Fast Algorithms for Uncertainty Propagation, and Their Applications to Structural Integrity

Andrzej Pownuk, Jakub Cerveny, and Jerald J. Brady
University of Texas at El Paso
El Paso, TX 79968
Contact email ampownuk@utep.edu

***Abstract*—In many practical situations, we need to know how uncertainty propagates through data processing algorithms, i.e., how the uncertainty in the inputs affects the results of data processing. This problem is important for all types of uncertainty: probabilistic, interval, and fuzzy. From the computational viewpoint, however, this problem is much more complex for interval and fuzzy uncertainty. Therefore, for these types of uncertainty, it is desirable to design faster algorithms.**

In this paper, we describe faster algorithms for two practically important situations:

- *linearization* **situations, when the approximation errors are small and therefore, the data processing algorithms can be replaced by a linear function, and**
- *monotonic* **situations, when the dependence of the result y of data processing on each of the inputs $x_1, \ldots, x_n$ is either monotonically increasing or monotonically decreasing.**

I. Practical Need for Uncertainty Propagation

In many practical situations, we are interested in the value of a quantity y which is difficult or even impossible to measure directly. To estimate this difficult-to-measure quantity y, we measure or estimate related easier-to-measure quantities $x_1, \ldots, x_n$ which are related to the desired quantity y by a known relation $y = f(x_1, \ldots, x_n)$. Then, we apply the relation f to the estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$ for x_i and produce an estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for the desired quantity y.

In the simplest cases, the relation $f(x_1, \ldots, x_n)$ may be an explicit expression: e.g., if we know the current x_1 and the resistance x_2, then we can measure the voltage y by using Ohm's law $y = x_1 \cdot x_2$. In many practical situations, the relation between x_i and y is much more complicated: the corresponding algorithm $f(x_1, \ldots, x_n)$ is not an explicit expression, but a complex algorithm for solving an appropriate non-linear equation (or system of equations).

Estimates are never absolutely accurate:

- measurements are never absolutely precise, and
- expert estimates can only provide approximate values of the directly measured quantities $x_1, \ldots, x_n$.

In both cases, the resulting estimates $\widetilde{x}_i$ are, in general, different from the actual (unknown) values x_i. Due to these estimation errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$, even if the relation $f(x_1, \ldots, x_n)$ is exact, the estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is different from the actual value $y = f(x_1, \ldots, x_n)$: $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y \neq 0$.

(In many situations, when the relation $f(x_1, \ldots, x_n)$ is only known approximately, there is an additional source of the approximation error in y caused by the uncertainty in knowing this relation.)

It is therefore desirable to find out how the uncertainty Δx_i in estimating x_i affects the uncertainty Δy in the desired quantity, i.e., how the uncertainties Δx_i *propagate* via the algorithm $f(x_1, \ldots, x_n)$.

II. Propagation of probabilistic uncertainty

Often, we know the probabilities of different values of Δx_i. For example, in many cases, we know that the approximation errors Δx_i are independent normally distributed with zero mean and known standard deviations σ_i; see, e.g., [14].

In this case, we can use known statistical techniques to estimate the resulting uncertainty Δy in y. For example, since we know the probability distributions, we can simulate them in the computer, i.e., use the Monte-Carlo simulation techniques to get a sample population $\Delta y^{(1)}, \ldots, \Delta y^{(N)}$ of the corresponding errors Δy. Based on this sample, we can then estimate the desired statistical characteristics of the desired approximation error Δy.

III. Propagation of interval uncertainty

In many other practical situations, we do not know these probabilities, we only know the upper bounds Δ_i on the (absolute values of) the corresponding measurement errors Δx_i: $|\Delta x_i| \leq \Delta$.

In this case, based on the known approximation $\widetilde{x}_i$, we can conclude that the actual (unknown) value of i-th auxiliary quantity x_i can take any value from the interval

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

To find the resulting uncertainty in y, we must therefore find the range $\mathbf{y} = [\underline{y}, \overline{y}]$ of possible values of y when $x_i \in \mathbf{x}_i$:

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=}$$

$$\{f(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

Computations of this range under interval uncertainty is called *interval computations*; see, e.g., [3], [4].

The corresponding computational problems are, in general, NP-hard [8]. Crudely speaking, this means that, in general, such problems require a large amount of computation time – and that therefore faster methods are needed.

IV. Propagation of fuzzy uncertainty

In many practical situations, the estimates $\widetilde{x}_i$ come from experts. Experts often describe the inaccuracy of their estimates in terms of imprecise words from natural language, such as "approximately 0.1", etc. A natural way to formalize such words is to use special techniques developed for formalizing this type of estimates – specifically, the technique of fuzzy logic; see, e.g., [5], [11].

