# How Multi-View Techniques Can Help In Processing Uncertainty

Olga Kosheleva and Vladik Kreinovich
University of Texas at El Paso, El Paso, Texas, USA
emails olgak@utep.edu, vladik@utep.edu

**Abstract**

Multi-view techniques help us reconstruct a 3-D object and its properties from its 2-D (or even 1-D) projections. It turns out that similar techniques can be used in processing uncertainty – where many problems can reduced to a similar task of reconstructing properties of a multi-D object from its 1-D projections. In this chapter, we provide an overview of these techniques.

## 1 Introduction

**What are multi-view techniques: a brief reminder.** Our world is 3-dimensional. However, in most practical situations, we only see 2-D projections of the real-world objects, and we need to reconstruct the properties of the 3-D object based on these multi-view 2-D projections.

Because of the ubiquity of this problem, many advanced and efficient multi-view techniques have been developed.

**It is advantageous to apply ideas behind multi-view techniques in other problems as well.** Because this area is well advanced, it can be advantageous to use its techniques to solve other problems – problems which are less ubiquitous, more recent, and which are, therefore, somewhat behind multi-view research areas – at least in terms of the existence of efficient techniques.

**What we do in this chapter.** In this chapter, we show that multi-view techniques can be used in uncertainty quantification (UQ) – which, by the way, is important to multi-view analysis as well.

**What is uncertainty quantification and what it is important.** The main need for uncertainty quantification comes from the fact that, in general, data for processing come from measurements, and measurements are never absolutely accurate, there is always measurement error – the difference between the measurement result and the actual (unknown) value of the corresponding quantity; see, e.g., [21].

Because of this, the value that we obtain by processing measurement results is, in general, different from what we would have got if we processed the actual values of the corresponding quantities. In many practical situations, it is very important to know how big the resulting inaccuracy can be. For example, if we are prospecting for oil, and we found out that a certain field contains 150 million tons of oil, then our actions depend on the accuracy of this estimate.

- If it is 150 plus minus 20, we should start exploiting this field right away.
- However, if it is 150 plus minus 200, maybe there is no oil at all, so we should perform more measurements before investing a lot of money in production.

**Challenges of uncertainty quantification.** The usual techniques for uncertainty quantification are based on the idea of sensitivity analysis: since we do not know the values of the measurement errors, we simulate different possible combinations of such errors and analyze how it affects the result of data processing. The question is what is the best way to simulate these errors and what is the best way to process the results of this simulation.

**How multi-view techniques can help.** It turns out that, under reasonable assumptions, the sensitivity of the data processing algorithm can be described by a multi-D vector. In this description, simulation results are 1-D projections of this vector, so the UQ problem means analyzing the property of the multi-D vector based on its 1-D projections. This problem is similar to the usual multi-view analysis:

- on the one hand, the uncertainty-related problem is somewhat easier that the usual multi-view analysis, since we have 1-D (and not 2-D) projections, and since, as we will show, the object whose properties we want to reconstruct from these projections is just a vector;
- on the other hand, the uncertainty-related problem is somewhat more complex that the usual multi-view analysis, since the object of interest is now multi-D (and not just 3-D).

We show that multi-view reformulation of UQ problems can be useful for solving these problems.

## 2 Need for Uncertainty Quantification

**What do we want.** What do we the humanity want? In a nutshell:

- we want to predict what will happen in the future – this is one of the main objectives of science, and
- we want to know what we can do to make the future better – which actions to perform, which actions to avoid, what gadgets to use and how; this is one of the main objectives of engineering.

Of course, these tasks go beyond the narrowly understood engineering. For example:

- When we go to a doctor because of a cough, we want to know when this cough will stop, and what we need to do to stop it faster.
- When we teach students, we want to know whether they will learn the required material, and if the prediction is that many of them will fail the class – how to change out teaching strategy to make sure that more students succeed.

**Need for data processing.** The state of the world, the states of all its objects, the actions and gadgets – all this is usually described by numbers, by the numerical values of the corresponding quantities:

- numerical values $x_1, \ldots, x_n$ describe the current state of the world,
- numerical values $y$, $\ldots$, describe the future state of the world and the necessary actions.

In these terms, our objective is to determine all the desired values $y$ based on the available data $x_1, \ldots, x_n$. Once the algorithm $y = f(x_1, \ldots, x_n)$ for this determination is found, the computations become straightforward:

- we plug in the known values $x_i$ into this algorithm, and
- we get the value $y = f(x_1, \ldots, x_n)$ of the desired quantity.

This is called *data processing*. This is what computers were designed to do in the first place, this is what computers still do a lot.

**Need for uncertainty quantification.** The above description – in which we assumed that we know the exact values of the quantities $x_1, \ldots, x_n$ – was somewhat oversimplified. Yes, we have information about these values, but this information comes:

- either from measurements,
- or from expert estimates.

Neither of these two procedures produces exact values: there is always a difference $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$ between the estimate $\widetilde{x}_i$ and the actual (unknown) value $x_i$ of the corresponding quantity.

Because of this difference, even in the ideal case, when the algorithm $f = f(x_1, \ldots, x_n)$ reflects the exact relation between the quantities $x_i$ and $y$, the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ that we obtained by processing the estimates $\widetilde{x}_i$ is, in general, different from the value $y = f(x_1, \ldots, x_n)$ that we would have obtained if we had access to the exact value $x_i$.

In most practical situations, it is important to understand how big can be the corresponding difference $\Delta y = \widetilde{y} - y$. For example, if we program the trajectory

of a self-driving car in a tunnel, and we conclude that in the next second, it will be at a distance $\widetilde{y} = 1$ m from the wall, then the car's reaction should depend on the accuracy of this estimate:

- if this is 1 m $\pm$ 0.5 m, we are safe;
- however, if this is 1 m $\pm$ 2 m, then we need to do something, since otherwise, the car may get too close to the wall and crash.

Finding out what values $\Delta y$ are possible is known as *uncertainty quantification* (UQ). Informally, uncertainty quantification solves the following task:

- *given:* an algorithm $y = f(x_1, \ldots, x_n)$, the measurement results $\widetilde{x}_i$, and some information about the uncertainties $\Delta x_i = \widetilde{x}_i - x_i$,
- *find out* what are the possible values of the difference

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

# 3 What Makes Uncertainty Quantification Easier and What Makes It More Complex

**Linearization.** The uncertainty quantification problem is made easier by the fact that the estimation errors are usually small. Therefore, taking into account that $x_i = \widetilde{x}_i - \Delta x_i$, we can expand the dependence

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

in Taylor series in $\Delta x_i$ and ignore terms which are quadratic (or higher order) in $\Delta x_i$. As a result, we get the following formula

$$\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i, \tag{1}$$

where we denoted

$$c_i \stackrel{\text{def}}{=} \frac{\partial f}{\partial x_i}(\widetilde{x}_1, \ldots, \widetilde{x}_n). \tag{2}$$

The fact that we can consider linear dependence on $\Delta x_i$ makes the corresponding computations easier; see, e.g., [2, 12, 21, 23].

