When is propagation of interval and fuzzy uncertainty feasible?

Vladik Kreinovich,[1*] Andrew M. Pownuk,[2] Olga Kosheleva,[3] and Aleksandra Belina[4]

[1] *Department of Computer Science, University of Texas at El Paso, USA*
[2] *Department of Mathematical Sciences, University of Texas at El Paso, USA*
[3] *Department of Teacher Education, University of Texas at El Paso, USA*
[4] *Department of Building Structures, Silesian University of Technology, Poland*
[*] Corresponding author: vladik[at]utep.edu

Abstract

In many engineering problems, to estimate the desired quantity, we process measurement results and expert estimates. Uncertainty in inputs leads to the uncertainty in the result of data processing. In this paper, we show how the existing feasible methods for propagating the corresponding interval and fuzzy uncertainty can be extended to new cases of potential practical importance.

Keywords: Interval uncertainty, fuzzy uncertainty, uncertainty propagation, feasible algorithms

1 Introduction

Need for data processing. In many practical situations, we are interested in a quantity y which is difficult (or even impossible) to measure directly.

For example, when we design a structure – a building, an airplane, etc. – we would like to know the maximal force that can be applied without breaking this structure. For a building, directly measuring this force by trying to topple the building would be too expensive.

Another example is prediction: we want to predict the mechanical properties of an airplane several years from now, without having to wait these several years.

In all such cases, what helps is that we usually know the relation $y = f(x_1, \ldots, x_n)$ between the desired quantity y and some easier-to-measure and/or easier-to-estimate quantities $x_1, \ldots, x_n$. This dependence is sometimes given in terms of an explicit expression, but more often, it is given as an algorithm for computing y based on x_i – e.g., for mechanical or thermal properties, as the algorithm for solving the corresponding partial differential equations.

Then, after measuring and/or estimating x_i, we apply the algorithm f to the measurement/estimation results $\widetilde{x}_i$, and get an estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for y. Such an application of the algorithm f is an important case of *data processing.*

Need for propagating uncertainty. Neither measurement nor expert estimates are absolutely precise. As a result, the estimates $\widetilde{x}_i$ are somewhat different from the actual (unknown) values x_i. So, even if the algorithm f is precise, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of applying this algorithm to the estimates $\widetilde{x}_i$ is, in general, different from the value $y = f(x_1, \ldots, x_n)$ that we would have gotten if we knew the exact values of the input quantities.

It is therefore desirable to estimate the corresponding "propagated" uncertainty $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$; see, e.g., [21]. How big can this approximation error be?

This is very important in many practical situations. For example, if we are designing a chemical plant, and, based on our computations, we conclude that the predicted level of undesired chemicals in the air $\widetilde{y}$ will not exceed the required threshold y_0, this does not necessarily mean that we can start building this plant: it all depends on how accurate this estimate is. If $\widetilde{y}$ is slightly smaller than y_0, but the approximation error Δy can be large, this means that there is a real possibility that the plant will not be functioning safely.

Similarly, if, based on the measurement results, we predict that the building will withstand the earthquake of given magnitude, this does not necessarily mean that people in this building will be safe: it all depend on how accurate were our predictions.

In all such situation, it is important to estimate the approximation error Δ, i.e., to propagate the uncertainty of measuring/estimating x_i through the data processing algorithm f.

Need for interval uncertainty. The desired approximation error Δy comes from the measurement/estimation errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$. Thus, to estimate Δy, we need to have information about the error Δx_i.

Ideal case is when for each i, we know which values Δx_i are possible and what is the probability of getting each possible values – i.e., when we know the probability distribution for each Δx_i. In principle, we can get this probability distribution if we *calibrate* the corresponding measuring instrument – i.e., compare its results with the results of a much more accurate ("standard') measuring instrument. However, calibration is a very expensive procedure. As a result, in many engineering applications, we do not know these probabilities. At best, we know the upper bound Δ_i on the corresponding measurement error: $|\Delta x_i| \le \Delta_i$. In this case, based on the measurement result $\widetilde{x}_i$, all we can conclude about the actual (unknown) value x_i is that this value is somewhere on the interval $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. This situation is therefore known as a situation of *interval uncertainty*; see, e.g., [11, 16, 17].

