Chapter I

How to Efficiently Process Uncertainty within a Cyberinfrastructure without Sacrificing Privacy and Confidentiality

Luc Longpré [I.1], Vladik Kreinovich [I.2]

In this paper, we propose a simple solution to the problem of estimating uncertainty of the results of applying a black-box algorithm – without sacrificing privacy and confidentiality of the algorithm.

I.1 Cyberinfrastructure and Web Services

I.1.1 Practical Problem: Need to Combine Geographically Separate Computational Resources

In different knowledge domains in science and engineering, there is a large amount of data stored in different locations, and there are many software tools for processing this data, also implemented at different locations. Users may be interested in different information about this domain.

[I.1] Department of Computer Science, University of Texas at El Paso, 500 W. University, El Paso, TX 79968, USA, `longpre@utep.edu, http://www.faculty.utep.edu/longpre`

[I.2] Department of Computer Science, University of Texas at El Paso, 500 W. University, El Paso, TX 79968, USA, `vladik@utep.edu, http://www.cs.utep.edu/vladik`

Sometimes, the information required by the user is already stored in *one of* the *databases.* For example, if we want to know the geological structure of a certain region in Texas, we can get his information from the geological map stored in Austin. In this case, all we need to do to get an appropriate response to the query is to get this data from the corresponding database.

In other cases, different pieces of the information requested by the user are *stored at different locations.* For example, if we are interested in the geological structure of the Rio Grande Region, then we need to combine data from the geological maps of Texas, New Mexico, and the Mexican state of Chihuahua. In such situations, a correct response to the user's query requires that we access these pieces of information from different databases located at different geographic notations.

In many other situations, the appropriate answer to the user's request requires that we not only collect the relevant data $x_1, \ldots, x_n$, but that we also use some *data processing* algorithms $f(x_1, \ldots, x_n)$ to process this data. For example, if we are interested in the large-scale geological structure of a geographical region, we may also use the gravity measurements from the gravity databases. For that, we need special algorithms to transform the values of gravity at different locations into a map that describes how the density changes with location. The corresponding data processing programs often require a lot of computational resources; as a result, many such programs reside on computers located at supercomputer centers, i.e., on computers which are physically separated from the places where the data is stored.

The need to combine computational resources (data and programs) located at different geographic locations seriously complicates research.

I.1.2 Centralization of Computational Resources – Traditional Approach to Combining Computational Resources; Its Advantages and Limitations

Traditionally, a widely used way to make these computational resources more accessible was to move all these resources to a *central location.* For example, in the geosciences, the US Geological Survey (USGS) was trying to become a central depository of all relevant geophysical data. However, this centralization requires a large amount of efforts: data are presented in different formats, the existing programs use specific formats, etc. To make the central data depository efficient, it is necessary:

- to reformat all the data,
- to rewrite all the data processing programs – so that they become fully compatible with the selected formats and with each other,
- etc.

The amount of work that is needed for this reformatting and rewriting is so large that none of these central depositories really succeeded in becoming an easy-to-use centralized database.

I.1.3 Cyberinfrastructure – A More Efficient Approach to Combining Computational Resources

Cyberinfrastructure technique is a new approach that provides the users with the efficient way to submit requests without worrying about the geographic locations of different computational resources – and at the same time avoid centralization with its excessive workloads. The main idea behind this approach is that *we keep all (or at least most) the computational resources*

- *at their current locations*,
- *in their current formats.*

To expedite the use of these resources:

- we supplement the local computational resources with the "metadata", i.e., with the information about the formats, algorithms, etc.,
- we "wrap up" the programs and databases with auxiliary programs that provide data compatibility into *web services*,

and, in general, we provide a cyberinfrastructure that uses the metadata to automatically combine different computational resources.

For example, if a user is interested in using the gravity data to uncover the geological structure of the Rio Grande region, then the system should automatically:

- get the gravity data from the UTEP and USGS gravity databases,
- convert them to a single format (if necessary),
- forward this data to the program located at San Diego Supercomputer Center, and
- move the results back to the user.

