How To Take Into Account Dependence Between the Inputs: From Interval Computations to Constraint-Related Set Computations, with Potential Applications to Nuclear Safety, Bio- and Geosciences

Martine Ceberio[1], Scott Ferson[2], Vladik Kreinovich[1],
Sanjeev Chopra[1,3], Gang Xiang[1], Adrian Murguia[1,4],
and Jorge Santillan[1]

[1]Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
mceberio@cs.utep.edu, vladik@utep.edu,
gxiang@utep.edu
[2]Applied Biomathematics
100 North Country Road
Setauket, New York 11733, USA
scott@ramas.com
[3]Lexmark International, Inc.
740 New Circle Road NW
Lexington, KY 40550, USA
sachopra@gmail.com
[4]XIMIS, Inc.
6006 N. Mesa, Suite 709
El Paso, TX 79912, USA

Abstract

In many real-life situations, in addition to knowing the intervals $\mathbf{x}_i$ of possible values of each variable x_i, we also know additional restrictions on the possible combinations of x_i; in this case, the set $\mathbf{x}$ of possible values of $x = (x_1, \ldots, x_n)$ is a proper subset of the original box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$. In this paper, we show how to take into account this dependence between the inputs when computing the range of a function $f(x_1, \ldots, x_n)$.

Keywords: constraints, interval computations, dependence between the inputs

1 Introduction

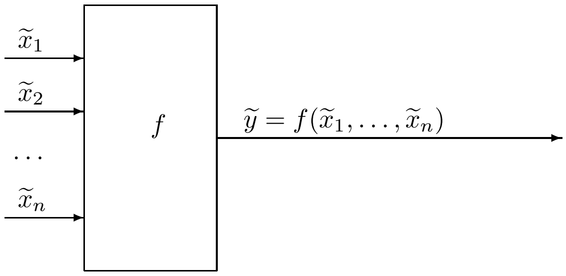
1.1 General Problem of Data Processing under Uncertainty

In many real-life situations, there exist quantities which are difficult (or even impossible) to measure directly: e.g., the amount of oil in an oil field, or the temperature inside a reactor. Since we cannot measure the corresponding quantity *directly*, we can measure it *indirectly:* by measuring the values of easier-to-measure quantities $x_1, \ldots, x_n$ which are related to the desired quantity y by a known dependence $y = f(x_1, \ldots, x_n)$.

The resulting indirect measurement consists of the following:

- first, we measure the quantities $x_1, \ldots, x_n$, and
- then, we apply the function f to the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of these measurements.

The resulting value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is our estimate for the desired quantity y.



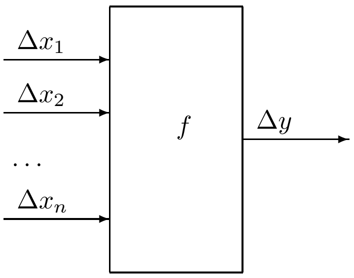
If measurements were absolutely accurate, then we would be able to get the exact values of x_i, and thus, compute the exact value of the desired quantity y. In reality, however, measurements are never 100% accurate; hence, the result $\widetilde{x}_i$ of i-th measurement is, in general, different from the actual value x_i of the corresponding quantity. In other words, we have a non-zero *measurement error* $\Delta x_i \neq 0$. Hence, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of applying the function f to the measured values is, in general, different from the actual (unknown) value y of the desired quantity – i.e., from the result $y = f(x_1, \ldots, y_n)$ of applying the function f to the actual (unknown) values of the quantities x_i.

A natural question is: what can we say about the error $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$ of indirect measurement?

Comment. In some real-life situations, we also do not know the exact function f, and this uncertainty in f needs to be added to the uncertainty caused by errors of direct measurements $\Delta x_i \neq 0$. In this paper, for simplicity, we consider only the cases when we know the exact expressions for the function f.

1.2 Probabilistic and Interval Uncertainty

The error Δy of indirect measurement is caused by the measurement errors Δx_i of direct measurements. Thus, to deduce the desired information about Δy, we must use the known information about Δx_i.



Traditionally, in engineering and science, we assume that we know the joint probability distribution for Δx_i. Usually, it is assumed that these measurement errors are independent and normally distributed, with 0 mean and known standard deviations; however, there are are also known ways of handling possible dependence and non-Gaussian (non-normal) distributions.

In many real-life situations, we do know these distributions: they come from the process of comparing the currently used measuring instruments (MI) with much more accurate "standard" MIs used in the national or international standards centers. Specifically, we repeatedly measure the same quantity by our MI and by the standard MI. The standard MI is, by definition, much more accurate than our MI, i.e., $|x_i^{\text{stand}} - x_i| \ll |\widetilde{x}_i - x_i|$. Hence, the difference $\widetilde{x}_i - x_i^{\text{stand}}$ between the results of these two measurements is very close to the actual (unknown) measurement error $\Delta x_i = \widetilde{x}_i - x_i$. Thus, by analyzing the sample of such differences, we can infer the probability distribution for the measurement error Δx_i.

This "calibration" of measuring instruments is indeed often performed. However, there are two important classes of situations where this calibration is not done.

The first such class is situations from *fundamental science.* If we are interested in the accuracy of a typical over-the-counter voltmeter, then it is possible to design a more accurate voltmeter and used this more accurate MI to calibrate our MI. However, when we are trying to analyze the accuracy of, say, measurements performed by using the newest particle super-collider, it would nice to have a much more accurate instrument available for calibration, but the existing instrument is the best we have. Similarly, to analyze the accuracy of measurements made by using the Hubble telescope, it would be nice to have a much more accurate instrument floating nearby, but the Hubble is the best we have so far.

Another class of situations is related to *manufacturing.* In manufacturing, in principle, it is possible to calibrate all the sensors. However, a detailed individual calibration of each sensor often costs orders of magnitude more than the sensors themselves. As a result, manufacturers are trying to avoid detailed calibration of all the sensors, and use whatever information is available without spending a lot of money.

In such cases, we *do not* know the probability distribution of the measurement errors Δx_i. What *do* we know in such situations? For sure, the manufacturer of the measuring instrument must supply us with an upper bound Δ_i on the (absolute value of) the measurement error $|\Delta x_i|$. Indeed, if such guaranteed bound is provided, this means that the actual value x_i of the measured quantity can be as far away as possible from the measured value $\widetilde{x}_i$. For example, we measure the current as 1 A, but the actual current current can be 1000 or 0. This is a wild guess, not a measurement. For an instrument to be called a measuring instrument, some bound has to be provided. The manufacturer *may* provide some additional information about Δx_i, but the upper bound *has* to be provided.

Once the upper bound Δ_i on $|\Delta x_i|$ is provided, then, based on the measured value $\widetilde{x}_i$, we can conclude that the actual (unknown) value x_i of the i-th quantity belongs to the interval

$$x_i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i].$$

In other words, we know the values x_i with *interval uncertainty.*

For example, if the measured current is 1.0 V and the upper bound on the measurement error is 0.1 V, then we are guaranteed that the actual (unknown) value of the current is in the interval $[1.0-0.1, 1.0+0.1] = [0.9, 1.1]$.

1.3 Interval Computations: A Problem

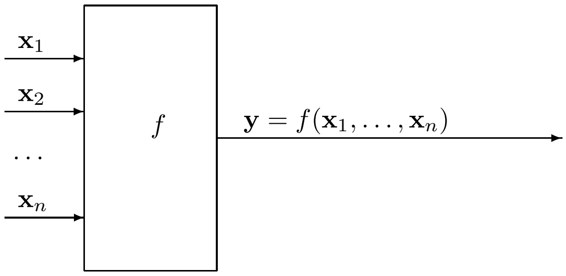
We have just mentioned that in many important real-life situations, we know x_i with interval uncertainty, i.e.:

- we know the ranges $\mathbf{x}_i$ of possible values of x_i, and
- we do not have any information about the probability of different values within these ranges.

In such situations, the only information that we can have about the desired quantity $y = f(x_1, \ldots, x_n)$ is the range of possible values of y when $x_i \in \mathbf{x}_i$. In other words, we face the following problem:

- *Given:*
  - an algorithm $y = f(x_1, \ldots, x_n)$ that transforms n real numbers x_i into a number y; and
  - n intervals $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$.
- *Compute:* the corresponding range of y:

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



The problem of computing this range is often called the *main problem* of interval computations; see, e.g., [10].

It is known that even for quadratic f, the problem of computing the exact range $\mathbf{y}$ is difficult to compute (in precise terms, NP-hard); see, e.g., [14, 20]. Crudely speaking, NP-hard means that[1] it is not possible to find an efficient algorithm that would compute the *exact* range for *all* possible problems. Since no such general algorithm is possible, to solve practical problems, we thus need to do the following:

- find classes of problems for which efficient algorithms are possible; and
- for problems outside these classes, find efficient techniques for *approximating* uncertainty of y.

This is what interval computations community has been doing for several decades.

1.4 Why Not Maximum Entropy?

From the engineering practical viewpoint, a natural question is: why not use the Maximum Entropy approach? Let us explain what this question means and how to answer it.

Our problems come from the fact that we do not know the exact probability distribution for $\Delta x = (\Delta x_1, \dots, \Delta x_n)$. In real life, this is a frequent situation: in many practical applications, it is very difficult to come up with the probabilities.

The traditional engineering approach recommends that we use probabilistic techniques. If we do not know the exact probability distribution, this means that there are many different probability distributions which are consistent with the same observations and measurements. The traditional engineering solution to this problem is to select one of these distributions – e.g., the one with the largest entropy; see, e.g., [11] for the detailed description of this Maximum Entropy (MaxEnt) approach.