Different values x_i from the corresponding intervals lead, in general, to different values y. It is therefore desirable to find the range of all possible values y, i.e., the interval

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

The problem of computing this range is known as the main problem of interval computations.

It is known that, in general, this problem is NP-hard; see, e.g., [14]. This means that, unless P=NP (which most computer scientists believe to be impossible), it is not possible to have a feasible algorithm that exactly computes the range $[\underline{y}, \overline{y}]$ for all possible functions $f(x_1, \ldots, x_n)$. It is therefore desirable to find cases when feasible algorithms are possible.

Need for fuzzy uncertainty. Often – e.g., for expert estimates – we do not even have a guaranteed upper bound. Instead, we have several bounds with different levels of certainty. As a result, instead of explicitly saying which values of Δx_i are possible and which are not – as in the case of interval uncertainty – we have, in effect, for each possible value Δx_i, a *degree* to which, according to the experts, this value is possible. In mathematical terms, this means that we have a function μ that maps every possible value of Δx_i into a degree $\mu(\Delta x_i)$ from the interval $[0, 1]$, so that:

- 1 corresponds to full confidence, and
- 0 corresponds to complete absence of confidence.

Such a function is known as a *fuzzy set*, and the corresponding uncertainty is known as *fuzzy uncertainty*; see, e.g., [1, 12, 19, 24].

What is known and what we do in this paper. Due to the fact that the problem of propagating interval and fuzzy uncertainty is of great practical importance, there exist many techniques for such a propagation. In this paper, we list some of the known techniques – and show how these known techniques can be extended to new cases of potential practical importance.

2 Linearized Case: Interval Uncertainty

Possibility of linearization. In many practical situations, the measurement errors $\Delta x_i = \widetilde{x}_i - x_i$ are small. In such situations, we can expand the expression

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

in Taylor series in Δx_i and ignore quadratic and higher order terms in this expansion. As a result, we get the following expression:

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

where

$$c_i \stackrel{\text{def}}{=} \frac{\partial f}{\partial x_i}_{|x_i = \widetilde{x}_i}.$$

Known result: in the linearized case, the main problem of interval computation is feasible to solve. In the linearized case (1), the dependence of Δy on Δx_i is linear. Each variable Δx_i takes all possible value from $-\Delta_i$ to Δ_i.

It is easy to see that:

- when $c_i > 0$, then Δy is an increasing function of Δx_i and thus, its largest possible value is attained when Δx_i takes its largest possible value $\Delta x_i = \Delta_i$;

- when $c_i < 0$, then Δy is a decreasing function of Δx_i and thus, its largest possible value is attained when Δx_i takes its smallest possible value $\Delta x_i = -\Delta_i$.

In both cases, the optimizing value of Δx_i is equal to $\text{sign}(c_i) \cdot \Delta_i$, where:

- $\text{sign}(c_i) = 1$ if $c_i > 0$, and
- $\text{sign}(c_i) = -1$ if $c_i < 0$.

For these optimizing values, each term $c_i \cdot \Delta x_i$ takes the form $c_i \cdot \text{sign}(c_i) \cdot \Delta_i$. One can easily see that $c_i \cdot \text{sign}(c_i) = |c_i|$, so the largest possible value of Δy is equal to

$$\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i. \tag{2}$$

Similarly:

- when $c_i > 0$, then Δy is an increasing function of Δx_i and thus, its smallest possible value is attained when Δx_i takes its smallest possible value $\Delta x_i = -\Delta_i$;
- when $c_i < 0$, then Δy is a decreasing function of Δx_i and thus, its smallest possible value is attained when Δx_i takes its largest possible value $\Delta x_i = \Delta_i$.

In both cases, the optimizing value of Δx_i is equal to $-\text{sign}(c_i) \cdot \Delta_i$. For this optimizing values, each term $c_i \cdot \Delta x_i$ takes the form $-|c_i| \cdot \Delta_i$, so the smallest possible value of Δy is equal to

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

The value Δy thus ranges from $-\Delta$ to Δ, so the range of possible values of $y = \widetilde{y} - \Delta y$ is the interval $[\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.

Computing Δ by using the expression (2) is definitely feasible: once we know c_i and Δ_i, this can be done by a simple linear-time algorithm.

