6-2014

Dealing with Uncertainties in Computing: from Probabilistic and Interval Uncertainty to Combination of Different Types of Uncertainty

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-14-15a
To appear in: Gerard Olivar Tost and Olga Vasilieva (eds.), *Analysis, Modelling, Optimization, and Numerical Techniques*, Springer Verlag, Berlin, Heidelberg, 2015.

Dealing with Uncertainties in Computing: from Probabilistic and Interval Uncertainty to Combination of Different Types of Uncertainty

Vladik Kreinovich

Abstract To predict values of future quantities, we apply algorithms to the current and past measurement results. Because of the measurement errors and model inaccuracy, the resulting estimates are, in general, different from the desired values of the corresponding quantities. There exist methods for estimating this difference, but these methods have been mainly developed for the two extreme cases: the case when we know the exact probability distributions of all the measurement errors and the interval case, when we only know the bounds on the measurement errors. In practice, we often have some partial information about the probability distributions which goes beyond these bounds. In this paper, we show how the existing methods of estimating uncertainty can be extended to this generic case.

Key words: error estimation, measurement errors, model inaccuracy, interval computations

1 Need to Deal with Uncertainty in Computing

Need for data processing. To make a proper decision, we need to be able to predict the results of making a certain decision (or of not making any decision at all). In many real-life situations, we know how the desired future value y of each corresponding quantities depends on the current values of relevant quantities $q_1, \ldots, q_n$; in other words, we have an algorithm that, given the values $q_1, \ldots, q_n$, produces the estimate $y = A(q_1, \ldots, q_n)$. This algorithm can be as simple as a straightforward computation by using an explicit formula, or it can be as complex as a solution of the corresponding system of partial differential equations (as in weather prediction).

Vladik Kreinovich
University of Texas at El Paso, El Paso, TX 79968, USA, e-mail: vladik@utep.edu

Sometimes, the quantities $q_1, \ldots, q_n$ can be measured directly; in such cases, to predict the future value y, we measure the current values of these quantities and use the algorithm f to predict the future value y.

In many practical situations, however, some of the quantities q_i are difficult (or even impossible) to measure directly. For example, to make predictions in geosciences, we must know the densities and stresses at different depths, including areas much deeper that current boreholes can reach. In such situations, instead of directly measuring the corresponding quantity q_i, we can measure it *indirectly*: namely, we measure the auxiliary quantities $a_1, \ldots, a_m$ which are related to q_i by a known dependence, and then use a known algorithm to estimate q_i based on the results of these measurements. For example, to estimate the density at different depths, we measure gravity at different Earth locations, we measure travel times of seismic waves, etc. As a result, we arrive at the following problem:

- First, we (directly) measure some quantities; we will denote these quantities by $x_1, \ldots, x_n$. Some of these quantities may be the easy-to-measures quantities q_i, some may be auxiliary quantities whose measurement is needed to estimate difficult-to-measure quantities q_i.
- Then, we use the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of measuring the quantities $x_1, \ldots, x_n$ to compute the estimate $\widetilde{y}$ for the desired future value y. We will denote the corresponding algorithm by f, so that $\widetilde{f} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$. This algorithm usually consists of two parts:
  - first, we use the values $\widetilde{x}_j$ to estimate the quantities q_i, and
  - then we use the estimated values of q_i to predict the value y.

Computation of $\widetilde{y}$ from $\widetilde{x}_i$ constitutes *data processing*.

Need to deal with uncertainty in data processing. Measurement are never absolutely accurate. As a result, the measurement results $\widetilde{x}_i$ are, in general, different from the actual (unknown) values x_i of the corresponding quantity. In other words, in general, we have a non-zero *measurement error* $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$. Because of this difference, even when the model is exact, i.e., when the actual values y and x_i satisfy the condition $y = f(x_1, \ldots, x_n)$, the estimated value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is, in general, different from the actual value y.

In some cases, the model itself is only approximate, in the sense that y is only approximately equal to $f(x_1, \ldots, x_n)$. In this case, there is an additional *model inaccuracy* $\Delta x_0 \stackrel{\text{def}}{=} f(x_1, \ldots, x_n) - y$, and hence, the estimate $\widetilde{y}$ is even more different from y.

To make a proper decision based on the estimate $\widetilde{y}$, it is important to know how accurate is this estimate. For example, if the estimate for the amount of water is an underground aquifer is 200 million tons, and it is 200 ± 10, then it is a good idea to start digging and exploiting this water; on the other hand, if it is 200 ± 300, then it may be that there is no water available at all – in which case, further measurements may be needed before we invest money in exploiting this possible source of water.

In general, it is important to get some information about the estimation error $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$.

2 Processing Uncertainty: General Formulation of the Problem

Towards the general formulation of the problem. We are interested in the difference $\Delta y = \widetilde{y} - y$.

- We know that $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.
- By definition of the model inaccuracy, we have $y = f(x_1, \ldots, x_n) - \Delta x_0$. By definition of the measurement error, we have $x_i = \widetilde{x}_i - \Delta x_i$, so

$$y = f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n) - \Delta x_0.$$

Substituting these expressions for $\widetilde{y}$ and y into the above formula for Δy, we conclude that

$$\Delta y = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n) + \Delta x_0. \tag{1}$$

Measurement errors are usually relatively small. The measurement errors are usually relatively small, we may have measurement accuracy 10%, 5%, 1%. In all these cases, the squares of the measurement errors can be safely ignored: e.g., for $\Delta x_i \approx 10\%$, we have $(\Delta x_i)^2 \approx 1\% \ll 10\%$. Because of this, we can expand the formula (1) in Taylor series in Δx_i and ignore terms which are quadratic (or of higher order) in Δx_i. We thus get $f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n) = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - \sum_{i=1}^{n} c_i \cdot \Delta x_i$, where $c_i \stackrel{\text{def}}{=} \dfrac{\partial f}{\partial x_i}$ and therefore, we get a linear dependence:

$$\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i + \Delta x_0. \tag{2}$$

Measurement errors corresponding to different measurements are usually independent. Measurement errors Δx_i corresponding to different measurements are usually independent from each other – and from the model inaccuracy Δx_0. Therefore, it makes sense to assume that all $n+1$ random variables $\Delta x_1, \ldots, \Delta x_n$, and Δx_0 are independent.

What we do in this paper. In this paper, we describe how to estimate Δy in a general situation, when we may have a combination of probabilistic and interval uncertainty. To provide this description, we need to first recall how uncertainty is usually estimated – so that it will be clear what are the assumptions underlying the usual techniques, and what needs to be modified when these assumptions are not satisfied.

3 Traditional Engineering Approach to Processing Uncertainty: Brief Reminder

Usual assumptions: that all distributions are normal with zero mean. In engineering practice, it is usually assumed that all the measurement errors are normally distributed with zero mean.

The normality distribution comes form the fact that for each measurement, the measurement error comes from many different sources. Usually, manufacturers of the measuring instrument try their best to eliminate all major sources of measurement errors. As a result, the remaining measurement error does not contain any large components, it is a joint effort of numerous small error components coming from different sources. According to the Central Limit Theorem (see, e.g., [17]), the distribution of the sum of a large amount of small independent random components is close to Gaussian – and the more components we have, the closer the resulting distribution to Gaussian. Thus, it make sense to assume that the measurement errors are normally distributed – and indeed, empirical analysis shows that more than half of measuring instruments have normal distribution [13, 14].

