Interval Computations as an Important Part of Granular Computing: An Introduction

Vladik Kreinovich

Abstract

This chapter provides a general introduction to interval computations, especially to interval computations as an important part of granular computing. This introduction is aimed at researchers who would like to learn more – and eventually to use – the main ideas and techniques of granular computing.

We explain how intervals naturally appear in data processing, which techniques exist for processing intervals, and how these techniques have been historically developed.

1 Brief Outline

The main goal of this chapter is to introduce interval computations to people who are interested in using the corresponding techniques. In view of this goal, we will not only *describe* these techniques, we will also do our best to outline the *problems* for which these techniques have been originally invented.

We start with explaining why computations in general are needed in practice. Then, we describe the uncertainty related to all these practical applications and, in particular, interval uncertainty. This will bring us to the main problem of interval computations.

In the following sections, we will briefly describe the history of interval computations, main interval techniques, and we list a few typical applications of these techniques.

2 Why Computations Are Needed in Practical Problems: A Brief Reminder

In accordance with the above outline, before we explain the specific role of *interval* computations, we will recall where and why *computations* in general are needed.

Let us recall what practical problems we need to solve in the first place. To understand why computations are needed in practice, let us recall

what practical problems we need to solve. Crudely speaking, most of the practical problems can be classified into three classes:

- we want to *learn* what is happening in the world; in particular, we want to know the numerical values of different quantities (distances, masses, charges, coordinates, etc.);
- based on these values, we would like to *predict* how the state of the world will change over time;
- finally, we would like to find out what *changes* we need to make in the world so that these changes will lead to the desired results.

It should be emphasized that this classification is very crude: a real-life problem often involves solving subproblems of all three above-described type.

The above classification is related to the distinction between science and engineering. The above classification may sound unusual, but in reality, it is related to the well-known classification of creative activity into engineering and science:

- The tasks of learning the current state of the world and predicting the future state of the world are usually classified as *science*.
- The tasks of finding the appropriate change are usually classified as *engineering*.

Example.

- Measuring the river flow at different locations and predicting how this river flow will change over time are problems of science.
- Finding the best way to change this flow (e.g., by building dams or levees) is a problem of engineering.

Computations are needed for all three classes of problems. In the following text, we will analyze the problems of these three types one by one. We will see that in all three cases, a large amount of computation is needed.

How we learn the current state of the world: sometimes, it is (relatively) straightforward. Let us start with the first class of practical problems: the problem of learning the state of the world. As we have mentioned, this means, in particular, that we want to know the numerical values of different quantities y that characterize this state.

Some quantities y we can simply directly measure. For example, when we want to know the current state of a patient in a hospital, we can measure the patient's body temperature, blood pressure, weight, and many other important characteristics.

In some situations, we do not even need to measure: we can simply ask an expert, and the expert will provide us with an (approximate) value $\widetilde{y}$ of the quantity y.

How we learn the current state of the world: sometimes, it is not easy. Some quantities we can simply directly measure. However, many other quantities of interest are difficult or even important to measure or estimate directly.

Examples. Examples of such quantities include the amount of oil in a given well or a distance to a star. Let us explain this situation on the example of measuring distances.

- We can estimate a distance between two nearby houses by simply placing a measuring tape between them.
- If we are interested in measuring the distance between the two cities, in principle, it is possible to do it directly, by driving or walking from one to another. (It is worth mentioning that while such a direct measurement is possible *in principle*, it is *not* a reasonable practical way.)
- If we are interested in measuring a distance to a star, then, at present, it is not possible to directly measure this distance.

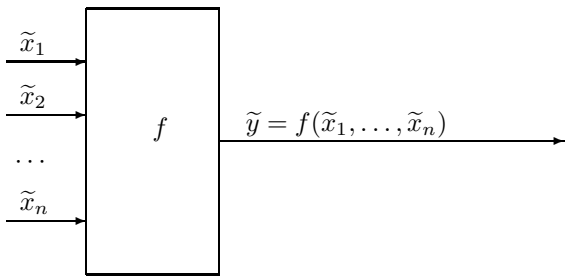
How we can measure difficult-to-measure quantities. Since we cannot directly measure the values of these quantities, the only way to learn some information about them is:

- to measure (or ask an expert to estimate) some other easier-to-measured quantities $x_1, \ldots, x_n$, and then
- to estimate y based on the measured values $\widetilde{x}_i$ of these auxiliary quantities x_i.

Examples.

- To estimate the amount of oil in a given well, we perform *seismic* experiments: we set up small explosions at some locations and measure the resulting seismic waves at different distances from the location of the explosion.
- To find the distance to a faraway star, we measure the direction to the star from different location on Earth (and/or at different seasons) and the coordinates of (and the distances between) the locations of the corresponding telescopes.

To learn the current value of the desired quantity, we often need a lot of computations. To estimate the value of the desired quantity y, we must know the relation between y and the easier-to-measure (or easier-to-estimate) quantities $x_1, \ldots, x_n$. Specifically, we want to use the estimates of x_i to come up with an estimate for y. Thus, the relation between y and x_i must be given in the form of an *algorithm* $f(x_1, \ldots, x_n)$ which transforms the values of x_i into an estimate for y. Once we know this algorithm f and the measured values $\widetilde{x}_i$ of the auxiliary quantities, we can estimate y as $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.



In different practical situations, we have algorithms f of different complexity. For example, to find the distance to star, we can usually have an explicit analytical formula coming from geometry. In this case, f is a simple formula.

On the other hand, to find the amount of oil, we must numerically solve a complex partial differential equation. In this case, f is a complex iterative algorithm of solving this equation.

There are many such practical cases when the algorithms f requires a lot of computations. Thus, the need to learn the current state of the world indeed often leads to the need to perform a large number of computations.

Comment: the notion of indirect measurement. We started with the situation in which we cannot estimate the value of the desired quantity y by simply directly measuring (or directly estimating) this value. In such situations, we can use the above two-stage process, as a result of which we get an indirect estimate for y.

In the case when the values x_i are obtained by measurement, this two-stage process does involve measurement. To distinguish it from *direct* measurements (i.e., measurements which directly measure the values of the desired quantity), the above two-stage process is called an *indirect* measurement.

Computations are needed to predict the future state of the world. Once we know the values of the quantities $y_1, \ldots, y_m$ which characterize the current state of the world, we can start predicting the future state of the world, i.e., the future values of these quantities.

To be able to predict the future value z of each of these quantities, we must know how exactly this value z depends on the current values $y_1, \ldots, y_m$. Specifically, we want to use the known estimates $\widetilde{y}_i$ for y_i to come up with an

estimate for z. Thus, the relation between z and y_i must be given in the form of an *algorithm* $g(y_1, \ldots, y_m)$ which transforms the values of y_i into an estimate for z. Once we know this algorithm g and the estimates $\widetilde{y}_i$ for the current values of the quantities, we can estimate z as $\widetilde{z} = g(\widetilde{y}_1, \ldots, \widetilde{y}_n)$.

Again, the corresponding algorithm g can be very complicated and time-consuming. So, we often need a large number of computations to make the desired predictions.

This is, e.g., how weather is predicted now: weather prediction requires so many computations that it can only be performed on fast supercomputers.

The general notion of data processing. So far, we have analyzed two different classes of practical problems:

- the problem of *learning* the current state of the world (i.e., the problem of indirect measurement), and
- the problem of *predicting* the future state of the world.

From the *practical* viewpoint, these two problems are drastically different. However, as we have seen, from the *computational* viewpoint, these two problems are very similar. In both problems,

- we start with the estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$ for the quantities $x_1, \ldots, x_n$, and then
- we apply the known algorithm f to these estimates, resulting in an estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for the desired quantity y.

In both cases, this algorithm can be very time-consuming. The corresponding (often time-consuming) computational part of each of these two classes of problems – applying a known algorithm to the known values – is called *data processing*.

Comment. Since the computational parts of these two classes of problems are similar, it is important to describe the difference between these two classes of problems. As we can see from the above descriptions, the only *difference* between the two classes is where the original inputs $\widetilde{x}_i$ come from:

- in the problem of learning the current state of the world, the inputs $\widetilde{x}_i$ come from *direct* measurements (or direct expert estimation);
- in contrast, in the problem of predicting the future state of the world, the inputs $\widetilde{y}_i$ come from the learning stage – e.g., they may come from *indirect* measurements.

Decision making, design, control. Once we know the current state of the world and we know how to predict the consequences of different decisions (designs, etc.), it is desirable to find the decision (design, etc.) which guarantees the given results.

Depending on what we want from this design, we can subdivide all the problems from this class into two subclasses.

In both subclasses, the design must satisfy some constraints. Thus, we are interested in finding a design that satisfies all these constraints.

- In some practical situations, satisfaction of all these constraints is all we want. In general, there may be several possible designs which satisfy given constraints. In the problems from the first subclass, we do not have any preferences for one of these designs – any one of them will suffice. Such problems are called the problems of *constraint satisfaction*.

- In other practical situations, we do have a clear preference between different designs x. This preference is usually described in terms of an *objective function* $F(x)$ – a function for which more preferable designs x correspond to larger values of $F(x)$. In such situation, among all the designs which satisfy given constraints, we would like to find a design x for which the value $F(x)$ of the given objective function is the largest. Such problems are called *optimization problems*.

Both constraint satisfaction and optimization often requires a large number of computations; see, e.g., [28].

Comment. Our main objective is to describe interval computations. They were originally invented for the first two classes of problems – i.e., for data processing, but they turned out to be very useful for the third class (constraint satisfaction and optimization) as well.

3 In Real-Life Computations, We Need to Take Uncertainty Into Account

Need for computations: reminder. In the previous section, we described the importance of computations. In particular, computations constituting data processing process the values which come from measurements (direct or indirect) and from expert estimations.

Let us start with the problem of learning the values of the physical quantities. Let us start with the problems from the first class, the problems of learning the values of the physical quantities. In these problems, computations are needed to transform the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of direct measurements (or direct expert estimations) into the estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of the desired quantity y.

Both in the case of measurements and in the case of expert estimates, the estimates $\widetilde{x}_i$ are only approximately equal to the (unknown) actual values x_i of the corresponding quantities. Let us elaborate on this statement.

Measurements are never exact.

- From the *philosophical* viewpoint, measurements cannot be exact because
  - the actual value of the quantity is a general real number; so, in general, we need *infinitely* many bits to describe the exact value, while
  - after every measurement, we only gain a *finite* number of bits of information (e.g., a finite number of binary digits in the binary expansion of the number).
- From the *physical* viewpoint, there is always some difficult-to-delete noise which is mixed with the measurement results.

Expert estimates are never absolutely exact either.

- First of all, as with the measurements, expert estimates cannot be absolutely exact, because an expert only generates a finite amount of information.
- Second, from the common sense viewpoint, experts are usually even less accurate than (sometimes super-precise) measuring instruments.

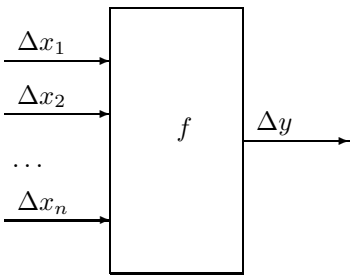
In both cases, there is usually a non-zero approximation error. The difference $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$ between the (approximate) estimate $\widetilde{x}_i$ and the (unknown) actual value x_i of the quantity x_i is called the *approximation error*.

In particular, if $\widetilde{x}_i$ is obtained by measurement, this difference is called the *measurement error*.

Uncertainty in inputs leads to uncertainty in the result of data processing. We assumed that the quantities $x_1, \ldots, x_n$ that we directly measure or directly estimate are related to the desired quantity y by a known relation $y = f(x_1, \ldots, x_n)$.

Because of this relation, we estimate the value y as $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$. Since the values $\widetilde{x}_i$ are, in general, different from the (unknown) actual values x_i, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of applying the algorithm f to the estimates $\widetilde{x}_i$ is, in general, different from the result $y = f(x_1, \ldots, x_n)$ of applying this algorithm to the actual values x_i. Thus, the estimate $\widetilde{y}$ is, in general, different from the actual value y of the desired quantity: $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y \neq 0$.

