Towards Fast and Accurate Algorithms for Processing Fuzzy Data: Interval Computations Revisited

Gang Xiang
The University of Texas at El Paso

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

International Journal of General Systems
Vol. 00, No. 00, October 2011, 1–25

RESEARCH ARTICLE

Towards Fast and Accurate Algorithms for Processing Fuzzy Data: Interval Computations Revisited

Gang Xiang[a] and Vladik Kreinovich[b]*

[a] *Philips Healthcare Informatics, 4050 Rio Bravo, Suite 200, El Paso, TX 79902, USA*;
[b] *Department of Computer Science, University of Texas, El Paso, TX 79968, USA*

(*received July 2011*)

In many practical applications, we need to process data – e.g., to predict the future values of different quantities based on their current values. Often, the only information that we have about the current values comes from experts, and is described in informal ("fuzzy") terms like "small". To process such data, it is natural to use fuzzy techniques, techniques specifically designed by Lotfi Zadeh to handle such informal information.

In this survey, we start by revisiting the motivation behind Zadeh's formulas for processing fuzzy data, explain how the algorithmic problem of processing fuzzy data can be described in terms of interval computations (α-cuts). Many fuzzy practitioners claim "I tried interval computations, they did not work" – meaning that they got estimates which are much wider than the desired α-cuts. We show that such statements are usually based on a (widely spread) misunderstanding – that interval computations simply means replacing each arithmetic operation with the corresponding operation with intervals. We show that while such *straightforward* interval techniques indeed often lead to over-wide estimates, the current *advanced* interval computations techniques result in estimates which are much more accurate.

We overview such advanced interval computations techniques, and show that by using them, we can efficiently and accurately process fuzzy data.

We wrote this survey with three audiences in mind. First, we want fuzzy researchers and practitioners to understand the current advanced interval computations techniques and to use them to come up with faster and more accurate algorithms for processing fuzzy data. For this "fuzzy" audience, we explain these current techniques in detail. Second, we also want interval researchers to better understand this important application area for their techniques. For this "interval" audience, we want to explain where fuzzy techniques come from, what are possible variants of these techniques, and what are the problems to which interval techniques can be applied. These readers needs to avoid their own frequent misunderstanding – that fuzzy techniques are "magical" heuristic tools that are only justified by intuition and that have no mathematical justification. Readers of both types can skip the parts they already know.

Finally, we also want to target people who solve practical data processing problems – and who may not be well familiar neither with the fuzzy nor with the interval techniques. We want these readers to get an understanding of both the problem of processing fuzzy data and of the interval techniques for solving this problem.

Keywords: fuzzy data processing, interval computations, algorithms

1. Need for Processing Fuzzy Data

Need for prediction. One of the main objectives of science and engineering is to predict the future state of the world, i.e., to predict the future values of the quantities that characterize this state. For example, we want to predict tomorrow's weather, the future position of a comet, the future location of a spaceship, the result of a chemical reaction, etc. To predict these values, we need to know the

*Corresponding author. Email: vladik@utep.edu

ISSN: 0308-1079 print/ISSN 1563-5104 online
© 2011 Taylor & Francis
DOI: 10.1080/0308107YYxxxxxxxx
http://www.informaworld.com

current values of these quantities, and we need to know how these values change with time.

In many situations, we know the equations that describe the time change. In such situations, for each of the desired future values y, we have an algorithm $y = f(x_1, \dots, x_n)$ that computes (and estimate for) this value based on the current (and past) values $x_1, \dots, x_n$ of this and related quantities. Thus, to predict the desired value, we:

- *estimate* the input data, i.e., the values $x_1, \dots, x_n$, and
- *process* the input data, i.e., apply the algorithm f to these estimates.

In simple cases, e.g., for a simple trajectory, this algorithm $f(x_1, \dots, x_n)$ may be as simple as a straightforward formula. In other cases – e.g., in weather prediction – this algorithm is based on solving a complex system of partial differential equations.

Need for indirect estimations. Another case when we need data processing is when we are interested in the value of a physical quantity which is very difficult (or even impossible) to measure or estimate directly.

For example, if we are interested in measuring the distance between two points on a desk, we can use a simple ruler. If we are interested in measuring the distance between two buildings on campus, we can simply walk from one to another, count the number of steps, and thus, measure the distance. If we are interested in the distance between the two nearby cities, we can drive from one to another and measure the distance covered by a car. However, if we want to measure the distance to a nearby star, it is not yet possible to measure it directly by simply traveling there :-(Such a distance is usually measured *indirectly* – e.g., by measuring the angle in the direction of this star in two different seasons, when the Earth is on the different sides of the Sun. Once we know these angles, we can use trigonometry to find the desired distance.

Similarly, when we want to know how much water is in a bottle, we can measure this amount directly – by weighing it. However, if we want to know how much oil is in an oil field, we cannot measure it directly, we have to do it *indirectly* – e.g., by sending seismic signals in different directions and recording the times that these signals take to travel between different locations, and then use these times to estimate the speed of sound at different points – and thus, to estimate the amount of oil by computing the total volume covered by all the location in which, based on the speed of sound, we conclude that this location contains oil.

In all such cases, in order to find the value of the desired difficult-to-measure quantity y, we estimate the values of several related easier-to-measure quantities $x_1, \dots, x_n$, and then use the known relation $y = f(x_1, \dots, x_n)$ between x_i and y to estimate the desired value y. In other words, in such cases, we also need data processing.

In this case too, the corresponding algorithm f can be straightforward – as in trigonometric formulas for the distance to a star – or complex – as in solving the system of partial differential equations that describes how a seismic signal propagates through different parts of the Earth.

Where do input values come from: need for processing imprecise data. Often, the estimates for the input values $x_1, \dots, x_n$ come from measurements. However, frequently, measurements are difficult to perform, so we have to rely on expert estimates for these values.

For example, in geosciences, it is rarely possible to get a reasonable description of the Earth structure based only on measurements – an expert geophysicist needs to supplement the easier-to-measure values in closer-to-surface areas with expert estimate of the difficult-to-measure deep structures. Similarly, in medicine, to diagnose a patient, it is often not enough to have the results of all the tests – it is often necessary to supplement these results with the expertise of medical doctors.

Expert estimates are usually imprecise, experts use informal words from natural language. For example, an expert can say that some value is “small”, or, more precisely “close to 3.0, with an accuracy about 0.5”.

Based on such informal, “fuzzy” information about the inputs $x_1, \ldots, x_n$, what can we say about the desired value $y = f(x_1, \ldots, x_n)$? In other words, how can we process such imprecise (fuzzy) data?

2. How to Describe Fuzzy Uncertainty: Reminder

Fuzzy techniques – a natural way of describing imprecise data. In the early 1960s, L. Zadeh proposed a technique that naturally describes such “fuzzy” data. The main idea behind these *fuzzy* techniques is as follows; see, e.g., Zadeh (1965), Klir and Yuan (1995), Nguyen and Walker (2006).

When we have a precise property, such as “less than 5”, then each possible value either satisfies this property or not. Thus, a precise property P can be described by its *characteristic function* $\chi_P(x)$, i.e., a function that assigns, to each possible real number x, the value “true” or “false” depending on whether the value x satisfies the property P. Our main objective is to process the data – i.e., to use computers. In the computers, everything is represented as 0s and 1s. In particular, “true” is usually represented as 1 and “false” as 0. As a result, it is reasonable to describe a characteristic function so that $\chi_P(x) = 1$ if the value x satisfies the property P and $\chi_P(x) = 0$ if the value x does not satisfy this property.

For an informal property like “small”, for some values x, we are absolutely sure that this value is small, for some other values x, we are absolutely sure that x is not small, but for many values x, a natural answer to the question “is x small?” is “to some extent”. How can we describe this extent? This problem is very similar to another problem for which the solution is well known: describing the degree of satisfaction with a customer service, with a product, etc., something that most of us describe in numerous polls. Usually, in such polls, we are asked to describe our degree of satisfaction on a scale from, say, 0 to 10, 0 being absolutely unsatisfied and 10 being fuzzy satisfied. This is also how we estimate the quality of a paper when refereeing, this is how students evaluate their professors, etc.

In other words, a natural way to estimate the degree to which a value x satisfies some informal property P is to assign a number. In a poll, this number goes from 0 to 5 or from 0 to 10, etc. However, we would like to make sure that in the case of precise knowledge, we get $\chi_P(x) = 0$ and $\chi_P(x) = 1$. In the poll, 0 corresponds to “absolutely false” ($\chi_P(x) = 0$) and, say, 10 corresponds to “absolutely true” ($\chi_P(x) = 1$). So, to match these two scales, we need to *re-scale* our values – e.g., by dividing the value marked by an expert (such as 7 on a scale from 0 to 10) by the largest possible value of this scale (i.e., by 10 in the above example).

No matter what re-scaling we use, we get a value from the interval $[0, 1]$ that describes to what extent the value x satisfies the given property. This value (degree) to which the value x satisfied the property P is usually denoted by $\mu_P(x)$, and the function μ_P that assigns, to each value x, this degree is called a *membership function* or a *fuzzy set*.

Need for logical operations with fuzzy degrees. Fuzzy (imprecisely defined) statements S, S', …, are often combined by logical connectives like “and” (&), “or” ($\vee$), and “not” ($\neg$): e.g., an expert rule for controlling a car can be that if the car in front is close *and* we are going fast, it is desirable to somewhat slow down. In order to formalize such rules, we need to be able to formalize such logical combinations $S \,\&\, S'$ and $S \vee S'$, i.e., to assign degree of belief to such statements.

When there is no imprecision, i.e., when each of the statements S and S' is either absolutely true or absolutely false, the truth values of S and S' uniquely determine the truth value of the logical combinations $S \,\&\, S'$ and $S \vee S'$. However, in case of uncertainty, the expert's degree of certainty in $S \,\&\, S'$ depends not only on the expert's degrees of certainty in statements S and S', but also on whether these statements are independent or related. For example, if the two witnesses made exactly opposite statements S and S', i.e., that the criminal was tall and that he was short, then the police's degree of confidence in the statement $S \,\&\, S'$ (meaning that both witnesses were correct) is 0. However, if two witnesses both claim with confidence that the criminal was tall (i.e., if $S = S'$), then the police's degree of confidence that both witnesses are correct is high.