**Additional complexity.** In many simple examples, we know all the steps of the algorithm, and we can use this knowledge when solving the corresponding uncertainty quantification problems. This happens when someone wrote the corresponding program "from scratch", then the lines of this program provide a clear idea of what exactly is being done.

However, for complex tasks, it is not possible for one person to write the whole code from scratch. When people write the code for solving such problems, they try to use off-the-shelf packages as much as possible. Many of these

packages are proprietary, and it makes sense: to design a huge-size software, it is necessary to employ many programmers, they all need to be paid – and if everything is open access, no one will pay. This makes sense from the economic viewpoint, but from the viewpoint of uncertainty quantification, it makes life more complicated – since now the algorithm $f(x_1, \ldots, x_n)$ is largely a "black box" in the following sense:

- we can plug in different values $x_i$ and get the result of applying the algorithm $f$, but
- we do not know what exactly steps this algorithm went through to produce these results.

# 4 How Uncertainty Quantification Is Related to Multi-View Techniques

**Traditional case of uncertainty quantification.** In the traditional approach to uncertainty quantification (see, e.g., [21, 22]), we assume:

- that all estimation errors $\Delta x_i$ are independent, and
- that each estimation error is normally distributed with 0 mean and known standard deviation $\sigma_i$.

In this case, the expression $\Delta y$ – as described by the formula (1) – is also normally distributed, since it is known that a linear combination of several independent normally distributed random variables is also normally distributed; see, e.g., [22]. The mean of this linear combination is equal to the linear combination of the means, i.e., to 0, and the variance $\sigma^2$ of the linear combination (1) is determined by the formula

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

It is known that a normal distribution is uniquely determined by its mean and its standard deviation $\sigma$ (or, equivalently, its variance $\sigma^2$). Thus, in this case, the uncertainty quantification problem is reduced to the problem of computing the expression (3).

**What we can do to estimate the desired variance.** Since the data processing algorithm $y = f(x_1, \ldots, x_n)$ is given as a black box, its expression is not known, we cannot differentiate the corresponding function and find the actual values $c_i$. What we *can* do, once we have computed the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of data processing, is to try different tuples $(\Delta x_1, \ldots, \Delta x_n)$; for each such tuple:

- first, we plug in the values $x_i = \widetilde{x}_i - \Delta x_i$ into the algorithm $f(x_1, \ldots, x_n)$, resulting in the value $y = f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n)$, and

- then, we compute the difference $\Delta y = \widetilde{y} - y$ (and we know that this difference is equal to $\sum\limits_{i=1}^{n} c_i \cdot \Delta x_i$).

A typical way to select the values $\Delta x_i$ is to use Monte-Carlo simulations, i.e., to select the values $\Delta x_i$ which are normally distributed with zero mean and standard deviation $\sigma_i$. Usually, programming languages and simulation packages has methods for simulating the "standard" normal distribution, with 0 mean and standard deviation 1. The resulting desired random variable can be obtained from the result $\xi_i$ of the standard normal random number generator if we multiply this result by $\sigma_i$.

From this viewpoint, instead of directly generating the values $\Delta x_i$, it makes sense to:

- first generate the values $\xi_i$, and
- then take $\Delta x_i = \sigma_i \cdot \xi_i$.

In terms of $\xi_i$, the procedure of generating the corresponding value $\Delta y$ takes the following form:

- first generate the values $\xi_i$;
- then, we plug in the values $x_i = \widetilde{x}_i - \sigma_i \cdot \xi_i$ into the algorithm $f(x_1, \ldots, x_n)$, resulting in the value $y = f(\widetilde{x}_1 - \sigma_1 \cdot \xi_1, \ldots, \widetilde{x}_n - \sigma_n \cdot \xi_n)$, and
- finally, we compute the difference $\Delta y = \widetilde{y} - y$ (which we know to be equal to $\sum\limits_{i=1}^{n} c_i \cdot \sigma_i \cdot \xi_i$).

In terms of the auxiliary quantities $\xi_i$, the expression (1) takes the form

$$\Delta y = \sum_{i=1}^{n} c_i \cdot \sigma_i \cdot \xi_i. \tag{4}$$

This expression can be simplified if we denote $a_i \stackrel{\text{def}}{=} c_i \cdot \sigma_i$, then the expression (4) takes the form

$$\Delta y = \sum_{i=1}^{n} a_i \cdot \xi_i. \tag{5}$$

Interestingly, in terms of $a_i$, the desired expression (3) also gets a simplified form:

$$\sigma^2 = \sum_{i=1}^{n} a_i^2. \tag{6}$$

Now, we are ready to describe the relation to multi-view techniques.

**Geometric formulation of the problem and its relation to multi-view techniques.** In geometric terms:

- the values $a_i$ form a vector $\vec{a} = (a_1, \ldots, a_n)$, and
- the values $\xi_i$ form a vector $\vec{\xi} = (\xi_1, \ldots, \xi_n)$.

In terms of these vectors:

- the value $\Delta y$ – as described by the expression (5) – is simply a scalar (dot) product $\vec{a} \cdot \vec{\xi}$ of these two vectors, and
- the value $\sigma^2$ – as describes by the formula (3) – is simply the square $\|\vec{a}\|^2$ of the length $\|\vec{a}\| = \sqrt{\sum_{i=1}^{n} a_i^2}$ of the vector $\vec{a}$.

So, the standard deviation $\sigma = \sqrt{\sigma^2}$ is simply equal to the length $\|\vec{a}\|$ of the vector $\vec{a}$.

Thus, in these geometric terms, the newly reformulated problem takes the following:

- there is a vector $\vec{a} = (a_1, \ldots, a_n)$ that we do not know;
- we want to find the length $\|\vec{a}\|$ of this vector;
- for this purpose, for different vectors $\vec{\xi}$, we can compute the scalar product $\vec{a} \cdot \vec{\xi}$.

Knowing the scalar product is equivalent to knowing the projection

$$\pi_{\vec{\xi}}(\vec{a}) = \frac{\vec{a} \cdot \xi}{\|\xi\|}$$

of the unknown vector $\vec{a}$ on the 1-D space generated by the vector $\vec{\xi}$.

Thus, the problem takes the following form:

- we want to estimate some characteristic of the unknown multi-D object $\vec{a}$
- by studying its projection on different 1-D spaces.

In this form, this is clearly a particular cases of the general multi-view reconstruction problem. The main difference from the usual cases of the multi-view problem is that:

- usually, we reconstruct a 3-D object from its 2-D projections;
- now, we reconstruct a multi-D object from its 1-D projections.

*Comment.* We explained the relation between uncertainty quantification and multi-view techniques on the example of the traditional approach to uncertainty quantification; however, as we will see in this paper, the same relation can be traced for all other types of uncertainty.

# 5 Straightforward ("Naive") Approach and Its Limitations

**Straightforward approach: main idea.** In terms of the multi-view reformulation of our problem, our goal is to find the length of the vector $\vec{a}$. According to the formula for the length, a straightforward way to compute this length is:

- to find all the components $a_1, \ldots, a_n$ of this vector, and
- then, to compute $\|\vec{a}\|$ as $\sqrt{\sum_{i=1}^{n} a_i^2}$.

