From Interval Computations to Constraint-Related Set Computations: Towards Faster Estimation of Statistics and ODEs under Interval and p-Box Uncertainty

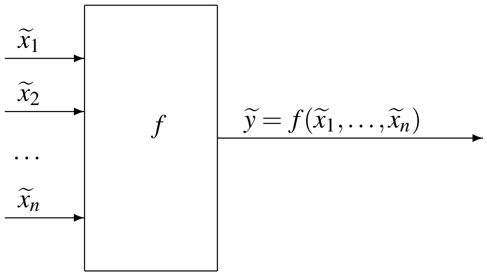
Vladik Kreinovich

Abstract Interval computations estimate the uncertainty of the result of data processing in situations in which we only know the upper bounds Δ on the measurement errors. In this case, based on the measurement result $\widetilde{x}$, we can only conclude that the actual (unknown) value x of the desired quantity is in the interval $[\widetilde{x}-\Delta,\widetilde{x}+\Delta]$. In interval computations, at each intermediate stage of the computation, we have intervals of possible values of the corresponding quantities. As a result, we often have bounds with excess width. To remedy this problem, in our previous papers, we proposed an extension of interval technique to *set computations*, where on each stage, in addition to intervals of possible values of the quantities, we also keep sets of possible values of pairs (triples, etc.). In this paper, we show that in several practical problems, such as estimating statistics (variance, correlation, etc.) and solutions to ordinary differential equations (ODEs) with given accuracy, this new formalism enables us to find estimates in feasible (polynomial) time.

1 Formulation of the Problem

Need for data processing. In many real-life situations, we are interested in the value of a physical quantity y that is difficult or impossible to measure directly. Examples of such quantities are the distance to a star and the amount of oil in a given well. Since we cannot measure y directly, a natural idea is to measure y *indirectly*. Specifically, we find some easier-to-measure quantities $x_1,\ldots,x_n$ which are related to y by a known relation $y = f(x_1,\ldots,x_n)$; this relation may be a simple functional transformation, or complex algorithm (e.g., for the amount of oil, numerical solution to a partial differential equation). Then, to estimate y, we first measure or estimate the values of the quantities $x_1,\ldots,x_n$, and then we use the results $\widetilde{x}_1,\ldots,\widetilde{x}_n$ of these measurements (estimations) to compute an estimate $\widetilde{y}$ for y as $\widetilde{y} = f(\widetilde{x}_1,\ldots,\widetilde{x}_n)$

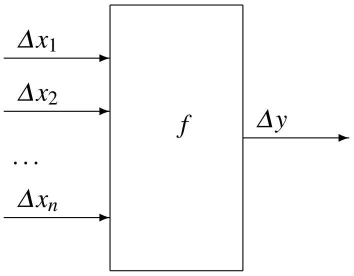
University of Texas at El Paso, El Paso, TX 79968, USA e-mail: vladik@utep.edu



Computing an estimate for y based on the results of direct measurements is called *data processing*; data processing is the main reason why computers were invented in the first place, and data processing is still one of the main uses of computers as number crunching devices.

Measurement uncertainty: from probabilities to intervals. Measurement are never 100% accurate, so in reality, the actual value x_i of i-th measured quantity can differ from the measurement result $\widetilde{x}_i$. Because of these *measurement errors* $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$, the result $\widetilde{y} = f(\widetilde{x}_1,\ldots,\widetilde{x}_n)$ of data processing is, in general, different from the actual value $y = f(x_1,\ldots,x_n)$ of the desired quantity y.

It is desirable to describe the error $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$ of the result of data processing. To do that, we must have some information about the errors of direct measurements.



What do we know about the errors Δx_i of direct measurements? First, the manufacturer of the measuring instrument must supply us with an upper bound Δ_i on the measurement error. If no such upper bound is supplied, this means that no accuracy is guaranteed, and the corresponding "measuring instrument" is practically useless. In this case, once we performed a measurement and got a measurement result $\widetilde{x}_i$, we know that the actual (unknown) value x_i of the measured quantity belongs to the interval $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$, where $\underline{x}_i = \widetilde{x}_i - \Delta_i$ and $\overline{x}_i = \widetilde{x}_i + \Delta_i$.

In many practical situations, we not only know the interval $[-\Delta_i, \Delta_i]$ of possible values of the measurement error; we also know the probability of different values Δx_i within this interval. This knowledge underlies the traditional engineering approach to estimating the error of indirect measurement, in which we assume that we know the probability distributions for measurement errors Δx_i.

