p_M \cdot M = \bar{N}$. From the first of these equations, we conclude that $p_M = 1 - p_m$. Substituting this expression into the second equation, we conclude that $p_m \cdot m + (1 - p_m) \cdot M = \bar{N}$ and hence,

$$p_m = \frac{M - \bar{N}}{M - m} \quad \text{and} \quad p_M = 1 - p_m = \frac{\bar{N} - m}{M - m}.$$

Since the probabilities have to be positive and $M > m$, we conclude that $\bar{N} \le M$. Similarly, from the fact that $p_M \ge 0$, we can conclude that $m \le \bar{N}$, so $m \le \bar{N} \le M$.

For these values p_m and p_M, the value of the expression (10a) depends on whether $M = n$ or $M < n$. Let us find the find for each of these cases, and then find the actual minimum by comparing the resulting two minima.

First Case: $M < n$. If $M < n$, then the minimized expression (10a) takes the form

$$J = \frac{p_m}{m} + \frac{p_M}{M} = \frac{p_m}{m} + \frac{1 - p_m}{M} = p_m \cdot \left(\frac{1}{m} - \frac{1}{M}\right) + \frac{1}{M} =$$

$$\frac{p_m \cdot (M - m)}{m \cdot M} + \frac{1}{M}.$$

Substituting the above expression for p_m, we conclude that

$$J = \frac{M - \bar{N}}{M \cdot m} + \frac{1}{M}.$$

For a fixed M, this expression is a decreasing function of m; thus, its minimum is attained when m takes the largest possible value. Since m can only take values from 1 to $\bar{N}$, we therefore conclude that $m = \lfloor \bar{N} \rfloor$.

Similarly, we can conclude that

$$J = \frac{p_m}{m} + \frac{p_M}{M} = \frac{1 - p_M}{m} + \frac{p_M}{M} = p_M \cdot \left(\frac{1}{M} - \frac{1}{m}\right) + \frac{1}{m} =$$

$$-\frac{p_M \cdot (M - m)}{m \cdot M} + \frac{1}{m}.$$

Substituting the above expression for p_M, we conclude that

$$J = -\frac{\bar{N} - m}{M \cdot m} + \frac{1}{m}.$$

For a fixed m, this expression is an increasing function of M; thus, its minimum is attained when M takes the smallest possible value. Since M can only take values from $\bar{N}$ to n, we therefore conclude that $M = \lceil \bar{N} \rceil$.

Second Case: $M = n$. In this case, the minimized expression (10a) takes the form

$$J = \frac{p_m}{m} = \frac{n - \bar{N}}{(n-m) \cdot m}.$$

Hence, its minimum is attained when the value $(n-m) \cdot m$ is the largest possible among the values $m \le \bar{N}$.

The function $(n-m) \cdot m$ is increasing until $m = n/2$ and then decreases. So:

- if $\bar{N} > n/2$, the maximum of this function is attained at $m = \lfloor n/2 \rfloor$, and
- if $\bar{N} \le n/2$, its maximum is attained when $m = \lfloor \bar{N} \rfloor$.

Comparing the Two Cases: General Idea. To find the optimal values of m and M, we must compare the optimal values corresponding to the two cases $M < n$ and $M = n$. Since in the second case, we get different values of m depending on whether $\bar{N} > n/2$ or $\bar{N} \le n/2$, let us consider these two situations separately.

Within each two situations, the formulas will be slightly different depending on whether $\bar{N}$ is an integer or not. In this proof, we will describe the generic situation when $\bar{N}$ is not an integer. In this generic case, we have $\lceil \bar{N} \rceil = \bar{n} + 1$.

Situations when $\bar{N}$ is an integer can be either considered separately (and similarly), or can be obtained from the generic situation as a limit case, when $\bar{N}$ tends to an integer.

Comparing the Two Cases When $\bar{N} > n/2$. Let us first show that when $\bar{N} > n/2$, then $M = n$ leads to a better estimate. In this proof, we will consider two subcases: when $\bar{N} > \lceil n/2 \rceil$ and when $n/2 < \bar{N} \le \lceil n/2 \rceil$.