Need to consider constraints. The above argument assumes that all possible combinations of values x_i are possible. In practice, we often have some dependence between these quantities.

In some cases, we have constraints like energy conservation, according to which some combination of the variables x_i is equal to a known value $g(x_1, \ldots, x_n) = g_0$. Substituting $x_i = \widetilde{x}_i - \Delta x_i$ into this formula, we get

$$g(\widetilde{x}_i - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n) = g_0.$$

Since we assume that the measurement errors Δx_i are small – and thus, that their squares can be safely ignored – we can linearize this constraint and get a linear constraint of the type

$$\sum_{i=1} g_i \cdot \Delta x_i = G_0, \tag{3}$$

where

$$g_i \stackrel{\text{def}}{=} \frac{\partial g}{\partial x_i}_{x_i = \widetilde{x}_i} \text{ and } G_0 = g_0 - g(\widetilde{x}_1, \ldots, \widetilde{x}_n).$$

In other cases, we have inequality constraints. For example, the second law of thermodynamics states that the entropy at the next moment of time must be smaller than or equal to the entropy at the previous moment of time. In such cases, we have constraints of the type $h(x_1, \ldots, x_n) \geq h_0$, which after linearization turn into linear inequalities

$$\sum_{i=1}^{n} h_i \cdot \Delta x_i \geq H_0, \tag{4}$$

where

$$h_i \stackrel{\text{def}}{=} \frac{\partial h}{\partial x_i}_{x_i = \widetilde{x}_i} \text{ and } H_0 = h_0 - h(\widetilde{x}_1, \ldots, \widetilde{x}_n).$$

In such situations, instead of finding the range of all possible values of $f(x_1, \ldots, x_n)$, we are interested in the range over all possible tuples $(x_1, \ldots, x_n)$ that satisfy the given constraints.

It turns out that in this case, we still have a feasible algorithm – although not as fast as in the no-constraints case.

First new (simple) result: in the linearized case, computing the range under constraints is also feasible. Indeed, in this case, we need to find the largest and the smallest possible values of a linear expression

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

under linear equalities and inequalities of the type (3) and (4).

The corresponding optimization problems are particular case of the general *linear programming* problem: optimize a linear function under linear constraints. Feasible algorithms are known for solving this problem; see, e.g., [15].

Comment. It should be noted, however, that while these algorithms are feasible, they are not as fast as the linear-time algorithm for computing the expression (2) – and there seems to be no hope of speeding them up by using, e.g., parallelization, since linear programming is known to be provably the most difficult to parallelize; see, e.g., [20].

Second new result: in the linearized case, computing the range under sparsity constraint is also feasible. Often, a natural constraint on the values is that the tuple $(x_1, \ldots, x_n)$ should be *sparse*, i.e., that no more than a certain number (K) of values x_i are different from 0; see, e.g., [2–6, 10].

We are interested in finding the range for $y = \widetilde{y} - \Delta y$. In the linearized case, Δy is described by the formula (1), where $\Delta x_i = \widetilde{x}_i - x_i$. Substituting the definition of Δx_i into the formula (1) and substituting the resulting expression for Δy into the formula for y, we get

$$y = \widetilde{y} - \sum_{i=1}^{n} c_i \cdot (\widetilde{x}_i - x_i),$$

i.e.,

$$y = Y + \sum_{i=1}^{n} c_i \cdot x_i,$$

where we denoted

$$Y \stackrel{\text{def}}{=} \widetilde{y} - \sum_{i=1}^{n} c_i \cdot \widetilde{x}_i.$$

The values x_i for which $0 \notin [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ clearly cannot be equal to 0, so when looking for 0 values of x_i we should look among values for which $0 \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$, i.e., for which $\widetilde{x}_i - \Delta_i \le 0 \le \widetilde{x}_i + \Delta_i$. No more than K of these values are different from 0.

For each i for which $c_i > 0$:

- the largest value of the corresponding term $c_i \cdot x_i$ is attained when x_i attains its largest possible value $x_i = \widetilde{x}_i + \Delta_i$, and
- the smallest value of the corresponding term $c_i \cdot x_i$ is attained when x_i attains its smallest possible value $x_i = \widetilde{x}_i - \Delta_i$.