A straightforward way to compute each component $a_i$ is to take into account that this component is simply a scalar product of the vector $\vec{a}$ and the vector $\vec{e}^{(i)} = (0, \ldots, 0, 1, 0, \ldots, 0)$ that has all components equal to 0 with the exception of the $i$-th component $e_i^{(i)}$ which is equal to 1: $a_i = \vec{a} \cdot \vec{e}^{(i)}$.

Thus, we arrive at the following straightforward algorithm for computing the desired length $\|\vec{a}\|$ – and thus, for solving the corresponding uncertainty quantification problem.

**Resulting algorithm.**

- for $i = 1, \ldots, n$, we take $\vec{\xi} = \vec{e}^{(i)}$ and compute the corresponding value $\Delta y$; we will denote the resulting value of $\Delta y$ by $a_i$;
- then, we compute the desired length as $\|\vec{a}\| = \sqrt{\sum_{i=1}^{n} a_i^2}$.

**Resulting algorithm: a detailed description.** If we explicitly describe how $\Delta y$ is computed, we arrive at the following detailed description of the above algorithm:

- for each $i$ from 1 to $n$, we prepare the values $\xi_1^{(i)}, \ldots, \xi_n^{(i)}$ for which $\xi_i^{(i)} = 1$ and $\xi_j^{(i)} = 0$ for all $j \neq i$;
- then, we apply the algorithm $f$ to the values

  $$x_1 = \widetilde{x}_1 - \sigma_1 \cdot \xi_1^{(i)}, \ldots, x_n = \widetilde{x}_n - \sigma_n \cdot \xi_n^{(i)},$$

  thus computing the value $y^{(i)} = f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i - \sigma_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n)$, and compute $a_i = \widetilde{y} - y^{(i)}$;
- finally, we compute $\sigma = \sqrt{\sum_{i=1}^{n} a_i^2}$.

**Limitations of the straightforward approach.** The above straightforward algorithm requires $n$ computations of the corresponding quantity $\Delta y$. As we have mentioned earlier, each of these computations means applying the algorithm $f(x_1, \ldots, x_n)$ to appropriate values $x_i$. Thus:

- in addition to the original call to the algorithm $f$ – to compute the original result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$,
- we need to call this algorithm $n$ more times.

We have also mentioned that:

- in many practical situations, this algorithm is complicated, each call may require hours on a high performance computer, and
- the number of inputs $n$ may be huge – for problems like weather predictions, we process thousands of inputs.

Thus, using the straightforward approach would mean spending thousands times more computation time that the actual computation – months instead of hours. This is clearly not realistic. That is why the above straightforward approach is sometimes called "naive" approach.

# 6 Monte-Carlo Approach: Traditional Probabilistic Case

**Main idea: reminder.** If the values $\xi_i$ are independent normally distributed random variables with 0 means and standard deviation (1), then their linear combination $\vec{a} \cdot \vec{\xi} = \sum_{i=1}^{n} a_i \cdot \xi_i$ is also normally distributed, with 0 mean and standard deviation $\sigma = \sqrt{\sum_{i=1}^{n} a_i^2}$.

So, if we simply simulate all these $\xi_i$, then the resulting value $\Delta y = \vec{a} \cdot \vec{\xi}$ will be normally distributed with mean 0 and standard deviation equation to the desired value $\|\vec{a}\|$. If we repeat this simulation several ($N$) times, we get a sample of values $\Delta y^{(k)}$, based on which we can compute the sample standard deviation

$$\sqrt{\frac{1}{N} \cdot \sum_{k=1}^{N} \left(\Delta y^{(k)}\right)^2}.$$

It is known (see, e.g., [22]), that the accuracy with which the sample standard deviation approximates the actual one is proportional to $1/\sqrt{N}$. So, e.g., if we want to compute the desired value $\sigma$ with the relative accuracy of 20%, it is sufficient to run just $N = 25$ simulations – since in this case, $1/\sqrt{N} = 20\%$. This is clearly much faster than thousands of calls to $f$ needed for the straightforward approach.

By the way, 20% relative accuracy in determining the uncertainty is very good – it means, e.g., distinguishing between 10% and 12% accuracy. Usually, an even lower accuracy is sufficient: we say that the accuracy is $\pm 10\%$ or $\pm 15\%$, but rarely $\pm 12\%$.

Let us describe the above idea in precise terms.

**Resulting algorithm.** For each $k = 1, \ldots, N$ (where $N$ is determined by the desired relative accuracy), we do the following:

- first, for each $i$ from 1 to $n$, we use the random number generator (generating standard normal distribution), and get the corresponding values $\xi_i^{(k)}$;
- then, we use the resulting vector $\vec{\xi}^{(k)}$ to compute the corresponding value $\Delta y^{(k)}$.

Once we have all these values, we compute $\|\vec{a}\| = \sqrt{\dfrac{1}{N} \cdot \sum_{k=1}^{N} \left(\Delta y^{(k)}\right)^2}$.

Let us describe this idea in detail.

# 7 Algorithm for the Probabilistic Case

**What is given.** We are *given*:

- the values $\widetilde{x}_1, \ldots, \widetilde{x}_n$;
- the standard deviations $\sigma_1, \ldots, \sigma_n$; and
- an algorithm $f(x_1, \ldots, x_n)$ given as a black box.

We also know the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

**What we want.** Our *goal* is to compute the standard deviation $\sigma$ of the difference $\Delta y = \widetilde{y} - f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n)$, where $\Delta x_i$ are independent random variables with 0 mean and standard deviation $\sigma_i$.

**Algorithm.** First, we select the number of iterations $N$ based on the desired relative accuracy $\varepsilon > 0$: we want $1/\sqrt{N} \approx \varepsilon$, so we take $N \approx \varepsilon^{-2}$.

Then, for each $k = 1, \ldots, N$, we do the following:

- first, for each $i$ from 1 to $n$, we use the random number generator (generating standard normal distribution), and get the corresponding values $\xi_i^{(k)}$;
- then, we plug in the values $x_i = \widetilde{x}_i - \sigma_i \cdot \xi_i^{(k)}$ into the algorithm $f$, thus computing the difference

$$\Delta y^{(k)} = \widetilde{y} - f\left(\widetilde{x}_1 - \sigma_1 \cdot \xi_1^{(k)}, \ldots, \widetilde{x}_n - \sigma_n \cdot \xi_n^{(k)}\right);$$

- once we have all these values, we compute $\sigma = \sqrt{\dfrac{1}{N} \cdot \sum_{k=1}^{N} \left(\Delta y^{(k)}\right)^2}$.

# 8 Need to Go Beyond the Traditional Probabilistic Case

**Traditional probabilistic case: reminder.** The above algorithms are intended for the case when:

- we know the probability distributions of all the approximation errors $\Delta x_i$,
- all these distributions are normal, with 0 mean and know standard deviation $\sigma_i$; and
- all approximation errors are independent.

**This case does frequently occur in practice.** In practice, there are indeed many situations where all these three conditions are satisfied.