In the first subcase, when $M < n$, we get $m = \lfloor \bar{N} \rfloor = \bar{n}$, $M = \lceil \bar{N} \rceil = \bar{n} + 1$, $p_m = 1 - \theta$, $p_M = \theta$, and the optimized expression J takes the value

$$J_< = \frac{1-\theta}{\bar{n}} + \frac{\theta}{\bar{n}+1}.$$

When $M = n$, we get $m = \lfloor n/2 \rfloor$, so

$$p_m = \frac{n - (\bar{n} + \theta)}{n - \lfloor n/2 \rfloor},$$

and the corresponding value of J is equal to

$$J_{=} = \frac{n - (\bar{n} + \theta)}{(n - \lfloor n/2 \rfloor) \cdot \lfloor n/2 \rfloor}.$$

We want to prove that $M = n$ leads to a better solution, i.e., that $J_{<} \geq J_{=}$. We will prove it in two steps: first, we will prove that $J_{<} \geq J_0$, where by J_0, we denoted $J_0 = 1/(\bar{n} + \theta)$, and then, we will prove that $J_0 \geq J_{=}$.

The inequality $J_{<} \geq J_0$ is equivalent to

$$\frac{1 - \theta}{\bar{n}} + \frac{\theta}{\bar{n} + 1} \geq \frac{1}{\bar{n} + \theta}.$$

Multiplying both sides of this desired inequality by the product of all three denominators, we get an equivalent inequality:

$$(1 - \theta) \cdot (\bar{n} + \theta) \cdot (\bar{n} + 1) + \theta \cdot \bar{n} \cdot (\bar{n} + \theta) \geq \bar{n} \cdot (\bar{n} + 1).$$

If we perform all the multiplications and group together terms proportional to $\bar{n}^2$, to $\bar{n}$, and to 1, we get an equivalent inequality

$$[(1 - \theta) + \theta] \cdot \bar{n}^2 + [(1 - \theta^2) + \theta^2] \cdot \bar{n} + (1 - \theta) \cdot \theta \geq \bar{n}^2 + \bar{n}.$$

Terms proportional to $\bar{n}^2$ and to $\bar{n}$ cancel each other, so we get the equivalent inequality $(1 - \theta) \cdot \theta \geq 0$ which is true for every $\theta \in [0, 1]$. So, $J_{<} \geq J_0$.

Let us now prove the second auxiliary inequality $J_0 \geq J_{=}$. Since $n - \lfloor n/2 \rfloor = \lceil n/2 \rceil$, the inequality $J_0 \geq J_{=}$ can be reformulated in the following equivalent form:

$$\frac{1}{\bar{n} + \theta} \geq \frac{n - (\bar{n} + \theta)}{\lceil n/2 \rceil \cdot (n - \lceil n/2 \rceil)}.$$

Multiplying both sides by the common denominator, we get an equivalent inequality

$$\lceil n/2 \rceil \cdot (n - \lceil n/2 \rceil) \geq (\bar{n} + \theta) \cdot (n - (\bar{n} + \theta)).$$

This inequality is true because, as we mentioned earlier, the function $z \cdot (n - z)$ is decreasing for $z \geq n/2$, and we are considering the case when $\bar{N} = \bar{n} + \theta \geq \lceil n/2 \rceil$. Thus, $J_0 \geq J_{=}$ and hence, $J_{<} \geq J_{=}$, i.e., the solution corresponding to the case $M = n$ is indeed optimal.

To complete the proof for the case $\bar{N} > n/2$, we must now consider the situation when $n/2 < \bar{N} \leq \lceil n/2 \rceil$. This is only possible if $n/2$ is not an integer; then, $\lfloor n/2 \rfloor = \lfloor \bar{N} \rfloor = \bar{n}$, so $\lceil n/2 \rceil = \bar{n} + 1$, and $n = 2\bar{n} + 1$. In this case, the desired inequality $J_{<} \geq J_{=}$ takes the following form:

$$\frac{1 - \theta}{\bar{n}} + \frac{\theta}{\bar{n} + 1} \geq \frac{\bar{n} + 1 - \theta}{\bar{n} \cdot (\bar{n} + 1)}.$$

Multiplying both sides by $\bar{n} \cdot (\bar{n}+1)$ and performing all the multiplications, we get the equality, so, in this case, we also have $J_< \geq J_=$.