For each i for which $c_i < 0$:

- the largest value of the corresponding term $c_i \cdot x_i$ is attained when x_i attains its smallest possible value $x_i = \widetilde{x}_i - \Delta_i$, and
- the smallest value of the corresponding term $c_i \cdot x_i$ is attained when x_i attains its largest possible value $x_i = \widetilde{x}_i + \Delta_i$.

In both cases:

- the largest value $\overline{y}_i$ of the corresponding term $c_i \cdot x_i$ is attained when $x_i = \widetilde{x}_i + \text{sign}(c_i) \cdot \Delta_i$, so that this largest value is equal to

$$\overline{y}_i = c_i \cdot x_i = c_i \cdot \widetilde{x}_i + c_i \cdot \text{sign}(c_i) \cdot \Delta_i = c_i \cdot \widetilde{x}_i + |c_i| \cdot \Delta_i;$$

and

- the smallest value $\underline{y}_i$ of the corresponding term $c_i \cdot x_i$ is attained when $x_i = \widetilde{x}_i - \text{sign}(c_i) \cdot \Delta_i$, so that this smallest value is equal to

$$\underline{y}_i = c_i \cdot x_i = c_i \cdot \widetilde{x}_i - c_i \cdot \text{sign}(c_i) \cdot \Delta_i = c_i \cdot \widetilde{x}_i - |c_i| \cdot \Delta_i.$$

For terms i for which $0 \notin [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$, these are the terms we take to find the largest $\overline{y}$ and the smallest $\underline{y}$ values of y.

For terms i which could be 0:

- to find $\overline{y}$, we take the K largest of these terms, and
- to find $\underline{y}$, we take K smallest of these terms.

Thus, we arrive at the following algorithm for computing $\underline{y}$ and $\overline{y}$. In this algorithm, we separate indices i for which $i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ as *possible zeros* and all other indices as *definitely non-zeros*. The algorithm is as follows:

- first, for all indices i, we compute $\overline{y}_i = c_i \cdot \widetilde{x}_i + |c_i| \cdot \Delta_i$ and $\underline{y}_i = c_i \cdot \widetilde{x}_i - |c_i| \cdot \Delta_i$;
- to compute $\overline{y}$, to Y, we add all the values $\overline{y}_i$ for all definitely non-zero i and K largest of the terms $\overline{y}_i$ corresponding to possible zeros;
- to compute $\underline{y}$, to Y, we add all the values $\underline{y}_i$ for all definitely non-zero i and K smallest of the terms $\underline{y}_i$ corresponding to possible zeros,

This algorithm is clearly feasible: it takes linear time + time $O(n \cdot \log(n))$ for sorting; see, e.g., [8].

3 Linearized Case: Fuzzy Uncertainty

Towards a precise formulation of the problem. In fuzzy approach, instead of simply indicating which values are possible and which are not, we have experts' degrees indicating to what extend each value is possible.

Once we know the corresponding degrees $\mu_i(\Delta x_i)$ for different possible values of the measurement errors Δx_i, we need to combine them into a degree $\mu(\Delta y)$ indicating to what extent each value Δy is possible.

A value Δy is possible if there exist values Δx_1, ..., Δx_n all of which are possible and for which the formula (1) holds. In other words, Δy is possible if and only if there exist $\Delta x_1, \ldots, \Delta x_n$ for which (1) is true and for which Δx_1 is possible *and* ... *and* Δx_n is possible.

We know the degree $\mu_i(\Delta x_i)$ to which each value Δx_i is possible. We get these degrees from the expert(s). Ideally, to find the degree to which Δx_1 is possible and Δx_2 is possible, etc., we should also ask experts – but there are so many combinations $(\Delta x_1, \ldots, \Delta x_n)$ that asking about all of them is not realistic.

Since we cannot elicit these degree directly, we need to estimate them based on what we know – i.e., based on the degrees to which each Δx_i is possible.

This is a typical situation in fuzzy reasoning:

- we know the degree a and b to which statements A and B are true, and
- we want to estimate to what extent their "and"-combination $A \,\&\, B$ is true.

Let $f_{\&}(a, b)$ denote this estimate; the operation $f_{\&}(a, b)$ is known as an *"and"-operation*, or, alternatively, a *t-norm*.