The zero mean assumption comes from the fact that the measuring instruments are usually calibrated before their use; see, e.g., [15]. One of the purposes of the calibration is to find the instrument's *bias* – i.e., the mean value of the measurement error – and to compensate for this bias. After the compensation, the mean is zero.

To describe a normal distribution, it is sufficient to describe the mean and the standard deviation. Since the mean of the variable Δx_i is zero, all we need to do to describe the measurement error is to provide the standard deviation σ_i. Similarly, we can eliminate the main sources of the model inaccuracy, and we can delete the model's bias as well. As a result, we can conclude that the model's inaccuracy Δx_0 is also normally distributed, with zero mean. We will denote its standard deviation by σ_0.

Estimating uncertainty under the usual assumptions: derivation of the resulting formulas. According to the formula (2), the estimation error Δy is a linear combination of measurement errors Δx_i and of the model inaccuracy Δx_0. These quantities are independent, and (under the above assumptions) normally distributed. It is known that a linear combination of independent Gaussian random variables is also normally distributed, so Δy is also normally distributed.

To describe a normal distribution, it is sufficient to describe the mean and the standard deviation. Since the means of all the variables Δx_i and Δx_0 are zeros, the mean value of Δy is also equal to 0. Thus, under the usual engineering assumptions, to describe the probability distribution for Δy, it is sufficient to describe its standard deviation σ. The variance of the sum of independent random variables is equal to the sum of the variances, so from (2), we conclude that

$$\sigma^2 = \sum_{i=1}^{n} c_i^2 \cdot \sigma_i^2 + \sigma_0^2. \tag{3}$$

How to actually estimate σ: towards the first algorithm. How can we actually estimate σ? To use this formula, we need to know the values c_i. These values are partial derivatives of the function $f(x_1,\ldots,x_n)$ describing the data processing algorithm. When this algorithm consists of a straightforward application of an explicit formula, we can simply differentiate this formula and get an explicit expression for the corresponding derivatives. However, in general, the function $f(x_1,\ldots,x_n)$ is given as a complex algorithm, so it is not possible to perform an explicit differentiation.

A reasonable alternative is to use *numerical differentiation*. Numerical differentiation is based on the definition of the derivative as a limit:

$$\frac{\partial f}{\partial x_i} = \lim_{h_i \to 0} \frac{f(x_1,\ldots,x_{i-1},x_i+h_i,x_{i+1},\ldots,x_n) - f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n)}{h_i}.$$

By the definition of the limit, this means that for small h, we have

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_1,\ldots,x_{i-1},x_i+h_i,x_{i+1},\ldots,x_n) - f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n)}{h_i}.$$

For small h_i, we expand the expression $f(x_1,\ldots,x_{i-1},x_i+h_i,x_{i+1},\ldots,x_n)$ in Taylor series and keep only terms which are linear in h, getting

$$f(x_1,\ldots,x_{i-1},x_i+h_i,x_{i+1},\ldots,x_n) = f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n) + h_i \cdot c_i.$$

From this formula, we can estimate c_i as the ratio

$$c_i = \frac{f(x_1,\ldots,x_{i-1},x_i+h_i,x_{i+1},\ldots,x_n) - f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n)}{h_i}. \qquad (4)$$

Substituting these expressions into the formula (3), we get

$$\sigma^2 = \sum_{i=1}^{n} \left(\frac{f(\ldots,x_{i-1},x_i+h_i,x_{i+1},\ldots) - f(\ldots,x_{i-1},x_i,x_{i+1},\ldots)}{h_i} \right)^2 \cdot \sigma_i^2 + \sigma_0^2.$$

Which values $h_1,\ldots,h_n$ should we use? Once we know the values of the function f, this formula uses subtraction, addition, multiplication, and division to estimate σ^2. In the computer, division is the most time-consuming operation, so ideally, we should select h_i so as to avoid divisions. Division can indeed be avoided if we take $h_i = \sigma_i$. In this case, the above formula takes the simplified form:

$$\sigma^2 = \sum_{i=1}^{n} (f(\ldots,x_{i-1},x_i+\sigma_i,x_{i+1},\ldots) - f(\ldots,x_{i-1},x_i,x_{i+1},\ldots))^2 + \sigma_0^2. \qquad (5)$$

Thus, we arrive at the following algorithm.

First algorithm: sensitivity analysis. We are given the values $\widetilde{x}_1,\ldots,\widetilde{x}_n$, the algorithm f, and the standard deviations $\sigma_1,\ldots,\sigma_n$, and σ_0.

- First, we perform the original data processing, i.e., compute the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.
- Then, for $i = 1, \ldots, n$, we compute $y_i \stackrel{\text{def}}{=} f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + \sigma_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n)$.
- Finally, we compute $\sigma^2 = \sum_{i=1}^{n} (y_i - \widetilde{y})^2 + \sigma_0^2$.

Comment. Our recommendation to use $h_i = \sigma_i$ differs from the usual numerical methods recommendation to use $h_i \approx \sqrt{\varepsilon}$, where ε is the machine epsilon. This usual recommendation makes perfect sense in the situations in which:

- the algorithm f provides very accurate computation of the corresponding function $f(x_1, \ldots, x_n)$, and
- the objective is to find the most accurate estimate of the derivatives.

In our application, none of these two conditions are satisfied.

First, since the input data come from measurement and are, thus, only approximately known, the data processing algorithms provide only approximate computation of the corresponding value – it makes no sense to compute, e.g., $\ln(x)$ with 8 digit accuracy if we only know x with accuracy 1% (which, by the way, means very accurate measurements). Such an approximate algorithm f may not even take into account the much smaller difference $h_i \approx \sqrt{\varepsilon}$ between the values x_i and $x_i + \sqrt{\varepsilon}$, but this algorithm will definitely react to the difference of order σ_i between the values x_i and $x_i + \sigma_i$, since a difference of this order of magnitude corresponds to practically distinguishable difference between data values.

Second, by applying linearization – i.e., by replacing the exact formula (1) with an approximate formula (2) – we have already ignored quadratic terms in the expression σ, terms which even for very accurate measurements, with accuracy 1%, leads to relative accuracy 10^{-4} of computing σ. Since the formula (3) is only valid with this accuracy, it does not make sense to spend additional computation time on estimating c_i too accurately.

In this case, as we have mentioned, the need to save computation time leads to $h_i = \sigma_i$.

Limitations of the first algorithm. As we have mentioned, the data processing algorithm f can be very time-consuming. Thus, the more times we call this algorithm, the longer our estimation of σ. The above algorithm requires $n + 1$ calls to the algorithm f (n more calls than a simple data processing). In many practical problems – e.g., in geosciences – we process thousands of data points, so n is in thousands. If it takes several hours on a high performance computer to estimate each value of f, then, to compute σ, the above algorithm requires thousands time more time – i.e., several months. This is not realistic, we need a faster method.

Towards a second algorithm. The possibility to process uncertainty faster comes from the fact that a similar expression for σ arises if we *simulate* normally distributed random errors. Namely, if we add, to the original values $\widetilde{x}_i$, simulated random errors δx_i which are normally distributed with 0 mean and standard deviation σ_i, and use a random variable δx_0 which is normally distributed with mean 0 and standard deviation σ_0, then the difference

$$f(\widetilde{x}_1+\delta x_1,\ldots,\widetilde{x}_n+\delta x_n)-f(\widetilde{x}_1,\ldots,\widetilde{x}_n)+\delta x_0=\sum_{i=1}^{n} c_i\cdot\delta x_i+\delta x_0$$

is also normally distributed with 0 mean and the desired standard deviation σ. We can thus use the standard formulas for estimating standard deviation from a sample to estimate σ. We therefore arrive at the following algorithm:

Second algorithm: Monte-Carlo simulations. We are given the values $\widetilde{x}_1,\ldots,\widetilde{x}_n$, the algorithm f, and the standard deviations $\sigma_1,\ldots,\sigma_n$, and σ_0.