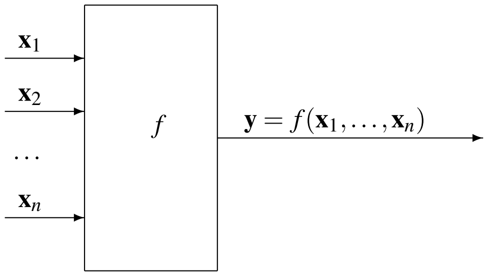
In practice, we can determine the desired probabilities of different values of Δx_i by comparing the results of measuring with this instrument with the results of measuring the same quantity by a standard (much more accurate) measuring instrument. Since the standard measuring instrument is much more accurate than the one use, the difference between these two measurement results is practically equal to the measurement error; thus, the empirical distribution of this difference is close to the desired probability distribution for measurement error. There are two cases, however, when this determination is not done:

- First is the case of cutting-edge measurements, e.g., measurements in fundamental science. When we use the largest particle accelerator to measure the properties of elementary particles, there is no "standard" (much more accurate) located nearby that we can use for calibration: our accelerator is the best we have.
- The second case is the case of measurements in manufacturing. In principle, every sensor can be thoroughly calibrated, but sensor calibration is so costly – usually costing ten times more than the sensor itself – that manufacturers rarely do it.

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

In this case, after we performed a measurement and got a measurement result $\widetilde{x}_i$, the only information that we have about the actual value x_i of the measured quantity is that it belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In such situations, the only information that we have about the (unknown) actual value of $y = f(x_1, \ldots, x_n)$ is that y belongs to the range $\mathbf{y} = [\underline{y}, \overline{y}]$ of the function f over the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$:

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



The process of computing this interval range based on the input intervals $\mathbf{x}_i$ is called *interval computations*; see, e.g., [4].

Outline. We start by recalling the basic techniques of interval computations and their drawbacks, then we will describe the new set computation techniques and describe a class of problems for which these techniques are efficient. Finally, we talk about how we can extend these techniques to other types of uncertainty (e.g., classes of probability distributions).

2 Interval Computations: Brief Reminder

Interval computations: main idea. Historically the first method for computing the enclosure for the range is the method which is sometimes called "straightforward" interval computations. This method is based on the fact that inside the computer, every algorithm consists of elementary operations (arithmetic operations, min, max, etc.). For each elementary operation $f(a,b)$, if we know the intervals $\mathbf{a}$ and $\mathbf{b}$ for a and b, we can compute the exact range $f(\mathbf{a},\mathbf{b})$. The corresponding formulas form the so-called *interval arithmetic*:

$$[\underline{a},\overline{a}]+[\underline{b},\overline{b}]=[\underline{a}+\underline{b},\overline{a}+\overline{b}];\ \ [\underline{a},\overline{a}]-[\underline{b},\overline{b}]=[\underline{a}-\overline{b},\overline{a}-\underline{b}];$$

$$[\underline{a},\overline{a}]\cdot[\underline{b},\overline{b}]=[\min(\underline{a}\cdot\underline{b},\underline{a}\cdot\overline{b},\overline{a}\cdot\underline{b},\overline{a}\cdot\overline{b}),\max(\underline{a}\cdot\underline{b},\underline{a}\cdot\overline{b},\overline{a}\cdot\underline{b},\overline{a}\cdot\overline{b})];$$

$$1/[\underline{a},\overline{a}]=[1/\overline{a},1/\underline{a}]\text{ if }0\not\in[\underline{a},\overline{a}];\ \ [\underline{a},\overline{a}]/[\underline{b},\overline{b}]=[\underline{a},\overline{a}]\cdot(1/[\underline{b},\overline{b}]).$$

In straightforward interval computations, we repeat the computations forming the program f step-by-step, replacing each operation with real numbers by the corresponding operation of interval arithmetic. It is known that, as a result, we get an enclosure $\mathbf{Y}\supseteq\mathbf{y}$ for the desired range.

From main idea to actual computer implementation. Not every real number can be exactly implemented in a computer; thus, e.g., after implementing an operation of interval arithmetic, we must enclose the result $[r^-,r^+]$ in a computer-representable interval: namely, we must round-off r^- to a smaller computer-representable value $\underline{r}$, and round-off r^+ to a larger computer-representable value $\overline{r}$.

Sometimes, we get excess width. In some cases, the resulting enclosure is exact; in other cases, the enclosure has excess width. The excess width is inevitable since straightforward interval computations increase the computation time by at most a factor of 4, while computing the exact range is, in general, NP-hard (see, e.g., [5]), even for computing the population variance $V=\frac{1}{n}\cdot\sum\limits_{i=1}^{n}(x_i-\overline{x})^2$, where $\overline{x}=\frac{1}{n}\cdot\sum\limits_{i=1}^{n}x_i$ (see [3]). If we get excess width, then we can use more sophisticated techniques to get a better estimate, such as centered form, bisection, etc.; see, e.g., [4].