There are several reasonable properties that such an "and"-operation should satisfy, For example, since $A \,\&\, B$ and $B \,\&\, A$ means the same, we should have $f_{\&}(a, b) = f_{\&}(b, a)$, i.e., the "and"-operation should be *commutative*. Similarly, since $A \,\&\, (B \,\&\, C)$ means the same as $(A \,\&\, B) \,\&\, C$, the "and"-operation must be associative:

$$f_{\&}(a, f_{\&}(b, c)) = f_{\&}(f_{\&}(a, b), c).$$

All the operations with these properties are known. The simplest such operation is $f_{\&}(a, b) = \min(a, b)$. This operation is widely used in fuzzy applications.

Other examples are operations of the type

$$f_{\&}(a, b) = g^{-1}(g(a) + g(b)), \tag{5}$$

where $g(x)$ is a decreasing function from $[0,1]$ to non-negative numbers, and $g^{-1}(x)$ is its inverse function. For example, for $g(x) = -\ln(x)$, we get $f_{\&}(a,b) = a \cdot b$. This operation is also widely used in applications.

In general, operations of type (5) are *universal approximators*, in the sense that for every "and"-operation $f_{\&}(a,b)$ and for every $\varepsilon > 0$, there exists an operation of type (5) which is ε-close to $f_{\&}(a,b)$ for all a and b; see, e.g., [18]. Thus, from the practical point, we can safely assume that the actually used "and"-operation is of type (5).

So, our estimate of the expert's degree of confidence that Δx_1 is possible *and* Δx_2 is possible etc. is equal to

$$f_{\&}(\mu_1(\Delta x_1), \mu_2(\Delta x_2), \ldots).$$

We have this degree for each tuple $(\Delta x_1, \ldots, \Delta x_n)$. We need to combine the degrees corresponding to different tuples that satisfy the property (1). The phrase "there exists" is, in essence, an "or": it means that either this property holds for one of the tuples, or it holds for another tuple, etc. Similarly to "and"-operations, in fuzzy logic, we also have "or"-operations $f_{\vee}(a,b)$ (also known as *t-conorms*). The simplest "or"-operation is $f_{\vee}(a,b) = \max(a,b)$ which is widely used. Other operations include operations of the type (5) with *increasing* functions $g(x)$. The mostly widely used example is $f_{\vee}(a,b) = a + b - a \cdot b$ which corresponds to $g(a) = -\ln(1-a)$.

At first glance, it may seem that, similar to the fact that we can use generic "and"-operations, we can also use general "or"-operations. However, there is a big difference here:

- we use "and"-operations to combine a *finite* number of degrees, while '
- the "or"-operation is used to combine degrees corresponding to *infinitely many* possible tuples.

For infinitely many degrees, "or"-operations of type (5) usually lead to the meaningless value 1, so the only meaningful choice is to use the max-operation $f_{\vee}(a,b) = \max(a,b)$.

Thus, we arrive at the following formula for the desired membership function $\mu(\Delta y)$:

$$\mu(\Delta y) = \max\left\{ f_{\&}(\mu_1(\Delta x_1), \ldots, \mu_n(\Delta x_n) : \sum_{i=1}^{n} c_i \cdot \Delta x_i = \Delta y. \right\}. \tag{6}$$

Known result: for min "and"-operation, the problem is feasible to solve. In the simplest case, when $f_{\&}(a,b) = \min(a,b)$, the formula (6) takes the form

$$\mu(\Delta y) = \max\left\{ \min(\mu_1(\Delta x_1), \ldots, \mu_n(\Delta x_n) : \sum_{i=1}^{n} c_i \cdot \Delta x_i = \Delta y. \right\}. \tag{7}$$

This formula was first derived by Zadeh himself and is thus known as *Zadeh's extension principle*.

It is known that to compute $\mu(\Delta y)$, it is convenient to use α-*cuts*, i.e., sets of the type $\mathbf{x}_i(\alpha) \stackrel{\text{def}}{=} \{\Delta x_i : \mu_i(\Delta x_i) \ge \alpha\}$ and $\mathbf{y}(\alpha) \stackrel{\text{def}}{=} \{\Delta y : \mu(\Delta y) \ge \alpha\}$.