It is therefore desirable to find out the uncertainty Δy caused by the uncertainties Δx_i in the inputs:



Comment. In the above argument, we assumed that the relation f provides the *exact* relation between the variables $x_1, \ldots, x_n$, and the desired value y. In this case, in the ideal case when we plug in the actual (unknown) values of x_i into the algorithm f, we get the exact value $y = f(x_1, \ldots, x_n)$ of y.

In many real-life situations, the relation f between x_i and y is only *approximately* known. In this case, even if we know the exact values of x_i, substituting these values into the approximate function f will not provide us with the exact value of y. In such situations, there is even more uncertainty in y:

- first, there is an uncertainty in y caused by the the uncertainty in the inputs;
- second, there is a *model uncertainty* caused by the fact that the known algorithm f only provides an approximate description of the dependence between the inputs and the output.

Interval computations enable us to estimate the uncertainty in y caused by the uncertainty of the inputs. If there is also a model uncertainty, it has to be estimated separately and added to the uncertainty produced by the interval computations techniques.

In many practical problems, it is important to estimate the inaccuracy of the results of data processing. In many practical applications, it is important to know not only the desired estimate for the quantity y, but also how accurate this estimate is.

For example, in geophysical applications, it is not enough to known that the amount of oil in a given oil files is about 100 million tons. It is important to know how accurate is this estimate.

If the amount is 100 ± 10, this means that the estimates are good enough, and we should start exploring this oil filed. On the other hand, if it is 100 ± 200, this means that it is quite possible that the actual value of the desired quantity y is 0, i.e., that there is no oil at all. In this case, it may be prudent to perform additional measurements before we invest a lot of money into drilling oil wells.

The situation becomes even more critical in medical emergencies: it is not enough to have an estimate of a blood pressure or a body temperature to make a decision (e.g., whether to perform a surgery), it is important that even with the measurement uncertainty, we are sure about the diagnosis – and if we are not, maybe it is desirable to perform more accurate measurements.

Problems of the second class (prediction-related): uncertainty in initial values leads to uncertainty of predicted values. In the prediction problems, we start with the estimates $\widetilde{y}_i$ of the current values of the known quantities; we then apply the prediction algorithm g and produce the prediction $\widetilde{z} = g(\widetilde{y}_1, \ldots, \widetilde{y}_m)$ for the desired future value z.

We have already mentioned that in general, the estimates $\widetilde{y}_i$ of the current values of the quantities y_i are different from the (unknown) actual values y_i of these quantities. Therefore, even if the prediction algorithm is absolutely exact, i.e., if the future value of z is equal to $g(y_1, \ldots, y_m)$, the prediction result $\widetilde{z}$ will be different from the actual future value z.

Comment. In many practical situations, the prediction algorithm is only approximately known, so in general (just as for the problems from the first class), there is also a model uncertainty – an additional component of uncertainty.

4 From Probabilistic to Interval Uncertainty: Case of Indirect Measurements

Let us start with the uncertainty of learning the values of the desired quantities. In the previous section, we have shown that the uncertainties in the results of direct measurements and/or direct expert estimations lead to an uncertainty in our estimates of the current values of the physical quantities. These uncertainties, in their turn, lead to an uncertainty in the predicted values.

We are interested in the uncertainties occurring in problems of both classes: learning the current values and predicting the future values. Since the uncertainty in the future values comes from the uncertainty in the current values, it is reasonable to start with analyzing the uncertainty of the learned values.

Let us start with indirect measurements. In the situation of learning the current values of the physical quantities, there are two possible situations:

- when the (estimates for the) values of the auxiliary quantities x_i come from direct measurements, and
- when these estimates come from the expert estimation.

(Of course, it is also possible that some estimates come from measurement and some from expert estimation.)

There is a lot of experience of handling measurement uncertainty, so we will start our analysis with measurement uncertainty. After that, we will explain how similar techniques can handle expert uncertainty.

Case of direct measurements: what can we know about Δx_i. To estimate the uncertainty Δy caused by the measurement uncertainties Δx_i, we need to have some information about these original uncertainties Δx_i.

The whole idea of uncertainty is that we do not know the exact value of x_i (hence, we do not know the exact value of Δx_i). In order words, there are several possible values of Δx_i. So, the first thing we would like to know is what is the *set* of possible values of Δx_i.

We may also know that some of these possible values are more frequent than the others. In other words, we may also have some information about the *probabilities* of different possible values Δx_i.

We need to go from theoretical possibility to practical situations. Up to now, we have analyzed the situation on a purely theoretical level: what kind of information can we have *in principle*.

From the viewpoint of practical applications, it is desirable to analyze what information we actually have.

First piece of information: upper bound on the measurement error. The manufacturers of a measuring device usually provide us with an upper bound Δ_i for the (absolute value of) possible measurement errors, i.e., with the bound Δ_i for which we are guaranteed that $|\Delta x_i| \le \Delta_i$.

The need for such a bound comes from the very nature of a measurement process. Indeed, if no such bound is provided, this means that the actual value x_i can be as different from the "measurement result" $\widetilde{x}_i$ as possible. Such a value $\widetilde{x}_i$ is not a measurement, it is a wild guess.

Enter intervals. Since the (absolute value of the) measurement error $\Delta x_i = \tilde{x}_i - x_i$ is bounded by the given bound Δ_i, we can therefore guarantee that the actual (unknown) value of the desired quantity belongs to the interval

$$\mathbf{x}_i \stackrel{\text{def}}{=} [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i].$$

Example. For example, if the measured value of a quantity is $\widetilde{x}_i = 1.0$, and the upper bound Δ_i on the measurement error is 0.1, this means that the (unknown) actual value of the measured quantity can be anywhere between $1 - 0.1 = 0.9$ and $1 + 0.1 = 1.1$, i.e., that it can take any value from the interval $[0.9, 1.1]$.

Often, we also know probabilities. In many practical situations, we not only know the interval $[-\Delta_i, \Delta_i]$ of possible values of the measurement error; we also know the probability of different values Δx_i within this interval [47].

In most practical applications, it is assumed that the corresponding measurement errors are normally distributed with 0 means and known standard deviation.

Numerous engineering techniques are known (and widely used) for processing this uncertainty; see, e.g., [47].

How we can determine these probabilities. In practice, we can determine the desired probabilities of different values of Δx_i by comparing

- the result $\widetilde{x}_i$ of measuring a certain quantity with this instrument and
- the result $\widetilde{x}_{i\,\mathrm{st}}$ of measuring the same quantity by a standard (much more accurate) measuring instrument.

Since the standard measuring instrument is much more accurate than the one we use, i.e., $|\widetilde{x}_{i\,\mathrm{st}} - x_i| \ll |\widetilde{x}_i - x_i|$, we can assume that $\widetilde{x}_{i\,\mathrm{st}} = x_i$, and thus, that the difference $\widetilde{x}_i - \widetilde{x}_{i\,\mathrm{st}}$ between these two measurement results is practically equal to the measurement error $\Delta x_i = \widetilde{x}_i - x_i$.

Thus, the empirical distribution of the difference $\widetilde{x}_i - \widetilde{x}_{i\,\mathrm{st}}$ is close to the desired probability distribution for measurement error.

In some important practical situations, we cannot determined these probabilities. In many practical cases, by using standard measuring instruments, we can determine the probabilities of different values of Δx_i. There are two cases, however, when this determination is not done:

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

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

Case of interval uncertainty. In this case, after performing a measurement and getting a measurement result $\widetilde{x}_i$, the only information that we have about the actual value x_i of the measured quantity is that it belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.

In other words, we do know not the actual value x_i of the i-th quantity. Instead, we know the *granule* $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ that contains x_i.

Resulting computational problem. In this situation, for each i, we know the interval $\mathbf{x}_i$ of possible values of x_i, and we need to find the range

$$\mathbf{y} \stackrel{\mathrm{def}}{=} \{f(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

of the given function $f(x_1, \ldots, x_n)$ over all possible tuples $x = (x_1, \ldots, x_n)$ with $x_i \in \mathbf{x}_i$.

The desired range is also usually an interval. Since the function $f(x_1, \ldots, x_n)$ is usually continuous, this range is also an interval, i.e., $\mathbf{y} = [\underline{y}, \overline{y}]$ for some $\underline{y}$ and $\overline{y}$. So, to find this range, it is sufficient to find the endpoints $\underline{y}$ and $\overline{y}$ of this interval.

From traditional (numerical) computations to interval computations. In traditional data processing, we know the estimates $\widetilde{x}_i$ of the input values, and we use these estimates to compute the estimate $\widetilde{y}$ for the desired quantity y. The corresponding algorithm is a particular case of *computations* (which often require a large amount of computing power).

When we take uncertainty in the account, then we have a similar problem in which,

- as inputs, instead of the *numerical* estimates $\widetilde{x}_i$ for x_i, we have *intervals* of possible values of x_i, and
- as an output, instead of a numerical estimate $\widetilde{y}$ for y, we want to compute the interval $[\underline{y}, \overline{y}]$ of possible values of y.

The corresponding computations are therefore called *interval computations.*

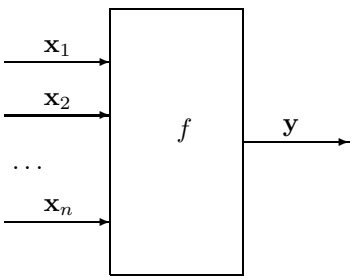
Let us formulate the corresponding problem of interval computations in precise terms.

The main problem of interval computations: a precise description. We are given:

- an integer n;
- n intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$, $\ldots$, $\mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$, and
- an algorithm $f(x_1, \ldots, x_n)$ which transforms n real numbers into a real number $y = f(x_1, \ldots, x_n)$.

We need to compute the endpoints $\underline{y}$ and $\overline{y}$ of the interval

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



Interval computations are also important for the second class of problems: predicting future. In the prediction problem, we start with the known information about the current values $y_1, \ldots, y_m$ of the physical quantities. Based on this information, we would like to derive the information about possible future value $z = g(y_1, \ldots, y_m)$ of each quantity of interest z.

We have already mentioned that in many practically important situations, we can only determine the intervals $[\underline{y}_i, \overline{y}_i]$ of possible values of z. In this case, the only information that we can deduce about z is that z belongs to the range

$$\mathbf{z} = \{g(y_1, \ldots, y_m) : y_1 \in [\underline{y}_1, \overline{y}_1], \ldots, y_m \in [\underline{y}_m, \overline{y}_m]\}.$$

The problem of computing this range is also the problem of interval computations:

- we know intervals of possible values of the input,
- we know the algorithm that transforms the input into the output, and
- we want to find the interval of possible values of the output.

Thus, interval computations are also important for the prediction problem.

5 Case of Expert Uncertainty

How can we describe expert uncertainty. So far, we have analyzed measurement uncertainty. As we have mentioned earlier, expert estimates also come with uncertainty. How can we estimate and process this uncertainty?

Probabilistic approach: its possibility and its limitations. For a measuring instrument, we know how to estimate the probability distribution of the measurement error:

- Ideally, we should compare the measurement results with the actual values of the measured quantity. The resulting differences form a sample from the actual distribution of measurement error. Based on this sample, we can determine the probability distribution for the measurement error.
- In practice, since we cannot determine the exact actual value of the quantity, we use an approximate value obtained by using a more accurate measuring instrument. Based on the sample of the corresponding differences, we can still determine the probability distribution for the measurement error.

In principle, we can do the same for expert estimates. Namely, to estimate the quality of expert estimates, we can consider the cases when the quantity estimates by an expert was consequently measured. Usually, measurements are much more accurate than expert estimates, i.e., $|\widetilde{x}_{\rm meas} - x| \ll |\widetilde{x} - x|$, where x is the (unknown) value of the estimated quantity, $\widetilde{x}$ is the expert's estimate

for this quantity, and $\widetilde{x}_{\rm meas}$ is the result of the consequent measurement of this same quantity. In comparison with expert estimates, we can therefore consider measurement results as approximately equal to the actual values of the quantity: $\widetilde{x}_{\rm meas} - \widetilde{x} \approx \widetilde{x} - x$. Thus, by considering the differences $\widetilde{x}_{\rm meas} - \widetilde{x}$ as a sample from the unknown probability distribution, we can determine the probability distribution for the expert estimation error.