For example, suppose that we have only one variable x, and all we know about the actual value of this variable is that it belongs to the interval $[\underline{x}, \overline{x}]$. Since we have no information about the relative probability of different values from this interval, there is no reason to assume that some values are more probable than the others. It is therefore reasonable to assume that all the values within this interval are equally probable, i.e., in precise terms, that we have a uniform distribution on this interval $[\underline{x}, \overline{x}]$. Not surprisingly, this is exactly what MaxEnt leads to.

In case we have several variables Δx_i and we have no information about their correlation, then we have no reason to assume that they are positively or negatively correlated; it is thus reasonable to assume that they are independent. For example, if all we know is that Δx_i belongs to the interval $[-\Delta_i, \Delta_i]$, then the only information that we have about the vector $\Delta x = (\Delta x_1, \dots, \Delta x_n)$ is that it is located in the box $[-\Delta_1, \Delta_1] \times \dots \times [-\Delta_n, \Delta_n]$. Since we have no reason to assume that some values from this box are more probable than the others, it seems reasonable to assume that all the values from the box are equally probable – i.e., in precise terms, that we have a uniform distribution on this box. One can easily see that the uniform distribution on the box means that:

- the variables Δx_i are independent, and
- each variable Δx_i is uniformly distributed in the corresponding interval.

Why should we not use this approach? Because, as we will show, this approach can sometimes seriously underestimate the error of indirect measurement. Indeed, let us consider the simplest possible case, when:

[1]unless P is equal to NP, which most computer scientists do not believe

- the desired quantity y is simply the sum of n values $x_1, \ldots, x_n$, i.e., $f(x_1, \ldots, x_n) = x_1 + \ldots + x_n$, and
- all direct measurements have the same error bound $\Delta_1 = \ldots = \Delta_n = \Delta$.

In this case, $\Delta y = \Delta x_1 + \ldots + \Delta x_n$, with $\Delta x_i \in [-\Delta_i, \Delta_i]$.

In practice, it is quite possible that all n measurement errors are caused by the same factor; in this case, it is possible that $\Delta x_1 = \ldots = \Delta x_n$ and thus, $\Delta y = n \cdot \Delta x_1$. Since the measurement error Δx_1 can be take any values from the interval $[-\Delta, \Delta]$, it is possible that $\Delta x_1 = \Delta$ and therefore, it is possible that $\Delta y = n \cdot \Delta$.

On the other hand, when we apply the MaxEnt approach to this situation, we thus assume that the values $\Delta x_i \in [-\Delta, \Delta]$ are independent identically distributed random variables uniformly distributed on the interval $[-\Delta, \Delta]$. For the uniform distribution, the mean is 0, and the variance is $\frac{1}{3} \cdot \Delta^2$. When we add independent random variables, their means and variances add up, so the sum Δy has a mean 0 and variance $V = \frac{1}{3} \cdot n \cdot \Delta^2$.

It is known that, due to the Central Limit Theorem (see, e.g., [21]), for large n, the sum Δy of n independent identically distributed random variables is almost normally distributed. Thus, within the MaxEnt approach, for large n, the measurement error Δy is (almost) normally distributed with 0 means and variance $V = \frac{1}{3} \cdot n \cdot \Delta^2$. It is also well known that for a normally distributed random variable, the probability of a value which is more than, say, 6σ away from the mean is negligibly small ($\approx 10^{-8}$). Thus, from the MaxEnt approach, we conclude that with probability $\geq 1 - 10^{-8}$ (i.e., practically, with certainty), the measurement error Δy is bounded by $6\sigma = 6 \cdot \sqrt{V} \sim \sqrt{n}$.

So, by using the MaxEnt approach, we get an error bound $\sim \sqrt{n}$, but in reality, due to possible correlations, we may have $\Delta y \sim n \gg \sqrt{n}$. Our conclusion is that using a single distribution – even the most reasonable one – can be very misleading, especially if we want guaranteed results, e.g., in high-risk application areas such as space exploration or nuclear engineering.

We therefore need to solve the original problem of interval computations.

1.5 General Approach: Interval-Type Step-by-Step Techniques

In this paper, we will modify the standard interval computation techniques. To explain the needed modification, let us recall these techniques in detail.

As we have mentioned, the main difficulty of solving the main problem of interval computations is that it is (provably) computationally difficult to compute the exact range $\mathbf{y}$ for an arbitrary function $f(x_1, \ldots, x_n)$. The solution provided by interval computations is to compute an *enclosure* $\mathbf{Y}$ for this range, i.e., a set $\mathbf{Y}$ for which $\mathbf{y} \subseteq \mathbf{Y}$.

Algorithms for computing an enclosure start with an observation that for arithmetic operations $f(x_1, x_2)$, we have explicit formulas for the range. When $x_1 \in \mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$ and $x_2 \in \mathbf{x}_2 = [\underline{x}_2, \overline{x}_2]$, then:

- The range $\mathbf{x}_1 + \mathbf{x}_2$ for $x_1 + x_2$ is $[\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2]$.
- The range $\mathbf{x}_1 - \mathbf{x}_2$ for $x_1 - x_2$ is $[\underline{x}_1 - \overline{x}_2, \overline{x}_1 - \underline{x}_2]$.
- The range $\mathbf{x}_1 \cdot \mathbf{x}_2$ for $x_1 \cdot x_2$ is $[\underline{y}, \overline{y}]$, where

$$\underline{y} = \min(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2); \quad \overline{y} = \max(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2).$$

- The range $1/\mathbf{x}_1$ for $1/x_1$ is $[1/\overline{x}_1, 1/\underline{x}_1]$ (if $0 \notin \mathbf{x}_1$).

These formulas are called formulas of *interval arithmetic.*

The main idea behind straightforward interval computations is that within a computer, only elementary arithmetic operations are hardware supported[2]. No matter how complex the function $f(x_1, \ldots, x_n)$ is, the compiler *parses* it, i.e., represents its computation as a sequence of elementary arithmetic operations. The

[2]Actually, only addition, subtraction, and multiplication are directly hardware supported; division a/b is usually implemented as $a \cdot (1/b)$.

main idea is that if we only know the inputs with interval uncertainty, then we perform the same arithmetic operations in the same order, but with intervals instead of numbers. It is known that the resulting interval is an enclosure for the desired range.

Let us consider a toy example of estimating the range of a function $f(x) = (x-2)\cdot(x+2)$ on the interval $x \in [1,2]$. How will the computer compute this function? It will first compute $x-2$, then $x+2$, and then multiply the results. If we denote i-th intermediate computational result by r_i, then we get the following sequence of elementary arithmetic operations:

- $r_1 := x - 2$;
- $r_2 := x + 2$;
- $r_3 := r_1 \cdot r_2$.

If we perform the same operations, but with *intervals* instead of *numbers*, then we get the following intervals:

- $\mathbf{r}_1 := [1,2] - [2,2] = [-1,0]$;
- $\mathbf{r}_2 := [1,2] + [2,2] = [3,4]$;
- $\mathbf{r}_3 := [-1,0] \cdot [3,4] = [-4,0]$.

As a result, we get an interval $[-4,0]$.

In this toy example, $f(x) = x^2 - 4$, so the actual range of this function on the interval $[1,2]$ is easy to compute: it is equal to $f(\mathbf{x}) = [-3,0]$. We can thus see that our computed range $\mathbf{Y} = [-4,0]$ is indeed the enclosure for the actual range $\mathbf{y} = [-3,0]$.

Comment. To avoid misunderstanding, we should emphasize that this is just a toy example. There exist more efficient ways of computing an enclosure $\mathbf{Y} \supseteq \mathbf{y}$ than straightforward interval computations (see, e.g., [10]); however, most of these more efficient and more sophisticated techniques are based on the main ideas of straightforward interval computations.

1.6 From "Theoretical" Interval Computations to Computer-Representable Interval Computations: The Need for Rounding

The above formulas for interval arithmetic assumed that all rational numbers can be exactly represented in a computer. In reality, only some binary-rational numbers can be represented. To represent numbers like 1/3 in a computer, we must therefore *round* these numbers, i.e., replace these theoretically correct numbers with nearby machine-representable ones.

To get a guaranteed enclosure, we must always:

- round the lower endpoint of the interval downwards (i.e., replace it with a smaller number), and
- round the upper endpoint of the interval upwards (i.e., replace it with a larger number).

1.7 Interval Computations: Analysis

As we have mentioned, the main problem with computing the *exact* range of the function under interval uncertainty is that this computation is NP-hard, which means that in the worst case, this computation probably require the time which is exponentially growing the size T of the expression f – i.e., grows as 2^T of faster. As a result, for reasonable size algorithms f, with T in hundreds, the required computation time will be unrealistic – e.g., it may exceed the lifetime of the universe.

From this viewpoint, a natural question to ask is: how long will computations take for the above straightforward computations techniques of computing the *enclosure* for the exact range. In straightforward interval computations, each original elementary arithmetic operation is replaced with one operation of interval arithmetic. Each interval arithmetic operation consists of several arithmetic operations with numbers: addition of two intervals means two additions of numbers, etc. The largest number of operation with numbers per single interval arithmetic operation is for interval multiplication, which requires 4 multiplications of numbers. Thus, when we move from the original computations to interval computations, we replace each arithmetic operation with ≤ 4 operations. As a result, the computation time for the straightforward computations is $\leq 4 \cdot T$, i.e., it is $O(T)$, where T is the number of operations in (i.e., in effect, the running time of) the original algorithm.

As a result of straightforward interval computations, we compute the enclosure $\mathbf{Y} \supseteq \mathbf{y}$, often with excess width. As we have seen on the toy example, the main reason why there is an excessive width is that:

- there is a relation between intermediate results, and
- in straightforward interval computations, we ignore this relation.