Expert estimates for degrees $\mu_i(\Delta x_i)$ usually decreases as we go further away from 0. In this case, the α-cuts are simply intervals

$$\mathbf{x}_i(\alpha) = [\underline{x}_i(\alpha), \overline{x}_i(\alpha)]$$

for appropriate endpoints $\underline{x}_i(\alpha)$ and $\overline{x}_i(\alpha)$.

By definition of $\mu(\Delta y)$, we have $\mu(\Delta y) \ge \alpha$ if and only there exist values $\Delta x_1, \ldots, \Delta x_n$ for which (1) holds and for which $\min(\mu_1(\Delta x_1), \ldots, \mu_n(\Delta x_n)) \ge \alpha$, i.e., equivalently, for which $\mu_1(\Delta x_1) \ge \alpha$ *and* $\mu_2(\Delta x_2) \ge \alpha$, etc. In other words, $\Delta y \in \mathbf{y}(\alpha)$ if and only if there exist $\Delta x_i \in \mathbf{x}_i(\alpha)$ for which $\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i$. Thus, for each α, the interval $\mathbf{y}(\alpha)$ is simply the range of the linear expression (1) when each Δx_i is in the interval $[\underline{x}_i(\alpha), \overline{x}_i(\alpha)]$.

To compute this range, we can represent each of these intervals as $[\widetilde{\Delta x}_i - \delta_i, \widetilde{\Delta x}_i + \delta_i]$, where

$$\widetilde{\Delta x}_i \stackrel{\text{def}}{=} \frac{\underline{x}_i(\alpha) + \overline{x}_i(\alpha)}{2}$$

and

$$\delta_i \stackrel{\text{def}}{=} \frac{\overline{x}_i(\alpha) - \underline{x}_i(\alpha)}{2}.$$

Then, the desired range is equal to

$$[\hat{y} - \delta, \hat{y} + \delta],$$

where

$$\hat{y} \stackrel{\text{def}}{=} \sum_{i=1}^{n} c_i \cdot \widetilde{\Delta x}_i$$

and

$$\delta \stackrel{\text{def}}{=} \sum_{i=1}^{n} |c_i| \cdot \delta_i.$$

All these explicit formula are clearly feasible.

Third new result: the problem is feasible for any "and"-operation. Let us show that the problem remains feasible if instead of min, we use any "and"-operation of type (3).

For such an "and"-operation, the formula (6) takes the form

$$\mu(\Delta y) = \max\left\{ g^{-1}(g(\mu_1(\Delta x_1)) + \ldots + g(\mu_n(\Delta x_n))) : \sum_{i=1}^{n} c_i \cdot \Delta x_i = \Delta y \right\}.$$

Since the function $g(x)$ is decreasing, we have $\mu(\Delta y) = g^{-1}(z)$, where

$$z \stackrel{\text{def}}{=} \min\left\{ g(\mu_1(\Delta x_1)) + \ldots + g(\mu_n(\Delta x_n)) : \sum_{i=1}^{n} c_i \cdot \Delta x_i = \Delta y \right\}.$$

Thus, if we can feasibly compute z, we can also feasible compute $\mu(\Delta y)$ as $g^{-1}(z)$.

To simplify our expression for z, let us denote

$$f_i(\Delta x_i) \stackrel{\text{def}}{=} g(\mu_i(\Delta x_i)).$$

In these terms, the above formula for z takes the form

$$z = \min\left\{ f_1(\Delta x_1) + \ldots + f_n(\Delta x_n) : \sum_{i=1}^{n} c_i \cdot \Delta x_i = \Delta y \right\}.$$

The problem of finding z is now a classical constraint optimization problem: minimize the sum

$$f_1(\Delta x_1) + \ldots + g_n(\Delta x_n)$$

under the constraint

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

To solve this problem, we can use the usual Lagrange multiplier method to reduce it to the following unconstrained optimization problem: minimize the expression

$$\sum_{i=1}^{n} f_i(\Delta x_i) + \lambda \cdot \left(\sum_{i=1}^{n} c_i \cdot \Delta x_i - \Delta y \right),$$

for an appropriate Lagrange multiplier λ. Differentiating the above objective function with respect to Δx_i and equating the derivative to 0, we get

$$f_i'(\Delta x_i) + \lambda \cdot c_i = 0,$$

i.e., that

$$\Delta x_i = f_i^{-1}(\lambda \cdot c_i).$$

Thus, if we know λ, we can feasibly compute all the values Δx_i.