- First, we perform the original data processing, i.e., compute the value $\widetilde{y}=f(\widetilde{x}_1,\ldots,\widetilde{x}_n)$.
- Then, we select the number of iterations N. For each k from 1 to N, we generate $n+1$ random numbers $r_{k1},\ldots,r_{kn},r_{k0}$ each of which is normally distributed with mean 0 and standard deviation 1.
- For each k, we compute $y_k=f(\widetilde{x}_1+\sigma_1\cdot r_{k1},\ldots,\widetilde{x}_n+\sigma_n\cdot r_{kn})+\sigma_0\cdot r_{k0}$.
- Finally, we estimate $\sigma^2=\dfrac{1}{N}\cdot\sum\limits_{k=1}^{N}(y_k-\widetilde{y})^2$.

Advantages and limitations of the second algorithm. The above method requires $N+1$ calls to the algorithm f. The number of iterations N depends on the accuracy with which we want to estimate σ. In general, the relative standard deviation of determining σ from a sample of size N is equal to $\sqrt{\dfrac{2}{N}}$; so, e.g., to find σ with accuracy 20% and reliability 95% (which corresponds to two standard deviations), we need to make sure that $2\cdot\sqrt{\dfrac{2}{N}}\le 0.2$, i.e., $N\ge 200$. For $n\gg 1{,}000$, this is much faster than the sensitivity analysis – this is the main advantage of this method.

The limitation is that, in contrast to the sensitivity analysis method, we do not get the exact value σ, only an approximate value.

Possibility of parallelization. In both methods for estimating σ, the most time-consuming step is calling the algorithm f. If we have at our disposal several processors which can work in parallel, then we can make all these calls in parallel and thus, drastically decrease the computation time.

Comment. In situations when we know the actual step-by-step code of the data processing algorithm f, there is another way to save computation time. Namely, we can apply, to the known code, the procedure of reverse differentiation (also known as *backpropagation* or *adjoint methods*) which allows us to compute the values x_i of all n partial derivatives c_i in time which, theoretically, is no more than 3 times longer than the time needed to compute the value f itself; see, e.g., [5, 18, 19]. Once we have computed all the values c_i of the gradient, we can use the formula (3) to compute the desired standard deviation σ.

This method is indeed effectively used, e.g., in neural networks [18, 19]. However, in many practical situations, the actual computational overhead of using reverse differentiation is much higher, to the extent than Monte-Carlo methods are faster.

Besides, in some practical situations, data processing uses proprietary programs, programs which for which the code is not provided to the user. The only way to use these programs is to treat the data processing algorithm as a "black box": the only thing we can compute are the output values $f(x)$ corresponding to different inputs x. For such programs, it is not possible to use reverse differentiation, and the only possibility to reduce the computation time in comparison with sensitivity analysis is to use Monte-Carlo techniques.

4 Case of Interval Uncertainty

Need for interval uncertainty. The traditional approach is based on the assumption that for each measuring instrument, we know the exact distribution of the corresponding measurement error Δx_i. In practice, this probability distribution can be established if we compare the results $\widetilde{x}_i^{(k)}$ produced by our measuring instrument with the results $\widetilde{x}_{i,\mathrm{st}}^{(k)}$ produced by a much more accurate ("standard") measuring instrument. Since the standard measuring instrument is much more accurate, we can ignore its measurement errors and assume that its measurement results are equal to the exact values of the corresponding quantity: $\widetilde{x}_{i,\mathrm{st}}^{(k)} \approx x_i^{(k)}$. In this approximation, the differences $\widetilde{x}_i^{(k)} - \widetilde{x}_{i,\mathrm{st}}^{(k)}$ are equal to the corresponding measurement errors $\Delta x_i^{(k)} = \widetilde{x}_i^{(k)} - x_i^{(k)}$. By accumulating a sample of such values, we get a probability distribution for Δx_i.

However, there are two situations when we cannot do it. First is the case of state-of-the-art measurements. For example, it would be nice if near the Hubble telescope, there would be another one, five times more accurate, which we could use to calibrate the Hubble telescope – but the Hubble telescope is the best we have. Similarly, it would be nice if we had geophysical methods which were five times more accurate than the current ones – but our methods are the best we have. In such situations, at best, we can have upper bounds Δ_i on the corresponding measurement errors. We know that $|\Delta x_i| \le \Delta_i$, i.e., that Δx_i is located on the interval $[-\Delta_i, \Delta_i]$, but we do not have any information about which values from this interval are more probable and which values are less probable. this situation is known as *interval uncertainty*; see, e.g., [6, 11, 15].

Interval uncertainty also occurs in *manufacturing*, where, in principle, we can calibrate every sensors, but since sensors are relatively cheap and their calibration is very expensive, they are not calibrated – instead, we rely on the upper bounds Δ_i provided by the manufacturer.

Similarly, we only know a bound Δ_0 on the model inaccuracy Δx_0: $|\Delta x_0| \le \Delta_0$.

Estimating uncertainty under interval uncertainty: derivation of the resulting formulas. The sum (2) is the largest when each term $c_i \cdot \Delta x_i$ attains its largest possible value on the corresponding interval $[-\Delta_i, \Delta_i]$.

- When $c_i \geq 0$, the function $c_i \cdot \Delta x_i$ is increasing, so its largest value is attained for the largest possible value $\Delta x_i = \Delta_i$. This largest value is equal to $c_i \cdot \Delta_i$.
- When $c_i \leq 0$, the function $c_i \cdot \Delta x_i$ is decreasing, so its largest value is attained for the smallest possible value $-\Delta x_i = \Delta_i$. This largest value is equal to $-c_i \cdot \Delta_i$.

In both cases, the largest possible value is $|c_i| \cdot \Delta_i$. Similarly, in both cases, the smallest possible value is $-|c_i| \cdot \Delta_i$. Thus, the range of possible values of Δy is equal to $[-\Delta, \Delta]$, where

$$\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i + \Delta_0. \tag{6}$$

Why not use Maximum Entropy approach? In statistics, situations when we do not know the exact probability distribution are frequent. In this case, if we have several possible distributions consistent with our knowledge, a reasonable idea is to select the distribution with the largest value of the *entropy* $S \stackrel{\text{def}}{=} -\int \rho(x) \cdot \ln(\rho(x))\, dx$, where $\rho(x)$ is the probability density; see, e.g., [7]. If we only know that the random variable is located on an interval, then this Maximum Entropy approach leads to a uniform distribution on this interval. (For several variables, if we know nothing about their correlation, the Maximum Entropy approach implies that they are independent.)

At first glance, this makes perfect sense – and this is how many practitioners deal with interval uncertainty. However, we can show that this approach can drastically underestimate the uncertainty Δy. We can illustrate it on the example of the simplest possible dependence, when $f(x_1, \ldots, x_n) = x_1 + \ldots + x_n$ and therefore, $\Delta y = \Delta x_1 + \ldots + \Delta x_n$. For simplicity, we can assume that all the upper bounds are the same: $\Delta_1 = \ldots = \Delta_n$. In this case, the formula (6) implies that $\Delta = n \cdot \Delta_1$. This is possible, e.g., if each measurement error is exactly equal to Δ_1.