If we have such a probability distribution, then we can use traditional well-developed statistical methods to process expert estimates – the same way we can process measurement results for which we know the distribution of measurement errors.

To determine a probability distribution from the empirical data, we need a large sample: the larger the sample, the more accurate the results.

- A measuring instrument takes a small portion of a second to perform a measurement. Thus, with a measuring instrument, we can easily perform dozens, hundreds, and even thousands of measurements. So, we can have samples which are large enough to determine the corresponding probability distribution with reasonable accuracy.
- On the other hand, for an expert, a single estimate may require a lot of analysis. As a result, for each expert, there are usually few estimates, and it is often not possible to determine the distributions from these estimates.

Experts can produce interval bounds. A measuring instrument usually simply produces a number; it cannot be easily modified to also produce an information about the measurement uncertainty, such as the upper bound on the measurement error.

In contrast, an expert is usually able not only to supply us with an estimate, but also to provide us with an accuracy of this estimate. For example, an expert can estimate the age of a person as $\widetilde{x} = 30$, and indicate that this is 30 plus minus $\Delta = 5$. In such a situation, what the expert is actually saying is that the actual (unknown) value of the estimated quantity should be in the interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$.

Interval computations are needed to handle interval uncertainty in expert estimates. Let us now consider a typical situation of data processing. We are interested in some quantity y which is difficult to estimate directly. To estimate y, we ask experts to estimate the values of the auxiliary quantities $x_1, \ldots, x_n$ which are related to y by a known dependence $y = f(x_1, \ldots, x_n)$.

Based on the experts' estimates $\widetilde{x}_i$ and on the expert estimates Δ_i of their inaccuracy, we conclude that the actual (unknown) value of the each quantity x_i belongs to the interval $\mathbf{x}_i \stackrel{\rm def}{=} [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. Thus, we can conclude that the actual value of $y = f(x_1, \ldots, x_n)$ belongs to the interval range

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

The problem of computing this range is exactly the problem of interval computations.

From interval to fuzzy uncertainty. Usually, experts can provide guaranteed bounds Δ_i on the inaccuracy of their estimates. Often, however, in addition to these (rather wide) bounds, experts can also produce narrower bounds – which are, however, only true with a certain degree of certainty.

For example, after estimating the age as 30,

- in addition to saying that an estimation inaccuracy is always ≤ 5 (with 100% certainty),
- an expert can also say that with 90% certainty, this inaccuracy is ≤ 4, and
- with 70% certainty, this inaccuracy is ≤ 2.

Thus, instead of a single interval $[30-5, 30+5] = [25, 35]$ that is guaranteed to contain the (unknown) age with certainty 100%, the expert also produces a narrower interval $[30-4, 30+4] = [26, 34]$ which contains this age with certainty 90%, and an even narrower interval $[30-2, 30+2] = [28, 32]$ which contains the age with certainty 70%. So, we have three intervals which are *nested* in the sense that every interval corresponding to a smaller degree of certainty is contained in the interval corresponding to the larger degree of certainty: $[28, 32] \subseteq [26, 34] \subseteq [25, 35]$.

In general, instead of single interval, we have a nested family of intervals corresponding to different degree of certainty. Such a nested family of intervals can be viewed as a *fuzzy number* [25, 44]: for every value x, we can define the degree $\mu(x)$ to which x is possible as 1 minus the largest degree of certainty α for which x belongs to the α-interval.

Interval computations are needed to process fuzzy data. For expert estimates, for each input i, we may have different intervals $\mathbf{x}_i(\alpha)$ corresponding to different degrees of certainty α. Our objective is then to produce the corresponding intervals for $y = f(x_1, \ldots, x_n)$.

For $\alpha = 1$, i.e., for intervals in which the experts are 100% confident, it is natural to take $\mathbf{y}(1) = f(\mathbf{x}_1(1), \ldots, \mathbf{x}_n(1))$. Similarly, for each α, if we want to consider believes at this level α, then we can combine the corresponding intervals $\mathbf{x}_i(\alpha)$ into the desired interval $\mathbf{y}(\alpha)$ for y: $\mathbf{y}(\alpha) = f(\mathbf{x}_1(\alpha), \ldots, \mathbf{x}_n(\alpha))$.

It turns out that the resulting fuzzy number is exactly what we would get if we simply apply Zadeh's extension principle to the fuzzy numbers corresponding to x_i [25, 43, 44]

So, in processing fuzzy expert opinions, we also need interval computations.

6 Interval Computations Are Sometimes Easy but In General, They Are Computationally Difficult (NP-Hard)

Interval computations are needed in practice: a reminder. In the previous sections, we have explained why interval computations are needed in many practical problems. In other words, in many practical situations, we know n intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$, we know an algorithm $f(x_1, \ldots, x_n)$, and need to find the range of the function f on these intervals:

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

Let us first analyze the computational complexity of this problem. Before we start explaining how to solve this problem, let us make a useful detour.

Until the 1930s, researchers believed that every mathematical problem can be solved. Under this belief, once we have a mathematical problem of practical importance, we should try to solve it in its entire generality.

Starting with the famous Gödel's result, it is well known that some mathematical problems cannot be solved in the most general case. For such problems, attempts to solve them in their most general form would be a futile waste of time. At best, we can solve some important class of such problems, or get an approximate solution. To avoid this waste of efforts, before we start solving a difficult problem, it is desirable to first analyze whether this problem can be solved in its utmost generality.

This strategy was further clarified in the 1970s, when it turned out, crudely speaking, that some problem cannot be *efficiently* solved; such difficult problems are called *NP-hard*; see [12, 28, 45] for detailed description. If a problem is NP-hard, then it is hopeless to search for a general efficient solution; we must look for efficient solutions to subclasses of this problem and/or approximate solutions.

Comment. Strictly speaking, NP-hardness does not necessarily mean that the problem if computationally difficult: this is only true under a hypothesis NP$\neq P$ which is widely believed but not proven yet (it is probably the most well known open problem in theoretical computer science).

Interval computations are sometimes easy: case of monotonicity. In some cases, it is easy to estimate the desired range. For example, the arithmetic average

$$E = \frac{x_1 + \ldots + x_n}{n}$$

is a monotonically increasing function of each of its n variables $x_1, \ldots, x_n$. So:

- the smallest possible value $\underline{E}$ of the average E is attained when each value x_i is the smallest possible ($x_i = \underline{x}_i$), and

- the largest possible value $\overline{E}$ of the average E is attained when $x_i = \overline{x}_i$ for all i.

In other words, the range $\mathbf{E}$ of E is equal to $[E(\underline{x}_1, \ldots, x_n), E(\overline{x}_1, \ldots, \overline{x}_n)]$, where,

$$\underline{E} = \frac{1}{n} \cdot (\underline{x}_1 + \ldots + \underline{x}_n)$$

and

$$\overline{E} = \frac{1}{n} \cdot (\overline{x}_1 + \ldots + \overline{x}_n).$$

In general, if $f(x_1, \ldots, x_n)$ is a monotonically increasing function of each of its n variables, then:

- The smallest possible value $\underline{y}$ of the function f over given intervals $[\underline{x}_i, \overline{x}_i]$ is attained when all its inputs x_i take the smallest possible values $x_i = \underline{x}_i$. In this case, $\underline{y} = f(\underline{x}_1, \ldots, \underline{x}_n)$.
- The largest possible value $\overline{y}$ of the function f over given intervals $[\underline{x}_i, \overline{x}_i]$ is attained when all its inputs x_i take the largest possible values $x_i = \overline{x}_i$. In this case, $\overline{y} = f(\overline{x}_1, \ldots, \overline{x}_n)$.

Thus, we have an explicit formula for the desired range:

$$[\underline{y}, \overline{y}] = [f(\underline{x}_1, \ldots, \underline{x}_n), f(\overline{x}_1, \ldots, \overline{x}_n)].$$

A similar formula can be written down if the function $f(x_1, \ldots, x_n)$ is increasing with respect to some of its inputs and decreasing with respect to some others. In this case, to compute $\overline{y}$, we must take:

- $x_i = \overline{x}_i$ for all the variables x_i relative to which f is increasing, and
- $x_j = \underline{x}_j$ for all the variables x_j relative to which f is decreasing.

Similarly, to compute $\underline{y}$, we must take:

- $x_i = \underline{x}_i$ for all the variables x_i relative to which f is increasing, and
- $x_j = \overline{x}_j$ for all the variables x_j relative to which f is decreasing.

Case of linear functions $f(x_1, \ldots, x_n)$. In the previous section, we showed how to compute the range of a function which is monotonic in each of its variable – and it can be increasing relative to some of them and decreasing relative to some others.

An example of such a function is a general linear function $f(x_1, \ldots, x_n) = c_0 + \sum_{i=1}^{n} c_i \cdot x_i$. Substituting $x_i = \widetilde{x}_i - \Delta x_i$ into this expression, we conclude that

$$y = f(x_1, \ldots, x_n) = c_0 + \sum_{i=1}^{n} c_i \cdot (\widetilde{x}_i - \Delta x_i) = c_0 + \sum_{i=1}^{n} c_i \cdot \widetilde{x}_i - \sum_{i=1}^{n} c_i \cdot \Delta x_i.$$

By definition, $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) = c_0 + \sum_{i=1}^{n} c_i \cdot \widetilde{x}_i$, so we have

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

The dependence of Δy on Δx_i is linear: it is increasing relative to x_i if $c_i \geq 0$ and decreasing if $c_i < 0$. So, to find the largest possible value Δ of Δy, we must take:

- the largest possible value $\Delta x_i = \Delta_i$ when $c_i \geq 0$, and
- the smallest possible value $\Delta x_i = -\Delta_i$ when $c_i < 0$.

In both cases, the corresponding term in the sum has the form $|c_i| \cdot \Delta_i$, so we can conclude that

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

Similarly, the smallest possible value of Δy is equal to $-\Delta$. Thus, the range of possible values of y is equal to $[\underline{y}, \overline{y}] = [\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.

Interval computations are, in general, computationally difficult. We have shown that for linear functions, we can easily compute the interval range.

Linear functions often occur in practice, because an arbitrary function can be usually expanded in Taylor series, and then we can only keep a few first terms to get a good description of the actual dependence. If we keep only linear terms, then we get a linear approximation to the original dependence.

If the accuracy of this linear approximation is not sufficient, then it is natural to also consider quadratic terms. A natural question is: is the corresponding interval computations problem still feasible?

Alas, it turns out that for quadratic functions, interval computations problem is, in general, NP-hard; this was first proved in [52]. Moreover, it turns out that it is NP-hard not just for some rarely-used exotic quadratic functions: it is known that the problem of computing the exact range $\mathbf{V} = [\underline{V}, \overline{V}]$ for the variance

$$V = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^2 = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - \left(\frac{1}{n} \cdot \sum_{i=1}^{n} x_i \right)^2$$

over interval data $x_i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ is, in general, NP-hard; see, e.g., [8, 9]. To be more precise, there is a polynomial-time algorithm for computing $\underline{V}$, but computing $\overline{V}$ is, in general, NP-hard.

Historical comment. NP-hardness of interval computations was first proven in [10, 11]. A general overview of computational complexity of different problems of data processing and interval computations is given in [28].

7 Maximum Entropy and Linearization: Useful Techniques for Solving Many Practical Cases of Interval Computations Problem, Their Advantages and Limitations

In many practical situations, an approximate estimate is sufficient. The NP-hardness result states that computing the *exact* range $[\underline{y}, \overline{y}]$, i.e., in other words, computing the *exact* values of the endpoints $\underline{y}$ and $\overline{y}$, is NP-hard.

In most practical problems, however, it is not necessary to produce the exact values of the range, good approximate values will be quite sufficient.

Computing the range with guaranteed accuracy is still NP-hard. Thus, we arrive at the following natural question. Suppose that we fix an accuracy ε, and we consider the problem of computing $\underline{y}$ and $\overline{y}$ with this accuracy, i.e., the problem of computing the values $\overline{Y}$ and $\underline{Y}$ for which $|\overline{Y} - \overline{y}| \leq \varepsilon$ and $|\underline{Y} - \underline{y}| \leq \varepsilon$.