Thus, when $\bar{N} > n/2$, the solution corresponding to the case $M = n$ is indeed optimal.

Comparing the Two Cases When $\bar{N} \leq n/2$. Let us now prove that when $\bar{N} \leq n/2$, then $M < n$ leads to a better estimate, i.e., that $J_< \leq J_=$. Here, $J_<$ is described by the same expression as in the case $\bar{N} > n/2$, but the expression for $J_=$ is now different, because for $M = n$, we now have $m = \bar{n}$ and hence

$$J_= = \frac{n - (\bar{n} + \theta)}{\bar{n} \cdot (n - \bar{n})}.$$

In this case, the desired inequality $J_< \leq J_+$ takes the following form:

$$\frac{1-\theta}{\bar{n}} + \frac{\theta}{\bar{n}+1} \leq \frac{n - (\bar{n} + \theta)}{\bar{n} \cdot (n - \bar{n})}.$$

Reducing the left-hand side to the common denominator, we get the equivalent inequality

$$\frac{\bar{n} + 1 - \theta}{\bar{n} \cdot (\bar{n}+1)} \leq \frac{n - (\bar{n} + \theta)}{\bar{n} \cdot (n - \bar{n})}.$$

Multiplying both sides by the common denominator, we get an equivalent inequality

$$(\bar{n} + 1 - \theta) \cdot (n - \bar{n}) \leq (\bar{n}+1) \cdot (n - \bar{n} - \theta).$$

If we perform all the multiplications and cancel equal terms on both sides of the inequality, we get an equivalent inequality $-\theta \cdot (n - \bar{n}) \leq -\theta \cdot \bar{n}$, which, in its turn, is equivalent to $n - \bar{n} \geq \bar{n}$, i.e., to $2\bar{n} \leq n$ and to $\bar{n} \leq n/2$. Since we consider the case $\bar{N} \leq n/2$ and $\bar{n} = \lfloor \bar{N} \rfloor \leq \bar{N}$, therefore, indeed $\bar{n} \leq n/2$, and hence, $J_< \leq J_=$.

Thus, when $\bar{N} \leq n/2$, the solution corresponding to the case $M < n$ is indeed optimal. The theorem is proven.

7.3 CORRELATIONS IN INTERVAL SETTING: JUSTIFICATION OF THE ALGORITHM

Let us show that the values $c^{(k)}$ are distributed according to Cauchy law with the parameter $\Delta_s = f_s(\vec{c})$ and the average $\Delta_a = f_a(\vec{c})$, i.e., that its probability density is equal to

$$\rho(z) = \frac{\Delta_s}{\pi \cdot ((z - \Delta_a)^2 + \Delta_s^2)}.$$

Indeed, by construction, $c = \vec{c} \cdot \vec{\delta}$, where δ is distributed according to the probability density $\rho(\vec{x})$. Thus, the characteristic function

$$\chi(\omega) = E[\exp(\mathrm{i} \cdot \omega \cdot c)]$$

of this distribution is equal to $\chi(\omega) = E[\exp(\mathrm{i} \cdot \omega \cdot \vec{c} \cdot \vec{\delta})]$, i.e., $\chi(\omega) = E[\exp(\mathrm{i} \cdot \vec{\omega} \cdot \vec{\delta})]$, where we denoted $\vec{\omega} = \omega \cdot \vec{c}$.