On the other hand, according to the Maximum Entropy approach, each value Δx_i is uniformly distributed on the interval $[-\Delta_1, \Delta_1]$. This distribution has mean 0 and variance $\frac{1}{3} \cdot \Delta_1^2$. For large n, the sum Δy of these independent random variables is approximately normally distributed (the same Central Limit Theorem that we cited earlier). The mean of Δy is equal to the sum of 0s, i.e., to 0, and its variance is equal to the sum of the variances, i.e., $\sigma^2 = \frac{n}{3} \cdot \Delta_1^2$. For a normal distribution, the values are located in the six-sigma interval with practically absolute certainty; thus, we can take $6\sigma \sim \sqrt{n}$ as an upper bound for Δy. For large n, this is much smaller than the above upper bound $n \cdot \Delta_1$. Thus, the Maximum Entropy approach is not applicable, and we have to use the formula (6).

How to actually estimate Δ: towards the first algorithm. How can we actually estimate Δ? If we substitute the above numerical differentiation formula for c_i into the formula (6), we conclude that

$$\Delta = \sum_{i=1}^{n} \left| \frac{f(\ldots, x_{i-1}, x_i + h_i, x_{i+1}, \ldots) - f(\ldots, x_{i-1}, x_i, x_{i+1}, \ldots)}{h_i} \right| \cdot \Delta_i + \Delta_0.$$

Which values $h_1,\ldots,h_n$ should we use? Similarly to the traditional case, we select the values h_i for which we can avoid dvision and thus, speed up computations. Division can indeed be avoided if we take $h_i=\Delta_i$. In this case, the above formula takes the simplified form:

$$\sigma^2=\sum_{i=1}^{n}|f(\ldots,x_{i-1},x_i+\Delta_i,x_{i+1},\ldots)-f(\ldots,x_{i-1},x_i,x_{i+1},\ldots)|+\Delta_0. \qquad (7)$$

Thus, we arrive at the following algorithm.

First algorithm: sensitivity analysis. We are given the values $\widetilde{x}_1,\ldots,\widetilde{x}_n$, the algorithm f, and the bounds deviations $\Delta_1,\ldots,\Delta_n$, and δ.

- First, we perform the original data processing, i.e., compute the value $\widetilde{y}=f(\widetilde{x}_1,\ldots,\widetilde{x}_n)$.
- Then, for $i=1,\ldots,n$, we compute $y_i\stackrel{\text{def}}{=}f(\widetilde{x}_1,\ldots,\widetilde{x}_{i-1},\widetilde{x}_i+\Delta_i,\widetilde{x}_{i+1},\ldots,\widetilde{x}_n)$.
- Finally, we compute $\Delta=\sum\limits_{i=1}^{n}|y_i-\widetilde{y}|+\Delta_0$.

Limitations of the first algorithm. Similarly to the traditional case, this algorithm requires $n+1$ calls to the algorithm f and is, thus, often too slow.

Towards a second algorithm. The possibility to process uncertainty faster comes from the fact that for random variables distributed according to the Cauchy distribution, with probability density $\rho(x)=\dfrac{1}{\pi\cdot\Delta}\cdot\dfrac{1}{(x/\Delta)^2+1}$, a linear combination (2) of variables Δx_i which are Cauchy distributed with parameters Δ_i is Cauchy-distributed with parameter Δ determined by the formula (7). We therefore arrive at the following algorithm [8, 9]:

Second algorithm: Monte-Carlo simulations. We are given the values $\widetilde{x}_1,\ldots,\widetilde{x}_n$, the algorithm f, and the bounds $\Delta_1,\ldots,\Delta_n$, and s_m.

- First, we perform the original data processing, i.e., compute the value $\widetilde{y}=f(\widetilde{x}_1,\ldots,\widetilde{x}_n)$.
- Then, we select the number of iterations N. For each k from 1 to N, we generate $n+1$ random numbers $r_{k1},\ldots,r_{kn}$ each of which is uniformly distributed on the interval $[0,1]$.
- Then, we compute Cauchy distributed values $c_{ki}=\tan(\pi\cdot(r_{ki}-0.5))$.
- We compute the maximum $K_k=\max\limits_i|c_{ki}|$ so that we will be able to normalize the simulated approximation errors and apply f to the values that are within the box of possible values.
- For each k, we compute

$$\Delta y_k=K_k\cdot\left(f\left(\widetilde{x}_1+\Delta_1\cdot\frac{c_{k1}}{K_k},\ldots,\widetilde{x}_n+\Delta_n\cdot\frac{c_{kn}}{K_k}\right)-\widetilde{y}\right).$$

- We compute Δ' by applying the bisection method to solve the equation

$$\frac{1}{1+\left(\frac{\Delta y^{(1)}}{\Delta'}\right)^2}+\ldots+\frac{1}{1+\left(\frac{\Delta y^{(N)}}{\Delta'}\right)^2}=\frac{N}{2}.$$

- Finally, we return $\Delta=\Delta'+\Delta_0$.

Advantages and limitations of the second algorithm. The above method requires $N+1$ calls to the algorithm f. Similarly to the usual Monte-Carlo method, the number of iterations N depends on the accuracy with which we want to estimate σ. For $n \gg 200$, this is much faster than the sensitivity analysis – this is the main advantage of this method. The limitation is that, in contrast to the sensitivity analysis method, we do not get the exact value Δ, only an approximate value.

Possibility of parallelization. Similarly to the statistical case, in both methods for estimating σ, the most time-consuming step is calling the algorithm f. So, if we have at our disposal several processors which can work in parallel, then we can make all these calls in parallel and thus, drastically decrease the computation time.

Comment. A numerical example of using this Cauchy-based Monte-Carlo method is given in [9].

5 Need to go Beyond Traditional and Interval Cases

What we have considered so far. Up to now, we considered two extreme cases:

- the traditional case, when all measurement errors are normally distributed with zero mean, and
- the interval case, when we only know the upper bounds on the measurement errors.

Need to go beyond these cases. In practice, we often have cases in between.

- In some cases, we know the distributions, and these distributions are non-Gaussian. This is actually the case for almost half (40%) of the measuring instruments; see, e.g., [13, 14].
- In some other cases, we do not know the exact probability distributions – but we have some partial information about these distributions which goes beyond the upper bounds.

What we do in this paper. In this paper, we describe how to estimate uncertainty in the general case.

6 Case of Known Non-Gaussian Distributions

Formulation of the problem. Let us first consider the case when we know the probability distributions of all the measurement errors Δx_i, and the probability distribution of the model error Δx_0. For example, these probability distributions are represented in terms of the probability density functions $\rho_i(\Delta x_i)$ and $\rho_0(\Delta x_0)$.

We know that the corresponding variables are independent. Our goal is to find the probability distribution of the quantity Δy – as described by the formula (2).

Two types of algorithms. Similarly to the above two cases, we will consider two types of algorithms for solving this problem: algorithms which produce the exact answer, and faster Monte-Carlo-type algorithms which produce approximate answers.

Algorithm for exact computation: general idea.