This example is exactly what we are designing under the NSF-sponsored Cyberinfrastructure for the Geosciences (GEON) project; see, e.g., [1, 2, 3, 6, 7, 17, 24, 28, 30, 31, 32, 35, 36]. This is similar to what other cyberinfrastructure projects are trying to achieve.

I.1.4 What Is Cyberinfrastructure: The Official NSF Definition

According to the final report of the National Science Foundation (NSF) Blue Ribbon Advisory Panel on Cyberinfrastructure, "a new age has dawned in scientific and engineering research, pushed by continuing progress in computing, information, and communication technology, and pulled by the expanding complexity, scope, and scale of today's challenges. The capacity of this technology has crossed thresholds that now make possible a comprehensive 'cyberinfrastructure' on which to build new types of scientific and engineering knowledge environments and organizations and to pursue research in new ways and with increased efficacy.

Such environments and organizations, enabled by cyberinfrastructure, are increasingly required to address national and global priorities, such as understanding global climate change, protecting our natural environment, applying genomics-proteomics to human health, maintaining national security, mastering the world of nanotechnology, and predicting and protecting against natural and human disasters, as well as to address some of our most fundamental intellectual questions such as the formation of the universe and the fundamental character of matter."

I.1.5 Web Services: What They Do – A Brief Summary

In different knowledge domains, there is a large amount of data stored in different locations; algorithms for processing this data are also implemented at different locations. Web services – and, more generally, cyberinfrastructure – provide the users with an efficient way to submit requests without worrying about the geographic locations of different computational resources (databases and programs) – and avoid centralization with its excessive workloads [12]. Web services enable the user to receive the desired data $x_1, \ldots, x_n$ and the results $y = f(x_1, \ldots, x_n)$ of processing this data.

I.2 Processing Uncertainty Within a Cyberinfrastructure

I.2.1 Formulation of the problem

The data x_i usually come from measurements or from experts. Measurements are never 100% accurate; as a result, the measured values $\widetilde{x}_i$ are, in general, somewhat different from the actual (unknown) values x_i of the corresponding quantities. Experts can also only provide us with approximate values of the desired quantities.

As a result of this measurement or expert uncertainty, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of data processing is, in general, different from the actual value $y = f(x_1, \ldots, x_n)$ of the desired quantity. It is desirable to gauge this difference. To do that, we must have some information about the errors of direct measurements.

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

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

These probabilities are often described by a normal distribution, so in standard engineering textbook on measurement, it is usually assumed that the distribution of Δx_i is normal, with 0 average and known standard deviation σ_i; see, e.g. [11, 29].

In general, we can determine the desired probabilities of different values of Δx_i by comparing the results of measuring with this instrument with the results of measuring the same quantity by a standard (much more accurate) measuring instrument. Since the standard measuring instrument is much more accurate than the one use, the difference between these two measurement results is practically equal to the measurement error; thus, the empirical distribution of this difference is close to the desired probability distribution for measurement error.

Case of interval uncertainty. There are two cases, however, when in practice, we do not determine the probabilities:

- First is the case of cutting-edge measurements, e.g., measurements in fundamental science. When a Hubble telescope detects the light from a distant galaxy, there is no "standard" (much more accurate) telescope floating nearby that we can use to calibrate the Hubble: the Hubble telescope is the best we have.
- The second case is the case of measurements on the shop floor. In this case, in principle, every sensor can be thoroughly calibrated, but sensor calibration is so costly – usually costing ten times more than the sensor itself – that manufacturers rarely do it.

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

In this case, after we performed a measurement and got a measurement result $\widetilde{x}_i$, the only information that we have about the actual value x_i of the measured quantity is that it belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In such situations, the only information that we have about the (unknown) actual value of $y = f(x_1, \ldots, x_n)$ is that y belongs to the range $\mathbf{y} = [\underline{y}, \overline{y}]$ of the function f over the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$:

$$\mathbf{y} = [\underline{y}, \overline{y}] = \{f(x_1, \ldots, x_n) \mid x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

The process of computing this interval range based on the input intervals $\mathbf{x}_i$ is called *interval computations*; see, e.g., [14, 16].