For example, in the above toy example, the intervals ranges for r_1 and r_2 were exact. However, when we multiplied the corresponding intervals $\mathbf{r}_1$ and $\mathbf{r}_2$, we used the general formulas for interval multiplication, formulas that implicitly assume that all pairs (r_1, r_2) from the corresponding box $\mathbf{r}_1 \times \mathbf{r}_2$ are possible. Thus, we ignored the fact that the values r_1 and r_2 are actually related – since they are both functions of the same variable x – and so, not all pairs (r_1, r_2) are possible.

In addition to algorithms for computing an enclosure, there also exist algorithms for computing the *exact* range; e.g., algorithms based on Tarski's ideas can be applicable for arbitrary algebraic functions f; see, e.g., [14] and references therein. These algorithms, however, require exponential time $\sim 2^T$ (or even higher) and are, thus, not applicable for large T.

1.8 Interval Computations: The First Problem

Summarizing the above discussion, we conclude that we have, in effect, two classes of algorithms for solving the main problem of interval computations:

- fast and efficient $O(T)$ algorithms – which often have large excess width;
- slow and inefficient (often non-feasible) algorithms – with no excess width.

In practice, we are often not satisfied with the excess width of a faster algorithm, but we do not have enough time to apply the algorithm for computing the exact range. To take care of such situations, it is desirable to develop a *sequence* of feasible algorithms with:

- longer and longer computation time and
- smaller and smaller excess width.

The development of such a sequence is one of the objectives of this paper.

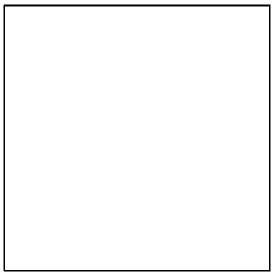
2 Formulation of the Main Problem

2.1 Interval Computations: Limitations

In traditional interval computations:

- we know the intervals $\mathbf{x}_i$ of possible values of different parameters x_i, and
- we assume that an arbitrary combination of these values is possible.

In geometric terms, this assumption means that the set of possible combinations $x = (x_1, \ldots, x_n)$ is a *box* $\mathbf{x} = \mathbf{x}_1 \times \ldots \times \mathbf{x}_n$.



In many real-life situations, in addition to knowing the intervals $\mathbf{x}_i$ of possible values of each variable x_i, we also know additional restrictions on the possible combinations of x_i. In this case, the set $\mathbf{x}$ of possible values of x is a (proper) *subset* of the original box. For example, in addition to knowing the bounds on x_1 and x_2, we may also know that the difference between x_1 and x_2 cannot exceed a certain amount. Informally speaking, the parameters x_i are no longer independent – in the sense that the set of possible values of x_i may depend on the values of other parameters.

In such situations, it is desirable to be able to compute the range of possible values of $f(x_1, \ldots, x_n)$ for all combinations $(x_1, \ldots, x_n)$ which satisfy the given restrictions. Computing this range is the main objective of this paper.

Comment. In interval computations, we start with independent inputs; as we follow computations, we get dependent intermediate results: e.g., for $x_1 - x_1^2$, the values of x_1 and $x_2 = x_1^2$ are strongly dependent in the sense that only values (x_1, x_1^2) are possible within the box $\mathbf{x}_1 \times \mathbf{x}_2$.

- In interval computations, there are many techniques for handling similar dependence between the *intermediate* computational *results*.
- In this paper, we extend these techniques to handle a different type of dependence – dependence between the *inputs*.

Before we start describing the corresponding ideas and algorithms, let us first give two examples of such restrictions.

2.2 Example from Geosciences

Our civilization greatly depends on the things we extract from the Earth, such as fossil fuels (oil, coal, natural gas), minerals, and water. Our need for these commodities is constantly growing, and because of this growth, they are being exhausted. Even under the best conservation policies, there is (and there will be) a constant need to find new sources of minerals, fuels, and water.

The only sure-proof way to guarantee that there are resources such as minerals at a certain location is to actually drill a borehole and analyze the materials extracted. However, exploration for natural resources using indirect means began in earnest during the first half of the 20th century. The result was the discovery of many large relatively easy to locate resources such as the oil in the Middle East.

However, nowadays, most easy-to-access mineral resources have already been discovered. For example, new oil fields are mainly discovered either at large depths, or under water, or in very remote areas – in short, in the areas where drilling is very expensive. It is therefore desirable to predict the presence of resources as accurately as possible before we invest in drilling.

From previous exploration experiences, we usually have a good idea of what type of structures are symptomatic for a particular region. For example, oil and gas tend to concentrate near the top of natural underground domal structures. So, to be able to distinguish between more promising and less promising locations, it is desirable to determine the structure of the Earth at these locations. To be more precise, we want to know the structure at different depths z at different locations (x, y).

Another vitally important application where the knowledge of the Earth structure is crucial is the assessment of earth hazards. Earthquakes can be very destructive, so it is important to be able to estimate the probability of an earthquake, where one is most likely to occur, and what will be the magnitude of the expected earthquake. Geophysicists have shown that earthquakes result from accumulation of mechanical stress; so if we know the detailed structure of the corresponding Earth locations, we can get a good idea of the corresponding stresses and faults present and the potential for occurrence of an earthquake. From this viewpoint, it is also very important to determine the structure of the Earth.

In general, to determine the Earth structure, we can use different measurement results that can be obtained without actually drilling the boreholes: e.g., gravity and magnetic measurements, analyzing the travel-times and paths of seismic ways as they propagate through the earth, etc.

The relation between the Earth structure and the related measurable quantities is usually known. So, when we know the exact structure at a given Earth location, we can predict, with reasonable accuracy, the corresponding values of the measured quantities – we can predict the local value of the gravity field, the time that a seismic signal needs to travel from its origin to the sensor, etc. Such problems are usually called *forward* problems.

Forward problems enable us, given a model of the Earth, to predict the values of different signals. What we need in the above geophysical applications is the opposite: given the measured values of different signals, we need to reconstruct the structure of the Earth at the location where the measurements have been made. Such problems are therefore called *inverse problems.*

Some measurements – like gravity and magnetic measurements – describe the overall effect of a large area. These measurements can help us determine the average mass density in the area, or the average concentration of magnetic materials in the area, but they often do not determine the detailed structure of this area. This detailed structure can be determined only from measurements which are narrowly focused on small sub-areas of interest.

The most important of these measurements are usually *seismic measurements.* Seismic measurements involve the recording of vibrations caused by distant earthquakes, explosions, or mechanical devices. For example, these records are what seismographic stations all over the world still use to detect earthquakes. However, the signal coming from an earthquake carries not only information about the earthquake itself, it also carries the information about the materials along the path from an earthquake to the station: e.g., by measuring the travel-time of a seismic wave, checking how fast the signal came, we can determine the velocity of sound v in these materials. Usually, the velocity of sound increases with increasing density, so, by knowing the velocity of sound at different 3-D points, we will be able to determine the density of materials at different locations and different depths.

The main problem with the analysis of earthquake data (i.e., *passive* seismic data) is that earthquakes are rare events, and they mainly occur in a few seismically active belts. Thus, we have a very uneven distribution of sources and receivers that results in a "fuzzy" image of earth structure in many areas.

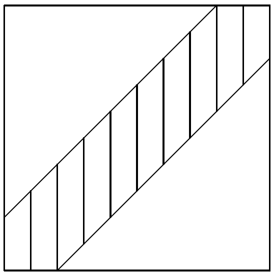
To get a better understanding of the Earth structure, we must therefore rely on *active* seismic data – in other words, we must make artificial explosions, place sensors around them, and measure how the resulting seismic waves propagate. The most important information about the seismic wave is the *travel-time* t_i, i.e., the time that it takes for the wave to travel from its source to the sensor. to determine the geophysical structure of a region, we measure seismic travel times and reconstruct velocities at different depths from these data. The problem of reconstructing this structure is called the *seismic inverse problem.* There are several algorithms for solving this inverse problem; see, e.g., [9, 16, 22].

In principle, we can determine the paths from the source to each sensor. The travel-time t_i along i-th path can then be determined as the sum of travel-times in different cells j through which this path passes: $t_i = \sum\limits_j \ell_{ij} v_j$, where ℓ_{ij} denotes the length of the part of i-th path within cell j. This formula can be somewhat simplified if we replace the velocities v_j by their inverses $s_j \stackrel{\text{def}}{=} \dfrac{1}{v_j}$, called *slownesses.* In terms of slownesses, the formula for the travel-time takes the simpler form $t_i = \sum\limits_j \ell_{ij} \cdot s_j$.

For each cell j, a geophysicist usually provides us with the smallest and largest possible value of slowness for this cell. In other words, for each cell j, the expert provides us with an interval $[\underline{s}_j, \overline{s}_j]$ that is guaranteed to contain the actual (unknown) value of slowness s_j. Based on these estimates, we can find the range $[\underline{t}_i, \overline{t}_i]$ of possible values of t_i, where $\underline{t}_i = \sum\limits_j \ell_{ij} \cdot \underline{s}_j$ and $\overline{t}_i = \sum\limits_j \ell_{ij} \cdot \overline{s}_j$. If the measured travel time $\widetilde{t}_i$ is outside this interval, this means that the observed travel-times are *inconsistent* with the intervals $[\underline{s}_j, \overline{s}_j]$.

This information should be reported back to the experts, so that the experts will be able to adjust their bounds for s_j in such a way that the new bounds will be consistent with the observations; see, e.g., [1].