Indeed, factors affecting the inaccuracy of different measurements – such as noise at the location of different sensors – are indeed reasonably independent.

Also, if we know the probability distributions, then the requirement that the mean is 0 makes sense:

- if the mean is not 0, i.e., in probabilistic terms, if we have a *bias*,
- then we can simply subtract this bias from all the measurement results and thus, end up with measurements for which the mean error is 0.

Even normality makes sense. Indeed, usually:

- each measurement error is the joint effect of many different independent factors, and,
- according to the Central Limit Theorem (see, e.g., [22]), the distribution of such a joint effect is close to normal.

Indeed, empirical studies [19, 20] show that for about 60% measuring instruments, the probability distribution of measurement errors is close to normal.

**Need to go beyond normal distributions.** On the other hand, the same statistics means that for about 40% of the measuring instruments the probability distribution is *not* close to normal.

So, the first thing we need to do is to extend out methods to this case – when we *know* the probability distributions, and we know that at least some of them are not normal.

**But do we always know the probability distributions?** To determine the probability distribution of the measurement error, we need to thoroughly study and test each individual sensor, each individual measuring instrument. Such a testing involves comparing the results of these measurements with the results of some much more accurate ("standard") measuring instrument. This is usually

possible, but it is a very expensive procedure – especially taking into account that many sensors are now very cheap.

This procedure makes sense if we are planning a manned space flight or a control system for a nuclear power station, where a wrong decision can lead to catastrophic consequences. However, if, for the purpose of weather prediction, we design a network of reasonably cheap temperature, wind, and humidity sensors, the corresponding expenses are not worth the effort.

**What do we know: possible information about uncertainty.** Since we do not always perform a thorough study of each sensor, in many situations:

- instead of knowing the exact probability distribution,
- we only have partial information about the corresponding probabilities.

In all cases, we should know a guaranteed upper bound $\Delta$ on the absolute value $|\Delta x|$ of the measurement error: $|\Delta x| \le \Delta$. Such an upper bound is needed, since otherwise, if there is no guaranteed upper bound, this would mean that the actual value can be as far from the measurement result as mathematically possible: this is not a measurement, this is a wild guess.

Once we know the upper bound $\Delta$, then, once we have the measurement result $\widetilde{x}$, the only thing that we can conclude about the actual (unknown) value $x$ is that this value belongs to the interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$, and we have no information about the probability of different values from this interval. This case is known as *interval uncertainty*; see, e.g., [3, 13, 15, 21].

In some cases, we have partial information about the probabilities: e.g., we know:

- the upper bound $\widetilde{\Delta}$ on the absolute value of the mean $E[\Delta x]$ – this mean is known as the *systematic error component*, and
- we know the upper bound $\widetilde{\sigma}$ on the standard deviation, i.e., the mean square value of the difference $\Delta x - E[\Delta x]$ between the measurement error and its mean; this difference is known as the *random error component*.

This is probably the most frequent type of information about the measurement accuracy [21].

There is also a case when some (or even all) estimates $\widetilde{x}_i$ come not from measurements, but from an expert estimates. In this case, the only information that we have about the accuracy of this estimate also comes from the expert, and the expert describes this information in imprecise ("fuzzy") terms of natural language, e.g., by saying that the accuracy is "most probably plus minus 0.5". Such situations are known as situations of *fuzzy uncerainty*; see, e.g., [1, 5, 14, 17, 18, 25].

Finally, there are cases when:

- for some measurements, we know the probability distributions of the measurement errors,

- for some other measurements, we only know interval bounds, and
- for yet other values, we only have fuzzy descriptions of their accuracy.

In this chapter, we show that in all these cases, multi-view ideas can be helpful. Let us describe how these ideas can be helpful – by considering all different types of uncertainty in the same order in which we have just listed them.

# 9 Cased When We Know the Probability Distributions but They Are Not Necessarily Normal

**Description of the case.** We consider the case when:

- we know the probability distributions of all the measurement errors $\Delta x_i$,
- for each of these distributions, the mean value is 0, and
- measurement errors corresponding to different measurements are independent.

**Analysis of the problem.** For large $n$, the value $\Delta y$ is the sum of a large number small independent random variables $c_i \cdot \Delta x_i$. Thus, due to the same Central Limit Theorem that explains the ubiquity of normal distributions, the distribution of $\Delta y$ is close to normal.

As we have mentioned earlier, a 1-D normal distribution is uniquely determined by its mean and standard deviation. Similarly to the normal cases, we can conclude that the mean is 0, and that the standard deviation is determined by the formula (3).

Thus, to compute $\sigma$, we can simply ignore all the information about the known distributions and only take into account the variances $\sigma_i$. Once we have found these values, we get the exact same mathematical problem as in the normal case – and the exact same multi-view reformulation of this problem. Thus, we can follow the same algorithm as for the normal case.

In other words, we arrive at the following algorithm.

**Resulting algorithm: detailed description.** First, for each $i$, we use our knowledge of the probability distribution of the corresponding measurement error $\Delta x_i$ to compute the corresponding standard deviation $\sigma_i$.

Then, for each $k = 1, \ldots, N$ (where $N$ is determined by the desired relative accuracy), we do the following:

- first, for each $i$ from 1 to $n$, we use the random number generator generating standard normal distribution, and get the corresponding values $\xi_i^{(k)}$;
- then, we plug in the values $x_i = \widetilde{x}_i - \sigma_i \cdot \xi_i^{(k)}$ into the algorithm $f$, thus computing the difference

$$\Delta y^{(k)} = \widetilde{y} - f\left(\widetilde{x}_1 - \sigma_1 \cdot \xi_1^{(k)}, \ldots, \widetilde{x}_n - \sigma_n \cdot \xi_n^{(k)}\right);$$

- once we have all these values, we compute $\sigma = \sqrt{\dfrac{1}{N} \cdot \sum\limits_{k=1}^{N} \left(\Delta y^{(k)}\right)^2}$.

**Important comment.** In the algorithm for the normal case, we adequately simulated the measurement errors, by emulating the exact same distribution as we know they have. In this case, however, the actual distributions are *not* normal, but we still use normal distributions.

In other words, in this case, the Monte-Carlo method that we use:

- is not a realistic simulation of measurement errors,
- it is a computational trick which leads to the same result as the actual simulation but which is computationally much faster.

This trick makes computations faster since, with this trick, there is no need to spend computation time emulating details of complex distributions: we can simply use standard (and fast) random number generator for normally distributed variables.

We make this comment because we will observe the same phenomenon in other algorithms as well:

- many of our Monte-Carlo simulations will *not* be adequately representing the actual distributions,
- they will serve as mathematical tricks helping us to compute the desired solution.

A reader should not be surprised by this: our main idea – reduction to a multi-view problem – is also, in effect, a mathematical trick and not a direct representation of the corresponding uncertainty quantification problem.

# 10 Case of Interval Uncertainty

**Formulation of the problem.**

- We know that the measurement errors $\Delta x_i$ can take any values from the interval $[-\Delta_i, \Delta_i]$.
- Under this condition, we need to find the range

$$\left\{\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i : |\Delta x_i| \le \Delta_i \text{ for all } i\right\}$$

of possible values of their linear combination $\Delta y = \sum\limits_{i=1}^{n} c_i \cdot \Delta x_i$.