Reason for excess width. The main reason for excess width is that intermediate results are dependent on each other, and straightforward interval computations ignore this dependence. For example, the actual range of $f(x_1)=x_1-x_1^2$ over $\mathbf{x}_1=[0,1]$ is $\mathbf{y}=[0,0.25]$. Computing this f means that we first compute $x_2:=x_1^2$ and then subtract x_2 from x_1. According to straightforward interval computations, we compute $\mathbf{r}=[0,1]^2=[0,1]$ and then $\mathbf{x}_1-\mathbf{x}_2=[0,1]-[0,1]=[-1,1]$. This excess width comes from the fact that the formula for interval subtraction implicitly assumes that both a and b can take arbitrary values within the corresponding intervals $\mathbf{a}$ and $\mathbf{b}$, while in this case, the values of x_1 and x_2 are clearly not independent: x_2 is uniquely determined by x_1, as $x_2=x_1^2$.

Why not use uniform distributions? Since we have no information about which values within a given interval are more probable and which are less probable, why not assume that these values are equally probable, i.e., that the distribution is uniform?

Similarly, for several variables, why not assume a uniform distribution on the corresponding box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$ – which is mathematically equivalent to assuming that we have n independent random variables x_i uniformly distributed in the corresponding intervals $\mathbf{x}_i$. This is indeed one of the main ways how interval uncertainty is treated in engineering practice.

To explain the limitations of this engineering approach, let us consider the simplest possible algorithm $y = f(x_1, \ldots, x_i, \ldots, x_n) = x_1 + \ldots + x_i + \ldots + x_n$. For simplicity, let us assume that the measured values of all n quantities are 0s $\widetilde{x}_1 = \ldots = \widetilde{x}_i = \ldots = \widetilde{x}_n = 0$, and that all n measurements have the same error bound Δ_x; $\Delta_1 = \ldots = \Delta x_i = \ldots = \Delta_n = \Delta_x$.

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

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

A normal distribution is uniquely determined by its mean and variance. When we add several independent variables, their means and variances add up. For each uniform distribution Δx_i on the interval $[-\Delta_x, \Delta_x]$ of width $2\Delta_x$, the mean is 0 and the variance is $V = \frac{1}{3} \cdot \Delta_x^2$. Thus, for the sum Δy of n such variables, the mean is 0, and the variance is equal to $(n/3) \cdot \Delta_x^2$. Hence, the standard deviation is equal to $\sigma = \sqrt{V} = \Delta_x \cdot \frac{\sqrt{n}}{\sqrt{3}}$.

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

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

3 Constraint-Based Set Computations

Main idea. The main idea behind constraint-based set computations (see, e.g., [1]) is to remedy the above reason why interval computations lead to excess width.

Specifically, at every stage of the computations, in addition to keeping the *intervals* $\mathbf{x}_i$ of possible values of all intermediate quantities x_i, we also keep several *sets*:

- sets $\mathbf{x}_{ij}$ of possible values of pairs (x_i, x_j);
- if needed, sets $\mathbf{x}_{ijk}$ of possible values of triples (x_i, x_j, x_k); etc.

In the above example, instead of just keeping two intervals $\mathbf{x}_1 = \mathbf{x}_2 = [0,1]$, we would then also generate and keep the set $\mathbf{x}_{12} = \{(x_1, x_1^2)\,|\,x_1 \in [0,1]\}$. Then, the desired range is computed as the range of $x_1 - x_2$ over this set – which is exactly $[0, 0.25]$.

To the best of our knowledge, in interval computations context, the idea of representing dependence in terms of sets of possible values of tuples was first described by Shary; see, e.g., [6, 7] and references therein.

How can we propagate this set uncertainty via arithmetic operations? Let us describe this on the example of addition, when, in the computation of f, we use two previously computed values x_i and x_j to compute a new value $x_k := x_i + x_j$. In this case, we set $\mathbf{x}_{ik} = \{(x_i, x_i + x_j)\,|\,(x_i, x_j) \in \mathbf{x}_{ij}\}$, $\mathbf{x}_{jk} = \{(x_j, x_i + x_j)\,|\,(x_i, x_j) \in \mathbf{x}_{ij}\}$, and for every $l \neq i, j$, we take

$$\mathbf{x}_{kl} = \{(x_i + x_j, x_l)\,|\,(x_i, x_j) \in \mathbf{x}_{ij}, (x_i, x_l) \in \mathbf{x}_{il}, (x_j, x_l) \in \mathbf{x}_{jl}\}.$$

From main idea to actual computer implementation. In interval computations, we cannot represent an arbitrary interval inside the computer, we need an enclosure. Similarly, we cannot represent an arbitrary set inside a computer, we need an enclosure.