- First, we use numerical differentiation (4) to estimate the coefficients c_i.
- For each i, we can then compute the probability density functions corresponding to $t_i \stackrel{\text{def}}{=} c_i \cdot \Delta x_i$ as $d_i(t_i) = \dfrac{1}{c_i} \cdot \rho_i\left(\dfrac{t_i}{c_i}\right)$.
- Then, we can apply several times the known convolution formula $\rho_c(x) = \int \rho_a(t) \cdot \rho_b(x-t)\,dt$ for the probability density of the sum $c = a + b$ of independent random variables to find the probability density corresponding to the sum $\Delta y = \sum\limits_{i=1}^{n} t_i + \Delta x_0$:
  - first, we combine the probability distributions of t_1 and t_2 to compute the probability density of the sum $t_1 + t_2$;
  - then, we combine the probability distributions of $t_1 + t_2$ and t_3 to compute the probability density of the sum $t_1 + t_2 + t_3$;
  - ...
  - finally, we combine the probability distributions of $\sum\limits_{i=1}^{n} t_i$ and Δx_0 to compute the probability density of $\Delta y = \sum\limits_{i=1}^{n} t_i + \Delta x_0$.

How to compute convolutions faster. One possibility to compute the probability density function of the sum is to perform a straightforward computation of each convolution integral $\rho_c(x) = \int \rho_a(t) \cdot \rho_b(x-t)\,dt$. If we represent each of the probability density functions by its values at M different points $\rho_a(v_k)$ and $\rho_b(v_k)$ for $v_k = k \cdot \Delta v$, then each computation takes the form $\rho_c(v_k) = \sum\limits_{\ell} \rho_a(v_\ell) \cdot \rho_b(v_{k\ell}) \cdot \Delta v$. This computation requires M^2 computational steps: M steps for each value k.

It is known, however, that we can speed up the computation of convolution if we use *Fourier transforms*, i.e., if instead of the original probability density functions $\rho_a(x)$ and $\rho_b(x)$, we use the corresponding *characteristic functions*

$$\chi_a(\omega) \stackrel{\text{def}}{=} E[\exp(\mathrm{i} \cdot \omega \cdot a)] = \int \exp(\mathrm{i} \cdot x \cdot \omega) \cdot \rho_a(x)\,dx$$

and

$$\chi_b(\omega) \stackrel{\text{def}}{=} E[\exp(\mathrm{i}\cdot\omega\cdot b)] = \int \exp(\mathrm{i}\cdot x\cdot\omega)\cdot\rho_b(x)\,dx.$$

Namely, it is known that the characteristic function of the sum is equal to the product of the characteristic functions. Thus, we can compute the convolution as follows; see, e.g., [1]:

- First, we the use the Fast Fourier Transform algorithm to compute the Fourier transforms $\chi_a(\omega)$ and $\chi_b(\omega)$ of the corresponding probability density functions. This computation takes $O(M\cdot\log(M))$ computational steps.
- Then, we multiply the corresponding values of the Fourier transform element-by-element to compute $\chi_c(\omega) = \chi_a(\omega)\cdot\chi_b(\omega)$. To compute M values of this new characteristic function, we need M computational steps.
- Finally, we apply the Inverse Fast Fourier Transform algorithm to the function $\chi_c(\omega)$ and thus, find the desired probability density function $\rho_c(x)$. This computation also takes $O(M\cdot\log(M))$ computational steps.

Thus, overall, we need $O(M\cdot\log(M)) + O(M) + O(M\cdot\log(M)) = O(M\cdot\log(M))$ computational steps to compute the convolution, which, for large M, is much smaller than M^2 steps needed for the straightforward computation of the convolution.

Faster computation of the convolution can speed up the computation of the probability density function $\rho(\Delta y)$. For the sum Δy of $n+1$ random variables $t_1,\ldots,t_n$, and Δx_0, the characteristic function $\chi(\omega)$ is equal to the product of the characteristic functions $\chi_i(\omega)$ and $\chi_0(\omega)$ of these random variables. Thus:

- First, we the use the Fast Fourier Transform algorithm to compute the Fourier transforms $\chi_i(\omega)$ and $x_m(\omega)$ of the corresponding probability density functions $d_i(t_i)$ and $\rho_0(\Delta x_0)$. This computation takes $(n+1)\cdot O(M\cdot\log(M))$ computational steps.
- Then, we multiply the corresponding values of the Fourier transform element-by-element to compute $\chi(\omega) = \chi_1(\omega)\cdot\ldots\chi_n(\omega)\cdot\chi_0(\omega)$. To compute M values of this new characteristic function, we need $n\cdot M$ computational steps.
- Finally, we apply the Inverse Fast Fourier Transform algorithm to the function $\chi(\omega)$ and thus, find the desired probability density function corresponding to Δy. This computation takes $O(M\cdot\log(M))$ computational steps.

Overall, we thus need $O(n\cdot M\cdot\log(M))$ computational steps.

Possible use of Central Limit Theorem: discussion. The larger the number n of inputs $x_1,\ldots,x_n$, the more computation time we need. On the other hand, when n is large, this means that most of the contributions $t_i = c_i\cdot\Delta x_i$ to the overall error Δy are relatively small. In this case, as we have mentioned earlier, we can invoke the Central Limit Theorem and conclude that the probability distribution for the sum of these small contributions is close to Gaussian.

A Gaussian distribution is uniquely determined by its mean and standard deviation (or, equivalently, variance), and the mean and variance of the sum of several independent random variables is equal to the sum of the corresponding means and

variances. Thus, for the small components, there is no need to use their full probability density functions: it is sufficient to estimate their means and variances, then add them, and then add the resulting Gaussian sum to the few non-small components.

Thus, we arrive at the following algorithm.

Use of the Central Limit Theorem: resulting algorithm. This algorithm requires that we know the list of non-small components. Without losing generality, let us assume that the components $t_1, \ldots, t_k$, $k \ll n$ (and Δx_0) are non-small, and that all the other components $t_{k+1}, \ldots, t_n$ are small.

For each small component t_i, we use the probability distribution $d_i(t_i)$ to compute the mean $\mu_i = \int x \cdot d_i(x)\,dx$ and the variance $\sigma_i^2 = \int (x-\mu_i)^2 \cdot d_i(x)\,dx$. Then, we compute the overall mean and variance of the sum of all the small components as $\mu = \sum_{i=k+1}^{n} \mu_i$ and $\sigma^2 = \sum_{i=k+1}^{n} \sigma_i^2$, and we form a probability distribution function

$$\rho_{\text{sm}} = \frac{1}{\sqrt{\pi}\cdot\sigma} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

This is a probability distribution for the sum $\sum_{i=k+1}^{n} t_i$ of all small components.

Then, we combine the probability distributions for $t_1, \ldots, t_k$, $\sum_{i=k+1}^{n} t_i$, and Δx_0.

Monte-Carlo-type algorithm. To decrease the number of calls to the algorithm f and thus, to speed up the computations, we can simulate the measurement errors. To simulate a measurement error t_i distributed according to the probability density $d_i(t_i)$, we can perform the following preliminary computations:

- form the cumulative distribution function $F_i(x) = \int^x d_i(t)\,dt$,
- form its inverse function $F_i^{-1}(p)$ – by computing, for every value $p \in [0,1]$, the value $x = F^{-1}(p)$ for which $F_i(x) = p$.

After that, on each iteration k, we generate a random number r_{ki} which is uniformly distributed on the interval $[0,1]$, and compute $c_{ki} = F^{-1}(r_{ki})$. Similarly, we simulate a number c_{k0} which is distributed according to the probability density function $\rho_0(\Delta x_0)$.