The only remaining problem is to find λ. The value λ must be found from the condition (1). Thus, we can use, e.g., bisection to find the appropriate value λ. Hence, the whole computation is feasible.

4 Beyond Linearized Case

For boxes, interval computation is NP-hard already for quadratic functions. For linear functions $f(x_1, \ldots, x_n)$, as we have mentioned, the problem of computing the interval range over a box

$$[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$$

is feasible. However, already for quadratic functions $f(x_1, \ldots, x_n)$, this problem is NP-hard; see, e.g., [14].

Beyond boxes. But why concentrate on boxes? Boxes correspond to the case when we have no constraints. In practice, as we have mentioned, we often have constraints, so we have a subset of the box.

Known result: for ellipsoids, the problem is feasible for quadratic functions $f(x_1, \ldots, x_n)$. One of the typical sets of possible values is an ellipsoid, i.e., set of the type $Q(x_1, \ldots, x_n) \le q_0$ for some quadratic function $Q(x_1, \ldots, x_n)$; see, e.g., [7, 13, 23] and references therein.

We want to find the range of a quadratic function $f(x_1, \ldots, x_n)$ over an ellipsoid. In other words, we want to find the maximum $\overline{y}$ and the minimum $\underline{y}$ of this function over an ellipsoid. The maximum or minimum occurs:

- either inside the ellipsoid
- or on its boundary $Q(x_1, \ldots, x_n) = q_0$.

If the maximum or minimum is attained inside, then all n partial derivatives of the quadratic function $f(x_1, \ldots, x_n)$ must be equal to 0. Derivatives of a quadratic function are linear, so we get an easy-to-solve system of n linear equations with n unknowns, for which we can easily find the corresponding tuple $(x_1, \ldots, x_n)$ – and check that this tuple is indeed inside the given ellipsoid.

If the maximum or minimum is attained on the border, then the location of this maximum or minimum can be found by solving the following constraint optimization problem: optimize the function $f(x_1, \ldots, x_n)$ under the constraint $Q(x_1, \ldots, x_n) = q_0$.

To solve this problem, we can also use the Lagrange multiplier method and solve the corresponding unconstrained optimization problem of optimizing the expression

$$f(x_1, \ldots, x_n) + \lambda \cdot (Q(x_1, \ldots, x_n) - q_0).$$

For each λ, by differentiating this expression with respect to x_i and equating each of these derivatives to 0, we get an easy-to-solve system of n linear equations with n unknowns. Thus we get the values $x_i(\lambda)$.

The only remaining problem is how to find λ from the condition that $Q(x_1(\lambda), \ldots, x_n(\lambda)) = q_0$. But this is an equation with one unknown, and such equations are feasibly solvable. (The problem becomes computationally complex when the number of variables grows.)

Fourth new result: for quadratic function, the problem is feasible also for intersection of ellipsoids. Ellipsoids are sets of a very specific shape. Can we extend the above results to generic shapes?

Of course, since the problem is NP-hard for boxes, we do not expect a feasible algorithm for all the sets, but what we would like to have is a sequence of families of sets, with more and more parameters, for each of which we have a feasible algorithm – although the complexity of such algorithm may increase as we add more and more parameters to the family.

It turns out that as such a sequence of families, we can take *intersections of ellipsoids*:

- the first family is the family of ellipsoids,
- the next family is intersections of 2 ellipsoids,
- then intersections of 3 ellipsoids, etc.

Let us show that this way, we indeed get a universal approximation family, i.e., that any convex set can be thus approximated. Indeed, each convex set can be approximated, with any given accuracy, by a polyhedron: e.g., by the convex hull of sufficiently points on its surface; see, e.g. [22]. A convex polyhedron is an intersection of half-spaces that contain it, and a half-space can be approximated by an ellipsoid – since in an appropriate limit, an ellipsoid tends to a half-space.