To describe such enclosures, we fix the number C of granules (e.g., $C = 10$). We divide each interval $\mathbf{x}_i$ into C equal parts $\mathbf{X}_i$; thus each box $\mathbf{x}_i \times \mathbf{x}_j$ is divided into C^2 subboxes $\mathbf{X}_i \times \mathbf{X}_j$. We then describe each set $\mathbf{x}_{ij}$ by listing all subboxes $\mathbf{X}_i \times \mathbf{X}_j$ which have common elements with $\mathbf{x}_{ij}$; the union of such subboxes is an enclosure for the desired set $\mathbf{x}_{ij}$.

This implementation enables us to implement all above arithmetic operations. For example, to implement $\mathbf{x}_{ik} = \{(x_i, x_i + x_j)\,|\,(x_i, x_j) \in \mathbf{x}_{ij}\}$, we take all the subboxes $\mathbf{X}_i \times \mathbf{X}_j$ that form the set $\mathbf{x}_{ij}$; for each of these subboxes, we enclosure the corresponding set of pairs $\{(x_i, x_i + x_j)\,|\,(x_i, x_j) \in \mathbf{X}_i \times \mathbf{X}_j\}$ into a set $\mathbf{X}_i \times (\mathbf{X}_i + \mathbf{X}_j)$. This set may have non-empty intersection with several subboxes $\mathbf{X}_i \times \mathbf{X}_k$; all these subboxes are added to the computed enclosure for $\mathbf{x}_{ik}$. Once can easily see if we start with the exact range $\mathbf{x}_{ij}$, then the resulting enclosure for $\mathbf{x}_{ik}$ is an $(1/C)$-approximation to the actual set – and so when C increases, we get more and more accurate representations of the desired set.

Similarly, to find an enclosure for

$$\mathbf{x}_{kl} = \{(x_i + x_j, x_l)\,|\,(x_i, x_j) \in \mathbf{x}_{ij}, (x_i, x_l) \in \mathbf{x}_{il}, (x_j, x_l) \in \mathbf{x}_{jl}\},$$

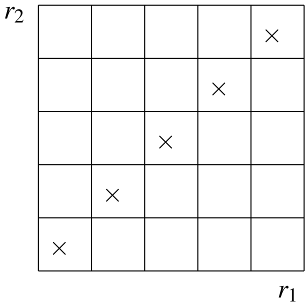
we consider all the triples of subintervals $(\mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_l)$ for which $\mathbf{X}_i \times \mathbf{X}_j \subseteq \mathbf{x}_{ij}$, $\mathbf{X}_i \times \mathbf{X}_l \subseteq \mathbf{x}_{il}$, and $\mathbf{X}_j \times \mathbf{X}_l \subseteq \mathbf{x}_{jl}$; for each such triple, we compute the box $(\mathbf{X}_i + \mathbf{X}_j) \times \mathbf{X}_l$; then, we add subboxes $\mathbf{X}_k \times \mathbf{X}_l$ which intersect with this box to the enclosure for $\mathbf{x}_{kl}$.

First example: computing the range of $x-x$. 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 $\mathbf{r}_{12}$:



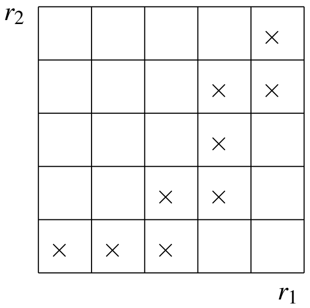
For each small box, we have $[-0.2,0.2]$, so the union is $[-0.2,0.2]$.

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

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 $\mathbf{R}_1^2=[0.16,0.36]$, so two sub-intervals $[0,0.2]$ and $[0.2,0.4]$ are affected, etc.

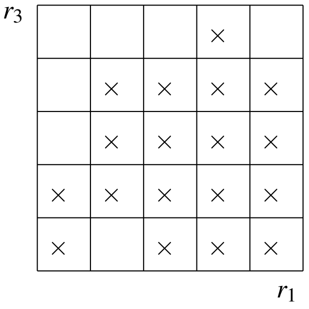


For each possible pair of small boxes $\mathbf{R}_1 \times \mathbf{R}_2$, we have $\mathbf{R}_1 - \mathbf{R}_2 = [-0.2, 0.2]$, $[0, 0.4]$, or $[0.2, 0.6]$, so the union of $\mathbf{R}_1 - \mathbf{R}_2$ is $\mathbf{r}_3 = [-0.2, 0.6]$.

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

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 $[-0.2, -0.04]$, $[-0.04, 0.12]$, and $[0.12, 0.28]$.
- 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.