We then compute simulated values

$$\Delta y^{(k)} = (f(\widetilde{x}_1 - c_{k1}, \ldots, \widetilde{x}_n - c_{kn}) + c_{k0}) - \widetilde{y}.$$

Based on the sample of these values, we can now determine the probability distribution for Δy.

Use of the Central Limit Theorem. Due to the Central Limit Theorem, for small components, instead of simulating their exact distributions, we can simulate normally distributed random variables with the same values of mean and standard deviation.

Parallelization can lead to a further speed-up. In all these methods, the most time-consuming step is calling the algorithm f. If we have at our disposal several

processors which can work in parallel, then we can make all these calls in parallel and thus, drastically decrease the computation time.

It is also possible to parallelize further processing of these values. For example, in the algorithm using Fourier transforms, we can compute each of $n+1$ Fourier transforms in parallel – and if we have more than $n+1$ processors, then we can also perform each Fast Fourier Transform in parallel. In the case of unlimited number of processors, this can be done in time $O(\log(M))$.

Similarly, each of the products $\chi(\omega)$ can be computed in parallel, and, if needed, each computation of a product can also be parallelized:

- first, we multiply all the neighboring pairs $\chi_{2i-1}(\omega) \cdot \chi_{2i}(\omega)$;
- then, we multiply product of neighboring pairs into products of neighboring 4-tuples,
- etc.

In this manner, in the ideal case of unlimited number of processors, we compute all the products in time $O(\log(M))$ – and thus, finish all the computations in time $O(\log(M))$.

7 Case of Partial Information about Probabilities: How to Represent this Partial Information?

Need to select a representation. In many real-life situations, we have only partial information about the probability distribution of measurement errors. How can we represent this partial information?

In principle, we can represent a probability distribution in many different forms:

- by its probability density function,
- by its cumulative distribution function,
- by its moments, etc.

Which representation should we use?

To select a representation, we need to take into account the ultimate goal: of decision making. As we have mentioned, one of the main reasons why we need to take into account uncertainty in data processing is that this uncertainty affects our decisions. From the viewpoint of decision making, what is the best way to represent partial information about the probabilities?

It is known that a consistent decision making can be described as optimizing an expected value of a special function $u(x)$ known as *utility*; see, e.g., [4, 10, 12, 16]. The utility function $u(x)$ describes the user preferences. Thus, it makes sense to select characteristics of the probability distribution which can help us compute this expected utility $\int \rho(x) \cdot u(x)\,dx$.

In particular, for measurement errors $\Delta x_i = \widetilde{x}_i - x_i$, the loss of utility is caused by the fact that while the only information that we can use about x_i is the measurement result $\widetilde{x}_i$, the actual value x_i is, in general, different from $\widetilde{x}_i$. For exam-

ple, if we want to dress appropriately for the weather, we must know the exact temperature; if we know it approximately, then there is a strong chance that we will dress either too warm or too cold. In general, the expected utility has the form $\int \rho_i(\Delta x_i) \cdot u(\Delta x_i)\, d\Delta x_i$.

Ideally, the perfect situation is when $\Delta x_i = 0$ and the actual value x_i is exactly equal to the measurement result $\widetilde{x}_i$. In this case, we prepare for exactly the proper conditions, so the utility attains its maximum value.

It is, however, possible that we know that the measuring instrument has a bias, and we know the approximate value of this bias b. In this case, when the measurement result is $\widetilde{x}_i$, we prepare for the de-biased value $x_i = \widetilde{x}_i - b$. So, even if $\Delta x_i = 0$, the actual condition $x_i = \widetilde{x}_i$ is somewhat different from the value $x_i = \widetilde{x}_i - b$ for which we are prepared.

Case of smooth utility functions: analysis of the problem. Let us first consider the case when the utility function smoothly changes with Δx_i. We consider the case when measurement errors are relatively small. This means that the values Δx_i are close to 0, so we can expand the utility function $u(\Delta x_i)$ in Taylor series and keep only the first few terms in this expansion.

In Section 2, we made a similar statement about the function f, and for this function, we decided to keep only linear terms, terms determined by its first derivatives c_i taken at the point $\widetilde{x}_i$ (i.e., at the point $x_i = \widetilde{x}_i - \Delta x_i$ corresponding to $\Delta x_i = 0$). For the utility function, this is not always possible: as we have mentioned, for the unbiased measuring instrument, the utility function attains its maximum when $\Delta x_i = 0$ and thus, its first derivative is equal to 0. So, for the utility function, we also need to take into account second-order terms: $u(\Delta x_i) = u_0 + u_1 \cdot \Delta x_i + u_2 \cdot (\Delta x_i)^2 + \ldots$, for some values u_0 and u_2.

Since the values Δx_i are assumed to be small, we can thus ignore cubic and higher order terms in this expansion, and conclude that $u(\Delta x_i) = u_0 + u_1 \cdot \Delta x_i + u_2 \cdot (\Delta x_i)^2$. For this utility function, the expected utility has the form

$$\int \rho_i(\Delta x_i) \cdot u(\Delta x_i)\, d\Delta x_i = u_0 + u_1 \cdot \int \Delta x_i \cdot \rho_i(\Delta x_i)\, d\Delta x_i + u_2 \cdot \int (\Delta x_i)^2 \cdot \rho_i(\Delta x_i)\, d\Delta x_i,$$

i.e., the form $u_0 + u_1 \cdot \mu_i + u_2 \cdot M_i$, where μ_i is the expected value of the measurement error (*bias*) and M_i is the second moment of the measurement error. So, in the case of a smooth utility function, to describe the probability distribution, it is reasonable to use its first two moments.

Our goal is not just to represent these measurement errors Δx_i, we also want to use this information to characterize the linear combination (2) of these measurement errors. From this viewpoint, it is more convenient, instead of the second moments M_i, to use variances $\sigma_i^2 = M_i - \mu_i^2$, since the variance is the easiest to process: the variance of the sum of two independent random variables is equal to the sum of the corresponding variances. Therefore, a reasonable representation of a probability distribution should consist of the mean μ_i and the standard deviation σ_i. Similarly, a reasonable way to describe the probability distribution of the model error Δx_0 is to describe its mean μ_0 and standard deviation σ_0.

In terms of metrology (measurement theory and practice), μ_i is a *systematic error component*, and σ_i is known as a strandard deviation of the *random error components*; see, e.g., [15].

Partial information means that we do not know the exact values of μ_i and σ_i. Instead, we only know the *bounds* on these values, i.e., we know the intervals $[\underline{\mu}_i, \overline{\mu}_i]$ and $[\underline{\sigma}_i, \overline{\sigma}_i]$ that contain the actual (unknown) values of mean and standard deviation.

Which characteristics of Δy should we compute based on these values? A similar analysis shows that we want to know the values of the corresponding mean μ and standard deviation σ.

Different possible values μ_i and σ_i from the corresponding intervals lead, in general, to different values of μ and σ; so, what we really want to compute are the ranges of possible values of μ and σ. Thus, we arrive at the following problem.

Case of a smooth utility function: precise formulation of the resulting computational problem. We know:

- the intervals $[\underline{\mu}_i, \overline{\mu}_i]$ and $[\underline{\sigma}_i, \overline{\sigma}_i]$ containing the means and standard deviations of $n+1$ independent random variables Δx_i, and
- the algorithm f.