The above bounds $\underline{t}_i$ and $\overline{t}_i$ were obtained under the assumption that the only information that we have about the slownesses s_j is that each slowness lies in the corresponding interval. In reality, in addition to bounds on slownesses s_j at different points, we also know that slowness cannot change too fast between the neighboring points. To be more precise, the experts usually provide us with a value Δ such that $|s_j - s_k| \le \Delta$ for all neighboring pairs (j, k):



It is therefore necessary to find the range of a linear function $t_i = \sum\limits_j \ell_{ij} \cdot s_j$ under such constraints.

2.3 Example from Safety-Critical Engineering

In engineering of safety-critical systems, e.g., in nuclear engineering, it is vitally important to provide safety, i.e., to guarantee that certain quantities y like temperature, pressure, radiation level, do not exceed the required thresholds y_0. The value of each such quantity y depends on several parameters $x_1, \ldots, x_n$, all of which may somewhat deviate from their nominal values. These parameters may include parameters of the design (such as the exact thickness of the protective layer) or external parameters such as the outdoors temperature.

We usually know the dependence $y = f(x_1, \ldots, x_n)$ of the desired quantity y on these parameters. So, the problem of guaranteeing safety means guaranteeing that the upper endpoint $\overline{y}$ of the range $\mathbf{y} = [\underline{y}, \overline{y}]$ of the function $f(x_1, \ldots, x_n)$ over all possible combinations $(x_1, \ldots, x_n)$ does not exceed y_0.

We usually know the ranges $\mathbf{x}_i$ of possible values of each of the parameters. Thus, we know that all possible combinations $(x_1, \ldots, x_n)$ are within the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$. So, in principle, we can guarantee safety if we guarantee that $f(x_1, \ldots, x_n) \le y_0$ for all possible values from this box. In other words, we can find the range $\widetilde{y} = [\underline{y}, \overline{y}]$ of the function $f(x_1, \ldots, x_n)$ on the box, and make sure that $\overline{y} \le y_0$.

This approach does lead to guarantee safety, but it may be too conservative. Indeed, the maximum of the function $f(x_1, \ldots, x_n)$ on the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$ is often attained at one its endpoints, i.e., at one of the possible combinations of extreme values of x_i. This fact is true, e.g., if the function $f(x_1, \ldots, x_n)$ is monotonic in each of its variables. However, experts often claim that combinations of extreme values are impossible. In other words, experts claim that the actual set S of possible values of $(x_1, \ldots, x_n)$ is a proper *subset* of the original box – i.e., that there are additional constraints which describe the relation between the parameters x_i.

How can we describe such a subset? In real life, whenever we have a cluster formed by real-life data points, this cluster has a reasonably smooth boundary. This cluster can be a disk (solid circle), a ball (solid sphere in multi-D space), an ellipsoid, or a more complex structure, but it is practically always smooth. The fact that it is smooth means that we can describe its border by an equation $b(x_1, \ldots, x_n) = C$ for some smooth function $b(x_1, \ldots, x_n)$ and for some constant C. As a result, the set S itself can be describe either by the inequality

$$b(x_1, \ldots, x_n) \le C_0 \tag{1}$$

or by the inequality $b(x_1, \ldots, x_n) \ge C_0$. In the second case, the inequality can be transformed into an equivalent form $b'(x_1, \ldots, x_n) \le C'$, where the function $b'(x_1, \ldots, x_n) = -b(x_1, \ldots, x_n)$ is also smooth, and $C' = -C_0$. So, without loss of generality, we can assume that the set S is described by the inequality (1), for some smooth function $b(x_1, \ldots, x_n)$.

An arbitrary smooth function can be approximated by a polynomial, so, instead of the the general set (1), we can consider the approximating set

$$a(x_1, \ldots, x_n) \le C_0, \tag{2}$$

where $a(x_1, \ldots, x_n)$ is a polynomial that approximates the smooth function $b(x_1, \ldots, x_n)$.

The simplest possible polynomials are linear polynomials $a(x_1, \ldots, x_n) = a_0 + a_1 \cdot x_1 + \ldots + a_n \cdot x_n$. However, for a linear function $a(x_1, \ldots, x_n)$, the set of all the vectors x for which $a(x) \le C_0$ is a half-space, i.e., a set that is not bounded in many directions, while we want a set S that is inside the box – and hence, bounded in all directions. Thus, if we restrict ourselves to only linear terms, we do not get a good approximation to the set (1).

To get a reasonable approximation, we must consider quadratic and higher order polynomial approximating functions $a(x_1, \ldots, x_n)$. In particular, for the simplest non-linear polynomials – quadratic polynomials – the approximating set (2) takes the following form:

$$a(x_1, \ldots, x_n) = a_0 + \sum_{i=1}^{n} a_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} \cdot x_i \cdot x_j \le C. \tag{3}$$

Ellipsoids indeed provide a reasonable description of the set of possible values of $(x_1, \ldots, x_n)$. To get an even better description of the actual set (1), we can, in principle, use 3rd, 4th, and higher order polynomials.

2.4 How This Information Is Processed Now

At present, to estimate the range of a given function over given constraints, we use problem-specific structure of the objective function $f(x_1, \ldots, x_n)$ and of the corresponding constraints.

In geophysical problems, to estimate the range of a linear function $t_i = \sum\limits_j \ell_{ij} \cdot s_j$ under linear constraints $\underline{s}_j \le s_j \le \overline{s}_j$ and $|s_j - s_k| \le \Delta$, we can use linear programming techniques – techniques that were specifically designed for such linear constraint optimization.

Another idea is used to estimate the range of a given function over an ellipsoid in safety-critical engineering; see, e.g., [13]. Usually, the range of each variable x_i is reasonably narrow, so we can expand the dependence $f(x_1, \ldots, x_n)$ in Taylor series around nominal values, and restrict ourselves to quadratic terms in this expansion. As a result, the problem of estimating the range of a given function $f(x_1, \ldots,_x)$ over the range S turns into the problem of estimating the range of the given quadratic function $f(x_1, \ldots, x_n)$ over an ellipsoid, i.e., over the range described by quadratic constraints $b(x_1, \ldots, x_n) \le C_0$.

For this constraint optimization problem, the Lagrange multiplier technique reduces it to the problem of unconstrained optimization of a quadratic function

$$F(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) + \lambda \cdot (b(x_1, \ldots, x_n) - C_0).$$

For this quadratic function, we can find the maximum by simply solving an easy-to-solve system of n linear equations with n unknowns: $\dfrac{\partial F}{\partial x_i} = 0$.

Both ideas can only be used for special objective functions and special constraints. It is therefore desirable to develop *general* techniques for estimating the range of a given function under given constraints.

3 Main Idea

3.1 Similar Situation: Statistics

In statistics, to get a *complete* description of a multi-dimensional probability distribution of n variables $x = (x_1, \ldots, x_n)$, ideally, we should take into account dependence between all the variables. It is, however, often too computationally taxing to find all these dependencies. Therefore, in statistics, it is often necessary to only use partial information about the n-dimensional distribution.

First, we need to find the probability distribution for each of n variables. As we have mentioned earlier, if we have no information about the dependence between these variables, then it is reasonable to assume that these variables are independent. This resulting probability distribution often forms a reasonable *first approximation* to the actual n-dimensional distribution.

To get a more accurate description, the *next* reasonable *step* is to take into account pairwise dependencies, i.e., dependencies between pairs of variables (x_i, x_j). In the traditional statistical practice in engineering and science, this is done by estimating correlation, covariance, and/or other characteristics of pairwise dependence.

To get an even better picture of the distribution, we can consider dependencies between triples, etc.

As a result, we get a sequence of methods – independent variables, pairwise dependence, dependence between the tripes, etc., all the way to a complete description of dependence between all n variables. As we go from independence to taking more and more information about the dependence into account, we get a sequence of methods which:

- require more and more time
- but at the same time lead to more and more accurate results.

3.2 Let Us Use a Similar Idea for Interval Uncertainty

How can we use a similar idea to take into account dependence between the inputs in interval computation?

In straightforward interval computations, we consider only intervals of possible values of x_i.

A natural next approximation is when we consider:

- sets $\mathbf{x}_i$ of possible values of x_i, and also
- sets $\mathbf{x}_{ij}$ of possible pairs (x_i, x_j).

Comment. This idea is similar to *constrained fuzzy arithmetic* developed by G. J. Klir; see, e.g., [12].

The third approximation is when we also consider possible sets of triples $\mathbf{x}_{ijk}$, etc., all the way to the situation when we completely describe the dependence between x_i by describing the set $\mathbf{x}_{12\ldots n}$ of possible values of $x = (x_1, x_2, \ldots, x_n)$.

Of course, the more dependence we take into account, the more information we need to store and process and thus, the more computation time the methods will take.

- For straightforward interval computations, all we need to store is intervals of possible values.
- For pairs, we need to store sets of possible values of pairs, i.e., subsets of 2-D boxes. To describe an arbitrary such set with accuracy ε, we must know, for each of $1/\varepsilon^2$ sub-boxes of size $\varepsilon \times \varepsilon$, whether this box belongs to the desired set or not. Thus, we need to store $1/\varepsilon^2$ bits of information.
- For triples, we similarly need $1/\varepsilon^3$ bits of information about whether each of $1/\varepsilon^3$ 3-D boxes of size $\varepsilon \times \varepsilon \times \varepsilon$ belongs to the desired set or not.
- For quadruples, we need $1/\varepsilon^4$ bits, etc.

As a result, we (hope to) get a sequence of methods which:

- require more and more time
- but at the same time lead to more and more accurate results.

3.3 How to Implement This Idea

In straightforward interval computations:

- First, we *describe* the initial uncertainty by intervals.
- Then, we show how, by using interval arithmetic, we can *propagate* this uncertainty through the algorithm f, so that at the end, we get an enclosure for the desired range.
- Finally, we show how to adjust operations of interval arithmetic so that all intermediate intervals are *computer-representable* – and at the same time the result is still a guaranteed enclosure.

