3-2011

Towards Faster Estimation of Statistics and ODEs Under Interval, P-Box, and Fuzzy Uncertainty: From Interval Computations to Rough Set-Related Computations

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

Comments:
Technical Report: UTEP-CS-11-09b
Published in In: Sergey O. Kuznetsov et al. (Eds.) *Proceedings of the Thirteenth International Conference on Rough Sets, Fuzzy Sets and Granular Computing RSFDGrC'2011 (Moscow, Russia, June 25-27, 2011),* Springer Lecture Notes on Artificial Intelligence LNAI, Springer-Verlag, Berlin, Heidelberg, 2011, Vol. 6743, pp. 3-10.
Proceedings of the Thirteenth International Conference on Rough Sets, Fuzzy Sets and Granular Computing RSFDGrC'2011, Moscow, Russia, June 25-27, 2011.

Towards Faster Estimation of Statistics and ODEs Under Interval, P-Box, and Fuzzy Uncertainty: From Interval Computations to Rough Set-Related Computations

Vladik Kreinovich

University of Texas at El Paso, El Paso, TX 79968, USA
vladik@utep.edu

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 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. In this paper, we show that one way to remedy this problem is to extend interval technique to *rough-set computations*, where at each stage, in addition to intervals of possible values of the quantities, we also keep rough sets representing possible values of pairs (triples, etc.).
The paper's outline is as follows: we formulate the main problem (Section 1), briefly overview interval computations techniques solve this problem (Section 2), and then explain how the main ideas behind interval computation techniques can be extended to computations with rough sets (Section 3).

Keywords: interval computations, interval uncertainty, rough sets, statistics under interval uncertainty

1 Formulation of the Problem

Need for interval computations. In many real-life situations, we need to process data, i.e., to apply an algorithm $f(x_1, \ldots, x_n)$ to measurement results $x_1, \ldots, x_n$.

Measurements 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.

In many practical situations, we only know the upper bound Δ_i on the (absolute value of) the measurement errors Δx_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].

Case of fuzzy uncertainty and its reduction to interval uncertainty. In addition to bounds, we can also have expert estimates on Δx_i. An expert usually describes his/her uncertainty by using words from a natural language, like "most probably, the value of the quantity is between 3 and 4". To formalize this knowledge, it is natural to use *fuzzy set theory*, a formalism specifically designed for describing this type of informal ("fuzzy") knowledge; see, e.g., [5].

In fuzzy set theory, the expert's uncertainty about x_i is described by a fuzzy set, i.e., by a function $\mu_i(x_i)$ which assigns, to each possible value x_i of the i-th quantity, the expert's degree of certainty that x_i is a possible value. A fuzzy set can also be described as a nested family of α-cuts $\mathbf{x}_i(\alpha) \stackrel{\text{def}}{=} \{x_i \,|\, \mu_i(x_i) \geq \alpha\}$.

Zadeh's extension principle can be used to transform the fuzzy sets for x_i into a fuzzy set for y. It is known that for continuous functions f on a bounded domain this principle is equivalent to saying that, for every α,

$$\mathbf{y}(\alpha) = f(\mathbf{x}_1(\alpha), \ldots, \mathbf{x}_n(\alpha)).$$

In other words, fuzzy data processing can be implemented as layer-by-layer interval computations. In view of this reduction, in the following text, we will mainly concentrate on interval computations.

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*:

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

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

$$1/[\underline{a}, \overline{a}] = [1/\overline{a}, 1/\underline{a}] \text{ if } 0 \notin [\underline{a}, \overline{a}]; \quad [\underline{a}, \overline{a}]/[\underline{b}, \overline{b}] = [\underline{a}, \overline{a}] \cdot (1/[\underline{b}, \overline{b}]).$$

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., [6]), even for computing the population variance $V = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^2$, where $\overline{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$ (see [3]). If we get excess width, then we can use techniques such as centered form, bisection, etc., to get a better estimate; 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 $x_2 = x_1^2$.

3 Rough Set Computations

Main idea. The main idea behind (rough) set computations (see, e.g., [1, 7, 8]) 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 $[0, 0.25]$.

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 enclosure is a P-upper approximation to the desired set.

This 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 $\mathbf{x}_{ik}$. One can easily see that 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

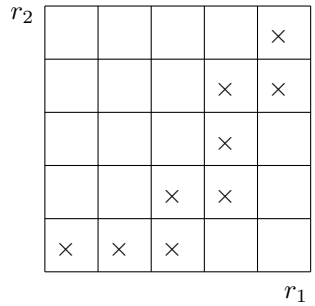
$$\mathbf{x}_{kl} = \{(x_i + x_j, x_l) \,|\, (x_i, x_j) \in \mathbf{x}_{ij}, (x_i, x_l) \in \mathbf{x}_{il}, (x_j, x_l) \in \mathbf{x}_{jl}\},$$

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 $\mathbf{x}_{kl}$.