**Analysis of the problem.** The largest possible value of the sum is attained when each of the terms $c_i \cdot \Delta x_i$ is the largest. Let us consider two possible cases: $c_i \geq 0$ and $c_i \leq 0$.

- If $c_i \geq 0$, then $c_i \cdot \Delta x_i$ is an increasing function of $\Delta x_i$. Thus, its largest value is attained when the variable $\Delta x_i$ is the largest possible, i.e., when $\Delta x_i = \Delta_i$. For this value $\Delta x_i$, the term is equal to $c_i \cdot \Delta_i$.
- If $c_i \leq 0$, then $c_i \cdot \Delta x_i$ is a decreasing function of $\Delta x_i$. Thus, its largest value is attained when the variable $\Delta x_i$ is the smallest possible, i.e., when $\Delta x_i = -\Delta_i$. For this value $\Delta x_i$, the term is equal to $c_i \cdot (-\Delta_i) = -c_i \cdot \Delta_i$.

In both cases, the largest possible value of each term is equal to $|c_i| \cdot \Delta_i$. Thus, the largest possible value $\Delta$ of $\Delta y$ is equal to the sum of these values

$$\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i. \tag{7}$$

Similarly, we can prove that the smallest possible value of $\Delta y$ is equal to $-\Delta$.

Thus, in the interval case, uncertainty quantification is reduced to the problem of computing the value (7).

**What we can do to estimate $\Delta$.** Similarly to the probabilistic case, since the data processing algorithm $y = f(x_1, \ldots, x_n)$ is given as a black box, the only thing we can do is to try different tuples $(\Delta x_1, \ldots, \Delta x_n)$; for each such tuple:

- first, we plug in the values $x_i = \widetilde{x}_i - \Delta x_i$ into the algorithm $f(x_1, \ldots, x_n)$, resulting in the value $y = f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n)$, and
- then, we compute the difference $\Delta y = \widetilde{y} - y$ (and we know that this difference is equal to $\sum\limits_{i=1}^{n} c_i \cdot \Delta x_i$).

**Reformulating the problem.** Similarly to the probabilistic case, instead of directly generating the value $\Delta x_i$, let us generate the auxiliary values $\xi_i$, and then take $\Delta x_i = \Delta_i \cdot \xi_i$.

In terms of $\xi_i$, the procedure of generating the corresponding value $\Delta y$ takes the following form:

- first generate the values $\xi_i$;
- then, we plug in the values $x_i = \widetilde{x}_i - \Delta_i \cdot \xi_i$ into the algorithm $f(x_1, \ldots, x_n)$, resulting in the value $y = f(\widetilde{x}_1 - \Delta_1 \cdot \xi_1, \ldots, \widetilde{x}_n - \Delta_n \cdot \xi_n)$, and
- finally, we compute the difference $\Delta y = \widetilde{y} - y$ (which we know to be equal to $\sum\limits_{i=1}^{n} c_i \cdot \Delta_i \cdot \xi_i$).

In terms of the auxiliary quantities $\xi_i$, the expression (1) takes the form

$$\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta_i \cdot \xi_i. \tag{8}$$

This expression can be simplified if we denote $a_i \stackrel{\text{def}}{=} c_i \cdot \Delta_i$, then the expression (8) takes the form

$$\Delta y = \sum_{i=1}^{n} a_i \cdot \xi_i. \tag{9}$$

Interestingly, in terms of $a_i$, the desired expression (7) also gets a simplified form:

$$\Delta = \sum_{i=1}^{n} |a_i|. \tag{10}$$

Now, we are ready to describe the relation to multi-view techniques.

**Relation to multi-view techniques.** Similarly to the probabilistic case, let us consider the two vectors:

- a vector $\vec{a} = (a_1, \ldots, a_n)$ formed by the values $a_i$, and
- a vector $\vec{\xi} = (\xi_1, \ldots, \xi_n)$ formed by the values $\xi_i$.

In terms of these vectors:

- the value $\Delta y$, as we have mentioned earlier, is simply a scalar (dot) product $\vec{a} \cdot \vec{\xi}$ of these two vectors, and
- the value $\Delta$ – as describes by the formula (10) – is simply the $\ell^1$-norm $\|\vec{a}\|_1 \stackrel{\text{def}}{=} \sum\limits_{i=1}^{n} |a_i|$ of the vector $\vec{a}$.

Thus, in these geometric terms, the newly reformulated problem takes the following:

- there is a vector $\vec{a} = (a_1, \ldots, a_n)$ that we do not know;
- we want to find the $\ell^1$-norm $\|\vec{a}\|_1$ of this vector;
- for this purpose, for different vectors $\vec{\xi}$, we can compute the scalar product $\vec{a} \cdot \vec{\xi}$.

Similarly to the probabilistic case, knowing the scalar product is equivalent to knowing the projection

$$\pi_{\vec{\xi}}(\vec{a}) = \frac{\vec{a} \cdot \xi}{\|\xi\|}$$

of the unknown vector $\vec{a}$ on the 1-D space generated by the vector $\vec{\xi}$. Thus, the problem takes the following form:

- we want to estimate some characteristic of the unknown multi-D object $\vec{a}$
- by studying its projection on different 1-D spaces.

In this form, this is clearly a particular cases of the general multi-view reconstruction problem.

**Straightforward algorithm.** In principle, similarly to the probabilistic case, we can use the following straightforward algorithm to compute the desired value $\Delta$:

- for each $i$ from 1 to $n$, we prepare the values $\xi_1^{(i)}, \ldots, \xi_n^{(i)}$ for which $\xi_i^{(i)} = 1$ and $\xi_j^{(i)} = 0$ for all $j \neq i$;
- then, we apply the algorithm $f$ to the values

  $$x_1 = \widetilde{x}_1 - \Delta_1 \cdot \xi_1^{(i)}, \ldots, x_n = \widetilde{x}_n - \Delta_n \cdot \xi_n^{(i)},$$

  thus computing the value $y^{(i)} = f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i - \Delta_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n)$, and compute $a_i = \widetilde{y} - y^{(i)}$;
- finally, we compute $\Delta = \sum_{i=1}^{n} |a_i|$.

However, this algorithm has the same main limitation as in the probabilistic case: it requires too much computation time.

**Monte-Carlo algorithm: main idea.** To estimate $\Delta$ faster, we can use the known fact about Cauchy distribution, a probability distribution characterised by the following probability density function:

$$f(x) = \frac{1}{\pi \cdot \Delta} \cdot \frac{1}{1 + \dfrac{x^2}{\Delta^2}}. \tag{11}$$

The standard case is when $\Delta = 1$.

This fact is that if $n$ independent random variables $\xi_1, \ldots, \xi_n$ are distributed according to the standard Cauchy distribution, then their linear combination $\sum_{i=1}^{n} a_i \cdot \xi_i$ is distributed according to the Cauchy distribution with parameter $\Delta = \sum_{i=1}^{n} |a_i|$. Thus, similarly to how we use normal distributions to estimate $\sigma$ in the probabilistic case, we can use Cauchy distribution in the interval case.