Similarly, to implement the new idea, we must be able to achieve the following:

- First, we must *describe* the initial uncertainty by sets of pairs etc.
- Second, we must learn how to *propagate* the corresponding uncertainty through algorithms, so that at the end, we will get a better enclosure for the desired range, an enclosure that takes into account the dependence between the inputs.
- Finally, we must learn how to *represent* and process sets of pairs etc, in the *computer*, so that the result will still be a guaranteed enclosure.

We have already decided on how to represent uncertainty by sets of pairs etc. In the following subsections, we will show how we can achieve the two remaining tasks.

3.4 How to Propagate This Uncertainty

In the beginning, we know the intervals $\mathbf{r}_1, \ldots, \mathbf{r}_n$ corresponding to the input variables $r_i = x_i$, and we know the sets $\mathbf{r}_{ij}$ for i, j from 1 to n.

The question: is how to propagate this information through an intermediate computation step, a step of computing $r_k = r_a * r_b$ for some arithmetic operation $*$ and for previous results r_a and r_b $(a, b < k)$. By the time we come to this step, we know the intervals $\mathbf{r}_i$ and the sets $\mathbf{r}_{ij}$ for $i, j < k$. We want to find the interval $\mathbf{r}_k$ for x_k, and the sets $\mathbf{r}_{ik}$ for $i < k$. The following is a natural way to find these sets:

- The range $\mathbf{r}_k$ can be naturally found as $\{r_a * r_b \,|\, (r_a, r_b) \in \mathbf{r}_{ab}\}$.
- The set $\mathbf{r}_{ak}$ is described as $\{(r_a, r_a * r_b) \,|\, (r_a, r_b) \in \mathbf{r}_{ab}\}$.
- The set $\mathbf{r}_{bk}$ is described as $\{(r_b, r_a * r_b) \,|\, (r_a, r_b) \in \mathbf{r}_{ab}\}$.
- For $i \neq a, b$, the set $\mathbf{r}_{ik}$ is described as $\{(r_i, r_a * r_b) \,|\, (r_i, r_a) \in \mathbf{r}_{ia}, (r_i, r_b) \in \mathbf{r}_{ib}\}$.

Comment. From the mathematical viewpoint, a subset $\mathbf{r}_{ij}$ of the set of all possible pairs $\mathbf{r}_i \times \mathbf{r}_j$ is a *relation*. It is therefore not surprising that processing this uncertainty is similar to processing relations in other application areas such as relational database systems; see, e.g., [19]. For example, a natural intermediate step in computing $\mathbf{r}_{ik}$ is when, given the relations $\mathbf{r}_{ia}$ and $\mathbf{r}_{ib}$, we form a new relation $\{(r_a, r_i, r_b) \,|\, (r_a, r_i) \in \mathbf{r}_{ai}, (r_i, r_b) \in \mathbf{r}_{ib}\}$. In relational algebra, this intermediate relation is called a *join* and denoted by $\mathbf{r}_{ai} \bowtie_i \mathbf{r}_{ib}$.

3.5 How to Represent Sets in a Computer

How can we represent a set of pairs or a set of triples in a computer? A natural idea is to do it in a way cumulative probability distributions (cdf) are represented in RiskCalc package [6]: by *discretization.*

In RiskCalc, we divide the interval $[0, 1]$ of possible values of probability into, say, 10 subintervals of equal width and represent cdf $F(x)$ by 10 values $x_1, \ldots, x_{10}$ at which $F(x_i) = i/10$.

Similarly, to describe a set $\mathbf{x}_{ij} \subseteq \mathbf{x}_i \times \mathbf{x}_j$, we:

- divide the box $\mathbf{x}_i \times \mathbf{x}_j$ into, say, 10×10 subboxes, and
- describe the set $\mathbf{x}_{ij}$ by listing all subboxes which contain possible pairs.

Comment. This representation of a set by the union of grid cells which intersect with this set is well known in data mining as an upper approximation in the sense of *rough set* theory; see, e.g., [17, 18].

Of course, in reality, there is no need to actually list these subboxes: to describe an arbitrary set, it is sufficient to store $10 \times 10 = 100$ bits of information describing whether each of the 10×10 subboxes belongs to the list. In other words, a set can be represented as 10×10 array of Boolean values. Similarly, for triples, we can represent the corresponding set as a 3-D array of size $10 \times 10 \times 10$, etc.

Comment. The above approach is a good way to describe generic sets, but in practice, the resulting description may be redundant.

- For example, even if we know that all the values (x_1, x_2) are possible, we still need 100 Boolean values to describe this set.
- Similarly, if the set consists of all the values for which $x_1 = x_2$, then out of 100 subboxes, only 10 diagonal boxes are affected, but we still need all 100 Boolean values.

A more efficient idea is to represent sets is by using a *paving* – in the style of [10]. In this approach, we start with a 2×2 subdivision. For each of the $2 \times 2 = 4$ subboxes, we:

- mark this subbox as "in" if it is completely inside the desired set;
- mark this subbox as "out" if it is completely outside the desired set;
- otherwise, if this subbox contains both points from the desired set and point outside the desired set, we subdivide this box into $2 \times 2 = 4$ subboxes, and repeat the procedure.

As a result, we get a list consisting of boxes of different sizes – starting with larger ones and only decreasing the size when necessary.

3.6 How to Propagate This Uncertainty: An Algorithm

Let us show how this representation can be propagated through an intermediate computational step, a step of computing $r_k = r_a * r_b$ for some arithmetic operation $*$ and for previous results r_a and r_b $(a, b < k)$. We start by dividing each original interval range into the same number C of equal sub-intervals. By the time we come to this step, we know the intervals $\mathbf{r}_i$ and the sets $\mathbf{r}_{ij}$ for $i, j < k$. Each of these sets is described as a union of the subboxes.

We want to find the interval $\mathbf{r}_k$ for x_k, and the sets $\mathbf{r}_{ik}$ for $i < k$. First, we compute the range $\mathbf{r}_k$:

- In our representation, the set $\mathbf{x}_{ab}$ consists of small 2-D boxes $\mathbf{X}_a \times \mathbf{X}_b$.
- For each small box $\mathbf{X}_a \times \mathbf{X}_b$, we use interval arithmetic to compute the range $\mathbf{X}_a * \mathbf{X}_b$ of the value $r_a * r_b$ over this box.
- Then, we take the union (interval hull) of all these ranges.

Then, we divide this range interval into C equal sub-intervls, and compute the sets $\mathbf{r}_{ik}$ as follows:

- We consider the sets $\mathbf{r}_{ab}$, $\mathbf{r}_{ai}$, and $\mathbf{r}_{bi}$.
- For each small box $\mathbf{R}_a \times \mathbf{R}_b$ from $\mathbf{r}_{ab}$, we:
  - consider all subintervals $\mathbf{R}_i$ for which $\mathbf{R}_a \times \mathbf{R}_i$ is in $\mathbf{r}_{ai}$ and $\mathbf{R}_b \times \mathbf{R}_i$ is in $\mathbf{r}_{bi}$, and then
  - we add $(\mathbf{R}_a * \mathbf{R}_b) \times \mathbf{R}_i$ to the set $\mathbf{r}_{ki}$.

To be more precise, since the interval $\mathbf{R}_a * \mathbf{R}_b$ may not have bounds exactly matching the subdivision of the range interval $\mathbf{r}_k$ into C parts, we may need to expand the interval $\mathbf{R}_a * \mathbf{R}_b$ to get within bounds of this subdivision (numerical examples are given in the following text).

Comment. How long does each computation take? For each i, we need to consider $\le C^2$ small boxes $\mathbf{R}_a \times \mathbf{R}_b$, and for each such subbox, we must consider C subintervals $\mathbf{R}_i$, so the computation of each new range $\mathbf{r}_{ik}$ requires $O(C^2) \cdot C = O(C^3)$ computational steps. Since C is a fixed constant, this number does not affect the asymptotic complexity of the proposed algorithm.

We repeat these computations step by step until we get the desired estimate for the range of the final result of the computations.

Comment. Our main objective is to be able to take into account the prior dependence between the inputs $x_1, \ldots, x_n$. However, as a side effect of this technique, in addition to taking into account dependence between the inputs, we also take care of the (more traditional) dependence between individual results. For example, when we compute the range of $x_1 - x_1^2$, we first compute $x_2 = x_1^2$ and then compute $x_3 = x_1 - x_2$; in our methodology, when we compute x_2, we automatically generate the set $\mathbf{x}_{12}$ of possible values of pairs (x_1, x_2). We will see that this set is close to the graph of the function x^2. On the next step, when we compute $x_3 = x_1 - x_2$, we take into account not only the intervals $\mathbf{x}_1$ and $\mathbf{x}_2$, but also the set $\mathbf{x}_{12}$, and thus, the resulting estimate for the range for x_3 is close to the ideal.

4 Examples

4.1 First Example: Computing the Range of $x - x$

Let us starts with the simplest example where straightforward interval computations lead to overestimation: the problem of estimating the range of the function $f(x) = x - x$ on the interval $[0, 1]$.

Of course, this function is identically 0, so its actual range is the degenerate interval $[0, 0]$. Let us trace what happens if we apply straightforward interval computations to this function. Parsing leads to the following sequence of elementary arithmetic operations: $r_1 = x$, $r_2 = r_1$, and $r_3 = r_1 - r_2$. So, if we replace each elementary arithmetic operation with the corresponding operation of interval arithmetic, we get $\mathbf{r}_1 = [0, 1]$, $\mathbf{r}_2 = [0, 1]$, and thus, the final range is $\mathbf{r}_3 = \mathbf{r}_1 - \mathbf{r}_2 = [0, 1] - [0, 1] = [-1, 1]$ – an enclosure with excess width.