Case of fuzzy uncertainty. Often, knowledge comes in terms of uncertain expert estimates. In the fuzzy case, to describe this uncertainty, for each value of estimation error Δx_i, we describe the degree $\mu_i(\Delta x_i)$ to which this value is possible.

For each degree of certainty α, we can determine the set of values of Δx_i that are possible with at least this degree of certainty – the *α-cut* $\{x \mid \mu(x) \geq \alpha\}$ of the original fuzzy set. In most cases, this α-cut is an interval.

Vice versa, if we know α-cuts for every α, then, for each object x, we can determine the degree of possibility that x belongs to the original fuzzy set [5, 18, 25, 26, 27]. A fuzzy set can be thus viewed as a nested family of its α-cuts.

So, if instead of a (crisp) interval $\mathbf{x}_i$ of possible values of the measured quantity, we have a fuzzy set $\mu_i(x)$ of possible values, then we can view this information as a family of nested intervals $\mathbf{x}_i(\alpha)$ – α-cuts of the given fuzzy sets.

I.2.2 Description of uncertainty: general formulas

In this paper, we will only consider a typical situation in which the direct measurements and/or expert estimates are accurate enough, so that the resulting approximation errors Δx_i are small, and terms which are quadratic (or of higher order) in Δx_i can be safely neglected. In such situations, 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.

Comment. 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., [16] 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 [22, 34]. 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%.

When approximation errors are small, 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

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

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

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

Case of probabilistic uncertainty. In the statistical setting, the desired measurement error Δy is a linear combination of independent Gaussian variables Δx_i. Therefore, Δy is also normally distributed, with 0 average and the standard deviation

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

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.

Case of interval uncertainty. In the interval setting, we do not know the probability of different errors Δx_i; instead, we only know that $|\Delta x_i| \le \Delta_i$. In this case, the sum (I.1) attains its largest possible value if each term $c_i \cdot \Delta x_i$ in this sum attains the largest possible value:

- If $c_i \ge 0$, 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

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

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 (I.4).

Case of fuzzy uncertainty. We have already mentioned that if instead of a (crisp) interval $\mathbf{x}_i$ of possible values of the measured quantity, we have a fuzzy set $\mu_i(x)$ of possible values, then we can view this information as a family of nested intervals $\mathbf{x}_i(\alpha)$ – α-cuts of the given fuzzy sets.

Our objective is then to compute the fuzzy number corresponding to this the desired value $y = f(x_1, \ldots, x_n)$. In this case, for each level α, to compute the α-cut of this fuzzy number, we can apply interval computations to the α-cuts $\mathbf{x}_i(\alpha)$ of the corresponding fuzzy sets. The resulting nested intervals form the desired fuzzy set for y.

So, e.g., if we want to describe 10 different levels of uncertainty, then we must solve 10 interval computation problems – i.e., apply the formula (I.4) 10 times.

In many practical situations, there is no need to perform 10 computations. For example, it is often reasonable to assume that all the membership functions $\mu_i(x_i)$ have the same shape and only differ by a scaling parameter, i.e., all have the form $\mu_i(\Delta x_i) = \mu_0(\Delta x_i/\Delta_i)$ for some fixed function $\mu_0(x)$ (e.g., triangular or Gaussian). In this case, as it is well known [8, 9, 10, 15], the membership function for Δy has a similar form $\mu_0(\Delta y/\Delta)$, where Δ is determined by the formula (I.4).