So, ideally, in addition to asking an expert to estimate the degree of certainty for each individual statement like "a person of height 175 cm is tall", we should also ask the experts to estimate the degree of confidence in all possible logical combinations of such statements, in particular, if we have n basic statements $S_1, \ldots, S_n$, we should ask the expert to estimate his or her degree of confidence in all 2^n combinations of the type $S_1^{\varepsilon_1} \,\&\, \ldots\, S_n^{\varepsilon_n}$, where each ε_i can take values $-$ and $+$, S_i^+ means S_i, and S_i^- means $\neg S_i$. In real life, the number of expert statements n is huge, and even for moderate $n \approx 300$, the number 2^n of such possible combinations exceeds the number of particles in the Universe – no way we can ask that many questions to an expert.

So, we face a following problem:

- We know the expert's degree of confidence $a = d(S)$ in a statement S, and we know the expert's degree of confidence $a' = d(S')$ in a statement S'.
- Based on the two values a and a', we must provide an estimate for the expert's degree of confidence in a statement $S \,\&\, S'$.

The estimate for this degree corresponding to given values a and a' is usually denoted by $f_\&(a, a')$. The function $f_\&(a, a')$ that provides such estimates is called an *and-operation*, or (for historic reasons) a *t-norm.*

Similarly, we need to estimate the expert's degree of confidence in a statement $S \vee S'$. This estimate is usually denoted by $f_\vee(a, a')$, and the corresponding function is called an *or-operation* or a *t-conorm.*

Comment. Of course, as we mentioned earlier, for the same values of $a = d(S)$ and $a' = d(S')$, we can get *different* actual expert's degree of confidence in a composite statement $S \,\&\, S'$. In other words, in reality, the degree of confidence in $S \,\&\, S'$ is *not* a function of a and a'. However, as we mentioned, we need to select *one* of these values as our estimate $f_\&(a, a')$ – i.e., to select a function.

Simplest selection of t-norms and t-conorms. In principle, we can have many different t-norms and t-conorms. In practice, it makes sense to select t-norms and t-conorms which are in good agreement with common sense.

Let us start with t-norms. If we already know that the statement S' is absolutely true (i.e., if $a' = d(S') = 1$), then the composite statement $S \,\&\, S'$ ("S and S'") is true if and only if is S is true. Thus, it is reasonable to require that in this case, the resulting estimate $f_\&(a, 1)$ for the expert's degree of confidence in $S \,\&\, S'$ should exactly coincide with the degree $a = d(S)$ of certainty in S: $f_\&(a, 1) = a$. Similarly, it is reasonable to require that $f_\&(1, a') = a'$ for all a'.

When we have two identical statements $S' = S$, with $a = d(S) = a' = d(S')$, then $S \,\&\, S'$ is simply equivalent to S. Thus, it is reasonable to require that the resulting degree of confidence $f_{\&}(a,a)$ in $S \,\&\, S'$ should be equal to the degree of confidence a in the statement S, i.e., that $f_{\&}(a,a)=a$ for all a.

Finally, if our degree of confidence in one of the statements S and S' increases, then intuitively, our degree of confidence in the composite statements $S \,\&\, S'$ should also increase (or at least not decrease). Thus, it is reasonable to require that the function $f_{\&}(a,a')$ be an increasing (or at least non-decreasing) function of both its variables: if $a \le a_1$ and $a' \le a'_1$, then $f_{\&}(a,a') \le f_{\&}(a_1,a'_1)$.

Let us show that these reasonable requirements uniquely determine the value of the t-norm $f_{\&}(a,a')$ for all possible combinations of degrees $a,a' \in [0,1]$.

- When $a \le a'$, hen, by monotonicity, $f_{\&}(a,a) \le f_{\&}(a,a') \le f_{\&}(a,1)$. We know that $f_{\&}(a,a)=a$ and that $f_{\&}(a,1)=a$, so $a \le f_{\&}(a,a') \le a$, i.e., $f_{\&}(a,a')=a$.
- Similarly, when $a' \le a$, we have $f_{\&}(a',a') \le f_{\&}(a,a') \le f_{\&}(1,a')$ hence $a' \le f_{\&}(a,a') \le a'$ and $f_{\&}(a,a')=a'$.

In both cases, $f_{\&}(a,a')=\min(a,a')$. This is the t-norm that we will be using.

For t-conorms (or-operations), we can use similar arguments. Indeed, if we know that a statement S' is false, then the statement "S or S'" is true if and only if S is true. In this case, it is reasonable to require that the resulting estimate $f_{\vee}(a,0)$ for the expert's degree of confidence in $S \vee S'$ should exactly coincide with the degree $a=d(S)$ of certainty in S: $f_{\vee}(a,0)=a$. Similarly, it is reasonable to require that $f_{\vee}(0,a')=a'$ for all a'.

When we have two identical statements $S'=S$, with $a=d(S)=a'=d(S')$, then $S \vee S'$ is simply equivalent to S. Thus, it is reasonable to require that the resulting degree of confidence $f_{\vee}(a,a)$ in $S \vee S'$ should be equal to the degree of confidence a in the statement S, i.e., that $f_{\vee}(a,a)=a$ for all a.

Finally, if our degree of confidence in one of the statements S and S' increases, then intuitively, our degree of confidence in the composite statements $S \vee S'$ should also increase (or at leats not decrease). Thus, it is reasonable to require that the function $f_{\vee}(a,a')$ be an increasing (or at least non-decreasing) function of both its variables: if $a \le a_1$ and $a' \le a'_1$, then $f_{\vee}(a,a') \le f_{\vee}(a_1,a'_1)$.

Let us show that these reasonable requirements uniquely determine the value of the t-conorm $f_{\vee}(a,a')$ for all possible combinations of degrees $a,a' \in [0,1]$. Indeed, we can have either $a \le a'$ or $a' \le a$.

- When $a' \le a$, then, by monotonicity, $f_{\vee}(a,0) \le f_{\vee}(a,a') \le f_{\vee}(a,a)$. We know that $f_{\vee}(a,0)=a$ and that $f_{\vee}(a,a)=a$, so $a \le f_{\vee}(a,a') \le a$, i.e., $f_{\vee}(a,a')=a$.
- Similarly, when $a \le a'$, we have $f_{\vee}(0,a') \le f_{\vee}(a,a') \le f_{\vee}(a',a')$ hence $a' \le f_{\vee}(a,a') \le a'$ and $f_{\vee}(a,a')=a'$.

In both cases, $f_{\vee}(a,a')=\max(a,a')$. This is the t-conorm that we will be using.

Comment. In our analysis, we considered the most general case, when we have no information about the possible relation between different statements. We showed that in this general case, if we need to select some t-norm and some t-conorm, then minimum and maximum are a reasonable choice.

In specific applications, however, we may have additional information about this relationship; in such situations, different t-norms and t-conorms are more adequate. For example, such situations occur in control applications; see, e.g., Nguyen and Kreinovich (1998).

3. How to Process Fuzzy Data: Derivation of Zadeh's Extension Principle

Processing fuzzy data: reminder. We are now ready to describe the problem of processing fuzzy data. In this problem:

- we know the algorithm $y = f(x_1, \ldots, x_n)$ that transforms the values of n input quantities into the estimate for the desired quantity y;
- we also have expert estimates about each of the inputs x_i, estimates which are described by the corresponding membership functions $\mu_i(x_i)$;
- our objective is to find, for every real number y, the degree $\mu(y)$ to which this number y is a possible value of the desired quantity.

Intuitively, y is a possible value of the desired quantity if for some values $x_1, \ldots, x_n$:

- x_1 is a possible value of the 1st input quantity,
- and x_2 is a possible value of the 2nd input quantity,
- $\ldots$,
- and $y = f(x_1 \ldots, x_n)$.

We know:

- that the degree of confidence that x_1 is a possible value of the 1st input quantity is equal to $\mu_1(x_1)$,
- that the degree of confidence that x_2 is a possible value of the 2nd input quantity is equal to $m_2(x_2)$, etc.

The degree of confidence $d(y, x_1, \ldots, x_n)$ in an equality $y = f(x_1 \ldots, x_n)$ is, of course, 1 or 0, depending on whether this equality is true or not.

As we have mentioned, a natural way to represent "and" is to use min. Thus, for each combination of values $x_1, \ldots, x_n$, the degree of confidence d in a composite statement

"x_1 is a possible value of the 1st input quantity, and x_2 is a possible value of the 2nd input quantity, $\ldots$, and $y = f(x_1 \ldots, x_n)$"

is equal to $d = \min(\mu_1(x_1), \mu_2(x_2), \ldots, d(y, x_1, \ldots, x_n))$. We can simplify this expression if we consider two possible cases:

- when $y = f(x_1 \ldots, x_n)$, we get $d = \min(\mu_1(x_1), \mu_2(x_2), \ldots, d(y, x_1, \ldots, x_n))$;
- otherwise, we get $d = 0$.

We want to combine these degrees of belief into a single degree of confidence that for some values $x_1, \ldots, x_n$,

- x_1 is a possible value of the 1st input quantity,
- and x_2 is a possible value of the 2nd quantity, $\ldots$,
- and $y = f(x_1 \ldots, x_n)$.

The words "for some values $x_1, \ldots, x_n$" means that the following composite property hold

- either for one combination of real numbers $x_1, \ldots, x_n$,
- or from another combination, etc.

As we have mentioned, a natural way to represent "or" is to use max. Thus, the desired degree of confidence $\mu(y)$ is equal to the largest of the degrees corresponding to different x_i:

$$\mu(y) = \sup_{x_1, \ldots, x_n} \min(\mu_1(x_1), \mu_2(x_2), \ldots, d(y, x_1, \ldots, x_n)).$$

We know that the degree $\min(\mu_1(x_1), \mu_2(x_2), \ldots, d(y, x_1, \ldots, x_n))$ is non-zero only when $y = f(x_1 \ldots, x_n)$. It is therefore sufficient to only take supremum over such combinations. For such combinations, we can omit the term $d(y, x_1, \ldots, x_n)$ in the maximized expression. So, we arrive at the following formula:

$$\mu(y) = \sup\{\min(\mu_1(x_1), \mu_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\}.$$

This formula was first proposed by L. Zadeh and is thus called *Zadeh's extension principle.* This is the main formula that describes knowledge processing under fuzzy uncertainty.