We want to find the ranges $[\underline{\mu}, \overline{\mu}]$ and $[\underline{\sigma}, \overline{\sigma}]$ of possible values of the mean μ and standard deviation σ of the quantity Δy described by the formulas (1) and (2).

How to compute the range of possible values of μ: analysis of the problem. The mean of a linear combination is equal to the linear combination of the means, so we have

$$\mu = \sum_{i=1}^{n} c_i \cdot \mu_i + \mu_0.$$

We want to use the above interval-computation formulas from Section 4 to find the range of values of this linear function. For that purpose, we need to represent the corresponding intervals in the form $[\widetilde{\mu}_i - \Delta_i, \widetilde{\mu}_i + \Delta_i]$. By equating $\widetilde{\mu}_i - \Delta_i$ with $\underline{\mu}_i$ and $\widetilde{\mu}_i + \Delta_i$ with $\overline{\mu}_i$, we get a system of two equations with two unknowns $\widetilde{\mu}_i$ and Δ_i, from which we can conclude that:

- the value $\widetilde{\mu}_i$ is equal to the midpoint and
- the value Δ_i is equal to the half-width of the corresponding interval:

$$\widetilde{\mu}_i = \frac{\underline{\mu}_i + \overline{\mu}_i}{2} \text{ and } \Delta_i = \frac{\overline{\mu}_i - \underline{\mu}_i}{2}.$$

For the differences $\Delta\mu_i \stackrel{\text{def}}{=} \widetilde{\mu}_i - \mu_i$, we have a limitation $|\Delta\mu_i| \le \Delta_i$. Thus, the general formulas for the range of a function f (from Section 4) lead to a conclusion that the range of possible values of μ is equal to $[\widetilde{\mu} - \Delta, \widetilde{\mu} + \Delta]$, where $\widetilde{\mu} = \sum_{i=1}^{n} c_i \cdot \widetilde{\mu}_i + \widetilde{\mu}_0$ and $\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i + \Delta_0$.

Because of the formulas (1) and (2), the value $\widetilde{\mu}$ can be computed simply as $\widetilde{y} - f(\widetilde{x}_1 - \widetilde{\mu}_1, \ldots, \widetilde{x}_n - \widetilde{\mu}_n) + \widetilde{\mu}_0$. The value Δ can be computed by using one of the two interval computations algorithms. Thus, we arrive at the following algorithms.

How to compute the range of possible values of μ: algorithm.

- First, we perform the original data processing, and compute the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.
- Then, for all i, we compute $\widetilde{\mu}_i = \dfrac{\underline{\mu}_i + \overline{\mu}_i}{2}$ and $\Delta_i = \dfrac{\overline{\mu}_i - \underline{\mu}_i}{2}$.
- After that, we compute the value $\widetilde{\mu} = \widetilde{y} - f(\widetilde{x}_1 - \widetilde{\mu}_1, \ldots, \widetilde{x}_n - \widetilde{\mu}_n) + \widetilde{\mu}_0$, and we use one of the two interval computation algorithms from Section 4 to compute $\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i + \Delta_0$.
- Finally, we compute the desired range $[\widetilde{\mu} - \Delta, \widetilde{\mu} + \Delta]$.

How to compute the range of possible values of σ: analysis of the problem. The variance of the sum is equal to the sum of the variances, so we have $\sigma^2 = \sum_{i=1}^{n} c_i^2 \cdot \sigma_i^2 + \sigma_0^2$. This expression is increasing in σ_i, so:

- its largest possible value $\overline{\sigma}$ is attained when each of the values σ_i attains its largest possible value $\overline{\sigma}_i$, so we have $(\overline{\sigma})^2 = \sum_{i=1}^{n} c_i^2 \cdot (\overline{\sigma}_i)^2 + (\overline{\sigma}_0)^2$;
- its smallest possible value $\underline{\sigma}$ is attained when each of the values σ_i attains its smallest possible value $\underline{\sigma}_i$, so we have $(\underline{\sigma})^2 = \sum_{i=1}^{n} c_i^2 \cdot (\underline{\sigma}_i)^2 + (\underline{\sigma}_0)^2$.

Each of these formulas is of type (3), so we can use the two algorithms from Section 3 to perform these computations. In other words, we arrive at the following algorithm.

How to compute the range of possible values of σ: algorithm.

- First, we use one of the algorithms from Section 3 to compute the value $\overline{\sigma}$ from the formula $(\overline{\sigma})^2 = \sum_{i=1}^{n} c_i^2 \cdot (\overline{\sigma}_i)^2 + (\overline{\sigma}_0)^2$.
- Then, we use the same algorithm to compute the value $\underline{\sigma}$ from the formula $(\underline{\sigma})^2 = \sum_{i=1}^{n} c_i^2 \cdot (\underline{\sigma}_i)^2 + (\underline{\sigma}_0)^2$.

Case of discontinuous utility function. In some cases, the utility function is not smooth, but discontinuous. For example, at a manufacturing plant, we want to make sure that the possible pollution does not exceed a certain legal level. In such situations, there are stiff penalties for exceeding the level.

The expected value of this utility function is thus proportional to the probability of exceeding (or not exceeding) a certain level. For a random variable η, the corresponding probabilities $F(x) \stackrel{\text{def}}{=} \text{Prob}(\eta \le x)$ form a *cumulative distribution function* (cdf). For such utility functions, an appropriate representation of the probability distribution is thus the cdf $F(x)$.

Partial information means that we may not know all the values $F(x)$ of the cdf; instead, we only know *bounds* $[\underline{F}(x), \overline{F}(x)]$. Such bounds are known as a *probability box*, or a *p-box*, for short; see, e.g., [2, 3]. So, we arrive at the following problem.

Case of a discontinuous utility function: precise formulation of the resulting computational problem. We know:

- the p-boxes $[\underline{F}_i(x), \overline{F}_i(x)]$ describing the probability distribution of $n+1$ independent random variables Δx_i, and
- the algorithm f.

We want to find the ranges $[\underline{F}(x), \overline{F}(x)]$ of possible values of the cdf $F(x)$ for the quantity Δy described by the formulas (1) and (2).

How to compute the corresponding p-box: analysis of the problem. The desired quantity Δ is the sum of several terms $t_i = c_i \cdot \Delta x_i$ and $t_0 = \Delta x_0$. Thus, it makes sense to first find the p-boxes $[\underline{G}_i(t), \overline{G}_i(t)]$ which describe the range of possible values of the cdf $G_i(x)$ characterizing each term t_i.

For $c_i > 0$, the inequality $c_i \cdot \Delta x_i \le t$ is equivalent to $\Delta x_i \le \dfrac{t}{c_i}$, so

$$G_i(t) = \text{Prob}(t_i \le t) = \text{Prob}\left(\Delta x_i \le \frac{t}{c_i}\right) = F_i\left(\frac{t}{c_i}\right).$$

In this case:

- the smallest possible value of $G_i(t)$ corresponding to the smallest possible values $\underline{F}_i$ of F_i, and
- the largest possible value of $G_i(t)$ corresponding to the largest possible values $\overline{F}_i$ of F_i.

So, we have $\underline{G}_i(t) = \underline{F}_i\left(\dfrac{t}{c_i}\right)$ and $\overline{G}_i(t) = \overline{F}_i\left(\dfrac{t}{c_i}\right)$.