Toy example: computing the range of $x - x^2$. In straightforward interval computations, we have $r_1 = x$ with the exact interval range $\mathbf{r}_1 = [0, 1]$, and we have $r_2 = x^2$ 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.

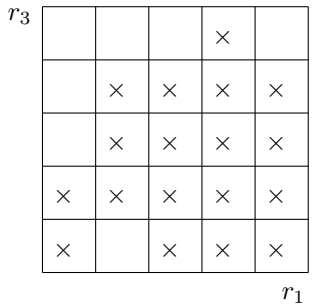


For each possible pair of small boxes $\mathbf{R}_1 \times \mathbf{R}_2$, we have $\mathbf{R}_1 - \mathbf{R}_2 = [-0.2, 0.2]$, $[0, 0.4]$, or $[0.2, 0.6]$, so the union of $\mathbf{R}_1 - \mathbf{R}_2$ is $\mathbf{r}_3 = [-0.2, 0.6]$.

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 $M \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i^2$ and $E \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i$.

A natural way to compute V is to compute the intermediate sums $M_k \stackrel{\text{def}}{=} \sum_{i=1}^{k} x_i^2$ and $E_k \stackrel{\text{def}}{=} \sum_{i=1}^{k} x_i$. 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 $(0, 0)$, and once we know the set $\mathbf{p}_k$ of possible values of (M_k, E_k), we can compute $\mathbf{p}_{k+1}$ as

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

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 $1/C$. So, 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; at 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 $C \sim n$, 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 seconds – 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 ($2^{10} \approx 10^3$

steps) for $n = 10$, but becomes unrealistically astronomical ($2^{100} \approx 10^{30}$ steps) already for $n = 100$.

Comment. When the accuracy increases to $\varepsilon = 10^{-k}$, 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 [1].

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 parameters whose values may differ 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 $b_j(t)$.

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 still polynomial in n.

Towards 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 (cdf) $F_i(z) = \mathrm{Prob}(x_i \le z)$. In these terms, a partial information means that instead of a single cdf, we have a *class* $\mathcal{F}$ 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 $\mathcal{F}_i$ of possible distributions for x_i, and data processing algorithms $f(x_1, \ldots, x_n)$, we would like to know the class $\mathcal{F}$ 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 $(x_1(t), \ldots, a_1, \ldots)$.

Conclusions. In many practical situations, for each quantity x_i, we only know the upper bound Δ_i on the measurement error $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$; in this case, once we know the measurement result $\widetilde{x}_i$, the only information that we have about the actual (unknown) value x_i is that it belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. For each quantity $y = f(x_1, \ldots, x_n)$, different values $x_i \in \mathbf{x}_i$ lead, in general, to different values y; it is therefore desirable to find the range $\mathbf{y}$ of all such values. In this paper, we show that for many problems, we can efficiently compute this range if we follow the original computation of y step-by-step with a rough set instead of a collection of exact values: we start with a box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$, and then estimate rough sets corresponding to each intermediate result.

Acknowledgments. This work was supported in part by the National Science Foundation grants HRD-0734825 and DUE-0926721 and by Grant 1 T36 GM078000-01 from the National Institutes of Health. The author is thankful to Dominik Slezak and Sergey Kuznetsov for the invitation and for the helpful editing advise.

References

1. Ceberio, C., Ferson, S., Kreinovich, V., Chopra, S., Xiang, G., Murguia, A., Santillan, J.: 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, 127–154 (2006); final version: Journal of Uncertain Systems, 1(1), 11–34 (2007)
2. Ferson, S.: *RAMAS Risk Calc 4.0*, CRC Press, Boca Raton, Florida (2002)
3. Ferson, S., Ginzburg, L., Kreinovich, V., Aviles, M.: Computing variance for interval data is NP-hard, ACM SIGACT News, 33(2), 108–118 (2002)
4. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis, Springer, London (2001)
5. Klir, G., Yuan, B.: Fuzzy Sets and Fuzzy Logic, Prentice Hall, Upper Saddle River, NJ (1995)
6. Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational Complexity and Feasibility of Data Processing and Interval Computations, Kluwer, Dordrecht (1997)
7. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data, Kluwer, Dordrecht (1991)
8. Shary, S. P.: Solving tied interval linear systems, Siberian Journal of Numerical Mathematics, 7(4), 363–376 (2004) in Russian