4. Reduction to Interval Computations

From general fuzzy sets to fuzzy numbers. General fuzzy sets can be very complex. In most practical situations, we are only interested in properties P describing numerical values, like "close to x_0" or "small" for which the corresponding degree of confidence $\mu_P(x)$ first increases, until some value $x = x_0$, and then starts decreasing. Since such properties come from describing *numerical* values, the fuzzy sets with such an increase-then-decrease property are called *fuzzy numbers.*

Because fuzzy numbers are most important in practice, in this paper, we will only consider the case when all the membership functions $\mu_i(x_i)$ are fuzzy numbers.

An alternative way to describe fuzzy numbers: α-cuts. By definition of a fuzzy number, for each fuzzy number $\mu(x)$ and for each real number $\alpha \in (0, 1]$, the set $\mathbf{x}(\alpha) \stackrel{\text{def}}{=} \{x : \mu(x) \geq \alpha\}$ is an interval. This interval is called an *α-cut* of the original fuzzy number.

It is worth mentioning that the α-cuts corresponding to different values α form an alternative way of representing this number, in the sense that if we know α-cuts for all α, then we can uniquely reconstruct the original membership function $\mu(x)$. Indeed, for each x, we can reconstruct $\mu(x)$ as the largest α for which $x \in \mathbf{x}(\alpha)$.

Thus, to describe the desired membership function $\mu(y)$, it is sufficient to describe the corresponding α-cuts $\mathbf{y}(\alpha)$. Similarly, instead of assuming that we know each original membership function $\mu_i(x_i)$, we can equivalently assume that for each i and for each α, we know the corresponding α-cut $\mathbf{x}_i(\alpha)$.

Zadeh's extension principle reformulated in terms of α-cuts: analysis. In order to use the α-cut representation for processing fuzzy data, we must reformulate Zadeh's extension principle in terms of α-cuts. In other words, for each α, we need to describe when $\mu(y) \geq \alpha$, i.e., when

$$\sup\{\min(\mu_1(x_1), \mu_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\} \geq \alpha.$$

One can show that in many reasonable cases, when the membership functions are continuous and the corresponding set of possible values are bounded, the largest value is attained for some values $x_1, \ldots, x_n$ for which $y = f(x_1, \ldots, x_n)$ – this is a known fact in mathematical analysis. In such cases, the largest value $\mu(y)$ is larger than or equal to α if and only one of the values $\min(\mu_1(x_1), \mu_2(x_2), \ldots)$ is $\geq \alpha$. Indeed, if one of the values $\min(\mu_1(x_1), \mu_2(x_2), \ldots)$ is $\geq \alpha$ for some $x_1, \ldots, x_n$, then $\mu(y)$, which is the largest of these values, is $\geq \alpha$. Vice versa, if the maximum $\mu(y)$ of the values $\min(\mu_1(x_1), \mu_2(x_2), \ldots)$ is $\geq \alpha$, then we have $\min(\mu_1(x_1), \mu_2(x_2), \ldots) \geq \alpha$ for the values $x_1, \ldots, x_n$ for which this maximum is attained.

When is $\min(\mu_1(x_1), \mu_2(x_2), \ldots) \geq \alpha$? Similarly to the above argument, the *smallest* of several numbers $\mu_i(x_i)$ is larger than or equal to α if and only *all* these numbers are $\geq \alpha$.

Indeed, if all the values $\mu_i(x_i)$ are $\geq \alpha$, then the smallest of these values is also $\geq \alpha$. Vice versa, if even the smallest of these numbers is larger than or equal to α, this means than every other value $\mu_i(x_i)$ – which is larger than or equal to this smallest number – is also larger than or equal to α.

Thus, $\mu(y) \geq \alpha$ if and only if there exist values $x_1, \ldots, x_n$ for which $y = f(x_1, \ldots, x_n)$, and for which $\mu_1(x_1) \geq \alpha$, ..., and $\mu_n(x_n) \geq \alpha$. In terms of α-cuts, $\mu(y) \geq \alpha$ means that $y \in \mathbf{y}(\alpha)$, and $\mu_i(x_i) \geq \alpha$ means that $x_i \in \mathbf{x}_i(\alpha)$. So, y is an element of the set $\mathbf{y}(\alpha)$ if and only if there exist values $x_i \in \mathbf{x}_i(\alpha)$ for which $y = f(x_1, \ldots, x_n)$. In other words, the α-cut for y is equal to the *range* of possible values of the function $f(x_1, \ldots, x_n)$ when x_i belongs to the corresponding α-cuts:

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

Zadeh's extension principle reformulated in terms of α-cuts: result. The above analysis shows to describe the result of fuzzy data processing, we must be able to solve, for each α, the following problem:

- *Given:* n intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and a data processing algorithm $y = f(x_1, \ldots, x_n)$.
- *Find:* the range $\{\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$.

In this problem, in effect, we need to perform computations on *intervals.* This problem is therefore called the problem of *interval computations.*

It is important to avoid a misunderstanding of the term "interval computations". Before we go further, we need to explain that there is a frequent misunderstanding of interval computations. As we have mentioned, interval computations is the name of a *problem.* No matter how we solve this problem, whether we use calculus or any specific interval techniques, or any new method that is still being developed now – in all these cases, we solve the problem of interval computations. However, often, when mentioning "interval computations", practitioners mean specific *techniques* for solving this problem — usually, the simplest of these techniques, so-called "naive" interval computations.

At first glance, this is a simple easy-to-resolve situation: two different groups – interval computations researchers and fuzzy practitioners – use the same term "interval computations" in two different meanings. To interval computations researchers, this is the name of a general problem – and thus, the name of all possible techniques that can be used to solve this problem. To fuzzy practitioners, this is a name for one specific technique. Such different meanings happen in science, and usually do not lead to any serious problems – once the difference in meaning is well understood.

However, in our case, the problem lies deeper: because of this misunderstanding, fuzzy practitioners often completely ignore all the efficient techniques that interval computation folks have invented to solve the general problem – because they mistakenly assume that anything published under the title "interval computations" simply repeats the simple technique that they know under this name. Not only fuzzy practitioners miss these efficient techniques, they also try to invent their own techniques for processing interval data – and sometimes "re-invent the wheel" and come up with "new" algorithms whose ideas have actually been known in the interval community for decades.

One of the main objectives of this paper is to explain this misunderstanding and to explain that interval computations go way beyond the naive techniques. So please do not assume that since we mentioned the term "interval computations", we will be only promoting naive techniques. Please continue reading, we are almost ready to start describing these efficient techniques.

Before we describe these techniques, we first want to explain where this problem of "interval computations" came from – a problem that was analyzed and solved before the main ideas of fuzzy were developed.

5. Practical Origins of Interval Computations

Need for data processing. In science and engineering, we want to *understand* how the world works, we want to *predict* the results of the world processes, and we want to *design* a way to control and change these processes so that the results will be most beneficial for the humankind.

For example, in meteorology, we want to know the weather now, we want to predict the future weather, and – if, e.g., floods are expected, we want to develop strategies that would help us minimize the flood damage.

Usually, we know the equations that describe how these systems change in time. Based on these equations, engineers and scientists have developed algorithms that enable them to predict the values of the desired quantities – and find the best values of the control parameters. As input, these algorithms take the current and past values of the corresponding quantities.

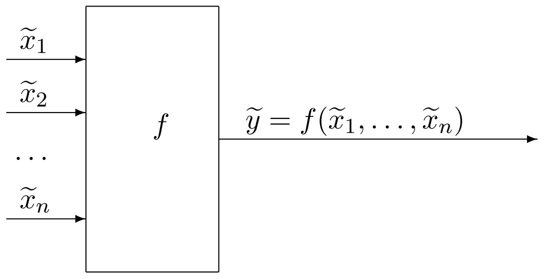
For example, if we want to predict the trajectory of the spaceship, we need to find its current location and velocity, the current position of the Earth and of the celestial bodies, then we can use Newton's equations to find the future locations of the spaceship.

In many situations – e.g., in weather prediction – the corresponding computations require a large amount of input data and a large amount of computations steps. Such computations (*data processing*) are the main reason why computers were invented in the first place – to be able to perform these computations in reasonable time.

Need to take input uncertainty into account. In all the data processing tasks, we start with the current and past values $x_1, \ldots, x_n$ of some quantities, and we use a known algorithm $f(x_1, \ldots, x_n)$ to produce the desired result $y = f(x_1, \ldots, x_n)$.

The values x_i come from measurements, and measurements are never absolutely accurate: the value $\widetilde{x}_i$ that we obtained from measurement is, in general, different from the actual (unknown) value x_i of the corresponding quantity. For example, if the clock shows 12:20, it does not mean that the time is *exactly* 12 hours, 20 minutes and 00.0000 seconds: it may be a little earlier or a little later than that.

As a result, in practice, we apply the algorithm f not to the actual values x_i, but to the *approximate* values $\widetilde{x}_i$ that come from measurements:



So, instead of the ideal value $y = f(x_1, \ldots, x_n)$, we get an approximate value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$. A natural question is: how do approximation errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$ affect the resulting error $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$? Or, in plain words, how to take input uncertainty into account in data processing?

From probabilistic to interval uncertainty. Manufacturers of the measuring instruments provide us with bounds Δ_i on the (absolute value of the) measurement errors: $|\Delta x_i| \le \Delta_i$. If now such upper bound is known, then the device is *not* a measuring instrument; see, e.g., Rabinovich (2005).

For example, a street thermometer may show temperature that is slightly different from the actual one. Usually, it is OK if the actual temperature is +24 but the thermometer shows +22 – as long as the difference does not exceed some reasonable value Δ. But if the actual temperature is +24 but the thermometer shows −5, any reasonable person would return it to the store and request a replacement.

Once we know the measurement result $\widetilde{x}_i$, and we know the upper bound Δ_i on the measurement error, we can conclude that the actual (unknown) value x_i belongs to the interval $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. For example, if the measured temperature is $\widetilde{x}_i = 22$, and the manufacturer guarantees the accuracy $\Delta_i = 3$, this means that the actual temperature is somewhere between $\widetilde{x}_i - \Delta_i = 22 - 3 = 19$ and $\widetilde{x}_i + \Delta_i = 22 + 3 = 25$.

Often, in addition to these bounds, we also know the *probabilities* of different possible values Δx_i within the corresponding interval $[-\Delta_i, \Delta_i]$. This is how uncertainty is usually handled in engineering and science – we assume that we know the probability distributions for the measurement errors Δx_i (in most cases, we assume that this distribution is normal), and we use this information to describe the probabilities of different values of Δy. However, there are two important situations when we do not know these probabilities:

- cutting-edge measurements, and
- cutting-cost manufacturing.