In this case, we can guarantee that $\underline{Y} - \varepsilon \leq \underline{y} \leq \underline{Y} + \varepsilon$ and $\overline{Y} - \varepsilon \leq \overline{y} \leq \overline{Y} + \varepsilon$. So, if we succeed in computing the estimate $\underline{Y}$ and $\overline{Y}$, then we do not have the exact range, but we have an ε-approximation for the (unknown) desired range $\mathbf{y}$: namely, we know that

$$[\underline{Y} + \varepsilon, \overline{Y} - \varepsilon] \subseteq \mathbf{y} \subseteq [\underline{Y} - \varepsilon, \overline{Y} + \varepsilon].$$

Is the problem of computing such values $\underline{Y}$ and $\overline{Y}$ computationally simpler? Alas, it turns out that this new problem is still NP-hard; see, e.g., [28].

In some practical problems, it is OK to have estimates which are not guaranteed. The difficulty of solving the general problem of interval computations comes from the fact that we are looking for *guaranteed* bounds for $\underline{y}$ and $\overline{y}$.

In some practical problems, we are not 100% sure that our algorithm $f(x_1, \ldots, x_n)$ is absolutely correct. This happens, e.g., in prediction problems, where the dynamic equations used for prediction are only approximately known anyway.

In such situations, it is OK to have estimates sometimes deviating from the desired range.

Possible approaches to this problem. In order to describe possible approaches to this problem, let us first recall what properties of our problem make it computationally complex. By relaxing these properties, we will be able to come up with computationally efficient algorithms.

We have mentioned that in some practical situations, we know the probability distributions of the estimation errors Δx_i. In such situations, the problem of

estimating the effect of these approximation errors Δx_i on the result of data processing is computationally easy. Namely, we can use Monte-Carlo simulations (see, e.g., [49]), when for several iterations $k = 1, \ldots, N$, we do the following:

- we simulate the inputs $\Delta x_i^{(k)}$ according to the known probability distributions;
- we substitute the resulting simulated values $x_i^{(k)} = \widetilde{x}_i - \Delta x_i^{(k)}$ into the algorithm f, producing $y^{(k)} = f(x_1^{(k)}, \ldots, x_n^{(k)})$, and then
- use the sample of the differences $\Delta y^{(k)} = \widetilde{y} - y^{(k)}$ to get the probability distribution of Δy.

Thus, the first difficulty of interval computations comes from the fact that we do not know the probability distribution. However, the mere fact that we do not know this distribution does not necessarily make the problem computational complex. For example, even when we restrict ourselves to interval uncertainty, for linear functions f, we still have a feasible algorithm for computing the range. Thus, the complexity of the general interval computations problem is caused by two following two properties of this general problem:

- first, that we do not know the probability distribution for the inputs Δx_i;
- second, that the function $f(x_1, \ldots, x_n)$ is non-linear.

To be able to perform efficient computations, we must relax one of these properties. Thus, we arrive at two possible ways to solve this problem:

- first, we can select one of the possible probability distributions;
- second, we can approximate the original function f by a linear one.

Let us describe these two ideas in more detail.

First idea: selecting a probability distribution. As we have mentioned, in many cases, we know the probability distribution for approximation errors Δx_i. Interval uncertainty corresponds to the case when we only have a partial information about this probability distribution: namely, the only thing we know about this distribution is that it is located (with probability 1) somewhere on the interval $[-\Delta_i, \Delta_i]$.

This distribution could be uniform on this interval, could be a truncated Gaussian distribution, could be a 1-point degenerate distribution in which the value Δx_i is equal to one fixed value from this interval with probability 1.

Situations in which we have partial information about the probability distributions are common is statistics. In such situations, we have several different probability distributions which are all consistent with the given knowledge. One way to handle these situations is to select one of the these distributions, the one which is, in some sense, the most reasonable to select.

Simplest case: Laplace's principle of indifference. The approach started with the early 19 century work of the famous mathematician Pierre Simon Laplace who analyzed the simplest of such situations, when we have finitely many (n) alternatives, and we have no information about their probabilities. In this simple situation, the original situation is invariant with respect to arbitrary permutations of the original alternatives. So, it is reasonable to select the probabilities which reflect this symmetry – i.e., equal probabilities $p_1 = \ldots = p_n$. Since the total probability $\sum_{i=1}^{n} p_i$ must be equal to 1, we thus conclude that $p_1 = \ldots = p_n = \dfrac{1}{n}$. This idea is called *Laplace principle of indifference.*

General case: Maximum Entropy approach. Laplace's simple idea can be naturally applied to the more general case, when we have partial information about the probabilities, i.e., when there are several possible distributions which are consistent with our knowledge. In this case, it is reasonable to view these distributions as possible alternatives. So, we discretize the variables (to make sure that the overall number of alternatives is finite), and then consider all possible distributions as equally probable. As the discretization constant tends to 0, we should get a distribution of the class of all (non-discretized) distributions.

It turns out that in the limit, only one such distribution has probability 1: namely, the distribution which has the largest possible value of the entropy $S \stackrel{\text{def}}{=} -\int \rho(x) \cdot \ln(\rho(x))\, dx$ (here $\rho(x)$ denotes the probability density). For details on this Maximum Entropy approach and its relation to interval uncertainty and Laplace's principle of indifference, see, e.g., [5, 21, 24]

Maximum entropy method for the case of interval uncertainty. One can easily check that for a single variable x_1, among all distributions located on a given interval, the entropy is the largest when this distribution is *uniform* on this interval. In the case of several variables, we can similarly conclude that the distribution with the largest value of the entropy is the one which is uniformly distributed in the corresponding box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$, i.e., a distribution in which:

- each variable Δx_i is uniformly distributed on the corresponding interval $[-\Delta_i, \Delta_i]$, and
- variables corresponding to different inputs are statistically independent.

This is indeed one of the main ways how interval uncertainty is treated in engineering practice: if we only know that the value of some variable is in the interval $[\underline{x}_i, \overline{x}_i]$, and we have no information about the probabilities, then we assume that the variable x_i is uniformly distributed on this interval.

Limitations of the maximum entropy approach. To explain the limitations of this engineering approach, let us consider the simplest possible algorithm $y = f(x_1, \ldots, x_n) = x_1 + \ldots + x_n$. For simplicity, let us assume that the

measured values of all n quantities are 0s $\widetilde{x}_1 = \ldots = \widetilde{x}_n = 0$, and that all n measurements have the same error bound Δ_x; $\Delta_1 = \ldots = \Delta_n = \Delta_x$.

In this case, $\Delta y = \Delta x_1 + \ldots + \Delta x_n$. Each of n component measurement errors can take any value from $-\Delta_x$ to Δ_x, so the largest possible value of Δy is attained when all of the component errors attain the largest possible value $\Delta x_i = \Delta_x$. In this case, the largest possible value Δ of Δy is equal to $\Delta = n \cdot \Delta_x$.

Let us see what the maximum entropy approach will predict in this case. According to this approach, we assume that Δx_i are independent random variables, each of which is uniformly distributed on the interval $[-\Delta, \Delta]$. According to the Central Limit theorem [50, 53], when $n \to \infty$, the distribution of the sum of n independent identically distributed bounded random variables tends to Gaussian. This means that for large values n, the distribution of Δy is approximately normal.

Normal distribution is uniquely determined by its mean and variance. When we add several independent variables, their means and variances add up. For each uniform distribution Δx_i on the interval $[-\Delta_x, \Delta_x]$ of width $2\Delta_x$, the probability density is equal to $\rho(x) = \dfrac{1}{2\Delta_x}$, so the mean is 0 and the variance is

$$V = \int_{-\Delta_x}^{\Delta_x} x^2 \cdot \rho(x)\, dx = \frac{1}{2\Delta_x} \cdot \int_{-\Delta_x}^{\Delta_x} x^2\, dx = \frac{1}{2\Delta_x} \cdot \frac{1}{3} \cdot x^3 \Big|_{-\Delta_x}^{\Delta_x} = \frac{1}{3} \cdot \Delta^2.$$

Thus, for the sum Δy of n such variables, the mean is 0, and the variance is equal to $(n/3) \cdot \Delta - x^2$. Thus, the standard deviation is equal to $\sigma = \sqrt{V} = \Delta_x \cdot \dfrac{\sqrt{n}}{\sqrt{3}}$.

It is known that in a normal distribution, with probability close to 1, all the values are located within the $k \cdot \sigma$ vicinity of the mean: for $k = 3$, it is true with probability 99.9%, for $k = 6$, it is true with probability 10^{-6}%, etc. So, practically with certainty, Δy is located within an interval $k \cdot \sigma$ which grows with n as $\sqrt{n}$.

For large n, we have $k \cdot \Delta_x \cdot \dfrac{\sqrt{n}}{\sqrt{3}} \ll \Delta_x \cdot n$, so we get a serious underestimation of the resulting measurement error. This example shows that estimates obtained by selecting a single distribution can be very misleading.

Linearization: main idea. As we have mentioned earlier, another way to handle the complexity of the general interval computations problem is to approximate the original expression $y = f(x_1, \ldots, x_n) = f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n)$ by linear terms in its Taylor expansion:

$$y \approx f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \cdot \Delta x_i,$$

where the partial derivatives are computed at the midpoint $\widetilde{x} = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$. Since $f(\widetilde{x}_1, \ldots, \widetilde{x}_n) = \widetilde{y}$, we conclude that $\Delta y = \widetilde{y} - y = \sum\limits_{i=1}^{n} c_i \cdot \Delta x_i$, where $c_i = \dfrac{\partial f}{\partial x_i}$.

We already know how to compute the interval range for a linear function, the resulting formula is $\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i$. Thus, to compute Δ, it is sufficient to know the partial derivatives c_i.

Linearization: how to compute. A natural way to compute partial derivatives comes directly from the definition. By definition, a partial derivative is defined as a limit

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1, \ldots, \widetilde{x}_n)}{h}.$$

In turn, a limit, by its definition, means that when the value of h is small, the corresponding ratio is very close to the partial derivative. Thus, we can estimate the partial derivative as the ratio

$$c_i = \frac{\partial f}{\partial x_i} \approx \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1, \ldots, \widetilde{x}_n)}{h}$$

for some small value h.

After we have computed n such ratios, we can then compute the desired bound Δ on $|\Delta y|$ as $\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i$.

Linearization: how to compute faster. The above algorithm requires that we call the data processing algorithm $n + 1$ times: first to compute the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$, and then n more times to compute the values

$$f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n)$$

and thus, the corresponding partial derivatives.

In many practical situations, the data processing algorithms are time-consuming, and we process large amounts of data, with the number n of data points in thousands. In this case, the use of the above linearization algorithm would require thousands time longer than data processing itself – which itself is already time consuming. Is it possible to estimate Δ faster?

The answer is "yes", it is possible to have an algorithm which estimates Δ by using only a constant number of calls to the data processing algorithm f; for details, see, e.g. [26, 27].

In some situations, we need a guaranteed enclosure. In many application areas, it is sufficient to have an approximate estimate of y. However, in some applications, it is important to guarantee that the (unknown) actual value y of a certain quantity does not exceed a certain threshold y_0. The only way to guarantee this is to have an interval $\mathbf{Y} = [\underline{Y}, \overline{Y}]$ which is guaranteed to contain y (i.e., for which $\mathbf{y} \subseteq \mathbf{Y}$) and for which $\overline{Y} \leq y_0$.

For example, in nuclear engineering, we must make sure that the temperatures and the neutron flows do not exceed the critical values; when planning a

space flight, we want to guarantee that the space ship lands on the planet and does not fly pass it, etc.

The interval $\mathbf{Y}$ which is guaranteed to contain the actual range $\mathbf{y}$ is usually called an *enclosure* for this range. So, in such situations, we need to compute either the original range or at least an enclosure for this range. Computing such an enclosure is also one of the main tasks of interval computations.

8 Interval Computations: A Brief Historic Overview

Before we start describing the main interval computations techniques, let us briefly overview the history of interval computations.