For the intersection of K ellipsoids $Q_k(x_1, \ldots, x_n) \le q_k$, the corresponding Lagrange multiplier problem takes the form

$$f(x_1, \ldots, x_n) + \sum_{k=1}^{K} \lambda_k \cdot (Q_k(x_1, \ldots, x_n) - q_k) = 0.$$

We also need to consider 2^K cases when the optimizing tuple $(x_1, \ldots, x_n)$ is inside some of the ellipsoids. In all these cases, we get a system of linear equations enabling us to find all the values x_i as functions of $\lambda_1, \ldots, \lambda_K$: $x_i = x_i(\lambda_1, \ldots, \lambda_K)$. The values λ_k must then be determined from the conditions that

$$Q_k(x_1(\lambda_1, \ldots, \lambda_K), \ldots, x_n(\lambda_1, \ldots, \lambda_K)) = q_k$$

for all k from 1 to K. We thus have a system of K nonlinear equations with K unknowns. When K is fixed, this system is feasible – although the complexity of solving this system grows exponentially with K.

Acknowledgments

This work was supported in part by the US National Science Foundation grant HRD-1242122 (Cyber-ShARE Center of Excellence).

The authors are thankful to the anonymous referees for valuable suggestions.

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] E. J. Candès, J. Romberg and T. Tao, Stable signal recovery from incomplete and inaccurate measurements, *Communications in Pure and Applied Mathematics*, **59**, 1207–1223 (2006).

[3] E. Candès, J. Romberg, and T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, *IEEE Transactions on Information Theory*, **52**(2), 489–509 (2006).

[4] E. J. Candès and T. Tao, Decoding by linear programming, *IEEE Transactions on Information Theory*, **51**(12), 4203–4215 (2005).

[5] E. J. Candès and M. B. Wakin, An introduction to compressive sampling, *IEEE Signal Processing Magazine*, **25**(2), 21–30 (2008).

[6] F. Cervantes, B. Usevitch, L. Valera, and V. Kreinovich, Why sparse? fuzzy techniques explain empirical efficiency of sparsity-based data- and image-processing algorithms, *Proceedings of the 2016 World Conference on Soft Computing* (Berkeley, California, May 22–25, 2016), 165–169.

[7] F. L. Chernousko, *State Estimation for Dynamic Systems* (CRC Press, Boca Raton, Florida, 1994).

[8] Th. H. Cormen, C. E. Leiserson, R. L., Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, Cambridge, Massachusetts, 2009).

[9] T. Dumrongpokaphan, O. Kosheleva, V. Kreinovich, and A. Belina, Why sparse?, In: O. Kosheleva, S. Shary, G. Xiang, and R. Zapatrin (eds.), *Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy, etc. Methods and Their Applications* (Springer, Cham, Switzerland, 2018) to appear.

[10] M. Elad, *Sparse and Redundant Representations* (Springer Verlag, 2010).

[11] 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).

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

[13] O. Kosheleva and V. Kreinovich, For describing uncertainty, ellipsoids are better than generic polyhedra and probably better than boxes: a remark, *Mathematical Structures and Modeling*, **27**, 38–41 (2013).

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

[15] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming* (Springer, Cham, Switzerland, 2016).

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

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

[18] H. T. Nguyen, V. Kreinovich, and P. Wojciechowski, Strict Archimedean t-norms and t-conorms as universal approximator, *International Journal of Approximate Reasoning*, **18**(3–4), 239–249 (1998).

[19] H. T. Nguyen and E. A. Walker, *A First Course in Fuzzy Logic* (Chapman and Hall/CRC, Boca Raton, Florida, 2006).
[20] C. H. Papadimitriou, *Computational Complexity* (Pearson, Boston, Massachusetts, 1993).
[21] S. G. Rabinovich, *Measurement Errors and Uncertainty: Theory and Practice* (Springer Verlag, Berlin, 2005).
[22] R. T. Rockafeller, *Convex Analysis* (Princeton University Press, Princeton, New Jersey, 1970).
[23] K. Villaverde, O. Kosheleva, and M. Ceberio, Why ellipsoid constraints, ellipsoid clusters, and Riemannian space-time: Dvoretzky's theorem revisited, In: M. Ceberio and V. Kreinovich (eds.), *Constraint Programming and Decision Making* (Springer Verlag, Berlin, Heidelberg, 2014) 203–207.
[24] L. A. Zadeh, Fuzzy sets, *Information and Control*, **8**, 338–353 (1965).