In straightforward interval computations, we have $r_1 = x$ with the exact interval range $\mathbf{r}_1 = [0, 1]$, we have $r_2 = x$ with the exact interval range $\mathbf{x}_2 = [0, 1]$. We get excess width because the variables r_1 and r_2 are dependent, but we ignore this dependence. In effect, when computing the range $\mathbf{r}_3$, we use formulas based on the assumption that the set of possible combinations of (r_1, r_2) is the entire box $\mathbf{r}_1 \times \mathbf{r}_2$.

In the new approach, we still have $\mathbf{r}_1 = \mathbf{r}_2 = [0, 1]$. However, since $r_2 = r_1$, we know that not all pairs (r_1, r_2) from the box $\mathbf{r}_1 \times \mathbf{r}_2$ are possible – the set $\mathbf{r}_{12}$ of possible values of (r_1, r_2) is the diagonal $\mathbf{r}_{12} = \{(r_1, r_2) \,|\, r_1, r_2 \in [0, 1], r_1 = r_2\}$.

When we compute the range $\mathbf{r}_3$ of $r_3 = r_1 - r_2$, we only use pairs (r_1, r_2) from the diagonal set $\mathbf{r}_{12}$. For each point from this diagonal set, $r_3 = r_1 - r_2 = 0$. Thus, with the new techniques, we get the exact range $[0, 0]$ for the function $f(x) = x - x$.

Comment. Similarly, the new method computes the exact range for $x \cdot x$: we have $r_1 = x$, $r_2 = r_1$, and $r_3 = r_1 \cdot r_2$. In contrast, if we use straightforward interval computations, then for $\mathbf{x} = [-, 1]$, instead of the correct range $[0, 1]$, we get an closure $[-1, 1] \cdot [-1, 1] = [-1, 1]$, with excess width.

4.2 Second Example: Computing the Range of $x - x^2$

In the example of the degenerate function $f(x) = x - x$, it is easy to avoid excess width without using any new techniques. Indeed, in this example, it is sufficient to simplify the expression for the function $f(x)$ to 0. Many existing compilers can detect the possibility of such a simplification and perform it.

There are less trivial examples of excess width, where a simplification is either impossible or at least is not so easy to find. A simple example of such a situation is the function $f(x) = x - x^2$ on the interval $[0, 1]$.

For this quadratic function, the range can be easily obtained by using the standard calculus technique: namely, according to calculus, to find the range of a function of one variable on a given interval, it is sufficient to find the values of this function on the endpoints and on all the stationary points (i.e., points where the derivative $f'(x)$ is equal to). The smallest of these values is the lower endpoint of the range, and the largest of these values is the upper endpoint of the range. For the given function, the only stationary point $f'(x) = 1 - 2x = 0$ is the point $x = 0.5$. So, to find the range of this function, it is sufficient to find its value for $x = 0$ (where $f(0) = 0$), for $x = 0.5$ (where $f(0.5) = 0.25$), and for $x = 1$ (where $f(1) = 0$). Thus, the actual range of this function is $[\min(0, 0.25, 0), \max(0, 0.25, 0)] = [0, 0.25]$.

In straightforward interval computations:

- we have $r_1 = x$ with interval $\mathbf{r}_1 = [0, 1]$;
- we have $r_2 = x^2$ with interval $\mathbf{x}_2 = [0, 1]$;
- the variables r_1 and r_2 are dependent, but we ignore this dependence and estimate $\mathbf{r}_3$ as $[0, 1] - [0, 1] = [-1, 1]$.

In the new approach, we still have $\mathbf{r}_1 = \mathbf{r}_2 = [0, 1]$, but, since $x_2 = x_1^2$, we now also have the set $\mathbf{r}_{12} = \{(x_1, x_2) \,|\, x_1, x_2 \in [0, 1], x_2 = x_1^2\}$. When we compute the range $\mathbf{r}_3$ of $r_3 = r_1 - r_2$, we only use pairs (r_1, r_2) from this set. For each point from this diagonal set, $r_3 = r_1 - r_2 = r_1 - r_1^2$. Thus, with the new techniques, the computed range $\mathbf{r}_3$ is exactly the range $[0, 0.25]$ of the original function $f(x) = x - x^2$ – with no excess width.

4.3 Distributivity: $a \cdot (b + c)$ vs. $a \cdot b + a \cdot c$

It is known that interval arithmetic is not distributive in the following sense: when we want to compute the range of the function $f(x_1, x_2, x_3) = x_1 \cdot (x_2 + x_3) = x_1 \cdot x_2 + x_1 \cdot x_3$, straightforward interval computations sometimes lead to different enclosures depending on which of the two equal expression we use.

This is true, e.g., when $x_1 \in \mathbf{x}_1 = [0, 1]$, $\mathbf{x}_2 = [1, 1]$, and $\mathbf{x}_3 = [-1, -1]$. In this case, $x_2 + x_3 = 0$, so $f(x_1, x_2, x_3) = x_1 \cdot (x_2 + x_3) = 0$ for all possible x_i. Hence, the actual range is $[0, 0]$.

For the expression $f(x_1, x_2, x_3) = x_1 \cdot (x_2 + x_3)$, straightforward interval computations lead to $\mathbf{x}_1 \cdot (\mathbf{x}_2 + \mathbf{x}_3) = [0, 1] \cdot [0, 0] = [0, 0]$, i.e., to the exact range. However, for $f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1 \cdot x_3$, we get $\mathbf{x}_1 \cdot \mathbf{x}_2 + \mathbf{x}_1 \cdot \mathbf{x}_3 = [0, 1] \cdot 1 + [0, 1] \cdot (-1) = [0, 1] + [-1, 0] = [-1, 1]$, i.e., excess width.

The reason for this excess width is that we have the exact ranges for $r_1 = x_1$, $r_2 = x_2$, $r_3 = x_3$, $r_4 = x_1 \cdot x_2$, and $r_5 = x_1 \cdot x_3$, but we ignore the dependence between r_4 and r_5 when computing the range of the final result $r_6 = r_4 + r_5$.

In the new approach, we start with the intervals $\mathbf{r}_1 = \mathbf{x}_1$, $\mathbf{r}_2 = \mathbf{x}_2$, and $\mathbf{r}_3 = \mathbf{x}_3$. Since we are not assuming any dependence between the variables r_1, r_2, and r_3, we thus assume that for these variables, all pairs are possible, i.e., $\mathbf{r}_{12} = \mathbf{r}_1 \times \mathbf{r}_2$, $\mathbf{r}_{23} = \mathbf{r}_2 \times \mathbf{r}_3$, and $\mathbf{r}_{13} = \mathbf{r}_1 \times \mathbf{r}_3$.

When we compute $r_4 = r_1 \cdot r_2$, we also compute the ranges $\mathbf{r}_{14}$, $\mathbf{r}_{24}$, and $\mathbf{r}_{34}$, as

$$\mathbf{r}_{14} = \{(r_1, r_1 \cdot r_2) \,|\, r_1 \in \mathbf{r}_1, r_2 \in \mathbf{r}_2\}, \quad \mathbf{r}_{24} = \{(r_2, r_1 \cdot r_2) \,|\, r_1 \in \mathbf{r}_1, r_2 \in \mathbf{r}_2\}, \quad \mathbf{r}_{34} = \mathbf{r}_3 \times \mathbf{r}_4.$$

When we compute $r_5 = r_1 \cdot r_3$, we also compute the range $\mathbf{r}_{45}$ for pairs (r_4, r_5) as

$$\{(r_4, r_1 \cdot r_3) \,|\, (r_1, r_4) \in \mathbf{r}_{14}, (r_3, r_4) \in \mathbf{r}_{34}\}.$$

From our description of $\mathbf{r}_{14}$ and $\mathbf{r}_{34}$, we conclude that

$$\mathbf{r}_{45} = \{(r_4, r_1 \cdot r_3) \,|\, \exists r_2 \in \mathbf{r}_2 \text{ s.t. } r_4 = r_1 \cdot r_2, r_3 \in \mathbf{r}_3\}.$$

Thus,

$$\mathbf{r}_{45} = \{(r_1 \cdot r_2, r_1 \cdot r_3) \,|\, r_1 \in \mathbf{r}_1, r_2 \in \mathbf{r}_2, r_3 \in \mathbf{r}_3\}.$$

Based on this set, the range of possible values of $r_6 = r_4 + r_5$ coincides with the set

$$\{r_1 \cdot r_2 + r_1 \cdot r_3 \,|\, r_1 \in \mathbf{r}_1, r_2 \in \mathbf{r}_2, r_3 \in \mathbf{r}_3\},$$

i.e., with the exact range of the function $f(x_1, x_2, x_3) = x_1 \cdot (x_2 + x_3)$.

4.4 Toy Example with Prior Dependence

Let us consider the problem of finding the range of $r_1 - r_2$ when $\mathbf{r}_1 = [0,1]$, $\mathbf{r}_2 = [0,1]$, and $|r_1 - r_2| \le 0.1$. In this case, the actual range of the difference $r_1 - r_2$ is, of course, $[-0.1, 0.1]$.

Straightforward interval computations cannot take the prior dependence into account. Thus, the only result we can get by using straightforward interval computations is the interval $\mathbf{r}_1 - \mathbf{r}_2 = [0,1] - [0,1] = [-1,1]$.

In the new approach, $\mathbf{r}_{12} = \{(r_1, r_2) \,|\, r_1 \in [0,1], r_2 \in [0,1], |r_1 - r_2| \le 0.1\}$. The range of the function $r_1 - r_2$ over this set is exactly the desired interval $[-0.1, 0.1]$.