Pre-history of interval computations: interval computations as a part of numerical mathematics. The notion of interval computations is reasonably recent, it dates from the 1950s, but the main problem is known since Archimedes who used guaranteed two-sided bounds to compute π; see, e.g., [1].

Since then, many useful guaranteed bounds have been developed for different numerical methods. There have also been several general descriptions of such bounds, often formulated in terms similar to what we described above. For example, in the early 20 century, the concept of a function having values which are bounded within limits was discussed by W. H. Young in [56]. The concept of operations with a set of multi-valued numbers was introduced by R. C. Young, who developed a formal algebra of multi-valued numbers [57]. The special case of closed intervals was further developed by P. S. Dwyer in [6].

Limitations of the traditional numerical mathematics approach. The main limitations of the traditional numerical mathematics approach to error estimation was that often, no clear distinction was made between approximate (non-guaranteed) and guaranteed (= interval) error bounds.

For example, for iterative methods, many papers on numerical mathematics consider the rate of convergence as an appropriate measure of approximation error. Clearly, if we know that the error decreases as $O(1/n)$ or as $O(a^{-n})$, we gain some information about the corresponding algorithms – and we also gain a knowledge that for large n, the second method is more accurate. However, in real life, we make a fixed number n of iterations. If the only information we have about the approximation error is the above asymptotics, then we still have no idea how close the result of n-th iteration is to the actual (desired) value.

It is therefore important to emphasize the need for guaranteed methods, and to develop techniques for producing *guaranteed* estimates. Such guaranteed estimates is what interval computations are about.

Origins of interval computations. Interval computations were independently invented by three researchers in three different parts of the world: by

M. Warmus in Poland [54, 55], by T. Sunaga in Japan [51], and by R. Moore in the USA [33, 38, 34, 35, 36, 37, 39, 40].

The active interest in interval computations started with Moore's 1966 monograph [39]. This interest was enhanced by the fact that in addition to estimates for general numerical algorithms, Moore's monograph also described *practical* applications which have already been developed in his earlier papers and technical reports: in particular, interval computations were used to make sure that even when we take all the uncertainties into account, the trajectory of a space flight is guaranteed to reach the Moon.

Since then, interval computations have been actively used in many areas of science and engineering [19, 20].

Comment. An early history of interval computations is described in detail in [31] and in [41]; early papers on interval computations can be found on the interval computations website [19].

9 Interval Computations: Main Techniques

General comment about algorithms and parsing. Our goal is to find the range of a given function $f(x_1, \ldots, x_n)$ on the given intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1], \ldots, \mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$.

This function $f(x_1, \ldots, x_n)$ is given as an algorithm. In particular, we may have an explicit analytical expression for f, in which case this algorithm consists of simply computing this expression.

When we talk about algorithms, we usually mean an algorithm (program) written in a high-level programming language like Java or C. Such programming languages allows us to use arithmetic expressions and many other complex constructions. Most of these constructions, however, are not directly implemented inside a computer. Usually, only simple arithmetic operations are implemented: addition, subtraction, multiplication, and $1/x$ (plus branching). Even division a/b is usually not directly supported, it is performed as a sequence of two elementary arithmetic operations:

- first, we compute $1/b$;
- then, we multiply a by $1/b$.

When we input a general program into a computer, the computer *parses* it, i.e., represents it a sequence of elementary arithmetic operations.

Since a computer performs this parsing anyway, we can safely assume that the original algorithm $f(x_1, \ldots, x_n)$ is already represented as a sequence of such elementary arithmetic operations.

Interval arithmetic. Let us start our analysis of the interval computation techniques with the simplest possible case when the algorithm $f(x_1, \ldots, x_n)$

simply consists of a single arithmetic operation: addition, subtraction, multiplication, or computing $1/x$.

Let us start by estimating the range of the addition function $f(x_1, x_2) = x_1 + x_2$ on the intervals $[\underline{x}_1, \overline{x}_1]$ and $[\underline{x}_2, \overline{x}_2]$. This function is increasing with respect to both its variables. We already know how to compute the range $[\underline{y}, \overline{y}]$ of a monotonic function. So, the range of addition is equal to $[\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2]$.

The desired range is usually denoted as $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$; in particular, for addition, this notation takes the form $\mathbf{x}_1 + \mathbf{x}_2$. Thus, we can define "addition" of two intervals as follows:

$$[\underline{x}_1, \overline{x}_1] + [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2].$$

This formula makes perfect intuitive sense: if one town has between 700 and 800 thousand people, and it merges with a nearby town whose population is between 100 and 200 thousand, then:

- the smallest possible value of the total population of the new big town is when both populations were the smallest possible, $700 + 100 = 800$, and
- the largest possible value is when both populations are the largest possible, i.e., $800 + 200 = 1000$.

The subtraction function $f(x_1, x_2) = x_1 - x_2$ is increasing with respect to x_1 and decreasing with respect to x_2, so we have

$$[\underline{x}_1, \overline{x}_1] - [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 - \overline{x}_2, \overline{x}_1 - \underline{x}_2].$$

These operations are also in full agreement with common sense. For example, if a warehouse originally had between 6.0 and 8.0 tons, and we moved between 1.0 and 2.0 tons to another location, then the smallest amount left is when we start with the smallest possible value 6.0 and move the largest possible value 2.0, resulting in $6.0 - 2.0 = 4.0$. The largest amount left is when we start with the largest possible value 8.0 and move the smallest possible value 1.0, resulting in $8.0 - 1.0 = 7.0$.

For multiplication $f(x_1, x_2) = x_1 \cdot x_2$, the direction of monotonicity depends on the actual values of x_1 and x_2: e.g., when $x_2 > 0$, the product increases with x_1, otherwise it decreases with x_1. So, unless we know the signs of the product beforehand, we cannot tell whether the maximum is attained at $x_1 = \underline{x}_1$ or at $x_1 = \overline{x}_1$. However, we know that it is always attained at one of these endpoints. So, to find the range of the product, it is sufficient to try all $2 \cdot 2 = 4$ combinations of these endpoints:

$$[\underline{x}_1, \overline{x}_1] \cdot [\underline{x}_2, \overline{x}_2] =$$

$$[\min(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2), \max(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2)].$$

Finally, the function $f(x_1) = 1/x_1$ is decreasing wherever it is defined (when $x_1 \neq 0$), so if $0 \notin [\underline{x}_1, \overline{x}_1]$, then

$$\frac{1}{[\underline{x}_1, \overline{x}_1]} = \left[\frac{1}{\overline{x}_1}, \frac{1}{\underline{x}_1}\right].$$

The formulas for addition, subtraction, multiplication, and reciprocal of intervals are called formulas of *interval arithmetic* .

Computational complexity of interval arithmetic. Interval addition requires two additions of numbers; interval subtraction requires two subtraction of numbers, and dividing 1 by an interval requires two divisions of 1 by a real number. In all these operations, we need twice longer time to perform the corresponding interval operation than to perform an operation with real numbers.

The only exception is interval multiplication which requires 4 multiplications of numbers. Thus, if we use the above formulas, we get, in the worst case, a 4 times increase in computation time.

Computational comment: interval multiplication can be performed faster. It is known that we can compute the interval product faster, by using only 3 multiplications [18, 16]. Namely:

- if $\underline{x}_1 \geq 0$ and $\underline{x}_2 \geq 0$, then $\mathbf{x}_1 \cdot \mathbf{x}_2 = [\underline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2]$;
- if $\underline{x}_1 \geq 0$ and $\underline{x}_2 \leq 0 \leq \overline{x}_2$, then $\mathbf{x}_1 \cdot \mathbf{x}_2 = [\overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2]$;
- if $\underline{x}_1 \geq 0$ and $\overline{x}_2 \leq 0$, then $\mathbf{x}_1 \cdot \mathbf{x}_2 = [\overline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2]$;
- if $\underline{x}_1 \leq 0 \leq \overline{x}_1$ and $\underline{x}_2 \geq 0$, then $\mathbf{x}_1 \cdot \mathbf{x}_2 = [\underline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2]$;
- if $\underline{x}_1 \leq 0 \leq \overline{x}_1$ and $\underline{x}_2 \leq 0 \leq \overline{x}_2$, then
$$\mathbf{x}_1 \cdot \mathbf{x}_2 = [\min(\underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2), \max(\underline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2)];$$
- if $\underline{x}_1 \leq 0 \leq \overline{x}_2$ and $\overline{x}_2 \leq 0$, then $\mathbf{x}_1 \cdot \mathbf{x}_2 = [\underline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2]$;
- if $\overline{x}_1 \leq 0$ and $\underline{x}_2 \geq 0$, then $\mathbf{x}_1 \cdot \mathbf{x}_2 = [\underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2]$;
- if $\overline{x}_1 \leq 0$ and $\underline{x}_2 \leq 0 \leq \overline{x}_2$, then $\mathbf{x}_1 \cdot \mathbf{x}_2 = [\underline{x}_1 \cdot \overline{x}_2, \underline{x}_1 \cdot \underline{x}_2]$;
- if $\overline{x}_1 \leq 0$ and $\overline{x}_2 \leq 0$, then $\mathbf{x}_1 \cdot \mathbf{x}_2 = [\overline{x}_1 \cdot \overline{x}_2, \underline{x}_1 \cdot \overline{x}_2]$.

We see that in 8 out of 9 cases, we need only 2 multiplications, and the only case when we still need 4 multiplications is when $0 \in \mathbf{x}_1$ and $0 \in \mathbf{x}_2$. In this case, it can be also shown that 3 multiplications are sufficient:

- If $0 \leq |\underline{x}_1| \leq \overline{x}_1$ and $0 \leq |\underline{x}_2| \leq \overline{x}_2$, then
$$\mathbf{x}_1 \cdot \mathbf{x}_2 = [\min(\underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2), \overline{x}_1 \cdot \overline{x}_2].$$
- If $0 \leq \overline{x}_1 \leq |\underline{x}_1|$ and $0 \leq \overline{x}_2 \leq |\underline{x}_2|$, then
$$\mathbf{x}_1 \cdot \mathbf{x}_2 = [\min(\underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2), \underline{x}_1 \cdot \underline{x}_2].$$
- If $0 \leq |\underline{x}_1| \leq \overline{x}_1$ and $0 \leq \overline{x}_2 \leq |\underline{x}_2|$, then
$$\mathbf{x}_1 \cdot \mathbf{x}_2 = [\overline{x}_1 \cdot \underline{x}_2, \max(\underline{x}_1 \cdot \underline{x}_2, \overline{x}_1, \overline{x}_2)].$$
- If $0 \leq \overline{x}_1 \leq |\underline{x}_1|$ and $0 \leq |\underline{x}_2| \leq \overline{x}_2$, then
$$\mathbf{x}_1 \cdot \mathbf{x}_2 = [\underline{x}_1 \cdot \overline{x}_2, \max(\underline{x}_1 \cdot \underline{x}_2, \overline{x}_1, \overline{x}_2)].$$

Straightforward ("naive") interval computations: idea. We know how to compute the range for each arithmetic operation. Therefore, to compute the range $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, it is reasonable to do the following:

- first, we parse the algorithm f (this is done automatically by a compiler);
- then, we repeat the computations forming the program f step-by-step, replacing each operation with real numbers by the corresponding operation of interval arithmetic.

It is known that, as a result, we get an enclosure $\mathbf{Y}$ for the desired range $\mathbf{y}$ [20, 39].

Example where straightforward interval computations work perfectly. Let us start with an example of computing the average of two values $f(x_1, x_2) = 0.5 \cdot (x_1 + x_2)$. This function is increasing in both variables, so its range on the intervals $[\underline{x}_1, \overline{x}_1]$ and $[\overline{x}_2, \overline{x}_2]$ is equal to $[0.5 \cdot (\underline{x}_1 + \underline{x}_2), 0.5 \cdot (\overline{x}_1 + \overline{x}_2)]$.

A compiler will parse the function f into the following sequence of computational steps:

- we start with x_1 and x_2;
- then, we compute an intermediate value $x_3 = x_1 + x_2$;
- finally, we compute $y = 0.5 \cdot x_3$.