Comment. These formulas correspond to the case when in the *extension principle* that describe how the uncertainty in Δx_i transforms into the uncertainty in Δy, we interpret "and" as min, i.e., if we consider

$$\mu(\Delta y) = \max_{\Delta x_1, \ldots, \Delta x_n} \min(\mu_1(\Delta x_1), \ldots, \mu_n(\Delta x_n)), \tag{I.5}$$

where maximum is taken over all the values $\Delta x_1, \ldots, \Delta x_n$ for which the expression (I.1) for Δy leads to the given value of Δy. In general, we can use a different t-norm to combine the values $\mu_i(\Delta x_i)$. For example, we may use the product and describe the resulting membership function as

$$\mu(\Delta y) = \max_{\Delta x_1,\ldots,\Delta x_n} \mu_1(\Delta x_1) \cdot \ldots \cdot \mu_n(\Delta x_n). \tag{I.6}$$

In this case, if we assume that all the membership functions $\mu_i(\Delta x_i)$ are Gaussian, i.e., have the form $\mu_i(\Delta x_i) = \mu_0(\Delta x_i/\sigma_i)$, where $\mu_0(z) = \exp(-z^2)$, then the resulting membership function for Δy is also Gaussian $\mu(\Delta y) = \mu_0(\Delta y/\sigma)$, where σ is determined by the formula (I.3); see, e.g., [8, 9, 10, 15].

I.2.3 Error Estimation for the Results of Data Processing: 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 (I.3) or (I.4).

Let us describe how this problem is solved now.

I.2.4 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 (I.3) and (I.4). This is the case which is typically analyzed in textbooks on measurement theory; see, e.g., [11, 29].

A more complex case: automatic 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 let the compute perform an explicit differentiation; for that, we can use one of the existing automatic differentiation tools (see, e.g., [4, 13]). These tools analyze the code of the program for computing $f(x_1,\ldots,x_n)$ and, as they perform their analysis, they produce the "differentiation code", i.e., a program that computes the partial derivatives c_i.

Once we know an algorithm that computes f in time T, automatic differentiation (AD) enables us to compute all partial derivatives in time $\le 3T$, hence we can compute σ or Δ in time $O(T+n)$.

I.3 Need for Privacy Makes the Problem More Complex

Privacy situation: description. In cyberinfrastructure, the owners of the program f may not want to disclose its code; instead, they may only allow to use f as a black box.

Related difficulty. If we do not know the code of f, then we cannot apply AD to compute all n partial derivatives $c_i = \dfrac{\partial f}{\partial x_i}$.

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 (I.3) or (I.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

$$c_i = \frac{f\left(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n\right) - \widetilde{y}}{h_i}. \qquad \text{(I.7)}$$

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 $\delta_i = \Delta_i$), $\delta_{i+1} = \ldots = \delta_n = 0$.

Problem: sometimes, numerical differentiation takes too long. Very often, the program f requires a reasonable time to compute (e.g., in the geological applications, 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 that it takes us to compute σ or Δ is (approximately) equal to the running time T 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 $T \cdot n$ (which is $\gg T + n$) 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 in minutes. In this case, $T \cdot n$ may take several weeks. This may be OK for a single measurement, but too long if we want more on-line results.

I.4 Solution for Statistical Setting: Monte-Carlo Simulations

Monte-Carlo simulations: main idea. In the statistical setting, we can use straightforward (Monte-Carlo) simulation, and drastically save the computation time. In this approach, 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

$$\sigma = \sqrt{\frac{1}{N-1} \cdot \sum_{k=1}^{N} \left(c^{(k)}\right)^2}. \tag{I.8}$$

Computation time required for Monte-Carlo simulation. The relative error of the above statistical estimate depends only on N (as $\approx 1/\sqrt{N}$), and not on the number of variables n. Therefore, the number N_f of calls to f that is 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 $\le 2\sigma_e = 2\sigma/\sqrt{2N}$.

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 $2\sigma_e = 2\sigma/\sqrt{2N} \le 0.2 \cdot \sigma$, 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 $3\sigma_e = 3 \cdot \sigma/\sqrt{2N} \le 0.2 \cdot \sigma$, 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 $n \le N_0$ (where $N_0 \approx 50$), and
- a randomized method if $n \ge N_0$.

Additional advantage: parallelization. In Monte-Carlo algorithm, we need 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. Fortunately, different calls to the function f are independent on each other, so we can run all the simulations in parallel.

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.