There are two additional computational problems here that we did not encounter in the probabilistic case:

- first, in contrast to the probabilistic case, we do not have a ready random number generator generating Cauchy distribution; this problem can be easily solved: we can take a random variable $u$ uniformly distributed on the interval $[0, 1]$ (for which the random number generator exists), and take

  $$\xi = \tan(\pi \cdot (u - 0.5)); \tag{12}$$

- second, once we get a Cauchy distributed sample of values $\Delta y^{(k)}$, there is no standard way to estimate the parameter $\Delta$; for this purpose, we can use the Maximum Likelihood approach of finding the most probable value $\Delta$; this leads to the need to solve the following system of equations:

$$\sum_{k=1}^{N} \frac{1}{1+\frac{\left(\Delta y^{(k)}\right)^2}{\Delta^2}} - \frac{N}{2} = 0. \tag{13}$$

As a result, we arrive at the following algorithm (see, e.g., [6, 7, 8, 9, 10, 11, 24, 16] for details):

**Resulting algorithm: first approximation.** For each $k = 1, \ldots, N$ (where $N$ is determined by the desired relative accuracy), we do the following:

- first, for each $i$ from 1 to $n$, we use the formula (12) and get the corresponding values $\xi_i^{(k)}$;
- then, we use the resulting vector $\vec{\xi}^{(k)}$ to compute the corresponding value $\Delta y^{(k)}$.

Once we have all these values, we compute $\Delta$ by solving the equation (13).

**First approximation: detailed description.** For each $k = 1, \ldots, N$ (where $N$ is determined by the desired relative accuracy), we do the following:

- first, for each $i$ from 1 to $n$, we use the random number generator for generating a random number $u_i^{(k)}$ which is uniformly distributed on the interval $[0, 1]$, and then compute $\xi_i^{(k)} = \tan\left(\pi \cdot \left(u_i^{(k)} - 0.5\right)\right)$;
- then, we plug in the values $x_i = \widetilde{x}_i - \Delta_i \cdot \xi_i^{(k)}$ into the algorithm $f$, thus computing the difference

$$\Delta y^{(k)} = \widetilde{y} - f\left(\widetilde{x}_1 - \Delta_1 \cdot \xi_1^{(k)}, \ldots, \widetilde{x}_n - \Delta_n \cdot \xi_n^{(k)}\right);$$

- once we have all these values, we compute $\Delta$ by solving the equation (13).

**Additional details: how to solve the equation (13).** To find $\Delta$, we can use, e.g., a bisection algorithm starting with the interval $\left[0, \max_{k=1,\ldots,N} \left|\Delta y^{(k)}\right|\right]$:

- for the value $\Delta$ corresponding to the left endpoint of this interval, the left-hand side of the equation (13) is negative, while
- for the right endpoint, the left-hand side is positive.

At each step, we take a midpoint of the previous interval, and select either the left or the right half-interval, so that in the endpoints of the selected half-interval, the left-hand side of (13) has different signs.

**Important computational comment.** Cauchy distributed values $\Delta x_i$ can be large, while linearization is only possible for small deviations $\Delta x_i$. To make sure that all deviations are within the linearity range, we need:

- to normalize all the simulated measurement errors by dividing them by a sufficiently large value $M$, and then
- re-scale the resulting values $\Delta y^{(k)}$ back, by multiplying them by the same value $M$.

A natural way to make sure that all simulated values $\Delta x_i$ are within the range $[-\Delta_i, \Delta_i]$ – or, equivalently, that all the values $\xi$ are within the interval $[-1, 1]$ – is to divide all the simulated values $\xi_i^{(k}$ by the largest of their absolute values.

As a result, the actual algorithm has the following modified form:

# 11 Final Algorithm for the Interval Case

**What is given.** We are *given*:

- the values $\widetilde{x}_1, \ldots, \widetilde{x}_n$;
- the values $\Delta_1, \ldots, \Delta_n$; and
- an algorithm $f(x_1, \ldots, x_n)$ given as a black box.

We also know the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

**What we want.** Our *goal* is to compute the range

$$\{y = f(x_1, \ldots, x_n) : x_i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i] \text{ for all } i\}.$$

**Algorithm.** To compute the desired range:

- first, for each $i$ from 1 to $n$, we use the random number generator for generating a random number $u_i^{(k)}$ which is uniformly distributed on the interval $[0, 1]$, and then compute $\xi_i^{(k)} = \tan\left(\pi \cdot \left(u_i^{(k)} - 0.5\right)\right)$;
- we then compute $M = \max\limits_{i,k} \left|\xi_i^{(k)}\right|$;
- then, we plug in the values

$$x_i = \widetilde{x}_i - \Delta_i \cdot \frac{\xi_i^{(k)}}{M}$$

into the algorithm $f$, thus computing the value

$$\Delta y^{(k)} = M \cdot \left( \widetilde{y} - f\left( \widetilde{x}_1 - \Delta_1 \cdot \frac{\xi_1^{(k)}}{M}, \ldots, \widetilde{x}_n - \Delta_n \cdot \frac{\xi_n^{(k)}}{M} \right) \right);$$

- once we have all these values, we compute $\Delta$ by solving the equation

$$\sum_{k=1}^{N} \frac{1}{1 + \frac{\left(\Delta y^{(k)}\right)^2}{\Delta^2}} - \frac{N}{2} = 0. \tag{13}$$

To find $\Delta$, we can use a bisection algorithm starting with the interval $\left[0, \max\limits_{k=1,\ldots,N} \left|\Delta y^{(k)}\right|\right]$:

- for the value $\Delta$ corresponding to the left endpoint of this interval, the left-hand side of the equation (13) is negative, while
- for the right endpoint, the left-hand side is positive.

At each step, we take a midpoint of the previous interval, and select either the left or the right half-interval, so that in the endpoints of the selected half-interval, the left-hand side of (13) has different signs.

The resulting range is equal to $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$.

**Methodological comment.** Here, as we warned, we have another example when a Monte-Carlo approach is not based on truthful simulation:

- we do not know the actual probability distribution, but
- we select a certain distribution for simulation – which is most probably different from the actual (unknown) probability distribution.

Similarly to the case of non-normal distributions, here too this use of Monte-Carlo simulations is a mathematical trick helping us to compute the result fast.

# 12 What If We Have Information about Systematic and Random Error Components

**Description of the case: reminder.** Let us consider the case which is most common in measurement practice, when for each measurement error $\Delta x_i$, we know:

- the upper bound $\widetilde{\Delta}_i$ on the absolute value $|E[\Delta x_i]|$ of its mean value, and
- the upper bound $\widetilde{\sigma}_i$ on its standard deviation.

What can we then conclude about the mean $m$ and the standard deviation $s$ of the value $\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i$?