Limitations of this approach. The main limitation of this approach is that when we need an accuracy ε, we must use $\sim 1/\varepsilon$ granules; so, if we want to compute the result with k digits of accuracy, i.e., with accuracy $\varepsilon = 10^{-k}$, we must consider exponentially many boxes ($\sim 10^k$). In plain words, this method is only applicable when we want to know the desired quantity with a given accuracy (e.g., 10%).

Cases when this approach is applicable. In practice, there are many problems when it is sufficient to compute a quantity with a given accuracy: e.g., when we detect an outlier, we usually do not need to know the variance with a high accuracy, an accuracy of 10% is more than enough.

Let us describe the case when interval computations do not lead to the exact range, but set computations do – of course, the range is "exact" modulo accuracy of the actual computer implementations of these sets.

Example: estimating variance under interval uncertainty. Suppose that we know the intervals $\mathbf{x}_1,\ldots,\mathbf{x}_n$ of possible values of $x_1,\ldots,x_n$, and we need to compute the range of the variance $V = \frac{1}{n}\cdot M - \frac{1}{n^2}\cdot E^2$, where $M \stackrel{\text{def}}{=} \sum\limits_{i=1}^{n} x_i^2$ and $E \stackrel{\text{def}}{=} \sum\limits_{i=1}^{n} x_i$.

This problem is important, e.g., in detecting outliers. Outliers are useful in many application areas. For example, in medicine, to detect possible illnesses, we analyze the healthy population, compute the averages $E[x]$ and the standard deviations $\sigma[x]$ of different characteristics x, and if for some person, the value of a blood pressure, weight, body temperature, etc., is outside the corresponding 2- or 3-sigma interval $[E[x]-k_0\cdot\sigma[x], E[x]+k_0\cdot\sigma[x]]$, then we perform additional tests to see if there is any hidden problem with this person's health. Similarly, in geophysics, when we look for rare minerals, we know the typical values for a given area, and if at some location, the values of the geophysical characteristics are outliers (i.e., they are outside the corresponding interval), then these area are probably the most promising.

Traditional algorithms for detecting outliers assume that we know the exact values x_i of the corresponding characteristics but in practice, these values often come from estimates or crude measurements. For example, most routine blood pressure measurements performed at health fairs, in drugstores, at the dentist office, etc., are very approximate, with accuracy 10 or more; their objective is not to find the exact values of the corresponding characteristics but to make sure that we do not miss a dangerous anomaly. When we estimate the mean and the standard deviations based on these approximate measurements, we need to take into account that these values are very approximate, i.e., that, in effect, instead of the exact value x_i (such as 110), we only know that the actual (unknown) value of the blood pressure is somewhere within the interval $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i] = [110-10, 110+10[= [100, 120]$.

In all these situations, we need to compute the range on the variance V under the interval uncertainty on x_i.

A natural way to to compute V is to compute the intermediate sums $M_k \stackrel{\text{def}}{=} \sum\limits_{i=1}^{k} x_i^2$ and $E_k \stackrel{\text{def}}{=} \sum\limits_{i=1}^{k} x_i$. We start with $M_0 = E_0 = 0$; once we know the pair (M_k, E_k), we compute $(M_{k+1}, E_{k+1}) = (M_k + x_{k+1}^2, E_k + x_{k+1})$. Since the values of M_k and E_k only depend on $x_1, \ldots, x_k$ and do not depend on x_{k+1}, we can conclude that if (M_k, E_k) is a possible value of the pair and x_{k+1} is a possible value of this variable, then $(M_k + x_{k+1}^2, E_k + x_{k+1})$ is a possible value of (M_{k+1}, E_{k+1}). So, the set $\mathbf{p}_0$ of possible values of (M_0, E_0) is the single point $(0,0)$; once we know the set $\mathbf{p}_k$ of possible values of (M_k, E_k), we can compute $\mathbf{p}_{k+1}$ as

$$\{(M_k + x^2, E_k + x) \mid (M_k, E_k) \in \mathbf{p}_k, x \in \mathbf{x}_{k+1}\}.$$

For $k = n$, we will get the set $\mathbf{p}_n$ of possible values of (M, E); based on this set, we can then find the exact range of the variance $V = \frac{1}{n} \cdot M - \frac{1}{n^2} \cdot E^2$.

What C should we choose to get the results with an accuracy $\varepsilon \cdot \overline{V}$? On each step, we add the uncertainty of $1/C$; to, after n steps, we add the inaccuracy of n/C. Thus, to get the accuracy $n/C \approx \varepsilon$, we must choose $C = n/\varepsilon$.

What is the running time of the resulting algorithm? We have n steps; on each step, we need to analyze C^3 combinations of subintervals for E_k, M_k, and x_{k+1}. Thus, overall, we need $n \cdot C^3$ steps, i.e., n^4/ε^3 steps. For fixed accuracy $C \sim n$, so we need $O(n^4)$ steps – a polynomial time, and for $\varepsilon = 1/10$, the coefficient at n^4 is still 10^3 – quite feasible.