By definition, the characteristic function $E[\exp(\mathrm{i} \cdot \vec{\omega} \cdot \vec{\delta})]$ is equal to the Fourier transform of the probability density function $\rho(\vec{x})$. Since this density function was defined as an inverse Fourier transform of the expression

$$\exp(\mathrm{i} \cdot f_a(\vec{\omega}) - f_s(\vec{\omega})),$$

we thus conclude that $E[\exp(\mathrm{i} \cdot \vec{\omega} \cdot \vec{\delta})] = \exp(\mathrm{i} \cdot f_a(\vec{\omega}) - f_s(\vec{\omega}))$. Substituting the expression $\vec{\omega} = \omega \cdot \vec{c}$ into this formula, we conclude that

$$\chi(\omega) = \exp(\mathrm{i} \cdot f_a(\omega \cdot \vec{c}) - f_s(\omega \cdot \vec{c})).$$

From the definition of the functions f_- and f_+, we can easily conclude that

- for $\omega > 0$, we have $f_-(\omega \cdot \vec{c}) = \omega \cdot f_-(\vec{c})$ and $f_+(\omega \cdot \vec{c}) = \omega \cdot f_+(\vec{c})$;
- for $\omega < 0$, we have $f_-(\omega \cdot \vec{c}) = \omega \cdot f_+(\vec{c})$ and $f_+(\omega \cdot \vec{c}) = \omega \cdot f_-(\vec{c})$.

Thus, for f_a and f_s, we have, for arbitrary real ω, that $f_a(\omega \cdot \vec{c}) = \omega \cdot f_a(\vec{c})$ and $f_s(\omega \cdot \vec{c}) = |\omega| \cdot f_s(\vec{c})$. Hence, $\chi(\omega) = \exp(\mathrm{i} \cdot \omega \cdot f_a(\vec{c}) - |\omega| \cdot f_s(\vec{c}))$.

This is a known characteristic function of Cauchy distribution with an average $f_a(\vec{c})$ and a parameter $f_s(\vec{c})$. Thus, we can estimate $f_a(\vec{c})$ as the average Δ_a of the sample values $c^{(k)}$. For the values $c^{(k)} - \Delta_a \approx c^{(k)} - f_a(\vec{c})$, the average is 0, so we can use the above maximum likelihood method to determine the parameter $f_s(\vec{c})$ of this distribution.

From the resulting estimates $\Delta_a \approx f_a(\vec{c})$ and $\Delta_s \approx f_s(\vec{c})$, we can conclude that $\Delta_- = \Delta_s - \Delta_a \approx f_s(\vec{c}) - f_a(\vec{c}) = f_-(\vec{c})$ and similarly that $\Delta_+ \approx f_+(\vec{c})$. The justification is completed.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by NSF grants No. DUE-9750858 and CDA-9522207, by United Space Alliance, grant No. NAS 9-20000 (PWO C0C67713A6), 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 number F49620-95-1-0518, and by the National Security Agency under Grant No. MDA904-98-1-0561.

The authors are thankful to the editors for their invitation, and to Chitta Baral and to R. Baker Kearfott for helpful discussions.

References

[1] J. Aczel, *Lectures on functional equations and their applications*, Academic Press, New York, London, 1966.

[2] S. Al Wakil, *Processes and design for manufacturing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[3] ANSI (American National Standards Institute), *Standard* ANSI Y14.5M-1982, Industrial Press, N.Y., 1982.

[4] G. Belforte and B. Bona, "An improved parameter identification algorithm for signal with unknown-but-bounded errors", *Proc. 7th IFAC Symposium on Identification and Parameter Estimation*, York, U.K., 1985.

[5] M. Beltran and V. Kreinovich, "How To Find Input Variables Whose Influence On The Result Is The Largest, or, How To Detect Defective Stages In VLSI Manufacturing?", *Reliable Computing*, 1995, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), pp. 34–37.

[6] A. Bernat, L. Cortes, V. Kreinovich, K. Villaverde, "Intelligent parallel simulation – a key to intractable problems of information processing." *Proceedings of the Twenty-Third Annual Pittsburgh Conference on Modelling and Simulation*, Pittsburgh, PA, 1992, Vol. 2, pp. 959–969.

[7] A. Bernat, E. Villa, K. Bhamidipati, V. Kreinovich, "Parallel interval computations as a background problem: when processors come and go", *International Conference on Interval and Computer-Algebraic Methods in Science and Engineering (Interval'94), St. Petersburg, Russia, March 7-10, 1994, Abstracts*, pp. 51–53.