Indeed, how do we determine the probabilities? Usually, to find the probabilities of different values of the measurement error $\Delta x_i = \widetilde{x}_i - x_i$, we bring our measuring instrument to a lab that has a "standard" (much more accurate) instrument, and compare the results of measuring the same quantity with two different instruments: ours and a standard one. Since the standard instrument is much more accurate, we can ignore its measurement error and assume that the value X_i that it measures is the actual value: $X_i \approx x_i$. Thus, the difference $\widetilde{x}_i - X_i$ between the two measurement results is practically equal to the measurement error $\Delta x_i = \widetilde{x}_i - x_i$. So, when we repeat this process several times, we get a histogram from which we can find the probability distribution of the measurement errors.

However, in the above two situations, this is not done. In the case of cutting-edge measurements, this is easy to explain. For example, if we want to estimate the measurement errors of the measurement performed by a Hubble space telescope (or by the newly built CERN particle collider), it would be nice to have a "standard", five times more accurate telescope floating nearby – but Hubble is the best we have. In manufacturing, in principle, we can bring every single sensor to the National Institute of Standards and determine its probability distribution – but this would cost a lot of money: most sensors are very cheap, and their "calibration" using the expensive super-precise "standard" measuring instruments would cost several orders of magnitude more. So, unless there is a strong need for such calibration – e.g., if we manufacture a spaceship – it is sufficient to just use the upper bound on the measurement error.

In both situations, after the measurements, the only information that we have about the actual value of x_i is that this value belongs to the interval $[\underline{x}_i, \overline{x}_i] = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.

Different possible values x_i from the corresponding intervals lead, in general, to different values of $y = f(x_1, \ldots, x_n)$. It is therefore desirable to find the range of all possible values of y, i.e., the set

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

(Since the function $f(x_1, \ldots, x_n)$ is usually continuous, its range is the interval.) Thus, we arrive at the same *interval computations* problem.

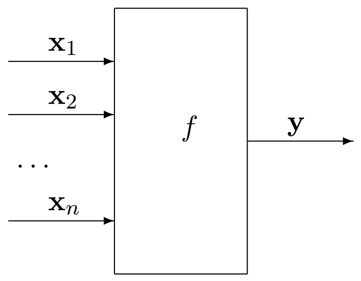
6. How to Solve Interval Computation Problems

The main problem: reminder. We are given:

- an integer n;
- n intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$, ..., $\mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$, 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

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



A brief history of interval computations. The need to provide guaranteed bounds can be traced to the ancient Greeks. For example, when estimating the value π – defined as the circumference of a circle with unit diameter – Archimedes computed the lower and upper bound for π by computing the circumferences of subscribing and circumscibing polygons. Particular cases of interval computations can be traced throughout the whole history of mathematics.

Systematic development started in the 1950s, with the development of modern computers. The main interval computation techniques were invented independently by three researchers from different countries: Ramon Moore from the US, Mieczysław Warmus from Poland, and Teruo Sunaga from Japan. Ramon Moore is probably the most well known, because he not only developed these ideas theoretically, he actively applied them at the Lokheed Company where he was working, and these applications – in particular, to computing trajectories of space flights – brought a lot of publicity to the interval computations ideas.

But why do we need interval computations? Cannot we just use calculus? What are these ideas? Because we start explaining them, we need to answer a natural question: why do we need any new ideas in the first place? From the purely mathematical viewpoint, solving the main problem of interval computations simply means that we solve two straightforward optimization problems:

- to find the lower endpoint $\underline{y}$ of the desired interval, we need to minimize $f(x_1, \ldots, x_n)$ under the constraints $\underline{x}_i \le x_i \le \overline{x}_i$;
- to find the upper endpoint $\underline{y}$ of the desired interval, we need to maximize $f(x_1, \ldots, x_n)$ under the constraints $\underline{x}_i \le x_i \le \overline{x}_i$.

From calculus, it is well known how to minimize a function: with respect to each variable x_i, the minimum is attained either at one of the endpoints $\underline{x}_i$ or $\overline{x}_i$ of the corresponding interval $[\underline{x}_i, \overline{x}_i]$, or – if this minimum is attained inside this interval – it is attained at a point where the partial derivative is equal to 0: $\dfrac{\partial f}{\partial x_i} = 0$. For example, if the function f is quadratic, the derivative is linear, and so we end up with a system of linear equations.

However, for each of n variables x_i, we have three possible linear equations:

- the equation $x_i = \underline{x}_i$ corresponding to the left endpoint of the interval,
- the equation $x_i = \overline{x}_i$ corresponding to the right endpoint of the interval, and
- the equation $\dfrac{\partial f}{\partial x_i} = 0$ corresponding to the interior points of the interval.

For two variables x_1 and x_2, for each of the three possibilities for x_1, we have three possibilities for x_2, so we have a total of $3 \times 3 = 3^2 = 9$ possible systems of linear equations. For three variables x_1, x_2, and x_3, we similarly have $3 \times 3 \times 3 = 3^3$ possible systems of linear equations. In general, for n variables, we have 3^n possible systems. While solving each system of linear equations is easy, solving 3^n of them requires exponential time and is – as we have mentioned earlier – not feasible for large n.

Interval computations is NP-hard. The standard calculus approach to the interval computation problem takes unrealistic exponential time. Is this a drawback of a method – so that other methods can solve this problem in feasible time? It turns out that it is the property of a problem itself. Specifically, it can be proven that, in general, the interval computation problem is NP-hard even for quadratic functions $f(x_1, \ldots, x_n)$; see, e.g., Ferson at el. (2002, 2005). Informally, NP-hard means that – provided that P≠NP, which most computer scientists believe to be true – no algorithm is possible that always solves the interval computation problem in feasible time.

Thus, this complexity is a property of a *problem* – no matter what method we propose for solving this algorithm, if this method is feasible, it will sometimes only provide an *approximate* range of the given function on given intervals.

What does the interval computations community do. Since the problem of interval computation is, in general, NP-hard, we cannot simply invent one feasible algorithm that would exactly solve all the cases of this problem. Since, as we have mentioned, interval computations problem do appear in practice – e.g., when processing fuzzy data – there is a need to solve such problems. So, researchers try their best:

- to find feasible algorithms that exactly solve practically important cases of the interval computation problems, and
- if that is not possible, at least come up with feasible algorithms that provide with as good an approximation to the desired range as possible.

In this paper, we provide a brief overview of some of the related techniques – techniques that are, in our opinion, under-used in fuzzy data processing.

Comment. In applications to fuzzy data processing, approximate algorithms make perfect sense – since the input intervals come from fuzzy (expert) estimates, and these estimates are approximate in the first place. In fuzzy techniques, we use exact values of the membership functions, but in reality, an expert cannot meaningfully distinguish between, say, degree of certainty 0.70 or degree of certainty 0.71.

Interval arithmetic: simplest case of interval computations. Let us start with the simplest case of interval computations, when data processing $f(x_1, x_2)$ consists of applying one arithmetic operation: addition, subtraction, multiplication, or division. In this case, we have explicit formulas for the range.

For addition and multiplication, these formulas comes from the fact that the corresponding functions $f(x_1, x_2)$ are monotonic in each of the variables. For example, addition $f(x_1, x_2) = x_1 + x_2$ is an increasing function of both variables x_1 and x_2. Thus, when x_1 is in the interval $[\underline{x}_1, \underline{x}_1]$ and x_2 is in the interval $[\underline{x}_2, \overline{x}_2]$, the smallest possible value $\underline{y}$ of $y = x_1 + x_2$ is attained when both variables attains their smallest possible values $\underline{x}_1$ and $\underline{x}_2$, i.e., $\underline{y} = \underline{x}_1 + \underline{x}_2$. Similarly, the largest possible value $\overline{y}$ of $y = x_1 + x_2$ is attained when both variables attains their largest possible values $\overline{x}_1$ and $\overline{x}_2$, i.e., $\overline{y} = \overline{x}_1 + \overline{x}_2$. Therefore, the range of the sum has the form

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

For example, if Gang has between 5 and 10 dollars, and Vladik has between 7 and 15 dollars, then together, we have between $5 + 7 = 12$ and $10 + 15 = 25$ dollars.

Subtraction $f(x_1, x_2) = x_1 - x_2$ is an increasing function of x_1 and a decreasing function of x_2. Thus, when x_1 is in the interval $[\underline{x}_1, \underline{x}_1]$ and x_2 is in the interval $[\underline{x}_2, \overline{x}_2]$, the smallest possible value $\underline{y}$ of $y = x_1 - x_2$ is attained when the variable x_1 attains its smallest possible values $\underline{x}_1$ and the variable x_2 attains its largest possible values $\overline{x}_2$, i.e., $\underline{y} = \underline{x}_1 - \overline{x}_2$. Similarly, the largest possible value $\overline{y}$ of $y = x_1 - x_2$ is attained when the variable x_1 attains its largest possible values $\overline{x}_1$ and the variable x_2 attains its smallest possible values $\underline{x}_2$, i.e., $\overline{y} = \overline{x}_1 - \underline{x}_2$. Therefore, the range of the difference has the form

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

For example, if Gang has between 5 and 10 dollars, and he owes Vladik between 2 and 4 dollars, then after paying his debt, he will have between $5 - 4 = 1$ and $10 - 2 = 8$ dollars.

For multiplication $f(x_1, x_2) = x_1 \cdot x_2$, the situation is not so simple since the product is an increasing function of x_1 when $x_2 \geq 0$ and a decreasing function of x_1 when $x_2 \leq 0$. However, for every x_2, the expression $x_1 \cdot x_2$ is a linear function of x_1, and it is known that a linear function attains its maximum and its minimum only at the endpoints. Similarly, when we view this expression as a function of x_2, we conclude that it attains its minimum and maximum only at the endpoints $x_2 = \underline{x}_2$ and $x_2 = \overline{x}_2$.

Thus, the minimum and maximum of $x_1 \cdot x_2$ are attained when each of the variables x_i is equal either to its minimum $\underline{x}_i$ or to its maximum $\overline{x}_i$. For each variable, there are two possibilities, so we have $2 \times 2 = 4$ possible combinations. Thus:

- to find the minimum, it is sufficient to find the smallest of the corresponding

four products, and
- to the find the maximum, it is sufficient to find the largest of the corresponding four products.