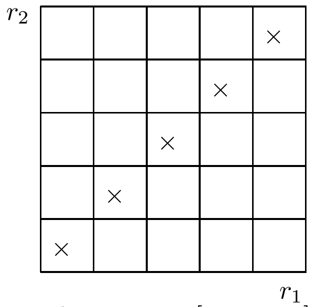
5 Numerical Examples

Let us show that the advantages of the new approach are preserved even when we take into consideration the need to approximate the sets.

5.1 First Example: Computing the Range of $x - x$

As we have mentioned, for $f(x) = x - x$ on $[0,1]$, the actual range is $[0,0]$, but straightforward interval computations lead to an enclosure $[0,1] - [0,1] = [-1,1]$. In straightforward interval computations, we have $r_1 = x$ with the exact interval range $\mathbf{r}_1 = [0,1]$, and we have $r_2 = x$ with the exact interval range $\mathbf{x}_2 = [0,1]$. The variables r_1 and r_2 are dependent, but we ignore this dependence.

In the new approach: we have $\mathbf{r}_1 = \mathbf{r}_2 = [0,1]$, and we also have $\mathbf{r}_{12}$:



For each small box, we have $[-0.2, 0.2]$, so the union is $[-0.2, 0.2]$.

If we divide into more pieces, we get an interval closer to 0.

5.2 Second Example: Computing the Range of $x - x^2$

In straightforward interval computations, we have $r_1 = x$ with the exact interval range interval $\mathbf{r}_1 = [0,1]$, and we have $r_2 = x^2$ with the exact interval range $\mathbf{x}_2 = [0,1]$. The variables r_1 and r_2 are dependent, but we ignore this dependence and estimate $\mathbf{r}_3$ as $[0,1] - [0,1] = [-1,1]$.

In the new approach: we have $\mathbf{r}_1 = \mathbf{r}_2 = [0,1]$, and we also have $\mathbf{r}_{12}$. First, we divide the range $[0,1]$ into 5 equal subintervals $\mathbf{R}_1$. The union of the ranges $\mathbf{R}_1^2$ corresponding to these 5 subintervals $\mathbf{R}_1$ is $[0,1]$, so $\mathbf{r}_2 = [0,1]$. We divide this interval $\mathbf{r}_2$ into 5 equal sub-intervals $[0, 0.2]$, $[0.2, 0.4]$, etc. We now compute the set $\mathbf{r}_{12}$ as follows:

- for $\mathbf{R}_1 = [0, 0.2]$, we have $\mathbf{R}_1^2 = [0, 0.04]$, so only sub-interval $[0, 0.2]$ of the interval $\mathbf{r}_2$ is affected;
- for $\mathbf{R}_1 = [0.2, 0.4]$, we have $\mathbf{R}_1^2 = [0.04, 0.16]$, so also only sub-interval $[0, 0.2]$ is affected;
- for $\mathbf{R}_1 = [0.4, 0.6]$, we have $\mathbf{R}_1^2 = [0.16, 0.25]$, so two sub-intervals $[0, 0.2]$ and $[0.2, 0.4]$ are affected, etc.

r_2

				×
			×	×
			×	
		×	×	
×	×	×		

r_1

For each possible pair of small boxes $\mathbf{R}_1 \times \mathbf{R}_2$, we have $\mathbf{R}_1 - \mathbf{R}_2 = [-0.2, 0.2]$, $[0, 0.4]$, or $[0.2, 0.6]$, so the union of $\mathbf{R}_1 - \mathbf{R}_2$ is $\mathbf{r}_3 = [-0.2, 0.6]$.

If we divide into more and more pieces, we get the enclosure which is closer and closer to the exact range $[0, 0.25]$.

5.3 How to Compute $\mathbf{r}_{ik}$

The above example is a good case to illustrate how we compute the range $\mathbf{r}_{13}$ for $r_3 = r_1 - r_2$. Indeed, since $\mathbf{r}_3 = [-0.2, 0.6]$, we divide this range into 5 subintervals $[-0.2, -0.04]$, $[-0.04, 0.12]$, $[0.12, 0.28]$, $[0.28, 0.44]$, $[0.44, 0.6]$.

- For $\mathbf{R}_1 = [0, 0.2]$, the only possible $\mathbf{R}_2$ is $[0, 0.2]$, so $\mathbf{R}_1 - \mathbf{R}_2 = [-0.2, 0.2]$. This covers $[-0.2, -0.04]$ and $[-0.04, 0.12]$.
- For $\mathbf{R}_1 = [0.2, 0.4]$, the only possible $\mathbf{R}_2$ is $[0, 0.2]$, so $\mathbf{R}_1 - \mathbf{R}_2 = [0, 0.4]$. This interval covers $[-0.04, 0.12]$, $[0.12, 0.28]$, and $[0.28, 0.44]$.
- For $\mathbf{R}_1 = [0.4, 0.6]$, we have two possible $\mathbf{R}_2$:
  - for $\mathbf{R}_2 = [0, 0.2]$, we have $\mathbf{R}_1 - \mathbf{R}_2 = [0.2, 0.6]$; this covers $[0.12, 0.28]$, $[0.28, 0.44]$, and $[0.44, 0.6]$;
  - for $\mathbf{R}_2 = [0.2, 0.4]$, we have $\mathbf{R}_1 - \mathbf{R}_2 = [0, 0.4]$; this covers $[-0.04, 0.12]$, $[0.12, 0.28]$, and $[0.28, 0.44]$.
- For $\mathbf{R}_1 = [0.6, 0.8]$, we have $\mathbf{R}_1^2 = [0.36, 0.64]$, so three possible $\mathbf{R}_2$: $[0.2, 0.4]$, $[0.4, 0.6]$, and $[0.6, 0.8]$, to the total of $[0.2, 0.8]$. Here, $[0.6, 0.8] - [0.2, 0.8] = [-0.2, 0.6]$, so all 5 subintervals are affected.
- Finally, for $\mathbf{R}_1 = [0.8, 1.0]$, we have $\mathbf{R}_1^2 = [0.64, 1.0]$, so two possible $\mathbf{R}_2$: $[0.6, 0.8]$ and $[0.8, 1.0]$, to the total of $[0.6, 1.0]$. Here, $[0.8, 1.0] - [0.6, 1.0] = [-0.2, 0.4]$, so the first 4 subintervals are affected.

r_3

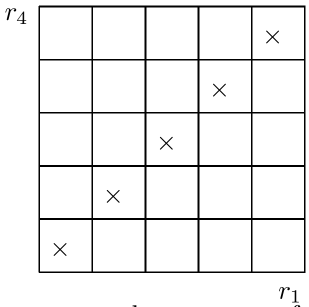
		×	×	
	×	×	×	×
	×	×	×	×
×	×	×	×	×
×		×	×	×

r_1

5.4 Distributivity: $a \cdot (b + c)$ vs. $a \cdot b + a \cdot c$

We want to estimate the range of the function $f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1 \cdot x_3$ when $x_1 \in \mathbf{x}_1 = [0, 1]$, $\mathbf{x}_2 = [1, 1]$, and $\mathbf{x}_3 = [-1, -1]$. The actual range is $[0, 0]$, but straightforward interval computations lead to $[0, 1] \cdot 1 + [0, 1] \cdot (-1) = [0, 1] + [-1, 0] = [-1, 1]$, i.e., to excess width. The reason is that we have exact ranges for $r_4 = x_1 \cdot x_2$ and $r_5 = x_1 \cdot x_3$, but we ignore the dependence between r_4 and r_5.

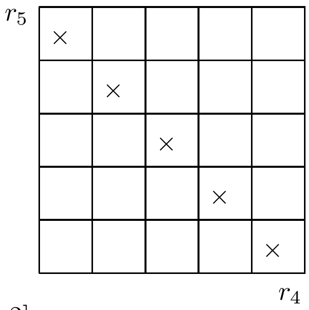
Here, parsing leads to $r_4 = r_1 \cdot r_2$, $r_5 = r_1 \cdot r_3$, and $r_6 = r_4 + r_5$. We start with $\mathbf{r}_1 = [0, 1]$, $r_2 = 1$, and $r_3 = -1$. In the new idea, when we get $r_4 = r_1 \cdot r_2$, we compute the ranges $\mathbf{r}_{14}$, $\mathbf{r}_{24}$, and $\mathbf{r}_{34}$; the only non-trivial range is $\mathbf{r}_{14}$:



For $r_5 = r_1 \cdot r_3$, we get $\mathbf{r}_5 = [-1, 0]$. To compute the range $\mathbf{r}_{45}$, for each possible box $\mathbf{R}_1 \times \mathbf{R}_3$, we:

- consider all boxes $\mathbf{R}_4$ for which $\mathbf{R}_4 \times \mathbf{R}_1$ is possible and $\mathbf{R}_4 \times \mathbf{R}_3$ is possible; and
- add $\mathbf{R}_4 \times (\mathbf{R}_1 \cdot \mathbf{R}_3)$ to the set $\mathbf{r}_{45}$.

The result is as follows:



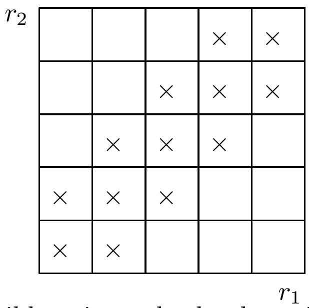
Hence, for $r_6 = r_4 + r_5$, we get $[-0.2, 0.2]$.

If we divide into more pieces, we get the enclosure closer to 0.