I.5 Solution for Interval and Fuzzy Setting: New Method Based on Cauchy Distribution

Can we use a similar idea in the interval setting? Since Monte-Carlo simulation speeds up computations, it is desirable to use a similar technique in interval setting as well.

There is a problem here. In the interval setting, we do not know the exact distribution, we may have different probability distributions – as long as they are located within the corresponding intervals. If we only use one of these distributions for simulations, there is no guarantee that the results will be valid for other distributions as well.

In principle, we could repeat simulations for several different distributions, but this repetition would drastically increase the simulation time and thus, eliminate the advantages of simulation as opposed to numerical differentiation.

Yes, we can. Luckily, there is a mathematical trick that enables us to use Monte-Carlo simulation in interval setting as well. This trick is based on using *Cauchy distribution* – i.e., probability distributions with the probability density

$$\rho(z) = \frac{\Delta}{\pi \cdot (z^2 + \Delta^2)}; \tag{I.9}$$

the value Δ is called the *scale parameter* of this distribution, or simply a *parameter*, for short.

Cauchy distribution has the following property that we will use: 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

$$c = f(\widetilde{x}_1 + \delta_1, \ldots, \widetilde{x}_n + \delta_n) - f(\widetilde{x}_1, \ldots, \widetilde{x}_n) = c_1 \cdot \delta_1 + \ldots + c_n \cdot \delta_n \tag{I.10}$$

is Cauchy distributed with the desired parameter (I.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:

$$\delta_i = \Delta_i \cdot \tan(\pi \cdot (r_i - 0.5)), \tag{I.11}$$

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

$$\frac{1}{1 + \left(\frac{c^{(1)}}{\Delta}\right)^2} + \ldots + \frac{1}{1 + \left(\frac{c^{(N)}}{\Delta}\right)^2} = \frac{N}{2}. \tag{I.12}$$

The left-hand side of (I.12) 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 (I.12) can be found by applying a bisection method to the interval $\left[0, \max \left|c^{(k)}\right|\right]$.

It is important to mention that we assumed that the function f is reasonably linear within the box

$$[\widetilde{x}_1 - \Delta_1, \widetilde{x}_1 + \Delta_1] \times \ldots \times [\widetilde{x}_n - \Delta_n, \widetilde{x}_n + \Delta_n]. \tag{I.13}$$

However, the simulated values δ_i may be outside the box. When we get such values, we do not use the function f for them, we use a normalized function that is equal to f within the box, and that is extended linearly for all other values (we will see, in the description of an algorithm, how this is done).

As a result, we arrive at the following algorithm (described, for a somewhat different problem, in [20, 21, 23, 33]):

Algorithm.

- Apply f to the results of direct measurements: $\widetilde{y} := f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$;
- For $k = 1, 2, \ldots, N$, repeat the following:
    - use the standard 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 Cauchy distributed values $c_i^{(k)} := \tan(\pi \cdot (r_i^{(k)} - 0.5))$;
    - compute the largest value of $|c_i^{(k)}|$ so that we will be able to normalize the simulated measurement errors and apply f to the values that are within the box of possible values: $K := \max_i |c_i^{(k)}|$;
    - compute the simulated measurement errors $\delta_i^{(k)} := \Delta_i \cdot c_i^{(k)} / K$;

- compute the simulated measurement results $x_i^{(k)} := \widetilde{x}_i + \delta_i^{(k)}$;
- apply the program f to the simulated measurement results and compute the simulated error of the indirect measurement:

$$c^{(k)} := K \cdot \left(f\left(x_1^{(k)}, \ldots, x_n^{(k)}\right) - \widetilde{y} \right);$$

- Compute Δ by applying the bisection method to solve the equation (I.12).

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 $\le 2\sigma_e = 2\Delta \cdot \sqrt{2/N}$. So, if we want to achieve a 20% accuracy in the error estimation, we must use the smallest N for which $2\sigma_e = 2\Delta \cdot \sqrt{2/N} \le 0.2 \cdot \Delta$, 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 $n \le N_0$ (where $N_0 \approx 200$), and
- a randomized algorithm if $n \ge N_0$.