For example, for $n = 10$ values and for the desired accuracy $\varepsilon = 0.1$, we need $10^3 \cdot n^4 \approx 10^7$ computational steps – “nothing” for a Gigaherz (10^9 operations per second) processor on a usual PC. For $n = 100$ values and the same desired accuracy, we need $10^4 \cdot n^4 \approx 10^{12}$ computational steps, i.e., 10^3 seconds (15 minutes) on a Gigaherz processor. For $n = 1000$, we need 10^{15} steps, i.e., 10^6 computational steps – 12 days on a single processor or a few hours on a multi-processor machine.

In comparison, the exponential time 2^n needed in the worst case for the exact computation of the variance under interval uncertainty, is doable ($2^{10} \approx 10^3$ step) for $n = 10$, but becomes unrealistically astronomical ($2^{100} \approx 10^{30}$ steps) already for $n = 100$.

Comment. When the accuracy increases $\varepsilon = 10^{-k}$, we get an exponential increase in running time – but this is OK since, as we have mentioned, the problem of computing variance under interval uncertainty is, in general, NP-hard.

Other statistical characteristics. Similar algorithms can be presented for computing many other statistical characteristics. For example, for every integer $d > 2$, the corresponding higher-order central moment $C_d = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^d$ is a linear combination of d moments $M^{(j)} \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i^j$ for $j = 1, \ldots, d$; thus, to find the exact range for C_d, we can keep, for each k, the set of possible values of d-dimensional tuples $(M_k^{(1)}, \ldots, M_k^{(d)})$, where $M_k^{(j)} \stackrel{\text{def}}{=} \sum_{i=1}^{k} x_i^j$. For these computations, we need $n \cdot C^{d+1} \sim n^{d+2}$ steps – still a polynomial time.

Another example is covariance $\mathrm{Cov} = \frac{1}{n} \cdot \sum\limits_{i=1}^{n} x_i \cdot y_i - \frac{1}{n^2} \cdot \sum\limits_{i=1}^{n} x_i \cdot \sum\limits_{i=1}^{n} y_i$. To compute covariance, we need to keep the values of the triples $(\mathrm{Cov}_k, X_k, Y_k)$, where $\mathrm{Cov}_k \stackrel{\text{def}}{=} \sum\limits_{i=1}^{k} x_i \cdot y_i$, $X_k \stackrel{\text{def}}{=} \sum\limits_{i=1}^{k} x_i$, and $Y_k \stackrel{\text{def}}{=} \sum\limits_{i=1}^{k} y_i$. At each step, to compute the range of

$$(\mathrm{Cov}_{k+1}, X_{k+1}, Y_{k+1}) = (\mathrm{Cov}_k + x_{k+1} \cdot y_{k+1}, X_k + x_{k+1}, Y_k + y_{k+1}),$$

we must consider all possible combinations of subintervals for Cov_k, X_k, Y_k, x_{k+1}, and y_{k+1} – to the total of C^5. Thus, we can compute covariance in time $n \cdot C^5 \sim n^6$.

Similarly, to compute correlation $\rho = \mathrm{Cov}/\sqrt{V_x \cdot V_y}$, we can update, for each k, the values of $(C_k, X_k, Y_k, X_k^{(2)}, Y_k^{(2)})$, where $X_k^{(2)} = \sum\limits_{i=1}^{k} x_i^2$ and $Y_k^{(2)} = \sum\limits_{i=1}^{k} y_i^2$ are needed to compute the variances V_x and V_y. These computations require time $n \cdot C^7 \sim n^8$.

Systems of ordinary differential equations (ODEs) under interval uncertainty. A general system of ODEs has the form $\dot{x}_i = f_i(x_1, \ldots, x_m, t)$, $1 \le i \le m$. Interval uncertainty usually means that the exact functions f_i are unknown, we only know the expressions of f_i in terms of parameters, and we have interval bounds on these parameters.

There are two types of interval uncertainty: we may have global parameters whose values are the same for all moments t, and we may have noise-like parameters whose values may different at different moments of time – but always within given intervals. In general, we have a system of the type

$$\dot{x}_i = f_i(x_1, \ldots, x_m, t, a_1, \ldots, a_k, b_1(t), \ldots, b_l(t)),$$

where f_i is a known function, and we know the intervals $\mathbf{a}_j$ and $\mathbf{b}_j(t)$ of possible values of a_i and $b_j(t)$.