In this technique, for each possible value of $x_i \in \mathbf{x}_i$, we describe the degree $\mu_i(x_i)$ to which this value is possible. For each degree of certainty α, we can determine the set of values of x_i that are possible with at least this degree of certainty – the *α-cut* $\mathbf{x}_i(\alpha) = \{x \,|\, \mu(x) \geq \alpha\}$ of the original fuzzy set. Vice versa, if we know α-cuts for every α, then, for each object x, we can determine the degree of possibility that x belongs to the original fuzzy set [2], [5], [9], [10], [11]. A fuzzy set can be thus viewed as a nested family of its (interval) α-cuts.

We already know how to propagate interval uncertainty. Thus, to propagate this fuzzy uncertainty, we can therefore consider, for each α, the fuzzy set y with the α-cuts

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

see, e.g., [2], [5], [9], [10], [11]. So, from the computational viewpoint, the problem of propagating fuzzy uncertainty can be reduced to several interval propagation problems.

V. Need for faster algorithms for uncertainty propagation

Summarizing the above analysis, we can conclude that in principle, we need to consider three possible types of uncertainty propagation: situations when we propagate probabilistic, interval, and fuzzy uncertainty.

For probabilistic uncertainty, there exist reasonable efficient uncertainty propagation algorithms such as Monte-Carlo simulations. In contrast, the problems of propagating interval and fuzzy uncertainty are, in general, computationally difficult. It is therefore desirable to design faster algorithms for propagating interval and fuzzy uncertainty.

The computational problem of propagating fuzzy uncertainty can be naturally reduced to the problem of propagating interval uncertainty. Because of this reduction, in the following text, we will mainly concentrate on faster algorithms for propagating interval uncertainty. Applications of the above algorithms to different structural integrity problems are described in [12], [13].

VI. Two situations

In this paper, we describe faster algorithms for two practically important situations:

- *linearization* situations, when the approximation errors are small and therefore, the data processing algorithms can be replaced by a linear function, and
- *monotonic* situations, when the dependence of the result y of data processing on each of the inputs $x_1, \ldots, x_n$ is either monotonically increasing or monotonically decreasing.

VII. Linearization situations: description

Due to the approximation errors $\Delta x_i = \widetilde{x}_i - x_i$, the unknown (actual) values $x_i = \widetilde{x}_i - \Delta x_i$ of the input quantities x_i are, in general, different from the approximate estimates $\widetilde{x}_i$. In many practical situations, the approximation errors Δx_i are small – e.g., when the approximations are obtained by reasonably accurate measurements. In such situations, we can ignore terms which are quadratic (and of higher order) in Δx_i.

VIII. Linearization situations: analysis

In the above situations, we can expand the expression for

$$\Delta y = \widetilde{y} - y = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(x_1, \ldots, x_n) =$$

$$f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n)$$

in Taylor series in Δx_i and keep only the linear terms in this expansion. In this case, we get

$$\Delta y = c_1 \cdot \Delta x_1 + \ldots + c_n \cdot \Delta x_n,$$

where we denoted

$$c_i \stackrel{\text{def}}{=} \frac{\partial f}{\partial x_i}(\widetilde{x}_1, \ldots, \widetilde{x}_n).$$

For a linear function, the largest possible value of Δy is obtained when each of the variables $\Delta x_i \in [-\Delta_i, \Delta_i]$ attains:

- either its largest value Δ_i (when $c_i \geq 0$)
- or its smallest value $-\Delta_i$ (when $c_i < 0$).

In both cases, the largest possible value of the corresponding term in Δy is equal to $|c_i| \cdot \Delta_i$. Thus, the largest possible value of Δy is equal to

$$\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n.$$

Similarly, the smallest possible value of Δy is obtained when each of the variables $\Delta x_i \in [-\Delta_i, \Delta_i]$ attains

- either its smallest value $-\Delta_i$ (when $c_i \geq 0$)
- or its largest value Δ_i (when $c_i < 0$).

In both cases, the smallest possible value of the corresponding term in Δy is equal to $-|c_i| \cdot \Delta_i$. Thus, the smallest possible value of Δy is equal to

$$-\Delta = -|c_1| \cdot \Delta_1 - \ldots - |c_n| \cdot \Delta_n.$$

Can can we transform these natural formulas into an algorithm? Due to the linearization assumption, we can estimate each partial derivative c_i as

$$c_i \approx \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}}{h_i}$$

for some small values h_i. So, we arrive at the following algorithm.

IX. Linearization situations: algorithm

To compute the range $\mathbf{y}$ of y, we do the following.