According to straightforward interval computations:

- we start with $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$ and $\mathbf{x}_2 = [\underline{x}_2, \overline{x}_2]$;
- then, we compute $\mathbf{x}_3 = \mathbf{x}_1 + \mathbf{x}_2 = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2]$;
- finally, we compute $\mathbf{y} = 0.5 \cdot \mathbf{x}_3$, and we get the desired range.

One can easily check that we also get the exact range for the general case of the arithmetic average, and, even more generally, for an arbitrary linear function $f(x_1, \ldots, x_n)$.

Can straightforward interval computations be always perfect? In straightforward interval computations, we replace each elementary arithmetic operation with the corresponding operation of interval arithmetic. We have already mentioned that this replacement increases the computation time at most by a factor of 4. So, if we started with the polynomial time, we still get polynomial time.

On the other hand, we know that the main problem of interval computations is NP-hard. This means, crudely speaking, that we cannot always compute the exact range by using a polynomial-time algorithm. Since straightforward interval computations is a polynomial-time algorithm, this means that in some cases, its estimates for the range are not exact. Let us describe a simple example when this happens.

Example where straightforward interval computations do not work perfectly. Let us illustrate straightforward interval computations on the example of a simple function $f(x_1) = x_1 - x_1^2$; we want to estimate its range when $x_1 \in [0, 1]$.

To be able to check how good is the resulting estimate, let us first find the actual range of f. According to calculus, the minimum and the maximum of a smooth (differentiable) function on an interval is attained either at one of the endpoints or at one of the extreme points where the derivative of this function is equal to 0. So, to find the minimum and the maximum, it is sufficient to compute the value of this function at the endpoints and at all the extreme points:

- the largest of these values is the maximum, and
- the smallest of these values is the minimum.

For the endpoints $x_1 = 0$ and $x_1 = 1$, we have $f(0) = f(1) = 0$. By differentiating this function and equating the derivative $1 - 2x_1$ to 0, we conclude that this function has only one extreme point $x_1 = 0.5$. At this point, $f(0.5) = 0.25$, so $\underline{y} = \min(0, 0, 0.25) = 0$ and $\overline{y} = \max(0, 0, 0.25) = 0.25$. In other words, the actual range is $\mathbf{y} = [0, 0.25]$.

Let us now apply straightforward interval computations. A compiler will parse the function into the following sequence of computational steps:

- we start with x_1;
- then, we compute $x_2 = x_1 \cdot x_1$;
- finally, we compute $y = x_1 - x_2$.

According to straightforward interval computations:

- we start with $\mathbf{x}_1 = [0, 1]$;
- then, we compute $\mathbf{x}_2 = \mathbf{x}_1 \cdot \mathbf{x}_1$;
- finally, we compute $\mathbf{Y} = \mathbf{x}_1 - \mathbf{x}_2$.

Here, $\mathbf{x}_2 = [0, 1] \cdot [0, 1] = [\min(0 \cdot 0, 0 \cdot 1, 1 \cdot 0, 1 \cdot 1), \max(0 \cdot 0, 0 \cdot 1, 1 \cdot 0, 1 \cdot 1) = [0, 1]$, and so $\mathbf{Y} = [0, 1] - [0, 1] = [0 - 1, 1 - 0] = [-1, 1]$.

The resulting interval is the enclosure for the actual range $[0, 0.25]$ but it is much wider than this range. In interval computations, we say that this enclosure has *excess width*.

Reason for excess width. On the above example, it is easy to see why we have excess width. The range $[0, 1]$ for x_2 is actually exact. However, when we compute the range for $\mathbf{y}$ as the difference $\mathbf{x}_1 - \mathbf{x}_2$, we use the general interval computations formulas which assume that x_1 and x_2 can independently take any values from the corresponding intervals $\mathbf{x}_1$ and $\mathbf{x}_2$ – i.e., that all pairs $(x_1, x_2) \in \mathbf{x}_1 \times \mathbf{x}_2$ are possible. In reality, $x_2 = x_1^2$, so only the pairs with $x_2 = x_1^2$ are possible.

Interval computations go beyond straightforward technique. People who are vaguely familiar with interval computations sometimes erroneously assume that the above straightforward ("naive") techniques is all there is in interval computations. In conference presentations (and even in published papers), one often encounters a statement: "I tried interval computations, and it did not work". What this statement usually means is that they tried the above straightforward approach and – not surprisingly – it did not work well.

In reality, interval computations is *not a single algorithm*, it is a *problem* for which many different techniques exist. Let us now describe some of such techniques.

Centered form. One of such techniques is the centered form technique. This technique is based on the same Taylor series expansion ideas as linearization. We start by representing each interval $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ in the form $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$, where $\widetilde{x}_i = (\underline{x}_i + \overline{x}_i)/2$ is the midpoint of the interval $\mathbf{x}_i$ and $\Delta_i = (\overline{x}_i - \underline{x}_i)/2$ is the half-width of this interval.

After that, we use the Taylor expansion. In linearization, we simply ignored quadratic and higher order terms. Here, instead, we use the Taylor form with a remainder term. Specifically, the centered form is based on the formula

$$f(x_1, \ldots, x_n) = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\eta_1, \ldots, \eta_n) \cdot (x_i - \widetilde{x}_i),$$

where each η_i is some value from the interval $\mathbf{x}_i$.

Since $\eta_i \in \mathbf{x}_i$, the value of the i-th derivative belongs to the interval range of this derivative on these intervals. We also known that $x_i - \widetilde{x}_i \in [-\Delta_i, \Delta_i]$. Thus, we can conclude that

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \cdot [-\Delta_i, \Delta_i].$$

To compute the ranges of the partial derivatives, we can use straightforward interval computations.

Example. Let us illustrate this method on the above example of estimating the range of the function $f(x_1) = x_1 - x_1^2$ over the interval $[0, 1]$. For this interval, the midpoint is $\widetilde{x}_1 = 0.5$; at this midpoint, $f(\widetilde{x}_1) = 0.25$. The half-width is $\Delta_1 = 0.5$. The only partial derivative here is $\dfrac{\partial f}{\partial x_1} = 1 - 2x_1$, its range on $[0, 1]$ is equal to $1 - 2 \cdot [0, 1] = [-1, 1]$. Thus, we get the following enclosure for the desired range $\mathbf{y}$:

$$\mathbf{y} \subseteq \mathbf{Y} = 0.25 + [-1, 1] \cdot [-0.5, 0.5] = 0.25 + [-0.5, 0.5] = [-0.25, 0.75].$$

This enclosure is narrower than the "naive" estimate $[-1, 1]$, but it still contains excess width.

How can we get better estimates? In the centered form, we, in effect, ignored quadratic and higher order terms, i.e., terms of the type

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \cdot \Delta x_i \cdot \Delta x_j.$$

When the estimate is not accurate enough, it means that this ignored term is too large. There are two ways to reduce the size of the ignored term:

- we can try to decrease this quadratic term, or
- we can try to explicitly include higher order terms in the Taylor expansion formula, so that the remainder term will be proportional to say Δx_i^3 and thus, be much smaller.

Let us describe these two ideas in detail.

First idea: bisection. Let us first describe the situation in which we try to minimize the second-order remainder term. In the above expression for this term, we cannot change the second derivative. The only thing we can decrease is the difference $\Delta x_i = x_i - \widetilde{x}_i$ between the actual value and the midpoint. This value is bounded by the half-width Δ_i of the box. So, to decrease this value, we can subdivide the original box into several narrower subboxes. Usually, we divide into two subboxes, so this subdivision is called *bisection*.

The range over the whole box is equal to the union of the ranges over all the subboxes. The widths of each subbox are smaller, so we get smaller Δx_i and hopefully, more accurate estimates for ranges over each of this subbox. Then, we take the union of the ranges over subboxes.

Example. Let us illustrate this idea on the above $x_1 - x_1^2$ example. In this example, we divide the original interval $[0, 1]$ into two subintervals $[0, 0.5]$ and $[0.5, 1]$. For both intervals, $\Delta x_1 = 0.25$.

In the first subinterval, the midpoint is $\widetilde{x}_1 = 0.25$, so $f(\widetilde{x}_1) = 0.25 - 0.0625 = 0.1875$. The range of the derivative is equal to $1 - 2 \cdot [0, 0.5] = 1 - [0, 1] = [0, 1]$, hence we get an enclosure $0.1875 + [0, 1] \cdot [-0.25, 0.25] = [-0.0625, 0.4375]$.

For the second interval, $\widetilde{x}_1 = 0.75$, $f(0.75) = 0.1875$, the range of the derivative is $1 - 2 \cdot [0.5, 1] = [-1, 0]$, hence we get an enclosure

$$0.1875 + [-1, 0] \cdot [-0.25, 0.25] = [-0.0625, 0.4375].$$

The union of these two enclosures is the same interval $[-0.0625, 0.4375]$. This enclosure is much more accurate than before.

Bisection: general comment. The more subboxes we consider, the smaller Δx_i and thus, the more accurate the corresponding enclosures. However, once we have more boxes, we need to spend more time processing these boxes. Thus, we have a trade-off between computation time and accuracy: the more computation time we allow, the more accurate estimates we will be able to compute.

Additional idea: monotonicity checking. If the function $f(x_1, \ldots, x_n)$ is monotonic over the original box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$, then we can easily compute its exact range. Since we used the centered form for the original box, this probably means that on that box, the function is not monotonic: for example, with respect to x_1, it may be increasing at some points in this box, and decreasing at other points.

However, as we divide the original box into smaller subboxes, it is quite possible that at least some of these subboxes will be outside the areas where the derivatives are 0 and thus, the function $f(x_1, \ldots, x_n)$ will be monotonic. So, after we subdivide the box into subboxes, we should first check monotonicity on each of these subboxes – and if the function is monotonic, we can easily compute its range.

In calculus terms, a function is increasing with respect to x_i if its partial derivative $\dfrac{\partial f}{\partial x_i}$ is non-negative everywhere on this subbox. Thus, to check monotonicity, we should find the range $[\underline{y}_i, \overline{y}_i]$ of this derivative (we need to do it anyway to compute the centered form expression):

- if $\underline{y}_i \geq 0$, this means that the derivative is everywhere non-negative and thus, the function f is increasing in x_i;
- if $\overline{y}_i \leq 0$, this means that the derivative is everywhere non-positive and thus, the function f is decreasing in x_i.

If $\underline{y}_i < 0 < \overline{y}_i$, then we have to use the centered form.

If the function is monotonic (e.g., increasing) only with respect to some of the variables x_i, then

- to compute $\overline{y}$, it is sufficient to consider only the value $x_i = \overline{x}_i$, and
- to compute $\underline{y}$, it is sufficient to consider only the value $x_i = \underline{x}_i$.

For such subboxes, we reduce the original problem to two problems with fewer variables, problems which are thus easier to solve.

Example. For the example $f(x_1) = x_1 - x_1^2$, the partial derivative is equal to $1 - 2 \cdot x_1$.

On the first subbox $[0, 0.5]$, the range of this derivative is $1 - 2 \cdot [0, 0.5] = [0, 1]$. Thus, the derivative is always non-negative, the function is increasing on this subbox, and its range on this subbox is equal to $[f(0), f(0.5)] = [0, 0.25]$.

On the second subbox $[0.5, 1]$, the range of the derivative is $1 - 2 \cdot [0.5, 1] = [-1, 0]$. Thus, the derivative is always non-positive, the function is decreasing on this subbox, and its range on this subbox is equal to $[f(1), f(0.5)] = [0, 0.25]$. The union of these two ranges is $[0, 0.25]$ – the exact range.

Comment. We got the exact range because of the simplicity of our example, in which the extreme point 0.5 of the function $f(x_1) = x_1 - x_1^2$ is exactly in the middle of the interval $[0, 1]$. Thus, when we divided the box in two, both

subboxes have the monotonicity property. In the general case, the extremal point will be inside one of the subboxes, so we will have excess width.

General Taylor techniques. As we have mentioned, another way to get more accurate estimates is to use so-called *Taylor techniques*, i.e., to explicitly consider second-order and higher-order terms in the Taylor expansion; see, e.g., [3, 42, 48] and references therein.