Example. For example, the case of a differential inequality when we only know the bounds $\underline{f}_i(x_1, \ldots, x_n, t)$ and $\overline{f}_i(x_1, \ldots, x_n, t)$ on $f_i(x_1, \ldots, x_n, t)$ can be described as

$$\widetilde{f}_i(x_1, \ldots, x_n, t) + b_1(t) \cdot \Delta(x_1, \ldots, x_n, t),$$

where $\widetilde{f}_i(x_1, \ldots, x_n, t) \stackrel{\text{def}}{=} (\underline{f}_i(x_1, \ldots, x_n, t) + \overline{f}_i(x_1, \ldots, x_n, t))/2$, $\Delta(x_1, \ldots, x_n, t) \stackrel{\text{def}}{=} (\overline{f}_i(x_1, \ldots, x_n, t) - \underline{f}_i(x_1, \ldots, x_n, t))/2$, and $\mathbf{b}_1(t) = [-1, 1]$.

Solving systems of ordinary differential equations (ODEs) under interval uncertainty. For the general system of ODEs, Euler's equations take the form

$$x_i(t + \Delta t) = x_i(t) + \Delta t \cdot f_i(x_1(t), \ldots, x_m(t), t, a_1, \ldots, a_k, b_1(t), \ldots, b_l(t)).$$

Thus, if for every t, we keep the set of all possible values of a tuple

$$(x_1(t), \ldots, x_m(t), a_1, \ldots, a_k),$$

then we can use the Euler's equations to get the exact set of possible values of this tuple at the next moment of time.

The reason for exactness is that the values $x_i(t)$ depend only on the previous values $b_j(t-\Delta t)$, $b_j(t-2\Delta t)$, etc., and not on the current values $b_j(t)$.

To predict the values $x_i(T)$ at a moment T, we need $n = T/\Delta t$ iterations.

To update the values, we need to consider all possible combinations of $m+k+l$ variables $x_1(t),\ldots,x_m(t),a_1,\ldots,a_k,b_1(t),\ldots,b_l(t)$; so, to predict the values at moment $T = n\cdot\Delta t$ in the future for a given accuracy $\varepsilon > 0$, we need the running time $n\cdot C^{m+k+l} \sim n^{k+l+m+1}$. This is is still polynomial in n.

Other possible cases when our approach is efficient. Similar computations can be performed in other cases when we have an iterative process where a fixed finite number of variables is constantly updated.

In such problems, there is an additional factor which speeds up computations. Indeed, in the modern computers, fetching a value from the memory, in general, takes much longer than performing an arithmetic operation. To decrease this time, computers have a hierarchy of memories – from registers from which the access is the fastest, to cash memory (second fastest), etc. Thus, to take full use of the speed of modern processors, we must try our best to keep all the intermediate results in the registers. In the problems in which, at each moment of time, we can only keep (and update) a small current values of the values, we can store all these values in the registers – and thus, get very fast computations (only the input values $x_1,\ldots,x_n$ need to be fetched from slower-to-access memory locations).

Additional advantage of our technique: possibility to take constraints into account. Traditional formulations of the interval computation problems assume that we can have arbitrary tuples $(x_1,\ldots,x_n)$ as long as $x_i \in \mathbf{x}_i$ for all i. In practice, we may have additional constraints on x_i. For example, we may know that x_i are observations of a smoothly changing signal at consequent moments of time; in this case, we know that $|x_i - x_{i+1}| \le \varepsilon$ for some small known $\varepsilon > 0$. Such constraints are easy to take into account in our approach.

For example, if know that $\mathbf{x}_i = [-1,1]$ for all i and we want to estimate the value of a high-frequency Fourier coefficient $f = x_1 - x_2 + x_3 - x_4 + \ldots - x_{2n}$, then usual interval computations lead to an enclosure $[-2n,2n]$, while, for small ε, the actual range for the sum $(x_1-x_2)+(x_3-x_4)+\ldots$ where each of n differences is bounded by ε, is much narrower: $[-n\cdot\varepsilon, n\cdot\varepsilon]$ (and for $x_i = i\cdot\varepsilon$, these bounds are actually attained).

Computation of f means computing the values $f_k = x_1 - x_2 + \ldots + (-1)^{k+1}\cdot x_k$ for $k = 1,\ldots$ At each stage, we keep the set $\mathbf{s}_k$ of possible values of (f_k,x_k), and use this set to find

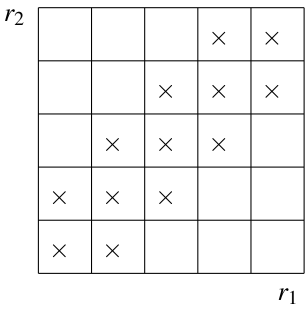
$$\mathbf{s}_{k+1} = \{(f_k + (-1)^k\cdot x_{k+1}, x_{k+1}) \,|\, (f_k,x_k) \in \mathbf{s}_k \,\&\, |x_k - x_{k+1}| \le \varepsilon\}.$$

In this approach, when computing f_{2k}, we take into account that the value x_{2k} must be ε-close to the value x_k and thus, that we only add $\le \varepsilon$. Thus, our approach leads to almost exact bounds – modulo implementation accuracy $1/C$.