Therefore, the range of the product has the form

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

The formula for the division comes from the fact that in most modern computers, division a/b is implemented as $a\cdot(1/b)$. Since we already know how to deal with the multiplication, to get the expression for a/b, it is sufficient to be able to deal with the inverse $1/b$. Inverse is defined only for values $b\neq 0$. When the interval does not contain 0, then $1/b$ is defined everywhere on this interval, and it is a decreasing function on this interval. Thus, when $x_1\in[\underline{x}_1,\overline{x}_1]$:

- the smallest possible value of $1/x_1$ is attained when x_1 is the largest ($x_1=\overline{x}_1$), and
- the largest possible value of $1/x_1$ is attained when x_1 is the smallest ($x_1=\underline{x}_1$).

Therefore, the range of $1/x_1$ has the form

$$1/[\underline{x}_1,\overline{x}_1]=[1/\overline{x}_1,1/\underline{x}_1].$$

Similar formulas can be written for simple elementary functions like $\ln(x)$, $\exp(x)$, or $\sqrt{x}$ since they are monotonic. For example, the function $\exp(x)$ is increasing, so

$$\exp([\underline{x}_1,\overline{x}_1])=[\exp(\underline{x}_1),\exp(\overline{x}_2)].$$

More complex functions like trigonometric functions $\sin(x_1)$ and $\cos(x_1)$ are only piece-wise monotonic, so to find their range on a given interval, we must first divide this interval into subintervals on which the function is monotonic.

Towards the general case: warning. We know how to estimate the range of the result of a simple arithmetic operation or an elementary function. A natural next question is: how to estimate the range of a general algorithm?

The first idea is called *straightforward* or *naive* interval computations. Before we explain what it is, we need to remind the reader that, contrary to a widely spread belief, this is *not* what interval computation is about. General interval computation algorithms – some of which will be described later – are much more sophisticated than that. So a reader may ask: why not start with these sophisticated practical algorithms, why study the naive technique? We need to study straightforward interval computations because more these sophisticated techniques use naive interval computation as a building block.

Straightforward interval computations: main idea. After this warning, let us explain the main idea behind straightforward interval computations. This idea is based on the fact that in the computer, only basic arithmetic operations and elementary functions are directly supported:

- either as a single hardware operations, as is he case with addition, subtraction, and multiplication,
- or as a pre-programmed sequence of hardware operations, as is the case of division and elementary functions.

Whatever more complex expression we write, a compiler will *parse* it, i.e., represent it as a sequence of elementary arithmetic operations and elementary functions.

Straightforward interval computations: example. As an illustrative example, let us consider a simple arithmetic expression $f(x) = (x-2)\cdot(x+2)$. How will the computer compute it? The same way we will compute it: first, we compute the two expressions in parentheses $x-2$ and $x+2$, then we multiply them. If we denote the i-th intermediate result by r_i, 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 want to estimate the range of this expression on the interval $[1,2]$, then we perform the same operations, but with *intervals* instead of *numbers*:

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

So, straightforward interval computations resulted in an estimate $\mathbf{Y} = [-4,0]$ for the desired range.

In our toy example, it is easy to find the actual range. Namely, if we represent the function $f(x)$ in the equivalent form $f(x) = x^2 - 4$, we can easily see that this function is increasing on the interval $[1,2]$. Thus, its smallest possible value is attained for $x = 1$, its largest for $x = 2$, and its range is equal to $\mathbf{y} = [f(1), f(2)] = [-3,0]$.

In this example, the range computed by straightforward interval computations *encloses* the actual range $\mathbf{Y} \supseteq \mathbf{y}$. One can prove, by induction, that this is always the case: that what we get as a result of straightforward interval computations is an *enclosure* for the actual range.

One can also see that in this example, the computed enclosure is *wider* that the actual range. It is usually said that this enclosure contains *excess width.*

Reason for excess width. On the above simple example, one can clearly explain why we get excess width. In this simple example, we have only three computational steps.

On the first step, we compute the interval $\mathbf{r}_1 = [-1,0]$ that contains all possible values of $r_1 = x - 2$. This is actually the exact range of $r_1 = x_1 - 2$ when $x_1 \in [1,2]$.

On the second step, we compute the interval $\mathbf{r}_2 = [3,4]$ that contains all possible values of $r_2 = x+2$. This is also the exact range of $r_2 = x_1 + 2$ when $x_1 \in [1,2]$. So far, there is no excess width.

Excess width appears on the third step, when we use the formula for interval multiplication $[-1,0]\cdot[3,4]$ to compute the enclosure $[-4,0]$ for the range of $r_3 = r_1 \cdot r_2$. The formula for interval multiplication correctly describes the range of possible values of $r_1 \cdot r_2$, when r_1 can take all possible values from the interval $\mathbf{r}_1$ and r_2 can take all possible values from the interval $\mathbf{r}_2$; in this case, all pairs (r_1, r_2) with $r_1 \in \mathbf{r}_1$ and $r_2 \in \mathbf{r}_2$ are possible. In particular, the value -4 is possible when $r_1 = -1$ and $r_2 = 4$.

However, in our applications, not all pairs (r_1, r_2) are possible: since $r_1 = x_1 - 2$ and $r_2 = x_1 + 2$, there is a dependence between r_1 and r_2: $r_2 = r_1 + 4$. In particular, the pair $(r_1, r_2) = (-1, 4)$ at which the value -4 is attained is no longer possible, as all the pairs that lead to values $r_3 = r_1 \cdot r_2 < -3$. When we apply interval multiplication to the intervals $\mathbf{r}_1$ and $\mathbf{r}_2$, we ignore this dependence and thus, get excess width.

In short, excess width is caused by dependence between intermediate results: if there is no dependence, if for each arithmetic operation, all pairs of inputs from the corresponding intervals are possible, interval arithmetic produces the exact range. Because of this observation, the problem of minimizing (and, ideally, eliminating) the excess width is sometimes called *dependency problem.* This is the main problem of interval computations: how to find the range with as little excess width as possible.

Excess width is inevitable. At this point, it is worth reminding that, in general, the main problem of interval computations, i.e., the problem of computing the exact range of a given function $f(x_1, \ldots, x_n)$ on given intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$, is NP-hard. This means that not only straightforward interval computations produce excess width: any feasible (polynomial-time) algorithm that always produces an enclosure has to have cases in which this enclosure is wider than the desired range – otherwise, we would be able to solve the NP-hard problem in feasible time, which most computer scientists believe to be impossible.

We need a good approximation. Since we cannot always efficiently find the exact range $\mathbf{y} = [\underline{y}, \overline{y}]$ of the given function $f(x_1, \ldots, x_n)$ on given intervals, we should at least find a good approximation $\mathbf{Y} = [\underline{Y}, \overline{Y}]$ for this range.

Why do we need an exclosure: engineering applications of interval computations. In some practical problems, we need enclosure. For example, assume that we want to prove that the system is stable for all possible combinations of parameters $x_i \in \mathbf{x}_i$, and stability is described by the condition $f(x_1, \ldots, x_n) \geq 0$ for a known function $f(x_1, \ldots, x_n)$. In terms of the exact range, the condition that $f(x_1, \ldots, x_n) \geq 0$ for all $x_i \in \mathbf{x}_i$ is equivalent to stating that the smallest possible value of $f(x_1, \ldots, x_n)$ is non-negative, i.e., that $\underline{y} \geq 0$. When we cannot compute the exact range, we compute an approximation $[\underline{Y}, \overline{Y}]$. In this case, a natural way to test stability is to check whether $\underline{Y} \geq 0$.

However, if $\underline{Y} > \underline{y}$, we may have $\underline{Y} \geq 0$ and still $\underline{y} < 0$. The only way to make sure that the test based on $\underline{Y}$ guarantees stability is to have $\underline{Y} \leq \underline{y}$.

Similarly, for criteria based on the opposite inequalities $f(x_1, \ldots, x_n) \leq y_0$, the only way to guarantee is to make sure that $\overline{y} \leq \overline{Y}$. Thus, to get guaranteed bounds, we must have $\underline{Y} \leq \underline{y} \leq \overline{y} \leq \overline{Y}$, i.e., we must have $\mathbf{y} = [\underline{y}, \overline{y}] \subseteq [\underline{Y}, \overline{Y}] = \mathbf{Y}$. In other words, the estimate $\mathbf{Y}$ must be an enclosure for the desired range $\mathbf{y}$.

Why do we need an exclosure: general case, including applications to fuzzy data processing. The main purpose of this paper is to describe applications to fuzzy data processing. In fuzzy data processing, the inputs are – well, fuzzy, so there seems to be not too much need for guarantees. In this case, it is enough to have an approximation $\mathbf{Y} \approx \mathbf{y}$. An interval is determined by its endpoints, so the fact that the interval $\mathbf{Y} = [\underline{Y}, \overline{Y}]$ approximates the desired range $\mathbf{y} = [\underline{y}, \overline{y}]$ means that $\underline{Y}$ approximates $\underline{y}$ and $\overline{Y}$ approximates $\overline{y}$.

Some things we do desire: we need some bounds on the accuracy of these approximations. If there are no bounds on accuracy, then the approximation does not make too much sense: whatever the value $\underline{Y}$, the actual value $\underline{y}$ can be any real number, as small as possible and as large as possible. Let us show that the existence of an approximation with known bounds leads to an enclosure.

Indeed, in general, when we have an approximation $\widetilde{x}$ for a number x, we may have two different bounds for the approximation accuracy: from below and form above. In other words, we have two numbers Δ^- and Δ^+ for which $\widetilde{x} - \Delta^- \leq x \leq \widetilde{x} + \Delta^+$. In our case, we would like to have such bounds for both approximations $\underline{Y}$ and for $\overline{Y}$, i.e., we would like to have values $\underline{\Delta}^-$, $\underline{\Delta}^+$, $\overline{\Delta}^-$, and $\overline{\Delta}^+$ for which $\underline{Y} - \underline{\Delta}^- \leq \underline{y} \leq \underline{Y} + \underline{\Delta}^+$ and $\overline{Y} - \overline{\Delta}^- \leq \overline{y} \leq \overline{Y} + \overline{\Delta}^+$. In this case, we have $\underline{Y} - \underline{\Delta}^- \leq \underline{y} \leq \overline{y} \leq \overline{Y} + \overline{\Delta}^+$, and thus, the interval $[\underline{Y} - \underline{\Delta}^-, \overline{Y} + \overline{\Delta}^+]$ is an enclosure for the desired range $[\underline{y}, \overline{y}]$.