Comment. If we use fewer than N_0 simulations, then we still get an approximate value of the range, but with worse accuracy – and the accuracy can be easily computed by using the above formulas.

This algorithm is naturally parallelizable. Similarly to the Monte-Carlo algorithm for statistical setting, we can run all N simulations in parallel and thus, speed up the computations.

Conclusion. When we know the code for f, then we can use AD and compute Δ and σ in time $O(T + n)$. If the owner of the program f only allows to use it as a black box, then we cannot use AD any more. In principle, we can compute each of n

derivatives $\partial f/\partial x_i$ by numerical differentiation, but this would require computation time $T \cdot n \gg T + n$.

For probabilistic uncertainty, one can use Monte-Carlo simulations and compute σ in time $O(T) \ll T \cdot n$. We have shown that for interval uncertainty, we can also compute Δ in time $O(T)$ by using an artificial Monte-Carlo simulations in which each Δx_i is Cauchy distributed with parameter Δ_i – then simulated Δy is Cauchy distributed with the desired parameter Δ.

Remark: the problem of non-linearity. In the above text, we assumed that the intervals $\mathbf{x}_i$ are narrow. In this case, terms quadratic in Δx_i are negligible, and so, we can safely assume that the desired function $f(x_1, \ldots, x_n)$ is linear on the box

$$\mathbf{x}_1 \times \ldots \times \mathbf{x}_n.$$

In practice, some intervals $\mathbf{x}_i$ may be wide, so even when restricted to the box, the function $f(x_1, \ldots, x_n)$ is non-linear. What can we do in this case?

Usually, experts (e.g., designers of the corresponding technical system) know for which variables x_i, the dependence is non-linear. For each of these variables, we can *bisect* the corresponding interval $[\underline{x}_i, \overline{x}_i]$ into two smaller subintervals – for which the dependence is approximately linear. Then, we estimate the range of the function f separately on each of the resulting subboxes, and take the union of these two ranges as the range over the entire box.

If one bisection is not enough and the dependence of f on x_i is non-linear over one or several subboxes, we can bisect these boxes again, etc.

This bisection idea has been successfully used in interval computations; see, e.g., [14, 16].

I.6 Summary

In different knowledge domains, there is a large amount of data stored in different locations; algorithms for processing this data are also implemented at different locations. Web services – and, more generally, cyberinfrastructure – provide the users with an efficient way to submit requests without worrying about the geographic locations of different computational resources (databases and programs) – and avoid centralization with its excessive workloads [12]. Web services enable the user to receive the desired data $x_1, \ldots, x_n$ and the results $y = f(x_1, \ldots, x_n)$ of processing this data.

The data x_i usually come from measurements or from experts. Measurements are never 100% accurate; as a result, the measured values $\widetilde{x}_i$ are, in general, somewhat different from the actual (unknown) values x_i of the corresponding quantities. Experts can also only provide us with approximate values of the desired quantities.

As a result of this measurement or expert uncertainty, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of data processing is, in general, different from the actual value $y = f(x_1, \ldots, x_n)$ of the desired quantity. It is desirable to gauge this difference.

Traditional methods for estimating the resulting uncertainty in y are based on the assumption that we know the code of the function f. In cyberinfrastructure, owners of the program f may not want to disclose its code; instead, they may only allow to use f as a black box. In this case, traditional techniques are not applicable.

There exist techniques for processing uncertainty under such a “black-box” situation, but these techniques require much longer computation time. In this paper, we describe new Monte-Carlo-type techniques that process uncertainty in such privacy-protecting black-box situations and that require the same amount of computation time as the traditional non-privacy-protecting techniques.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, NSF grants EAR-0225670 and DMS-0532645, Star Award from the University of Texas System, and Texas Department of Transportation grant No. 0-5453.

Bibliography

[1] M.S. Aguiar, G.P. Dimuro, A.C.R. Costa, R.K.S. Silva, F.A. Costa, and V. Kreinovich, The Multi-Layered Interval Categorizer Tesselation-Based Model, In: C. Iochpe and G. Câmara (eds.), IFIP WG2.6 Proceedings of the 6th Brazilian Symposium on Geoinformatics Geoinfo'2004, Campos do Jordão, Brazil, November 22–24, 2004, pp. 437–454. ISBN 3901882200.