In this simplified example, the problem is linear, so we could use linear programming to get the exact range, but set computations work for similar non-linear problems as well.

Toy example with a constraint. The problem is to find the range of $r_1 - r_2$ when $\mathbf{r}_1 = [0,1]$, $\mathbf{r}_2 = [0,1]$, and $|r_1 - r_2| \le 0.1$. 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:



r_2					
				×	×
			×	×	×
		×	×	×	
	×	×	×		
	×	×			
					r_1

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 $[-0.1, 0.1]$.

Towards possible extension to p-boxes and classes of probability distributions. Often, in addition to the interval $\mathbf{x}_i$ of possible values of the inputs x_i, we also have partial information about the probabilities of different values $x_i \in \mathbf{x}_i$. An exact probability distribution can be described, e.g., by its cumulative distribution function $F_i(z) = \text{Prob}(x_i \le z)$. In these terms, a partial information means that instead of a single cdf, we have a *class* $\mathscr{F}$ of possible cdfs.

A practically important particular case of this partial information is when, for each z, instead of the exact value $F(z)$, we know an interval $\mathbf{F}(z) = [\underline{F}(z), \overline{F}(z)]$ of possible values of $F(z)$; such an "interval-valued" cdf is called a *probability box*, or a *p-box*, for short; see, e.g., [2].

Propagating p-box uncertainty via computations: a problem. Once we know the classes $\mathscr{F}_i$ of possible distributions for x_i, and a data processing algorithms $f(x_1, \ldots, x_n)$, we would like to know the class $\mathscr{F}$ of possible resulting distributions for $y = f(x_1, \ldots, x_n)$.

Idea. For problems like systems of ODES, it is sufficient to keep, and update, for all t, the set of possible joint distributions for the tuple $(x_1(t), \ldots, a_1, \ldots)$.

From idea to computer implementation. We would like to estimate the values with some accuracy $\varepsilon \sim 1/C$ and the probabilities with the similar accuracy $1/C$. To describe a distribution with this uncertainty, we divide both the x-range and the probability (p-) range into C granules, and then describe, for each x-granule, which p-granules are covered. Thus, we enclose this set into a finite union of p-boxes which assign, to each of x-granules, a finite union of p-granule intervals.

A general class of distributions can be enclosed in the union of such p-boxes. There are finitely many such assignments, so, for a fixed C, we get a finite number of possible elements in the enclosure.

We know how to propagate uncertainty via simple operations with a finite amount of p-boxes (see, e.g., [2]), so for ODEs we get a polynomial-time algorithm for computing the resulting p-box for y.

For p-boxes, we need further improvements to make this method practical. Formally, the above method is polynomial-time. However, it is not yet practical beyond very small values of C. Indeed, in the case of interval uncertainty, we needed C^2 or C^3 subboxes. This amount is quite feasible even for $C = 10$.

To describe a p-subbox, we need to attach one of C probability granules to each of C x-granules; these are $\sim C^C$ such attachments, so we need $\sim C^C$ subboxes. For $C = 10$, we already get an unrealistic 10^{10} increase in computation time.

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 from the Science and Technology Centre in Ukraine (STCU), funded by European Union.

The authors are thankful to all the participants of the International Union for Theoretical and Applied Mechanics (IUTAM) Symposium on The Vibration Analysis of Structures with Uncertainties, Saint Petersburg, Russia, July 6–10, 2009.

References

1. M. Ceberio, 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”, *Proc. 2nd Int’l Workshop on Reliable Engineering Computing*, Savannah, Georgia, February 22–24, 2006, pp. 127–154; final version in *Journal of Uncertain Systems*, 2007, Vol. 1, No. 1, pp. 11–34.
2. S. Ferson. *RAMAS Risk Calc 4.0.* CRC Press, Boca Raton, Florida, 2002.
3. S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles, “Computing Variance for Interval Data is NP-Hard”, *ACM SIGACT News*, 2002, Vol. 33, No. 2, pp. 108–118.
4. L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*, Springer, London, 2001.
5. V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1997.
6. S. P. Shary, “Parameter partitioning scheme for interval linear systems with constraints”, *Proceedings of the International Workshop on Interval Mathematics and Constraint Propagation Methods (ICMP’03)*, July 8–9, 2003, Novosibirsk, Akademgorodok, Russia, pp. 1–12 (in Russian).
7. S. P. Shary, “Solving tied interval linear systems”, *Siberian Journal of Numerical Mathematics*, 2004, Vol. 7, No. 4, pp. 363–376 (in Russian).