Let us illustrate the main ideas of Taylor analysis on the case when we allow second order terms. In this case, the formula with a remainder takes the form

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

$$\frac{1}{2}\cdot\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial^2 f}{\partial x_i \partial x_j}(\eta_1,\ldots,\eta_n)\cdot(x_i-\widetilde{x}_i)\cdot(x_j-\widetilde{x}_j).$$

Thus, we get the enclosure

$$f(\mathbf{x}_1,\ldots,\mathbf{x}_n) \subseteq f(\widetilde{x}_1,\ldots,\widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\widetilde{x}_1,\ldots,\widetilde{x}_n)\cdot[-\Delta_i,\Delta_i] +$$

$$\frac{1}{2}\cdot\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}_1,\ldots,\mathbf{x}_n)\cdot[-\Delta_i,\Delta_i]\cdot[-\Delta_j,\Delta_j].$$

Example. Let us illustrate this idea on the above example of $f(x_1) = x_1 - x_1^2$. Here, $\widetilde{x}_1 = 0.5$, so $f(\widetilde{x}_1) = 0.25$ and $\dfrac{\partial f}{\partial x_1}(\widetilde{x}_1) = 1 - 2\cdot 0.5 = 0$. The second derivative is equal to -2, so the Taylor estimate takes the form $\mathbf{Y} = 0.25 - [-0.5, 0.5]^2$.

Strictly speaking, if we interpret Δx_1^2 as $\Delta x_1 \cdot \Delta x_1$ and use the formulas of interval multiplication, we get the interval $[-0.5, 0.5]\cdot[-0.5, 0.5] = [-0.25, 0.25]$ and thus, the range $\mathbf{Y} = 0.25 - [-0.25, 0.25] = [0, 0.5]$ with excess width. However, we can view x^2 as a special function, for which the range over $[-0.5, 0.5]$ is known to be $[0, 0.25]$. In this case, the above enclosure $0.25 - [0, 0.25] = [0, 0.25]$ is actually the exact range.

Taylor methods: general comment. The more terms we consider in the Taylor expansion, the smaller the remainder term and thus, the more accurate the corresponding enclosures. However, once we have more terms, we need to spend more time computing these terms. Thus, for Taylor methods, we also have a trade-off between computation time and accuracy: the more computation time we allow, the more accurate estimates we will be able to compute.

An alternative version of affine and Taylor arithmetic. The main idea of Taylor methods is to approximate the given function $f(x_1, \ldots, x_n)$ by a polynomial of a small order plus an interval remainder term.

In these terms, straightforward interval computations can be viewed as 0-th order Taylor methods in which all we have is the corresponding interval (or, equivalently, the constant term plus the remainder interval). To compute this interval, we repeated the computation of f step by step, replacing operations with numbers by operations with intervals.

We can do the same for higher-order Taylor expansions as well. Let us illustrate how this can be done for the first order Taylor terms. We start with the expressions $x_i = \widetilde{x}_i - \Delta x_i$. Then, at each step, we keep a term of the type $a = \widetilde{a} + \sum\limits_{i=1}^{n} a_i \cdot \Delta x_i + \mathbf{a}$. (To be more precise, the keep the coefficients $\widetilde{a}$ and a_i and the interval $\mathbf{a}$.)

Addition and subtraction of such terms are straightforward:

$$(\widetilde{a}+\sum_{i=1}^{n} a_i\cdot\Delta x_i+\mathbf{a})+(\widetilde{b}+\sum_{i=1}^{n} b_i\cdot\Delta x_i+\mathbf{b}) = (\widetilde{a}+\widetilde{b})+\sum_{i=1}^{n}(a_i+b_i)\cdot\Delta x_i+(\mathbf{a}+\mathbf{b});$$

$$(\widetilde{a}+\sum_{i=1}^{n} a_i\cdot\Delta x_i+\mathbf{a})-(\widetilde{b}+\sum_{i=1}^{n} b_i\cdot\Delta x_i+\mathbf{b}) = (\widetilde{a}-\widetilde{b})+\sum_{i=1}^{n}(a_i-b_i)\cdot\Delta x_i+(\mathbf{a}-\mathbf{b}).$$

For multiplication, we add terms proportional to $\Delta x_i \cdot \Delta x_j$ to the interval part:

$$(\widetilde{a}+\sum_{i=1}^{n} a_i\cdot\Delta x_i+\mathbf{a})\cdot(\widetilde{b}+\sum_{i=1}^{n} b_i\cdot\Delta x_i+\mathbf{b}) = (\widetilde{a}\cdot\widetilde{b})+\sum_{i=1}^{n}(\widetilde{a}\cdot b_i+\widetilde{b}\cdot a_i)\cdot\Delta x_i+$$

$$(\widetilde{a}\cdot\mathbf{b}+\widetilde{b}\cdot\mathbf{a}+\sum_{i=1}^{n} a_i\cdot b_i\cdot[0,\Delta_i^2]+\sum_{i=1}^{n}\sum_{j\neq i} a_i\cdot b_j\cdot[-\Delta_i,\Delta_i]\cdot[\Delta_j\cdot\Delta_j]).$$

At the end, we get an expression of the above type for the desired quantity y: $y = \widetilde{y} + \sum\limits_{i=1}^{n} y_i \cdot \Delta x_i + \mathbf{y}$. We already know how to compute the range of a linear function, so we get the following enclosure for the final range: $\mathbf{Y} = \widetilde{y} + [-\Delta, \Delta] + \mathbf{y}$, where $\Delta = \sum\limits_{i=1}^{n} |y_i| \cdot \Delta_i$.

Example. For $f(x_1) = x_1 - x_1^2$, we first compute $x_2 = x_1^2$ and then $y = x_1 - x_2$. We start with the interval $\mathbf{x}_1 = \widetilde{x}_1 - \Delta x_1 = 0.5 + (-1)\cdot\Delta_1 + [0,0]$.

On the next step, we compute the square of this expression. This square is equal to $0.25 - \Delta x_1 + \Delta x_1^2$. Since $\Delta x_1 \in [-0.5, 0.5]$, we conclude that $\Delta x_1^2 \in [0, 0.25]$ and thus, that $x_2 = 0.25 + (-1)\cdot\Delta x_1 + [0, 0.25]$.

For $y = x_1 - x_2$, we now have

$$y = (0.5 - 0.25) + ((-1) - (-1))\cdot\Delta x_1 + ([0,0] - [0, 0.25]) =$$

$$0.25 + [-0.25, 0] = [0, 0.25].$$

This is actually the exact range for the desired function $f(x_1)$.

10 Applications of Interval Computations

General overview. Interval computations have been used in almost all areas of science and engineering in which we need guaranteed results, ranging from space exploration to chemical engineering to robotics to supercollider design. Many applications are listed in [20, 23]; some other described in numerous books and articles (many of which are cited in the interval computations website [19]). Many important applications are described in the interval-related chapters of this handbook.

Most of these applications use special software tools and packages specifically designed for interval computations; see, e.g, [15]; a reasonably current list of such tools is available from the interval website [19].

Applications to control. One of the areas where guaranteed bounds are important is the area of control. Robust control methods, i.e., methods which stabilize a system (known with interval uncertainty) for all possible values of the parameters from the corresponding intervals, are presented, e.g., in [2, 4].

Applications to optimization: practical need. As we have mentioned earlier, one of the main objectives of engineering s to find the alternative which is the best (in some reasonable sense).

In many real-life situations, we have a precise description of what is the best, i.e., we have an *objective functions* which assigns to each alternative $x = (x_1, \ldots, x_n)$ a value $F(x_1, \ldots, x_n)$ characterizing the overall quality of this alternative, and our goal is to find the alternative for which this quality metric attains the largest possible value.

In mathematical terms, we want to find the maximum M of a function $F(x_1, \ldots, x_n)$ on a given set S, and we are also interested in finding out where exactly this maximum is attained.

Applications to optimization: idea. The main idea of using interval computations in optimization is as follows. If we compute the value of F at several points from S and then take the maximum m of the computed values, then we can be sure that the maximum M over *all* points from S is not smaller than m: $m \le M$. Thus, if we divide the original set into subboxes, and on one of these subboxes the range $[\underline{y}, \overline{y}]$ for f is $< m$, then we can guarantee that the desired maximum does not occur on this subbox. Thus, this subbox can be excluded from the future search.

This idea is implemented as the following branch-and-bound algorithm.

Applications to optimization: simple algorithm. For simplicity, let us describe this algorithm for the case when the original set S is a box.

On each step of this algorithm, we have:

- a collection of subboxes,

- interval enclosures for the range of F on each subbox, and
- a current lower bound m for the desired maximum M.

We start with the original box; as the initial estimate m, we take, e.g., the value of F at the midpoint of the original box.

On each step, we subdivide one or several of the existing subboxes into several new ones. For each new subbox, we compute the value of F at its midpoint; then, as a new bound m, we take the maximum of the old bound and of these new results. For each new subbox, we use interval computations to compute the enclosure $[\underline{Y}, \overline{Y}]$ for the range. If $\overline{Y} < m$, then the corresponding subbox is dismissed.

This procedure is repeated until all the subboxes concentrate in a small vicinity of a single point (or of a few points); this point is the desired maximum.

Example. Let us show how this algorithm will find the maximum of a function $F(x_1) = x_1 - x_1^2$ on the interval $[0, 1]$. We start with the midpoint value $m = 0.5 - 0.5^2 = 0.25$, so we know that $M \geq 0.25$.

For simplicity, let us use the centered form to compute the range of F. On the entire interval, as we have shown earlier, we get the enclosure $[-0.25, 0.75]$.

Let us now subdivide this box. In the computer, all the numbers are binary, so the easiest division is by 2, and the easiest subdivision of a box is bisection (division of one of the intervals into two equal subintervals). Since we use the decimal system, for us, it is easier to divide by 5, so let us divide the original box into 5 subboxes $[0, 0.2]$, $[0, 2, 0.4]$, $\ldots$, $[0.8, 1]$. All the values at midpoints are $\leq m$, so the new value of m is still 0.25.

The enclosure over $[0, 0.2]$ is

$$(0.1 - 0.1^2) + (1 - 2 \cdot [0, 0.2]) \cdot [-0.1, 0.1] = 0.09 - [-0.1, 0.1] = [-0.01, 0.19].$$

Since $0.19 < 0.25$, this subbox is dismissed. Similarly, the subbox $[0.8, 1]$ will be dismissed. For the box $[0.2, 0.4]$, the enclosure is

$$(0.3 - 0.3^2) + (1 - 2 \cdot [0.2, 0.4]) \cdot [-0.1, 0.1] = 0.21 - [-0.06, 0.06] = [0.15, 0.27].$$

Since $m = 0.25 < 0.27$, this subbox is not dismissed. Similarly, we keep boxes $[0.4, 0.6]$ and $[0.6, 0.8]$ – the total of three.

On the next step, we subdivide each of these three boxes, dismiss some more boxes, etc. After a while, the remaining subboxes will concentrate around the actual maximum point $x = 0.5$.

Applications to optimization: more sophisticated algorithms. Interval techniques are actually used in the best optimization packages which produce guaranteed results. Of course, these interval methods go beyond the above simple branch-and-bound techniques: e.g., they check for monotonicity to weed out subboxes where local maxima are only possible at the endpoints, they look for solutions to the equation $\dfrac{\partial f}{\partial x_i} = 0$, etc.; see, e.g., [17, 22].

Optimization: granularity helps. In the above text, we assumed that we know the exact value of the objective function $F(x)$ for each alternative x. In reality, we often only have approximate predictions of this value $F(x)$, with some accuracy ε. In such situations, it does not make sense to waste time and optimize the function beyond this accuracy.

For example, in the simplest interval-based optimization algorithm, at each stage, we not only get the lower bound m for the desired maximum. We can also compute the upper bound $\overline{M}$ – which can be found as the largest of the endpoints $\overline{Y}$ of all subbox enclosures. Thus, $m \le M = \max F(x) \le \overline{M}$. Once we get $|\overline{M} - m| \le \varepsilon$, we can guarantee that every value from the interval $[m, \overline{M}]$ is ε-close to M. Thus, we can produce any alternative from any of the remaining subboxes as a good enough solution.