[2] R. Aldouri, G.R. Keller, A. Gates, J. Rasillo, L. Salayandia, V. Kreinovich, J. Seeley, P. Taylor, and S. Holloway, GEON: Geophysical data add the 3rd dimension in geospatial studies, Proceedings of the ESRI International User Conference 2004, San Diego, California, August 9–13, 2004, Paper 1898.

[3] M.G. Averill, K.C. Miller, G.R. Keller, V. Kreinovich, R. Araiza, and S.A. Starks, Using Expert Knowledge in Solving the Seismic Inverse Problem, Proceedings of the 24nd International Conference of the North American Fuzzy Information Processing Society NAFIPS'2005, Ann Arbor, Michigan, June 22–25, 2005, pp. 310–314.

[4] M. Berz, C. Bischof, G. Corliss, A. Griewank, Computational Differentiation: Techniques, Applications, and Tools, SIAM, Philadelphia, 1996.

[5] G. Bojadziev and M. Bojadziev, Fuzzy Sets, Fuzzy Logic, Applications, World Scientific, Singapore, 1995.

[6] M. Ceberio, S. Ferson, V. Kreinovich, S. Chopra, G. Xiang, A. Murguia, and J. Santillan, How to take into account dependence between the inputs: from interval computations to constraint-related set computations, with potential applications to nuclear safety, bio- and geosciences, Proceedings of the Second International Workshop on Reliable Engineering Computing, Savannah, Georgia, February 22–24, 2006, pp. 127–154.

[7] M. Ceberio, V. Kreinovich, S. Chopra, and B. Ludäscher, Taylor Model-Type Techniques for Handling Uncertainty in Expert Systems, with Potential Applications to Geoinformatics, Proceedings of the 17th World Congress of the International Association for Mathematics and Computers in Simulation IMACS'2005, Paris, France, July 11–15, 2005.

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

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

[10] D. Dubois and H. Prade, Fuzzy Sets and Systems: Theory and Applications, Academic Press, N.Y., London, 1980.

[11] W.A. Fuller, Measurement Error Models, J. Wiley & Sons, New York, 1987.

[12] A. Gates, V. Kreinovich, L. Longpré, P. Pinheiro da Silva, and G.R. Keller, Towards Secure Cyberinfrastructure for Sharing Border Information, in: Proceedings of the Lineae Terrarum: International Border Conference, El Paso, Las Cruces, and Cd. Juárez, March 27–30, 2006.

[13] A. Griewank, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, SIAM, Philadelphia, 2000.

[14] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, Applied Interval Analysis, Springer Verlag, London, 2001.

[15] A. Kauffman and M.M. Gupta, Introduction to Fuzzy Arithmetic: Theory and Applications, Van Nostrand, N.Y., 1985.

[16] R.B. Kearfott and V. Kreinovich (Eds.), Applications of Interval Computations, Kluwer, Dordrecht, 1996.

[17] G.R. Keller, T.G. Hildenbrand, R. Kucks, M. Webring, A. Briesacher, K. Rujawitz, A.M. Hittleman, D.R. Roman, D. Winester, R. Aldouri, J. Seeley, J. Rasillo, R. Torres, W.J. Hinze, A. Gates, V. Kreinovich, and L. Salayandia, A community effort to construct a gravity database for the United States and an associated Web portal, In: A. K. Sinha (ed.), Geoinformatics: Data to Knowledge, Geological Society of America Publ., Boulder, Colorado, 2006, pp. 21–34.

[18] G. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic, Prentice Hall, New Jersey, 1995.

[19] V. Kreinovich, J. Beck, C. Ferregut, A. Sanchez, G.R. Keller, M. Averill, and S.A. Starks, Monte-Carlo-type techniques for processing interval uncertainty, and their potential engineering applications, Reliable Computing (to appear).