**Analysis of the problem.** From the formula (1), we conclude that

$$m = E[\Delta y] = \sum_{i=1}^{n} c_i \cdot E[\Delta x_i]. \tag{14}$$

We only information that we have about each value $E[\Delta x_i]$ is that this value is somewhere in the interval $\left[-\widetilde{\Delta}_i, \widetilde{\Delta}_i\right]$. Thus, from the mathematical viewpoint, this is exactly the problem of uncertainty quantification under interval uncertainty – so we can use the above-described interval algorithm to find the largest possible value $\widetilde{\Delta}$ of the absolute value $|m|$ of the mean $m$.

For the standard deviation, we have the formula (3), i.e., we have $\sigma^2 = \sum_{i=1}^{n} c_i^2 \cdot \sigma_i^2$. However, here, in contrast to the previously described probabilistic case:

- we do not know the standard deviations $\sigma_i$,
- we only know the upper bounds $\widetilde{\sigma}_i$ for which $\sigma_i \le \widetilde{\sigma}_i$.

The expression $\sigma^2 = \sum_{i=1}^{n} \sigma_i^2$ is increasing with respect to each of the unknowns $\sigma_i$. Thus, its largest possible value is attained when each of the values $\sigma_i$ is the largest possible, i.e., when $\sigma_i = \widetilde{\sigma}_i$. For $\sigma_i = \widetilde{\sigma}_i$, the resulting largest possible value $\widetilde{\sigma}$ has the form $(\widetilde{\sigma})^2 = \sum_{i=1}^{n} c_i^2 \cdot (\widetilde{\sigma}_i)^2$. This is the same formula as for the probabilistic case – so, to compute $\widetilde{\sigma}$, we can use the above-described probabilistic algorithm.

Thus, we arrive at the following algorithm;

**Algorithm.** First, we apply the interval algorithm to transform:

- the bounds $\widetilde{\Delta}_i$ on the absolute value of the means of the measurement errors into
- the bound $\widetilde{\Delta}$ on the absolute value of the mean of the resulting approximation error $\Delta y$.

Then, we apply the probabilistic-case algorithm to transform:

- the bounds $\widetilde{\sigma}_i$ on the standard deviations of the measurement errors into
- the bound $\widetilde{\sigma}$ for the standard deviation of the resulting approximation error $\Delta y$.

# 13 Fuzzy Case

**Formulation of the problem.** Suppose now that for each $i$, we only have fuzzy information about each estimate $\widetilde{x}_i$, i.e., that we have a fuzzy number $\mu_i(x_i)$ that describes the corresponding uncertainty. We want to find the membership function $\mu(y)$ that describes the result of applying the algorithm $y = f(x_1, \ldots, x_n)$ to these uncertain inputs.

**Analysis of the problem.** In fuzzy techniques, the usual way to transform the initial uncertainty into the uncertainty of the result of data processing is to use the so-called Zadeh's extension principle, which basically means that for each $\alpha \in (0, 1]$, the corresponding "$\alpha$-cut" $\mathbf{y}(\alpha) \stackrel{\text{def}}{=} \{y : \mu(y) \geq \alpha\}$ is obtained from the $\alpha$-cuts of the inputs $\mathbf{x}_i(\alpha) = \{x_i : \mu_i(x_i) \geq \alpha\}$ by the usual interval formula

$$\mathbf{y}(\alpha) = \{f(x_1, \ldots, x_n) : x_i \in \mathbf{x}_i(\alpha) \text{ for all } i\}.$$

Thus, for each $\alpha$, we can apply the above interval algorithm to the corresponding $\alpha$-cuts.

For this purpose, each $\alpha$-cut interval needs to be represented in the center-radius form, i.e., as $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$, where:

- $\widetilde{x}_i$ is the interval's center and
- $\Delta_i$ its half-width ("radius").

Thus, we arrive at the following algorithm.

**Algorithm.** For each $\alpha$, we:

- represent each $\alpha$-cut $\mathbf{x}_i(\alpha)$ in the center-radius form, and
- then use the interval algorithm to compute the range.

This range will be the desired $\alpha$-cut $\mathbf{y}(\alpha)$ for $y$.

# 14 General Case: What If We Know Different Inputs with Different Uncertainty

**Formulation of the problem.** Let us consider the most general case, when we have inputs of all possible types. For the following three types of uncertainty, we know the measurement result $\widetilde{x}_i$, and we also have the following additional information:

- For some inputs $i$, we know that the mean value of the measurement error is 0, and we know the standard deviation $\sigma_i$ of the measurement error. We will denote the set of all such inputs by $I_{\text{p}}$, where p stands for "probabilistic".

- For other inputs $i$, we know the upper bound $\Delta_i$ on the absolute value of the mean of the measurement error, and we know the upper bound $\sigma_i$ on the standard deviation of the measurement error. We will denote the set of all such inputs by $I_{\rm m}$, where m stands for "measurements".
- For some inputs $i$, we only know the upper bound $\Delta_i$ on the absolute value of the measurement error. We will denote the set of all such inputs by $I_{\rm i}$, where i stands for "interval".

Finally, for some inputs, instead of the measurement result, we know only the fuzzy number $\mu_i(x_i)$. We will denote the set of all such inputs by $I_{\rm f}$, where f stands for "fuzzy".

**Analysis of the problem and the resulting algorithm.** If the set $I_f$ is non-empty, let us pick some values $\alpha \in (0,1]$.

Then, we extend the definitions of $\Delta_i$, $\sigma_i$, and $\widetilde{x}_i$ to all indices $i$:

- for $i \in I_{\rm p}$, we take $\Delta_i = 0$;
- for $i \in I_{\rm i}$, we take $\sigma_i = 0$; and
- for $i \in I_{\rm f}$, we take $\sigma_i = 0$; as $\Delta_i$, we take the radius of the $\alpha$-cut $\mathbf{x}_i(\alpha)$, and as $\widetilde{x}_i$, we take the center of the $\alpha$-cut.

Under this definition, for each $i$, the corresponding value $\Delta x_i \stackrel{\rm def}{=} \widetilde{x}_i - x_i$ can be represented as the sum

$$\Delta x_i = \Delta x_{si} + \Delta x_{ri}, \tag{15}$$

of "systematic" and "random" components, where:

- the only thing we know about $\Delta x_{si}$ is that $|\Delta_{si}| \le \Delta_i$, and
- the only thing we know about $\Delta x_{ri}$ is that it is a random variable with 0 mean and standard deviation not exceeding $\sigma_i$.

Substituting the expression (15) into the formula (1), we conclude that:

$$\Delta y = \Delta y_r + \Delta y_s,$$

where we denoted $\Delta y_r \stackrel{\rm def}{=} \sum\limits_{i=1}^{n} c_i \cdot \Delta x_{ri}$ and $\Delta y_s \stackrel{\rm def}{=} \sum\limits_{i=1}^{n} c_i \cdot \Delta x_{si}$.

We can therefore conclude that:

- the only thing we know about $\Delta y_s$ is that $|\Delta y_s| \le \Delta$, and
- the only thing we know about $\Delta y_r$ is that it is a random variable with 0 mean and standard deviation not exceeding $\sigma$,

where:

- $\Delta$ is obtained by applying the interval algorithm to the values $\Delta_i$, and

- $\sigma$ is obtained by applying the probabilistic algorithm to the values $\sigma_i$.