This simple idea can often drastically decrease computation time.

Applications to mathematics. In addition to practical applications, there have been several examples when interval computations help in solving long-standing mathematical open problems.

The first such problem was the *double bubble* problem. It is well known that of all sets with a given volume, a ball has the smallest surface area. What if we consider two sets of equal volumes, and count both the area of the outside boundaries and of the boundary between the two sets? It has been conjectured that the smallest overall area is attained for the "double bubble": we take two spheres, use a plane to cut off a top of one of them, do a similar cut with the second sphere, and bring them together at the cut (so that the boundary between them in the disk). The actual proof required proving that for this configuration, the area is indeed larger than for all possible other configurations. This proof was done by Haas et al. in [13] who computed an interval enclosure $[\underline{Y}, \overline{Y}]$ for all other configurations, and showed that $\overline{Y}$ is smaller than the area Y_0 corresponding to the double bubble.

Another well-known example is the *Kepler's conjecture*. Kepler conjectured that the standard way of stacking cannonballs (or oranges), when we place some balls on a planar grid, place the next layer in the holes between them, etc., has the largest possible density. This hypothesis was proved in 1998 by T. Hales, who, in particular, used interval computations to prove that many other placements lead to a smaller density [14].

Beyond interval computations, towards general granular computing. In the previous text, we consider situations when we have either probabilistic, or interval, or fuzzy uncertainty.

In practice, we often have all kinds of uncertainty. For example, we may have *partial* information about probabilities: e.g., instead of the cumulative distribution function (cdf) $F(x) \stackrel{\text{def}}{=} \text{Prob}(\xi \le x)$, we only know *bounds* $\underline{F}(x)$ and $\overline{F}(x)$ on this cdf. In this case, all we know about the probability distribution is that the actual (unknown) cdf $F(x)$ belongs to the corresponding interval $[\underline{F}(x), \overline{F}(x)]$. This probability-related interval is called a *probability box*, or *p-*

box, for short. In data processing, once we know the p-boxes corresponding to the auxiliary quantities x_i, we need to find the p-box corresponding to the desired quantity $y = f(x_1, \ldots, x_n)$; such methods are described, e.g., in [7] (see also [29, 30]).

Similarly, in fuzzy logic, we considered the case when for every property A and for every value x, we know the exact value of the degree $\mu_A(x)$ to which x satisfies the property. In reality, as we have mentioned, experts can only produce interval of possible values of their degrees. As a result, *interval-valued* fuzzy sets more adequately describe expert opinions and thus, often, lead to better applications; see, e.g., [32] as well as the corresponding chapters from this handbook.

Overall, we need a *combination* of all these types of tools, a combination which is able to handle all kinds of granules, a combination termed *granular computing* (see, e.g., [46]).

Our hopes. One of the main objectives of this handbook is that interested readers learn the techniques corresponding to different parts of granular computing – and when necessary, combine them. We hope that this handbook will further enhance the field of granular computing.

Acknowledgments. This work was supported in part by NSF grant EAR-0225670, by Texas Department of Transportation grant No. 0-5453, and by the Japan Advanced Institute of Science and Technology (JAIST) International Joint Research Grant 2006-08. This work was partly done during the author's visit to the Max Planck Institut für Mathematik.

References

[1] Archimedes, On the measurement of the circle, In: T. L. Heath (ed.), *The Works of Archimedes*, Cambridge University Press, Cambridge, 1897; Dover edition, 1953.

[2] B.R. Barmish, *New Tools for Robustness of Linear Systems*, McMillan, New York, 1994.

[3] M. Berz and G. Hoffstätter, "Computation and Application of Taylor Polynomials with Interval Remainder Bounds", *Reliable Computing* 4(1) (1998) 83–97.

[4] S.P. Bhattacharyya, H. Chapellat, and L. Keel, *Robust Control: The Parametric Approach*, Prentice-Hall, 1995.

[5] B. Chokr and V. Kreinovich. "How far are we from the complete knowledge: complexity of knowledge acquisition in Dempster-Shafer approach." In R. R. Yager, J. Kacprzyk, and M. Pedrizzi (Eds.), *Advances in the Dempster-Shafer Theory of Evidence*, Wiley, N.Y., 1994, pp. 555–576.

[6] P. S. Dwyer, *Linear Computations*, J. Wiley, N.Y., 1951.

[7] S. Ferson, *Risk assessment with uncertainty numbers: Risk Calc*, CRC Press, Boca Raton, Florida, 2002.

[8] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles, Computing Variance for Interval Data is NP-Hard, *ACM SIGACT News* 33(2) (2002) 108–118.

[9] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles, Exact Bounds on Finite Populations of Interval Data, *Reliable Computing*, 11(3) (2005) 207–233.

[10] A.A. Gaganov, *Computational complexity of the range of the polynomial in several variables*, Leningrad University, Math. Department, M.S. Thesis, 1981 (in Russian).

[11] A.A. Gaganov, Computational complexity of the range of the polynomial in several variables, *Cybernetics* (1985) 418–421.

[12] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, W. F. Freeman, San Francisco, 1979.

[13] J. Haas, M. Hutchings, and R. Schlafy, The Double Bubble Conjecture, *Electron. Res. Announc. Amer. Math. Soc.* 1 (1995) 98–102.

[14] T.C. Hales, A proof of the Kepler conjecture, *Ann. Math.* 162 (2005) 1065–1185.

[15] R. Hammer, M. Hocks, U. Kulisch, D. Ratz, *Numerical toolbox for verified computing. I. Basic numerical problems*, Springer Verlag, Heidelberg, N.Y., 1993.

[16] C. Hamzo and V. Kreinovich, On Average Bit Complexity of Interval Arithmetic, *Bulletin of the European Association for Theoretical Computer Science (EATCS)* 68 (1999) 153–156.

[17] E.R. Hansen and G.W. Walster, *Global Optimization Using Internal Analysis*, MIT Press, Cambridge, Massachusetts, 2004.

[18] G. Heindl, *An improved algorithm for computing the product of two machine intervals*, Interner Bericht IAGMPI- 9304, Fachbereich Mathematik, Gesamthochschule Wuppertal, 1993.

[19] Interval computations website http://www.cs.utep.edu/interval-comp

[20] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied interval analysis: with examples in parameter and state estimation, robust control and robotics*, Springer Verlag, London, 2001.

[21] E.T. Jaynes and G.L. Bretthorst, *Probability Theory: the logic of science*, Cambridge University Press, Cambridge, UK, 2003.

[22] R.B. Kearfott, *Rigorous global search: continuous problems*, Kluwer, Dordrecht, 1996.

[23] R.B. Kearfott and V. Kreinovich (eds.), *Applications of Interval Computations*, Kluwer, Dordrecht, 1996.

[24] G.J. Klir, *Uncertainty and Information: Foundations of Generalized Information Theory*, J. Wiley, Hoboken, New Jersey, 2005.

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

[26] V. Kreinovich, J. Beck, C. Ferregut, A. Sanchez, G.R. Keller, M. Averill, and S.A. Starks, Monte-Carlo-type techniques for processing interval uncertainty, and their potential engineering applications, *Reliable Computing* 13(1) (2007) 25–69.

[27] V. Kreinovich and S. Ferson, A New Cauchy-Based Black-Box Technique for Uncertainty in Risk Analysis, *Reliability Engineering and Systems Safety* 85(1–3) (2004) 267–279.

[28] V. Kreinovich, A. Lakeyev, J. Rohn, P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1997.

[29] V. Kreinovich, L. Longpré, S. A. Starks, G. Xiang, J. Beck, R. Kandathi, A. Nayak, S. Ferson, and J. Hajagos, "Interval Versions of Statistical Techniques, with Applications to Environmental Analysis, Bioinformatics, and Privacy in Statistical Databases", *Journal of Computational and Applied Mathematics*, 2007, Vol. 199, No. 2, pp. 418–423.

[30] V. Kreinovich, G. Xiang, S. A. Starks, L. Longpré, M. Ceberio, R. Araiza, J. Beck, R. Kandathi, A. Nayak, R. Torres, and J. Hajagos, Towards combining probabilistic and interval uncertainty in engineering calculations: algorithms for computing statistics under interval uncertainty, and their computational complexity, *Reliable Computing* 12(6) (2006) 471-501.

[31] S. Markov and K. Okumura, The Contribution of T. Sunaga to Interval Analysis and Reliable Computing, In: T. Csendes (ed.), *Developements in Reliable Computing*, Kluwer, Dordrecht, 1999, pp. 167–188.

[32] J.M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice-Hall, 2001.

[33] R.E. Moore, *Automatic error analysis in digital computation*, Technical Report, Space Div. Report LMSD84821, Lockheed Missiles and Space Co., 1959.

[34] R.E. Moore, The automatic analysis and control of error in digital computing based on the use of interval numbers. In: L.B. Rall (ed.), *Error in Digital Computation*, Wiley, NY, 1965, Vol. I, pp. 61–130.

[35] R.E. Moore, Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solutions of ordinary differential equations. In: L.B. Rall (ed.), *Error in Digital Computation*, Wiley, NY, 1965, Vol. II, pp. 103–140.

[36] R.E. Moore, W. Strother, and C.T. Yang, *Interval integrals*, Technical Report Space Div. Report LMSD703073, Lockheed Missiles and Space Co., 1960.

[37] R.E. Moore and C.T. Yang, *Interval analysis I*, Technical Report Space Div. Report LMSD285875, Lockheed Missiles and Space Co., 1959.

[38] R.E. Moore, *Interval Arithmetic and Automatic Error Analysis in Digital Computing*, Ph.D. Dissertation, Department of Mathematics, Stanford University, Stanford, California, 1962. Published as Applied Mathematics and Statistics Laboratories Technical Report No. 25.

[39] R. E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs N. J., 1966.

[40] R. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[41] R.E. Moore, The dawning, *Reliable Computing* 5 (1999) 423–424.

[42] A. Neumaier, Taylor forms – use and limits, *Reliable Computing* 9 (2002) 43–79.

[43] H.T. Nguyen, A note on the extension principle for fuzzy sets, *J. Math. Anal. and Appl.* 64 (1978) 359–380.

[44] H.T. Nguyen and E.A. Walker, *A first course in fuzzy logic*, CRC Press, Boca Raton, Florida, 2005.

[45] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[46] W. Pedrycz (ed.), *Granular Computing: An Emerging Paradigm*, Springer-Verlag, 2001.

[47] S. G. Rabinovich, *Measurement Errors and Uncertainty. Theory and Practice*, Springer Verlag, Berlin, 2005.

[48] N. Revol, K. Makino, and M. Berz, Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in COSY, *J. Log. Algebr. Program.* 64(1) (2005) 135–154.

[49] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*, Springer-Verlag, New York, 2004.

[50] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, Boca Raton, Florida, 2004.

[51] T. Sunaga, "Theory of interval algebra and its application to numerical analysis", In: *RAAG Memoirs, Ggujutsu Bunken Fukuy-kai*, Tokyo, 1958, Vol. 2, 1958, pp. 29–46 (547–564).

[52] S. Vavasis, *Nonlinear Optimization: Complexity Issues*, Oxford University Press, 1991.

[53] H. M. Wadswort, (ed.), *Handbook of statistical methods for engineers and scientists*, McGraw-Hill Publishing Co., New York, 1990.

[54] M. Warmus, Calculus of Approximations, *Bulletin de l'Academie Polonaise de Sciences* 4(5) (1956) 253–257.

[55] M. Warmus, Approximations and Inequalities in the Calculus of Approximations. Classification of Approximate Numbers, *Bull. Acad. Polon. Sci., Ser. Sci. Math. Astronom. Phys.* 9 (1961) 241–245.

[56] W.H. Young, Sull due funzioni a piu valori constituite dai limiti d'una funzione di variable reale a destra ed a sinistra di ciascun punto, *Rendiconti Academia di Lincei, Classes di Scienza Fiziche*, 17(5) (1908) 582–587.

[57] R.C. Young, The algebra of multi-valued quantities, *Mathematische Annalen* 104 (1931) 260–290.