[8] M. Berz, C. Bischof, G. Corliss, and A. Griewank, *Computational differentiation: techniques, applications, and tools*, SIAM, Philadelphia, 1996.

[9] K. Bhamidipati, "PVM estimates errors caused by imprecise data", In: *Proceedings of the 1994 PVM Users' Group Meeting, Oak Ridge, TN, May 19–20*, Center for Research on Parallel Computations, Session 2A, 1994.

[10] C.-C. Chang, *Fast Algorithm That Estimates the Precision of Indirect Measurements*, Master Project, Computer Science Dept., University of Texas at El Paso, 1992.

[11] F. L. Chernousko, *Estimation of the phase space of dynamic systems*, Nauka publ., Moscow, 1988 (in Russian).

[12] F. L. Chernousko, *State estimation for dynamic systems*, CRC Press, Boca Raton, FL, 1994.

[13] A. A. Clifford, *Multivariate error analysis*, J. Wiley & Sons, N.Y., 1973.

[14] L. A. Cortes, "How to design an expert system that for a given query Q, computes the interval of possible values of probability $p(Q)$ that Q is true", *Abstracts for a Workshop on Interval Methods in Artificial Intelligence, International Conference on Numerical Analysis with Automatic Result Verification: Mathematics, Application and Software, February 25–March 1, 1993,* Lafayette, LA, 1993, p. 11.

[15] L. A. Cortes, *Calculating belief probabilities and intervals*, Master Thesis, Department of Computer Science, University of Texas at El Paso, May 1994.

[16] D. I. Doser, K. D. Crain, M. R. Baker, V. Kreinovich, and M. C. Gerstenberger "Estimating uncertainties for geophysical tomography", *Reliable Computing*, 1998, Vol. 4, No. 3, pp. 241–268.

[17] D. Dubois and H. Prade, "Operations on fuzzy numbers", *International Journal of Systems Science*, 1978, Vol. 9, pp. 613–626.

[18] D. Dubois and H. Prade, "Fuzzy real algebra: some results", *Fuzzy Sets and Systems*, 1979, Vol. 2, pp. 327–348.

[19] D. Dubois and H. Prade, *Fuzzy sets and systems: theory and applications*, Academic Press, N.Y., London, 1980.

[20] D. Dubois and H. Prade, *Possibility theory. An approach to computerized processing of uncertainty*, Plenum Press, N.Y. and London, 1988.

[21] C. Ferregut, S. Nazarian, K. Vennalganti, C.-C. Chang, and V. Kreinovich, "Fast Error Estimates For Indirect Measurements: Applications To Pavement Engineering", *Reliable Computing*, 1996, Vol. 2, No. 3, pp. 219–228.

[22] A. F. Filippov, "Ellipsoidal estimates for a solution of a system of differential equations", *Interval Computations*, 1992, No. 2(4), pp. 6–17.

[23] A. Finkelstein, O. Kosheleva, and V. Kreinovich, "Astrogeometry, error estimation, and other applications of set-valued analysis", *ACM SIGNUM Newsletter*, 1996, Vol. 31, No. 4, pp. 3–25.

[24] E. Fogel and Y. F. Huang, "On the value of information in system identification. Bounded noise case", *Automatica*, 1982, Vol. 18, pp. 229–238.

[25] W. A. Fuller, *Measurement error models*, J. Wiley & Sons, New York, 1987.

[26] C. F. Gerald and P. O. Wheatley, *Applied Numerical Analysis*, Addison-Wesley, Reading, MA, 1994.

[27] A. Griewank, *Evaluating derivatives: Principles and techniques of algorithmic differentiation*, SIAM, Philadelphia, 2000.

[28] H. G. Hecht, *Mathematics in chemistry. An introduction to modern methods*, Prentice Hall, Englewood Cliffs, NJ, 1990.