Vice versa, every enclosure be viewed as an approximation with known bounds: indeed, if we have an enclosure $[\underline{Y}, \overline{Y}] \supseteq [\underline{y}, \overline{y}]$, this means that $\underline{Y} \le \underline{y} \le \overline{y} \le \overline{Y}$, i.e., that we have $\underline{Y} - \underline{\Delta}^- \le \underline{y} \le \underline{Y} + \underline{\Delta}^+$ and $\overline{Y} - \overline{\Delta}^- \le \overline{y} \le \overline{Y} + \overline{\Delta}^+$ for $\underline{\Delta}^- = \overline{\Delta}^+ = 0$ and $\underline{\Delta}^+ = \overline{\Delta}^- = \overline{Y} - \underline{Y}$.

Because of this, in the following text, we will concentrate on techniques for computing enclosures.

Comment. When we computed an enclosure from an approximation, we only used two of the four bounds, we did not use the inequalities $\underline{y} \le \underline{Y} + \underline{\Delta}^+$ and $\overline{Y} - \overline{\Delta}^- \le \overline{y}$. When $\underline{Y} + \underline{\Delta}^+ < \overline{Y} - \overline{\Delta}^-$, i.e., when $\underline{y} \le \underline{Y} + \underline{\Delta}^+ \le \overline{Y} - \overline{\Delta}^- \le \overline{y}$, we get a second interval $[\underline{Y} + \underline{\Delta}^+, \overline{Y} - \overline{\Delta}^-]$ which is guaranteed to be contained in the desired range. In this overview, we just wanted to mention that such intervals exist and that they are actively used in some efficient interval-related problems. Such problems include the problems of controllability.

For example, we may want to make sure that by appropriately changing the parameters $x_1, \ldots, x_n$ from the given intervals $[\underline{x}_i, \overline{x}_i]$, we can achieve all the values of a certain quantity $f(x_1, \ldots, x_n)$ ranging from y^- to y^+. If we know the exact range $[\underline{y}, \overline{y}]$, then the checking is simple: we just check that $[y^-, y^+] \subseteq [\underline{y}, \overline{y}]$. However, if we know the enclosure $[\underline{Y}, \overline{Y}] \supseteq [\underline{y}, \overline{y}]$, and this enclosure contains $[y^-, y^+]$, nothing is guaranteed, because the actual range may be narrower than $[y^-, y^+]$. To guarantee that $[\underline{Y}, \overline{Y}] \supseteq [y^-, y^+]$ for an approximation interval $[\underline{Y}, \overline{Y}]$ implies that $[\underline{y}, \overline{y}] \supseteq [y^-, y^+]$, we must make sure that $[\underline{Y}, \overline{Y}]$ is a *subset* of the actual range.

Resulting problem: computing most accurate approximations. Since we cannot always compute the exact range, we need to produce an approximation which is as accurate as possible. As we have mentioned, computing an approximation is equivalent to computing an enclosure, For enclosures, inaccuracy means excess width. Thus, in terms of enclosures, the problem is to reduce the excess width as much as possible.

How can we reduce excess width? In the following subsections, we will describe the main techniques for reducing excess width. The efficient software packages for solving interval computation problems usually incorporate all these techniques. Before we go into the details of each of these techniques, let us briefly explain the logic behind these techniques.

We started our description of interval computations with the example of interval addition and interval subtraction. In these two cases, the corresponding function $f(x_1, x_2)$ is monotonic in each variable and thus, computing its range is easy. In general, monotonicity seems to be the simplest case for computing the interval range.

For example, intuitively, to compute the range of a function $f(x_1)$ on one variable on an interval $[\underline{x}_1, \overline{x}_1]$, we need to compute at least the values $f(\underline{x}_1)$ and $f(\overline{x}_1)$ of this function $f(x_1)$ at the endpoints $\underline{x}_1$ and $\overline{x}_1$. The simplest case is when these two values are sufficient to describe the range, i.e., when we are sure that all values $f(x_1)$ for $x_1 \in [\underline{x}_1, \overline{x}_1]$ like between these values $f(\underline{x}_1)$ and $f(\overline{x}_1)$. This is guaranteed only if the function is monotonic – increasing or decreasing, because otherwise, it may have a local maximum or a local minimum at which its value is larger or smaller than the endpoint values; see, e.g., Koshelev and Kreinovich (1996).

Thus, once we are given a function and n intervals, it may not be a bad idea to check whether this function is monotonic. If the function is monotonic, computing its range is easy.

What if it is not monotonic – or at least we cannot prove that it is monotonic? In this case, a good idea is to *approximate* the original function by a monotonic one, so that at least the range for this monotonic component will be computed exactly. For smooth functions, the possibility of such approximation comes from the fact each smooth function $f(x_1, \ldots, x_n)$ on intervals $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ – i.e., in the vicinity of a point $\widetilde{x} = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$ – can be well approximation by a linear function (its tangent):

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

The approximating linear function $f(\widetilde{x}_1, \ldots, \widetilde{x}_n) + \sum\limits_{i=1}^{n} \dfrac{\partial f}{\partial x_i} \cdot \Delta x_i$ is always monotonic. The use of this approximation is the main idea behind the *mean value form*.

What if the mean valued form is still not accurate enough? The accuracy of the linear approximation can be estimated if we consider the first two terms in the Taylor expansion:

$$f(x_1, \ldots, x_n) = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \cdot \Delta x_i + \frac{1}{2} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j} \cdot \Delta x_i \cdot \Delta x_j + \ldots$$

Since the main component of the approximation error is the quadratic part, this approximation is of order Δ_i^2. So, to decrease this excess width, we need to decrease Δ_i. This can be done if we divide one or more of the original intervals into smaller pieces, find the range of $f(x_1, \ldots, x_n)$ over all these pieces, and then compute the union of all these ranges. In the simplest case when we divide into two pieces, this method is called *bisection*.

All these methods are based on explicitly using the linear terms in Taylor expansion. Sometimes, we can get better results by using quadratic and higher order terms; this idea is called *Taylor arithmetic*.

Comment. In this overview, we have just listed the main interval techniques. There are many other useful techniques, many of them specifically designed for special classes of functions.

First idea: use of monotonicity. For arithmetic operations, we had exact ranges. The possibility to compute the exact ranges came from the fact that, e.g., addition $x_1 + x_2$ and subtraction $x_1 - x_2$ are monotonic in each variable.

In general, if the $f(x_1, \ldots, x_n)$ is (non-strictly) increasing in each variable x_i, then its smallest value is attained when all the inputs take their smallest possible values, and its largest value is attained when all the inputs take their largest possible values. Thus, the desired range has the form:

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

Similarly, we can easily find the explicit formulas for the range when the function $f(x_1, \ldots, x_n)$ is increasing in some variables x_i and decreasing in some other variables x_j.

How can we check whether a function is increasing or decreasing with respect to each variable? From calculus, it is known that a function f is increasing in x_i if its derivative with respect to x_i is non-negative: $\dfrac{\partial f}{\partial x_i} \geq 0$. If we know the exact range $[\underline{d}_i, \overline{d}_i]$ of this derivative $\dfrac{\partial f}{\partial x_i}$ on the interval $\mathbf{x}_i$, then checking monotonicity is easy: all the values of the derivative are non-negative if and only if the smallest of these values $\underline{d}_i$ is non-negative: $\underline{d}_i \geq 0$.

To apply this idea, we must estimate the range of the derivative. Estimating the range is exactly what interval computations is about. To apply interval computations, we first need to find the derivative itself. We are considering functions that consist of several well-defined steps. For such functions, we can find the derivative step-by-step if we follow the original scheme and apply the standard rules for differentiation: $(u + v)' = u' + v'$, $(u - v)' = u' - v'$, $(u \cdot v)' = u' \cdot v + u \cdot v'$, and the "chain" rule for differentiating the composition: if $h(x) = f(g(x))$, then $h'(x) = f'(g(x)) \cdot g'(x)$. The resulting procedure has been automatically implemented in several *automatic differentiation* (AD) tools; see, e.g., Griewank and Walther (2008). Once we apply interval computations to the resulting derivative, we get an enclosure $[\underline{D}_i, \overline{D}_i]$ for the actual range $[\underline{d}_i, \overline{d}_i]$. The fact that we have an enclosure means, in particular, that $\underline{d}_i \geq \underline{D}_i$. So, if $\underline{D}_i \geq 0$, we know that $\underline{d}_i \geq 0$ and thus, that the function $f(x_1, \ldots, x_n)$ is increasing in x_i.

Similarly, if $\overline{D}_i \leq 0$, we conclude that $\overline{d}_i \leq 0$ and thus, that the function $f(x_1, \ldots, x_n)$ is decreasing in x_i. Thus, we arrive at the following technique.

Monotonicity: resulting technique. When we need to compute the range of a function $f(x_1, \ldots, x_n)$ over intervals $[\underline{x}_i, \overline{x}_i]$, then, for each variable x_i, we first use automatic differentiation techniques to find the derivative $\dfrac{\partial f}{\partial x_i}$ and use straightforward interval computation techniques to find the range $[\underline{D}_i, \overline{D}_i]$ of this derivative.

If for every i, we have $\underline{D}_i \geq 0$ or $\overline{D}_i \leq 0$, then we can conclude that the function $f(x_1, \ldots, x_n)$ is monotonic with respect to each of its variables. So, to find its largest possible value $\overline{y}$, we apply the algorithm f to the following values:

- $x_i = \overline{x}_i$ when $\underline{D}_i \geq 0$, and
- $x_i = \underline{x}_i$ when $\overline{D}_i \geq 0$.

Similarly, to find its smallest possible value $\underline{y}$, we apply the algorithm f to the following values:

- $x_i = \underline{x}_i$ when $\underline{D}_i \geq 0$, and
- $x_i = \overline{x}_i$ when $\overline{D}_i \geq 0$.

Comment. It should be mentioned that since straightforward interval computations produce an enclosure for the actual range for the derivative, we may have $\underline{d}_i \geq 0$ but $\underline{D}_i < 0$. In such situations, the function is actually monotonic, but we do not detect this monotonicity. To detect monotonicity better, we can thus use more accurate interval computation techniques – e.g., the techniques based on checking monotonicity of the derivative itself, or the mean valued form techniques that we will describe in the following text.