[20] V. Kreinovich, A. Bernat, E. Villa, and Y. Mariscal, Parallel computers estimate errors caused by imprecise data, Interval Computations, No. 2, pp. 21–46, 1991.

[21] V. Kreinovich and S. Ferson, A New Cauchy-Based Black-Box Technique for Uncertainty in Risk Analysis, Reliability Engineering and Systems Safety, Vol. 85, No. 1–3, pp. 267–279, 2004.

[22] V. Kreinovich, A. Lakeyev, J. Rohn, P. Kahl, Computational Complexity and Feasibility of Data Processing and Interval Computations, Kluwer, Dordrecht, 1998.

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

[24] L. Longpré, V. Kreinovich, E. Freudenthal, M. Ceberio, F. Modave, N. Baijal, W. Chen, V. Chirayath, G. Xiang, and J.I. Vargas, Privacy: Protecting, Processing, and Measuring Loss, Abstracts of the 2005 South Central Information Security Symposium SCISS'05, Austin, Texas, April 30, 2005, p. 2.

[25] R.E. Moore and W.A. Lodwick, Interval Analysis and Fuzzy Set Theory, Fuzzy Sets and Systems, Vol. 135, No. 1, pp. 5–9, 2003.

[26] H.T. Nguyen and V. Kreinovich, Nested Intervals and Sets: Concepts, Relations to Fuzzy Sets, and Applications, In [16], pp. 245–290.

[27] H.T. Nguyen and E.A. Walker, First Course in Fuzzy Logic, CRC Press, Boca Raton, Florida, 2005.

[28] E. Platon, K. Tupelly, V. Kreinovich, S.A. Starks, and K. Villaverde, Exact Bounds for Interval and Fuzzy Functions Under Monotonicity Constraints, with Potential Applications to Biostratigraphy, Proceedings of the 2005 IEEE International Conference on Fuzzy Systems FUZZ-IEEE'2005, Reno, Nevada, May 22–25, 2005, pp. 891–896.

[29] S. Rabinovich, Measurement Errors and Uncertainties: Theory and Practice, American Institute of Physics, N.Y., 2005.

[30] C.G. Schiek, R. Araiza, J.M. Hurtado, A.A. Velasco, V. Kreinovich, and V. Sinyansky, Images with Uncertainty: Efficient Algorithms for Shift, Rotation, Scaling, and Registration, and Their Applications to Geosciences, In: M. Nachtegael, D. Van der Weken, and E.E. Kerre (eds.), Soft Computing in Image Processing: Recent Advances, Springer Verlag (to appear).

[31] A.K. Sinha, Geoinformatics: Data to Knowledge, Geological Society of America Publ., Boulder, Colorado, 2006.

[32] R. Torres, G.R. Keller, V. Kreinovich, L. Longpré, and S. A. Starks, Eliminating Duplicates Under Interval and Fuzzy Uncertainty: An Asymptotically Optimal Algorithm and Its Geospatial Applications, Reliable Computing, Vol. 10, No. 5, pp. 401–422, 2004.

[33] R. Trejo and V. Kreinovich, Error Estimations for Indirect Measurements: Randomized vs. Deterministic Algorithms For 'Black-Box' Programs, In: S. Rajasekaran, P. Pardalos, J. Reif, and J. Rolim (eds.), Handbook on Randomized Computing, Kluwer, pp. 673–729, 2001.

[34] S.A. Vavasis, Nonlinear optimization: complexity issues, Oxford University Press, New York, 1991.

[35] Q. Wen, A. Q. Gates, J. Beck, V. Kreinovich, and G. R. Keller, Towards automatic detection of erroneous measurement results in a gravity database, Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference, Tucson, Arizona, October 7–10, 2001, pp. 2170–2175.

[36] H. Xie, N. Hicks, G.R. Keller, H. Huang, and V. Kreinovich, An IDL/ENVI implementation of the FFT based algorithm for automatic image registration, Computers and Geosciences, Vol. 29, No. 8, pp. 1045–1055, 2003.