[29] H. M. Hersch and A. Caramazza, “A fuzzy-set approach to modifiers and vagueness in natural languages", *J. Exp. Psychol.: General*, 1976, Vol. 105, pp. 254–276.

[30] A. Kauffman and M. M. Gupta, *Introduction to fuzzy arithmetic: Theory and applications*, Van Nostrand, N.Y., 1985.

[31] R. B. Kearfott and V. Kreinovich (eds.), *Applications of interval computations*, Kluwer, Dordrecht, 1996.

[32] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications*, Prentice Hall, Upper Saddle River, NJ, 1995.

[33] E. Koltik, V. G. Dmitriev, N. A. Zheludeva, and V. Kreinovich, “An optimal method for estimating a random error component,” *Investigations in Error Estimation*, Proceedings of the Mendeleev Metrological Institute, Leningrad, pp. 36–41, 1986 (in Russian).

[34] V. Ya. Kreinovich, “A General Approach to Analysis of Uncertainty in Measurements”, *Proceedings of the the 3-rd USSR National Symposium on Theoretical Metrology*, Leningrad, Mendeleev Metrology Institute (VNIIM), 1986, pp. 187–188 (in Russian).

[35] V. Kreinovich, *A method of error estimation for indirect measurements for the case when the set of values of input variables forms an ellipsoid*, Center for New Information Technology “Informatika”, Leningrad, 1989 (in Russian).

[36] V. Kreinovich, *In simulation modeling, sometimes simulation with distortions is useful*, Center for New Information Technology “Informatika”, Leningrad, 1989 (in Russian).

[37] V. Kreinovich, “A simplified version of the tomography problem can help to estimate the errors of indirect measurements”, In: A. Mohamad-Djafari (ed.), *Bayesian Inference for Inverse Problems*, Proceedings of the SPIE/International Society for Optical Engineering, Vol. 3459, San Diego, CA, 1998, pp. 106–115.

[38] V. Kreinovich, “Error estimation for indirect measurements is exponentially hard,” *Neural, Parallel, and Scientific Computations*, 1994, Vol. 2, No. 2, pp. 225–234.

[39] V. Kreinovich and A. Bernat, “Parallel algorithms for interval computations: an introduction”, *Interval Computations*, 1994, No. 3, pp. 6–62.

[40] V. Kreinovich, A. Bernat, E. Villa and Y. Mariscal, “Parallel computers estimate errors caused by imprecise data", *Proceedings of the Fourth ISMM (International Society on Mini and Micro Computers) International Conference on Parallel and Distributed Computing and Systems*, Washington, 1991, Vol. 1, pp. 386–390.

[41] V. Kreinovich, A. Bernat, E. Villa, and Y. Mariscal, “Parallel computers estimate errors caused by imprecise data”, *Interval Computations*, 1991, Vol. 2, pp. 21–46.

[42] V. Kreinovich, A. P. Bernat, E. Villa, and Y. Mariscal, “Parallel computers estimate errors caused by imprecise data", *Technical Papers of the the Society of Mexican American Engineers and Scientists 1992 National Symposium*, San Antonio, Texas, April 1992, pp. 192–199.

[43] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

[44] V. Kreinovich, M. I. Pavlovich, “Error estimate of the result of indirect measurements by using a calculational experiment”, *Measurement Techniques*, 1985, Vol. 28, No. 3, pp. 201–205.

[45] V. Kreinovich, S. A. Starks, and R. Trejo, “Automatic Differentiation or Monte-Carlo Methods: Which is Better for Error Estimation?”, *Abstracts of the SIAM Annual Meeting*, Toronto, July 13–17, 1998, p. 51.

[46] V. Kreinovich, C. Quintana, L. Reznik, “Gaussian membership functions are most adequate in representing uncertainty in measurements", *Proceedings of NAFIPS’92: North American Fuzzy Information Processing Society Conference*, Puerto Vallarta, Mexico, December 15–17, 1992, NASA Johnson Space Center, Houston, TX, 1992, Vol. II, pp. 618–624.