What if we only have monotonicity with respect to some variables? Once we know that the function $f(x_1, \ldots, x_n)$ is, e.g., increasing with respect to x_1, then we know that the maximum of $f(x_1, \ldots, x_n)$ on given intervals is attained when $x_1 = \overline{x}_1$ and the minimum of $f(x_1, \ldots, x_n)$ is attained when $x_1 = \underline{x}_1$. Thus, to find $\overline{y}$, it is sufficient to find the largest possible value of a function $f(\overline{x}_1, x_2, \ldots, x_n)$ of $n - 1$ variables $x_2, \ldots, x_n$, and to find $\underline{y}$, it is sufficient to find the largest possible values of a function $f(\underline{x}_1, \ldots, x_n)$ of $n - 1$ variables.

Similarly, if we know that a function $f(x_1, \ldots, x_n)$ is monotonic in k of its n variables, then it is sufficient to consider two functions of $n - k$ variables. As a we have mentioned, the range-computing problem is NP-hard, which means, crudely speaking, that its computational complexity grows exponentially with the number of variables, as $\approx 2^n$. From this viewpoint, reducing the number of variables from n to $n-k$ means decreasing computation time by a factor of 2^k. For large k, this is a big speed-up.

Monotonicity: example. Let us consider the same example on which we showed excess width: finding the range of a function $f(x) = (x-2)\cdot(x+2)$ on the interval $\mathbf{x} = [1,2]$. For this function, differentiation leads to $\dfrac{df}{dx} = 1\cdot(x+2)+(x-2)\cdot 1 = 2x$. By using straightforward interval arithmetic, we can estimate the range of this function for $x \in [1,2]$ as $[2,2]\cdot[1,2] = [2,4]$. The lower endpoint of this range is non-negative, so we can conclude that the original function $f(x)$ is increasing on the interval $[1,2]$. Thus, its range is equal to $f([1,2]) = [f(1), f(2)] = [-3,0]$. On this example, we get the exact range.

Not every function is monotonic. Of course, not every function is monotonic. Let us see what happens when we apply straightforward interval computations to a non-monotonic function. As a simple example, let us take the function $f(x) = x\cdot(1-x)$ on the range $x \in [0,1]$. To describe what naive interval computations will do for this function, let us first parse this expression, i.e., describe the sequence of elementary step that a computer will perform when computing $f(x)$. First, it will compute $r_1 = 1-x$, and then it will compute $r_2 = x\cdot r_1$. Replacing arithmetic operations by the corresponding operations with intervals, we get $\mathbf{r}_1 = [1,1] - [0,1] = [0,1]$ and then $\mathbf{r}_2 = [0,1]\cdot[0,1] = [0,1]$.

The actual range can be computed based on the fact that for every function of one variable, its minimum and maximum on a given interval are attained either at one of the endpoints, or inside the interval. If the minimum or the maximum are attained inside, then, according to calculus, its derivative is 0. Thus, to find the minimum and the maximum of a function on a given interval $[\underline{x}, \overline{x}]$, it is sufficient to find its values at the endpoints $\underline{x}$, $\overline{x}$, and at the points where $\dfrac{df}{dx} = 0$. The smallest of these values is the minimum $\underline{y}$ of the function $f(x)$ on the given interval, the largest of these values is its maximum $\overline{y}$.

For our function, $\dfrac{df}{dx} = 1-2x = 0$ for $x = 0.5$, so we compute $f(0) = 0$, $f(0.5) = 0.25$, and $f(1) = 0$, and then find $\underline{y} = \min(0, 0.25, 0) = 0$ and $\overline{y} = \max(0, 0.25, 0) = 0.25$. So, the actual range is $f(\mathbf{x}) = [\underline{y}, \overline{y}] = [0, 0.25]$. This range shows that the estimate $[0,1]$ obtained by using straightforward interval computations has excess width.

Mean valued form: second idea. How do we decrease excess width for such non-monotonic functions? As we have mentioned, a natural idea is to approximate the original function by its linear part – since every linear function is monotonic. A general idea behind this approximation is well-known: we expand the dependence $f(x_1,\ldots,x_n) = f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n)$ into Taylor series and keep only linear terms

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

where ... stands for quadratic and higher order terms that need to be estimated.

Let us show that we can come up with an even simpler expression – known as the *mean valued form* – if instead of the values of the partial derivatives at the central point $(\widetilde{x}_1,\ldots,\widetilde{x}_n)$ we consider its values at nearby points. The advantage of the mean valued form is that it does not have quadratic or higher order terms.

Let us start our explanation of the mean-valued form with the simplest case of a function $f(x)$ of one variable. To make this explanation clear, let us use the standard calculus interpretation of the derivative: if $f(x)$ is the position of the object at moment x, then the derivative $\dfrac{df}{dx}$ is the velocity at the moment x. In general, the motion is not uniform: at some moments, the velocity is larger, at some other moments, the velocity is smaller. In addition to the velocities at different moments of time, we can also consider the average velocity on the time interval $[\widetilde{x}-\Delta x, \widetilde{x}]$, i.e., the value $\dfrac{f(\widetilde{x})-f(\widetilde{x}-\Delta x)}{\Delta x}$. The average velocity, by definition, is smaller than or equal to the largest velocity and larger than or equal to the smallest velocity. Thus, as we go from the moment when the velocity is the smallest to the moment when the velocity is the largest, we will, at some point, reach the moment χ at which the velocity is exactly equal to the average velocity:

$$\frac{f(\widetilde{x})-f(\widetilde{x}-\Delta x)}{\Delta x}=\frac{df}{dx}(\chi).$$

Multiplying both sides of this equality by the difference Δx, we conclude that $f(\widetilde{x})-f(\widetilde{x}-\Delta x)=\dfrac{df}{dx}(\chi)\cdot\Delta x$, i.e., that

$$f(\widetilde{x}-\Delta x)=f(\widetilde{x})-\frac{df}{dx}(\chi)\cdot\Delta x$$

for some $\chi\in[\widetilde{x}-\Delta x,\widetilde{x}]$. Since both values $\widetilde{x}-\Delta x$ and $\widetilde{x}$ belong to the interval $\mathbf{x}=[\underline{x},\overline{x}]$, we conclude that the intermediate point χ belongs to the same interval.

For a function $f(x_1,x_2)$ of two variables, we can apply this argument first to x_1 and then to x_2. As a result, we get the following two formulas:

$$f(\widetilde{x}_1-\Delta x_1,\widetilde{x}_2-\Delta x_2)=f(\widetilde{x}_1,\widetilde{x}_2-\Delta x_2)-\frac{\partial f}{\partial x_1}(\chi_1,\widetilde{x}_2-\Delta x_2)\cdot\Delta x_1;$$

$$f(\widetilde{x}_1,\widetilde{x}_2-\Delta x_2)=f(\widetilde{x}_1,\widetilde{x}_2)-\frac{\partial f}{\partial x_2}(\widetilde{x}_1,\chi_2)\cdot\Delta x_2.$$

Substituting the expression for $f(\widetilde{x}_1,\widetilde{x}_2-\Delta x_2)$ from the second formula into the first one, we conclude that

$$f(\widetilde{x}_1-\Delta x_1,\widetilde{x}_2-\Delta x_2)=f(\widetilde{x}_1,\widetilde{x}_2)-\frac{\partial f}{\partial x_1}(\chi_1,\widetilde{x}_2-\Delta x_2)\cdot\Delta x_1-\frac{\partial f}{\partial x_2}(\widetilde{x}_1,\chi_2)\cdot\Delta x_2,$$

i.e., that

$$f(\widetilde{x}_1-\Delta x_1,\widetilde{x}_2-\Delta x_2)=f(\widetilde{x}_1,\widetilde{x}_2)-\frac{\partial f}{\partial x_1}(\chi^{(1)})\cdot\Delta x_1-\frac{\partial f}{\partial x_2}(\chi^{(2)})\cdot\Delta x_2$$

for some $\chi^{(1)},\chi^{(2)}\in\mathbf{x}_1\times\mathbf{x}_2$.

In general, for n variables, we get a similar formula

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

This formula can be used to find an enclosure for the function $f(x_1, \ldots, x_n)$ on given intervals $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. Namely, since $\chi^{(i)} \in \mathbf{x}_1 \times \times \times \mathbf{x}_n$, each derivative $\dfrac{\partial f}{\partial x_i}(\chi^{(i)})$ belongs to the range of this derivative on these intervals. Also, by definition, $\Delta x_i \in [-\Delta_i, \Delta_i]$. Thus, each product $\dfrac{\partial f}{\partial x_i}(\chi^{(i)}) \cdot \Delta x_i$ belongs to the product of the corresponding intervals, and the value $f(x_1, \ldots, x_n) = f(\widetilde{x}_1 - \Delta x_1, \ldots \widetilde{x}_n - \Delta x_n)$ belongs to the corresponding linear combination of such intervals. In short, we arrive at the following conclusion.

Mean valued form: resulting algorithm. When $x_i \in \mathbf{x}_i$, then $f(x_1, \ldots, x_n) \in \mathbf{Y}$, where

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

So, we arrive at the following method for estimating the range of a given function $f(x_1, \ldots, x_n)$ on given intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$:

- first, we use the Automatic Differentiation (AD) tools to find all n partial derivatives of the given function;
- then, we estimate the range of each of these derivatives on the given intervals;
- finally, we use the above formula to compute the enclosure.

How do we estimate the ranges of the partial derivatives?

- if appropriate, we can use monotonicity;
- if the derivatives are not monotonic, we can use straightforward interval computations;
- alternatively, we can estimate the ranges of each derivative by applying the same mean values form; this way, we spend more time on computations but hopefully get more accurate results.

Mean valued form is usually combined with checking monotonicity. In the process of computing the mean valued form, we find the range for each partial derivative. Once we have this range, we can check whether the function is monotonic with respect to the corresponding variables x_i – in the above text, we describe how we can do it. If the function is monotonic with respect to some of the variables, then we can use this monotonicity to reduce the problem to the one with fewer variables – and then apply the mean valued form only to this function with fewer variables.

Mean valued form: example when it leads to a better enclosure. Let us first give an example when the mean valued method leads to a much narrower enclosure than straightforward interval computations. As such an example, we will consider the problem of computing the range of the function $f(x) = x \cdot (1 - x)$ on the interval $\mathbf{x} = [\underline{x}, \overline{x}] = [0.4, 0.6]$. To represent an interval $[\underline{x}, \overline{x}]$ in the form $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$, we can take $\widetilde{x} = \dfrac{\underline{x} + \overline{x}}{2}$ and $\Delta = \dfrac{\overline{x} - \underline{x}}{2}$. In our case, we get $\widetilde{x} = 0.5$ and $\Delta = 0.1$.