If some of the inputs are described by fuzzy uncertainty, the procedure of estimating $\Delta$ needs to be repeated for several different values $\alpha$ (e.g., for $\alpha = 0.1, 0.2, \ldots, 1.0$), so that $\Delta$ becomes a fuzzy number.

**Important comment.** Instead of treating systematic and random components separately (as we did), a seemingly reasonable idea is:

- to transform them into a single type of uncertainty, and then
- to combine the transformed uncertainties.

Such a transformation is, in principle, possible:

- If all we know is that the measurement error is located on the interval $[-\Delta_i, \Delta_i]$, but we have no reason to believe that some values on this intervals are more probable than others, then it is reasonable to assume that they are equally probable – i.e., to consider a uniform distribution on this interval; see, e.g., [4]. For the uniform distribution, the mean is 0, and the standard deviation is equal to $\frac{1}{\sqrt{3}} \cdot \Delta_i$.
- On the other hand, if we have a normally distributed random variable $\Delta x_i$ with mean 0 and standard deviation $\sigma_i$, then, with high certainty, we can conclude that this value is located within the $3\sigma$ interval $[-3\sigma_i, 3\sigma_i]$; see, e.g., [22].

The problem is that these seemingly reasonable transformations may drastically change the result of uncertainty quantification. Let us show this on the simplest example when $f(x_1, \ldots, x_n) = x_1 + \ldots + x_n$, so that $c_1 = \ldots = c_n = 1$, and all the initial values $\Delta_i$ and $\sigma_i$ are equal to 1.

In this case, for interval uncertainty we get $\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i = n$. However:

- if we transform it into the probabilistic uncertainty, with $\sigma_i = \frac{1}{\sqrt{3}} \cdot \Delta_i = \frac{1}{\sqrt{3}}$ and
- process this probabilistic information,

then we will get $\sigma^2 = \sum_{i=1}^{n} c_i^2 \cdot \sigma_i^2 = \frac{1}{3} \cdot n$, so $\sigma = \frac{1}{3} \cdot \sqrt{n}$. If we now form an interval bound based on this $\sigma$, we will get $\Delta = 3\sigma = \sqrt{3} \cdot \sqrt{n}$ – a value which is much smaller than $\Delta = n$. So, if we use this transformation, we will drastically underestimate the uncertainty – which, in many practical situations, can lead to a disaster.

Similarly, in the case of probabilistic uncertainty, we get $\sigma = \sqrt{\sum_{i=1}^{n} c_i^2 \cdot \sigma_i^2} = \sqrt{n}$. However:

- if we first transform it into interval uncertainty, with $\Delta_i = 3\sigma_i = 3$, and then
- apply interval estimate to this new uncertainty,

we will get $\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i = 3n$. If we transform this value back into standard deviations, we get $\sigma = \frac{1}{3} \cdot \Delta = \sqrt{3} \cdot n$ – a value which is much larger than $\sigma = \sqrt{n}$. So, if we use this transformation, we will drastically overestimate the uncertainty – and thus, fail to make conclusions about $y$ which could have made if we estimated the uncertainty correctly.

Bottom line: let 100 flowers bloom, do not try to reduce all uncertainties to a single one.

## Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), and HRD-1834620 and HRD-2034030 (CAHSI Includes), and by the AT&T Fellowship in Information Technology.

It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478.

## References

[1] R. Belohlavek, J. W. Dauben, and G. J. Klir, *Fuzzy Logic and Mathematics: A Historical Perspective*, Oxford University Press, New York, 2017.

[2] R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Addison Wesley, Boston, Massachusetts, 2005.

[3] L. Jaulin, M. Kiefer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control, and Robotics*, Springer, London, 2001.

[4] E. T. Jaynes and G. L. Bretthorst, *Probability Theory: The Logic of Science* (Cambridge University Press, Cambridge, UK, 2003).

[5] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[6] V. Kreinovich, "Application-Motivated Combinations of Fuzzy, Interval, and Probability Approaches, with Application to Geoinformatics, Bioinformatics, and Engineering", *Proceedings of the International Conference on Information Technology InTech'07*, Sydney, Australia, December 12–14, 2007, pp. 11–20.

[7] V. Kreinovich, "Application-motivated combinations of fuzzy, interval, and probability approaches, and their use in geoinformatics, bioinformatics, and engineering", *International Journal of Automation and Control (IJAAC)*, 2008, Vol. 2, No. 2/3, pp. 317–339.

[8] V. Kreinovich, "Decision making under interval uncertainty (and beyond)", In: P. Guo and W. Pedrycz (eds.), *Human-Centric Decision-Making Models for Social Sciences*, Springer Verlag, 2014, pp. 163–193.

[9] V. Kreinovich, "Interval computations and interval-related statistical techniques: estimating uncertainty of the results of data processing and indirect measurements", In: F. Pavese, W. Bremser, A. Chunovkina, N. Fisher, and A. B. Forbes (eds.), *Advanced Mathematical and Computational Tools in Metrology and Testing AMTCM'X*, World Scientific, Singapore, 2015, pp. 38–49.

[10] V. Kreinovich, "How to deal with uncertainties in computing: from probabilistic and interval uncertainty to combination of different approaches, with applications to engineering and bioinformatics", In: J. Mizera-Pietraszko, R. Rodriguez Jorge, D. M. Almazo Pérez, and P. Pichappan (eds.), *Advances in Digital technologies. Proceedings of the Eighth International Conference on the Applications of Digital Information and Web Technologies ICADIWT'2017*, Ciudad Juarez, Chihuahua, Mexico, March 29–31, 2017, IOS Press, Amsterdam, 2017, pp. 3–15.

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

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

[13] G. Mayer, *Interval Analysis and Automatic Result Verification*, de Gruyter, Berlin, 2017.

[14] J. M. Mendel, *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions*, Springer, Cham, Switzerland, 2017.

[15] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, SIAM, Philadelphia, 2009.

[16] H. T. Nguyen, V. Kreinovich, B. Wu, and G. Xiang, *Computing Statistics under Interval and Fuzzy Uncertainty*, Springer Verlag, Berlin, Heidelberg, 2012.

[17] H. T. Nguyen, C. L. Walker, and E. A. Walker, *A First Course in Fuzzy Logic*, Chapman and Hall/CRC, Boca Raton, Florida, 2019.

[18] V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical Principles of Fuzzy Logic*, Kluwer, Boston, Dordrecht, 1999.

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

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

[21] S. G. Rabinovich, *Measurement Errors and Uncertainties: Theory and Practice*, Springer, New York, 2005.

[22] D. J. Sheskin, *Handbook of Parametric and Non-Parametric Statistical Procedures*, Chapman & Hall/CRC, London, UK, 2011.

[23] K. S. Thorne and R. D. Blandford, *Modern Classical Physics: Optics, Fluids, Plasmas, Elasticity, Relativity, and Statistical Physics*, Princeton University Press, Princeton, New Jersey, 2017.

[24] 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, 2001, pp. 673–729.

[25] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 1965, Vol. 8, pp. 338–353.