[47] D. Morgenstein and J. Murphy, “An application of parallel interval techniques to geophysics”, *Reliable Computing*, 1995, Supplement (Extended Abstracts of APIC’95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), p. 155

[48] *MU* 25.750 − 85. *Methods of calibration, estimation, and testing for metrological characteristics of information processing algorithms*, Industrial Standard, Leningrad, VNIIEP, 1985 In Russian).

[49] S. Nesterov and V. Kreinovich, “The worse, the better: a survey of paradoxical computational complexity of interval computations", In: M. A. Campos (ed.), *Abstracts of the II Workshop on Computer Arithmetic, Interval and Symbolic Computation (WAI’96)*, Recife, Pernambuco, Brazil, August 7-8, 1996, pp. 61A–63A.

[50] H. T. Nguyen and V. Kreinovich, *Applications of continuous mathematics to computer science*, Kluwer, Dordrecht, 1997.

[51] J. P. Norton, “Identification and application of bounded parameter models", *Proc. 7th IFAC Symposium on Identification and Parameter Estimation*, York, U.K., 1985.

[52] P. V. Novitskii and I. A. Zograph, *Estimating the measurement errors*, Energoatomizdat, Leningrad, 1991 (in Russian).

[53] G. C. Oden, “Integration of fuzzy logical information", *Journal of Experimental Psychology: Human Perception Perform.*, 1977, Vol. 3, No. 4, pp. 565–575.

[54] A. I. Orlov, “How often are the observations normal?”, *Industrial Laboratory*, 1991, Vol. 57, No. 7, pp. 770–772.

[55] D. E. Punccohar, *Interpretation of geometric dimensioning and tolerancing*, Society of Manufacturing Engineers (SME) Press, MI, 1990.

[56] S. Rabinovich, *Measurement errors: theory and practice*, American Institute of Physics, N.Y., 1993.

[57] F. C. Schweppe, “Recursive state estimation: unknown but bounded errors and system inputs", *IEEE Transactions on Automatic Control*, 1968, Vol. 13, p. 22.

[58] F. C. Schweppe, *Uncertain dynamic systems*, Prentice Hall, Englewood Cliffs, NJ, 1973.

[59] S. G. Shina, *Concurrent engineering and design for manufacturing of electronics products*, Van Nostrand Reinhold, N.Y., 1991.

[60] S. T. Soltanov, “Asymptotic of the function of the outer estimation ellipsoid for a linear singularly perturbed controlled system", In: S. P. Shary and Yu. I. Shokin (eds.), *Interval Analysis*, Krasnoyarsk, Academy of Sciences Computing Center, Publication No. 17, 1990, pp. 35–40 (in Russian).

[61] G. S. Utyubaev, “On the ellipsoid method for a system of linear differential equations", In: S. P. Shary (ed.), *Interval Analysis*, Krasnoyarsk, Academy of Sciences Computing Center, Publication No. 16, 1990, pp. 29–32 (in Russian).

[62] S. A. Vavasis, *Nonlinear optimization: complexity issues*, Oxford University Press, N.Y., 1991.

[63] E. Villa, A. Bernat, and V. Kreinovich, “Estimating errors of indirect measurement on realistic parallel machines: routings on 2-D and 3-D meshes that are nearly optimal”, *Interval Computations*, 1993, No. 4, pp. 154–175.

[64] O. R. Wade, *Tolerance control in design and manufacturing*, Industrial Press, N.Y., 1989.

[65] Website on interval computations: http://cs.utep.edu/interval-comp.

[66] L. Zadeh, “Fuzzy sets", *Information and control*, 1965, Vol. 8, pp. 338–353.

[67] N. A. Zheludeva and V. Kreinovich, *A method of error estimation for indirect measurements for the case when the set of values of input variables forms a domain*, Center for New Information Technology “Informatika”, Leningrad, 1989 (in Russian).

[68] H. J. Zimmerman, “Results of empirical studies in fuzzy set theory". In: G. Klir, ed., *Applied General System Research*, Plenum, New York, 1978, pp. 303–312.

[69] H. J. Zimmerman, *Fuzzy set theory and its applications*, Kluwer, Dordrecht, 1985.