The actual range can be found if we compute the value of this function at the endpoints 0.4 and 0.6, and at the point where the derivative is equal to 0. Here, automatical differentiation leads to $\dfrac{df}{dx} = 1 \cdot (1 - x) + x \cdot (-1) = 1 - 2x$, so the derivative is equal to 0 when $x = 0.5$. At the resulting three values of x, the function $f(x)$ takes the values $f(0.4) = f(0.6) = 0.4 \cdot 0.6 = 0.24$ and $f(0.5) = 0.5 \cdot (1 - 0.5) =$

0.25. Thus, the smallest possible value of $f(x)$ in the given interval is equal to $\underline{y} = \min(0.24, 0.25) = 0.24$, the largest value is equal to $\overline{y} = \max(0.24, 0.25) = 0.25$, and so, the desired range is equal to $[\underline{y}, \overline{y}] = [0.24, 0.25]$.

For this problem, straightforward interval computations lead to $1 - [0.4, 0.6] = [1, 1] - [0.4, 0.6] = [1 - 0.6, 1 - 0.4] = [0.4, 0.6]$ and $[0.2, 0.6] \cdot [0.4, 0.6] = [0.16, 0.26]$.

For the case of a single variable $n = 1$, the general formula for the mean valued method takes the form $\mathbf{Y} = f(\widetilde{x}) + \dfrac{df}{dx}(\mathbf{x}) \cdot [-\Delta, \Delta]$. Here, automatical differentiation leads to $\dfrac{df}{dx} = 1 \cdot (1 - x) + x \cdot (-1) = 1 - 2x$. The range of this derivative can be estimated by using straightforward interval computations, as

$$\frac{df}{dx}(\mathbf{x}) = 1 - 2 \cdot [0.4, 0.6] = [-0.2, 0.2].$$

Thus, the mean valued method leads to the following enclosure:

$$\mathbf{Y} = f(0.5) + [-1, 1] \cdot [-0.2, 0.2] = 0.25 + [-0.02, 0.02] = [0.23, 0.27].$$

It is easy to check that this interval is indeed an enclosure for the actual range $[0.24, 0.25]$, and that its width $0.27 - 0.23 = 0.04$ is much smaller than the width $0.36 - 0.16 = 0.2$ of the interval obtained by using straightforward interval computations.

Mean valued form: example when it does not lead to a better enclosure. Let us now give an example when the mean valued method is still not perfect. As such an example, we can take the above case of computing the range of the function $f(x) = x \cdot (1 - x)$ on the interval $\mathbf{x} = [\underline{x}, \overline{x}] = [0, 1]$. To represent an interval $[\underline{x}, \overline{x}]$ in the form $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$, we take $\widetilde{x} = \dfrac{\underline{x} + \overline{x}}{2}$ and $\Delta = \dfrac{\overline{x} - \underline{x}}{2}$. In our case, we get $\widetilde{x} = 0.5$ and $\Delta = 0.5$.

In the previous example, we have shown that for the function $f(x) = x \cdot (1 - x)$, we have $\mathbf{Y} = f(\widetilde{x}) + \dfrac{df}{dx}(\mathbf{x}) \cdot [-\Delta, \Delta]$, where $\dfrac{df}{dx} = 1 - 2x$. The range of this derivative can be estimated by using straightforward interval computations, as $\dfrac{df}{dx}(\mathbf{x}) = 1 - 2 \cdot [0, 1] = [-1, 1]$. Thus, the mean valued method leads to the following enclosure:

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

It is easy to check that this is indeed an enclosure for the actual range $[0, 0.25]$. The right endpoint 0.75 of the new enclosure is somewhat better than the corresponding endpoint 1 of the enclosure $[0, 1]$ that we obtained by using straightforward interval computations, but the left endpoint is worse than the previous one.

Bisection: third idea. How can we further decrease the excess width? The main idea behind the mean valued method is that we approximated a function by linear terms of its Taylor expansion. The accuracy of this approximation is of the same order as quadratic terms $\dfrac{\partial^2 f}{\partial x_i \partial x_j} \cdot \Delta x_i \cdot \Delta x_j$, i.e., of order $O(\Delta_i^2)$. Thus, a natural way to make the estimations more accurate is to decrease the half-widths Δ_i of the corresponding intervals. For example, if we split each interval in two, with half-width $\Delta_i/2$, the approximation error decreases from Δ_i^2 to $\Delta_i^2/4$.

Thus, the idea is:

- to bisect the original domain,
- to use one of the above techniques to compute the enclosure of the original function of each of sub-domains, and
- take the union of the resulting enclosures.

Comment. Bisection is an example of a general divide-and-conquer strategy for algorithm design, where in order to solve a complex problem, we first divide it into several simpler ones; see, e.g., Cormen et al. (2009). The term itself comes from history, where divide-and-conquer strategy was one of the main ways how big empires were built. For example, when in the 13th century the Mongols conquered Russia and China, they did not do it by winning a big battle in which the Mongol army overcame the Russian army: in such a battle underpopulated Mongolia did not have a chance. They won because both Russia and China were divided into smaller states hostile to each other. By exploiting these divisions, the Mongols succeeded in conquering these smaller states one by one. This is also how England conquered India and, going further into history, how a small city of Rome became the center of a powerful Roman empire.

In politics, the divide-and-conquer strategy was often used for "negative" things like conquering, but in computer science, similar ideas are used for "positive" purposes – to solve difficult problems.

Bisection: example. Let us consider the same example for which the mean valued form did not work well: estimating the range of the function $f(x) = x \cdot (1-x)$ when $x \in \mathbf{x} = [0,1]$. Applying bisection means that we split the original interval $\mathbf{x} = [0,1]$ into two half-intervals: $\mathbf{x}' = [0, 0.5]$ and $\mathbf{x}'' = [0.5, 1]$.

On the first half-interval, the range of the derivative is equal to $1 - 2 \cdot \mathbf{x} = 1 - 2 \cdot [0, 0.5] = [0, 1]$. The lower endpoint of this range is non-negative, so the function $f(x)$ is increasing on this interval and thus, its range is equal to $f(\mathbf{x}') = [f(0), f(0.5)] = [0, 0.25]$.

On the second half-interval, the range of the derivative is equal to $1 - 2 \cdot \mathbf{x} = 1 - 2 \cdot [0.5, 1] = [-1, 0]$. The upper endpoint of this range is non-positive, so the function $f(x)$ is decreasing on this interval and thus, its range is equal to $f \downarrow$ and $f(\mathbf{x}'') = [f(1), f(0.5)] = [0, 0.25]$.

The union of these ranges $f(\mathbf{x}') \cup f(\mathbf{x}'') = [0, 0.25]$ is an enclosure for the whole interval $[0, 1]$. One can easily check that in this example, this enclosure coincides with the desired range.

Bisection: discussion. Of course, in general, we may have excess width. In this case, if we want a more accurate enclosure, we can further subdivided one or both sub-intervals, etc. In the general case of functions of several variables, we can bisect by each of these variables – there are several semi-heuristic techniques for selecting the variable over which we bisect.

Fourth idea: Taylor arithmetic. The above methods are based on explicitly using the linear terms in Taylor expansion. Sometimes, we can get better results by using quadratic and higher order terms; this idea is called *Taylor arithmetic*; see, e.g., Neumaier (2002) and references therein.

Acknowledgments

This work was supported in part by the National Science Foundation grants HRD-0734825 and DUE-0926721, by Grant 1 T36 GM078000-01 from the National Institutes of Health, by Grant MSM 6198898701 from MŠMT of Czech Republic, and by Grant 5015 "Application of fuzzy logic with operators in the knowledge based systems" from the Science and Technology Centre in Ukraine (STCU), funded by European Union.

The authors are greatly thankful to George J. Klir for his encouragement.

References

Cormen, Th. H., Leiserson, C. E., Rivest, R. L., and Stein, S., 2009. *Introduction to Algorithms*, Cambridge, MA: MIT Press.

Ferson, F., Ginzburg, L., Kreinovich, V., Longpré, L., and Aviles, M., 2002. Computing variance for interval data is NP-hard, *ACM SIGACT News*, 33(2), 108–118.

Ferson, F., Ginzburg, L., Kreinovich, V., Longpré, L., and Aviles, M., 2005. Exact bounds on finite populations of interval data, *Reliable Computing*, 11(3), 207–233.

Griewank, A., and Walther, A., 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Philadelphia, PA: SIAM Publ.

Klir, G. and Yuan, B., 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Upper Saddle River, NJ: Prentice Hall.

Koshelev, M., and Kreinovich, V., 1996. Why Monotonicity in Interval Computations? A Remark, *ACM Special Interest Group on Numerical Mathematics (SIGNUM) Newsletter*, 31(3), 4–8.

Neumaier, A., 2000. Taylor forms - use and limits, *Reliable Computing*, 9: 43–79.

Nguyen, H.T., 1978. A note on the extension principle for fuzzy sets, *J. Math. Anal. and Appl.*, 64: 369–380.

Nguyen, H.T., and Kreinovich, V, 1996. Nested intervals and sets: concepts, relations to fuzzy sets, and applications. In: Kearfott, R.B., and Kreinovich, V. (eds.) *Applications of Interval Computations*, Norwell, MA, USA, and Dordrecht, The Netherlands: Kluwer Academic Publishers, 245–290.

Nguyen, H.T., and Kreinovich, V., 1998. Methodology of fuzzy control: an introduction. In: Nguyen, H.T., and Sugeno, M. (eds.) *Fuzzy Systems: Modeling and Control*, Norwell, MA, USA, and Dordrecht, The Netherlands: Kluwer Academic Publishers, 19–62.

Nguyen, H.T., Kreinovich, V., and Zuo, Q. 1997. Interval-valued degrees of belief: applications of interval computations to expert systems and intelligent control. *International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems (IJUFKS)* 5, 317–358.

Nguyen, H.T. and Walker, E.A., 2006. *First Course in Fuzzy Logic*, Boca Raton, FL:CRC Press.

Rabinovich, S., 2005. *Measurement Errors: Theory and Practice*, N.Y.:American Institute of Physics.

Zadeh, L.A., 1965. Fuzzy sets, *Information and control*, 8: 338–353.