5.5 Toy Example with Prior Dependence

The problem is to find the range of $r_1 - r_2$ when $\mathbf{r}_1 = [0, 1]$, $\mathbf{r}_2 = [0, 1]$, and $|r_1 - r_2| \leq 0.1$. Here, the actual range is $[-0.1, 0.1]$, but straightforward interval computations return $[0, 1] - [0, 1] = [-1, 1]$.

In the new approach, first, we describe the constraint in terms of subboxes:



Next, we compute $\mathbf{R}_1 - \mathbf{R}_2$ for all possible pairs and take the union. The result is $[-0.6, 0.6]$.

If we divide into more pieces, we get the enclosure closer to $[-0.1, 0.1]$.

6 Discussion

When we apply straightforward interval computations to a T-step algorithm,

- we need to compute T intervals $\mathbf{r}_i$, $i = 1, \ldots, T$;
- so, it requires $O(T)$ steps.

In the new approach:

- we need to compute T^2 sets $\mathbf{r}_{ij}$, $i, j = 1, \ldots, T$;
- so, it requires $O(T^2)$ steps.

Thus, the new method takes longer than straightforward interval computations, but it is still feasible.

We have already mentioned that the range estimation problem is, in general, NP-hard (even without any dependency between the inputs). This means that no feasible method can completely avoid excess width. In particular, this means that our quadratic time method cannot completely avoid excess width. So sometimes, we will need better estimates.

To get better estimates, in addition to sets of pairs, we can also consider sets of *triples* $\mathbf{r}_{ijk}$. This will be a T^3 time version of our approach. If the use of a full subdivision of each box $\mathbf{r}_i \times \mathbf{r}_j \times \mathbf{r}_k$ into $C \times C \times C$ subboxes requires too much computation time, then, instead of using the full 3-D approach, we can use an intermediate "$2\frac{1}{2}$-D" approach in which we divide each box into $C \times C \times c$ subboxes, with $c \ll C$.

We can also go to *quadruples* with time $O(T^4)$, etc. When we have tuples with as many elements as the number of variables, we get the exact range. Thus, as we planned, we have a sequence of more and more accurate feasible algorithms for estimating the range, the sequence whose algorithm require longer and longer computation time as the accuracy improves.

Comment. Similar ideas can be applied to the case of expert systems, when we have partial information about probabilities [3, 4, 5].

Traditionally, expert systems use technique similar to straightforward interval computations: we parse F and replace each computation step with corresponding probability operation. The problem with this approach is that at each step, we ignore the dependence between the intermediate results F_j. As a result, the resulting intervals of possible values of probability are too wide (or, if we use numerical estimates instead of intervals, these numerical estimates can be way off).

This phenomenon can be illustrated on the simple example of estimating the probability $P(A \vee \neg A)$ when $P(A) = 0.5$. In reality, $A \vee \neg A$ is always true, so this probability should be equal to 1. In the interval-type approach, we parse the expression $A \vee \neg A$ into the following sequence: $F_1 = A$, $F_2 = \neg F_1$, and $F_3 = F_1 \vee F_2$. So, first we conclude that $P(F_1) = 0.5$, then that $P(F_2) = 1 - P(F_1) = 1 - 0.5 = 0.5$. However, when we compute the probability $P(F_1 \vee F_2)$, we ignore the dependence between F_1 and F_2 and only use the fact that $P(F_1) = P(F_2) = 0.5$. In this case, the probability $P(F_1 \vee F_2)$ can take any value from the interval $[0.5, 1]$. This interval is what the system returns – with excess width.

A solution to this problem is that, similarly to the above algorithm, on each intermediate step, besides $P(F_j)$, we also compute $P(F_j \,\&\, F_i)$ (or $P(F_{j_1} \,\&\, \ldots \,\&\, F_{j_k})$). On each step, we use all combinations of l such probabilities to get new estimates. As a result, we get a new technique in which, e.g., $P(A \vee \neg A)$ is always estimated as 1.

The fact that similar ideas work in interval and in probabilistic cases should not be surprising, because the set of possible values $\mathbf{x}_{ij}$ which described the dependence between two interval-valued quantities is a natural analog between copulas – which describe dependence between two random variables; see, e.g., [15].

Acknowledgments

This work was largely inspired by suggestions from Luc Jaulin, Arnold Neumaier, and Bill Walster during the 2005 Scandinavian Workshop on Interval Computations.

This work was supported in part by NASA under cooperative agreement NCC5-209, NSF grants EAR-0225670 and DMS-0532645, Army Research Lab grant DATM-05-02-C-0046, Star Award from the University of Texas System, and Texas Department of Transportation grant No. 0-5453. The authors are thankful to all the participants of the Second International Workshop on Reliable Engineering Computing (Savannah, Georgia, February 22–24, 2006) for valuable discussions.

References

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

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

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

[4] Ceberio, M., V. Kreinovich, S. Chopra, L. Longpré, B. Ludäscher, and C. Baral, Interval-Type and Affine Arithmetic-Type Techniques for Handling Uncertainty in Expert Systems, Journal of Computational and Applied Mathematics, 2007, Vol. 199, No. 2, pp. 403–410.

[5] Chopra, S. *Affine Arithmetic-Type Techniques for Handling Uncertainty in Expert Systems*. Master's Thesis, Department of Computer Science, University of Texas at El Paso, 2005.

[6] Ferson, S. *RAMAS RiskCalc: Risk Assessment with Uncertain Numbers*. CRC Press, Boca Raton, Florida, 2002.

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

[8] Hansen, E. Sharpness in interval computations, *Reliable Computing*, 3:7–29, 1997.

[9] Hole, J. A. Nonlinear High-Resolution Three-Dimensional Seismic Travel Time Tomography. *J. Geophysical Research*, 97(B5):6553–6562, 1992.

[10] Jaulin, L., 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.

[11] Jaynes, E. T. *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, Massachusetts, 2003.

[12] Klir, G. J. *Fuzzy Sets: An Overview of Fundamentals, Applications, and Personal Views*. Beijing Normal University Press, Beijing, 2000.

[13] Kreinovich, V., J. Beck, and H. T. Nguyen, Ellipsoids and Ellipsoid-Shaped Fuzzy Sets as Natural Multi-Variate Generalization of Intervals and Fuzzy Numbers: How to Elicit Them from Users, and How to Use Them in Data Processing, *Information Sciences*, 2007, Vol. 177, No. 2, pp. 388–407.

[14] Kreinovich, V., A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Kluwer, Dordrecht, 1997.

[15] Nelsen, R. B. *Introduction to Copulas*. Springer Verlag, New York, 1999.

[16] Parker, R. L. *Geophysical Inverse Theory*. Princeton University Press, Princeton, New Jersey, 1994.

[17] Pawlak, Z. *Rough Sets*. Kluwer Academic Publishers, Dodrecht, 1991.

[18] Polkowski, L. *Rough sets. Mathematical foundations*. Physica Verlag, A Springer-Verlag Co., Heidelberg, New York, 2002.

[19] Ullman, J. D., and J. Widom, *A First Course in Database Systems*, Prentice Hall, Upper Saddle River, New Jersey, 2002.

[20] Vavasis, S. A. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York, 1991.

[21] Wadsworth Jr., H. M. *Handbook of Statistical Methods for Engineers and Scientists*. McGraw-Hill, N.Y., 1990.

[22] Zelt, C. A., and P. J. Barton, Three-dimensional seismic refraction tomography: A comparison of two methods applied to data from the Faeroe Basin, *J. Geophysical Research*, 103(B4):7187–7210, 1998.

A Open Questions

When is the New Method Exact? It is known that straightforward interval computations produce the exact range for single-use expressions (SUE), in which each variable occurs exactly once; see, e.g., [8, 10]. A natural question is: is there a similar syntactic class of expressions for which our pair-wise method leads to the exact range?

One seemingly natural hypothesis does not work here. Namely, we have shown that our new method leads to the exact range for expressions $x - x$, $x - x^2$, and $x_1 \cdot x_2 + x_1 \cdot x_3$. In all these expressions, each variable occurs no more than twice. It may therefore seem natural to conjecture that the new method is exact for all such "double-use" expressions. Alas, this is not true: it is known (see, e.g., [7]) that computing the range of the variance $V = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - \left(\frac{1}{n} \cdot \sum_{i=1}^{n} x_i\right)^2$ on interval data $\mathbf{x}_i$ is NP-hard. Since variance is an example of a double-use expression, and our algorithm is feasible, we can thus conclude that for some double-use problems, it must lead to excess width.

If we allow prior constraints, then the problem of estimating the range become NP-hard even for SUE expressions with linear SUE constraints. Indeed, we can take an arbitrary non-SUE algebraic expression, replace each occurrence of each variable x_i with different new variables x_{i1}, x_{i2}, $\ldots$ – this will make this expression SUE, and then add SUE linear constraint $x_{i1} = x_{i2}$, $x_{i2} = x_{i3}$, $\ldots$ Under these constraints, the range of the new expression is exactly the same as the range of the original expression, and we already know that computing the range of even quadratic expressions is NP-hard.

What Are the Possible Shapes of $\mathbf{r}_{ij}$? It is easy to show that for 1-D ranges, for algebraic functions $f(x_1, \ldots, x_n)$ (i.e., solutions of polynomial equations with polynomial coefficients), the endpoints of the range intervals are algebraic numbers, and that, vice versa, every interval with algebraic endpoints is a range of an appropriate algebraic function; see, e.g. [14].

It is easy to show that when we have two algebraic functions $f(x_1, \ldots, x_n)$ and $g(x_1, \ldots, x_n)$, then the set of possible values of pairs (f, g) is semi-algebraic (i.e., is described by a finite set of polynomial equalities and inequalities). A natural question is: can every semi-algebraic set in $I\!R^2$ be thus represented? What about sets in $I\!R^3$? in $I\!R^n$ for an arbitrary n?