For $c_i < 0$, the inequality $c_i \cdot \Delta x_i \le t$ is equivalent to $\Delta x_i \ge \dfrac{t}{c_i}$, so

$$G_i(t) = \text{Prob}(t_i \le t) = 1 - \text{Prob}\left(\Delta x_i \ge \frac{t}{c_i}\right) = 1 - F_i\left(\frac{t}{c_i}\right).$$

In this case:

- the smallest possible value of $G_i(t)$ corresponding to the largest possible values $\overline{F}_i$ of F_i, and
- the largest possible value of $G_i(t)$ corresponding to the smallest possible values $\underline{F}_i$ of F_i.

So, we have $\underline{G}_i(t) = 1 - \overline{F}_i\left(\dfrac{t}{c_i}\right)$ and $\overline{G}_i(t) = 1 - \underline{F}_i\left(\dfrac{t}{c_i}\right)$.

In general, the lower bound $\underline{F}(x)$ corresponds to the smallest possible probability of smaller values – and thus, to the largest possible probability of larger values.

Similarly, the upper bound $\overline{F}(x)$ corresponds to the largest possible probability of smaller values. Thus:

- To find the lower bound $\underline{F}(x)$ corresponding to Δy, we must use probability distributions $G_i(\Delta x_i)$ for which the values t_i are the largest, i.e., the values $\underline{G}_i(t)$.
- Similarly, to find the upper bound $\overline{F}(x)$, we must use probability distributions $G_i(\Delta x_i)$ for which the values t_i are the smallest, i.e., the values $\overline{G}_i(t)$.

So, we arrive at the following algorithm.

Algorithm for exact computation of p-box $[\underline{F}(x),\overline{F}(x)]$: general idea.

- First, we use numerical differentiation (4) to estimate the coefficients c_i.
- For each i, we can then compute the p-boxes $[\underline{G}_i(t),\overline{G}_i(t)]$ corresponding to $t_i \stackrel{\text{def}}{=} c_i \cdot \Delta x_i$ as follows:
  - when $c_i > 0$, we take $\underline{G}_i(t) = \underline{F}_i\left(\dfrac{t}{c_i}\right)$ and $\overline{G}_i(t) = \overline{F}_i\left(\dfrac{t}{c_i}\right)$;
  - when $c_i < 0$, we take $\underline{G}_i(t) = 1-\overline{F}_i\left(\dfrac{t}{c_i}\right)$ and $\overline{G}_i(t) = 1-\underline{F}_i\left(\dfrac{t}{c_i}\right)$.
- Then, to find the p-box $[\underline{F}(x),\overline{F}(x)]$ corresponding to the sum $\Delta y = \sum\limits_{i=1}^{n} t_i + \Delta x_0$, we do the following:
  - to compute $\underline{F}(x)$, we apply the convolution formula

$$\rho_c(x) = \int \rho_a(t)\cdot\rho_b(x-t)\,dt$$

    for the probability density of the sum $c = a+b$ to independent random variables with cdf's $\underline{G}_i(t)$; and
  - to compute $\overline{F}(x)$, we apply the same convolution formula to independent random variables with cdf's $\overline{G}_i(t)$.

To compute convolutions, we use the above algorithm based on Fast Fourier Transform.

Possible use of the Central Limit Theorem. Similarly to the case when we know the exact non-Gaussian distributions, we can speed up computations if we know the list of non-small components. In this case, we know the sum $t_{k+1}+\ldots+t_n$ of small components is normally distributed. Normal distribution can be described by the mean μ and standard deviation σ; ranges $[\underline{\mu},\overline{\mu}]$ and $[\underline{\sigma},\overline{\sigma}]$ for μ and σ can be found by using the same methods as in the case of smooth utility function.

In general, cdf for a normal distribution has the form $F(t) = F_0\left(\dfrac{t-\mu}{\sigma}\right)$, where $F_0(t)$ is the cdf of the "standard" normal distribution, with mean 0 and standard deviation 1. The function $F_0(t)$ is increasing. Thus, if we know the bounds on μ and on σ:

- the smallest value of $F(t)$ is attained when μ and σ are the largest, and

- the largest value of $F(t)$ is attained when μ and σ are the smallest.

In other words, $\underline{F}_{\rm sm}(x) = F_0\left(\dfrac{t-\overline{\mu}}{\overline{\sigma}}\right)$ and $\overline{F}_{\rm sm}(x) = F_0\left(\dfrac{t-\underline{\mu}}{\underline{\sigma}}\right)$.

The p-box for Δy can then be obtained by combining p-boxes corresponding to $t_1, \ldots, t_k$, t_0, and the above Gaussian p-box $[\underline{F}_{\rm sm}(x), \overline{F}_{\rm sm}(x)]$ for $\sum\limits_{i=k+1}^{n} t_i$.

Acknowledgments. This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721.

The author is greatly thankful to all the participants of the International Conference on Applied Mathematics and Informatics ICAMI'2013 (San Andres Island, Colombia, November 24–28, 2013), especially to Anibal Sosa, for valuable suggestions, and to the anonymous referees for their useful comments.

References

1. Cormen, T.H., et al., Introduction to Algorithms, MIT Press, Cambridge, MA (2009)
2. Ferson, S.: Risk Assessment with Uncertainty Numbers: RiskCalc, CRC Press, Boca Raton, Florida (2002)
3. Ferson, S., et al.: Experimental Uncertainty Estimation and Statistics for Data Having Interval Uncertainty. Sandia National Laboratories, Report SAND2007-0939, May (2007); available as http://www.ramas.com/intstats.pdf
4. Fishburn, P.C.: Nonlinear Preference and Utility Theory, John Hopkins Press, Baltimore, Maryland (1988)
5. Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, SIAM, Philadelphia, Pennsylvania (2000)
6. Jaulin, L., et al.: Applied Interval Analysis. Springer Verlag, London (2001)
7. Jaynes, E.T., Bretthorst, G.L.: Probability Theory: The logic of science, Cambridge University Press, Cambridge, UK (2003)
8. Kreinovich, V.: Interval computations and interval-related statistical techniques: tools for estimating uncertainty of the results of data processing and indirect measurements, In: Pavese, F., Forbes, A.B. (eds.), Data Modeling for Metrology and Testing in Measurement Science, pp. 117–145, Birkhauser-Springer, Boston (2009)
9. Kreinovich, V., Ferson, S.: A new Cauchy-based black-box technique for uncertainty in risk analysis. Reliability Engineering and Systems Safety **85**(1–3), 267–279 (2004)
10. Luce, R.D., Raiffa, R.: Games and Decisions: Introduction and Critical Survey, Dover, New York (1989)
11. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis, SIAM Press, Philadelphia, Pennsylviania (2009)
12. Nguyen, H.T., et al.: Computing Statistics under Interval and Fuzzy Uncertainty, Springer Verlag, Berlin, Heidelberg (2012)
13. Novitskii, P.V., Zograph, I.A.: Estimating the Measurement Errors, Energoatomizdat, Leningrad (1991), in Russian
14. Orlov, A.I.: How often are the observations normal?, Industrial Laboratory **57**(7), 770–772 (1991)
15. Rabinovich, S.G.: Measurement Errors and Uncertainty. Theory and Practice, Springer Verlag, Berlin (2005)
16. Raiffa, H.: Decision Analysis, McGraw-Hill, Columbus, Ohio (1997)

17. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & Hall/CRC, Boca Raton, Florida (2011)
18. Werbos, P.J.: Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University (1974)
19. Werbos, P.J.: The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting, Wiley, New York (1994).