- First, we apply the algorithm f to the original estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$, resulting in the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.
- Second, for all i from 1 to n, we compute $f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n)$ for some small h_i and then compute
$$c_i = \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}}{h_i}.$$
- Finally, we compute
$$\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n$$
and the desired range $\mathbf{y} = [\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.

X. Linearization situations: computational complexity

The main computation time is spent on calling the time-consuming algorithm f. In the above uncertainty propagation algorithm, after one call to f to compute $\widetilde{y}$, we need n calls to f to compute the corresponding partial derivatives c_i and then, we can estimate the desired uncertainty Δ in y by using the above simple formula.

Overall, we thus need $n + 1$ calls to the algorithm f.

Comment. For large n, we can further reduce the number of calls to f if we use a special technique of Cauchy-based Monte-Carlo simulations, which enables us to use a fixed number of calls to f (≈ 200) for all possible values n; see, e.g., [6], [7].

XI. Monotonic situation with known directions of monotonicity: description

In the previous case, we considered situations in which we can safely ignore terms which are quadratic and of higher order in Δx_i and in which, therefore, the actual dependence $y = f(x_1, \ldots, x_n)$ can be safely approximated by a linear function $\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i$.

In many practical situations – especially in situations related to expert (fuzzy) estimates when approximation errors may not be small – we can no longer ignore quadratic and higher order terms in Δx_i, so we must consider approximating functions which are more general than the linear ones.

A natural more general class of functions comes from the fact that linear functions are monotonic in each of their variables Δx_i. This class is general enough, since in the generic case, all partial derivatives $\dfrac{\partial f}{\partial x_i}$ are non-zeros at the point $\widetilde{x} \stackrel{\text{def}}{=} (\widetilde{x}_1, \ldots, \widetilde{x}_n)$. Hence, these derivatives are also different from 0 in some vicinity of this point – and thus, in this vicinity, the derivatives retain their signs and thus, the function f is monotonic with respect to each of these variables.

In practice, we are indeed often sure that the dependence is monotonic. For example, the Ohm's law $y = f(x_1, x_2) = x_1 \cdot x_2$ is not a linear function of its two variables, but it is monotonic for $x_1, x_2 \geq 0$.

XII. Monotonic situation with known directions of monotonicity: analysis

For a monotonic function $f(x_1, \ldots, x_n)$, the largest possible value over the intervals $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ is attained when:

- the variables x_i w.r.t. which the function f is increasing attain their largest possible values $\widetilde{x}_i + \Delta_i$, and
- the variables x_i w.r.t. which the function f is decreasing attain their smallest possible values $\widetilde{x}_i - \Delta_i$.

In other words, if we denote, for each input x_i, the "monotonicity sign" ε_i as $\varepsilon_i = 1$ if f is increasing in x_i and $\varepsilon_i = -1$ if f is decreasing in x_i, then the largest possible value $\overline{y}$ of $y = f(x_1, \ldots, x_n)$ is attained when $x_i = \widetilde{x}_i + \varepsilon_i \cdot \Delta_i$ for all i, i.e.,
$$\overline{y} = f(\widetilde{x}_1 + \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n + \varepsilon_n \cdot \Delta_n).$$

Similarly, the smallest possible value $\underline{y}$ of $y = f(x_1, \ldots, x_n)$ is attained when $x_i = \widetilde{x}_i - \varepsilon_i \cdot \Delta_i$ for all i, i.e.,
$$\underline{y} = f(\widetilde{x}_1 - \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n - \varepsilon_n \cdot \Delta_n).$$

So, if we know with respect to which variables x_i it is increasing and with respect to which variables x_j this dependence is decreasing, then we can find the desired range $[\underline{y}, \overline{y}]$ of possible values of y by using the following algorithm.

XIII. Monotonic situation with known directions of monotonicity: algorithm

- First, we compute $\underline{y}$ as
$$\underline{y} = f(\widetilde{x}_1 - \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n - \varepsilon_n \cdot \Delta_n).$$
- Then, we compute $\overline{y}$ as
$$\overline{y} = f(\widetilde{x}_1 + \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n + \varepsilon_n \cdot \Delta_n).$$

The resulting range of possible values of y is $[\underline{y}, \overline{y}]$.

XIV. Monotonic situation with known directions of monotonicity: computational complexity

The above algorithm requires two calls to f.

XV. General monotonic situation (with unknown directions of monotonicity): description

In more complex situations, when the function $f(x_1, \ldots, x_n)$ is not an explicit expression but rather a complex algorithm, we may still know that the dependence is monotonic, but we do not know a priori with respect to which variables the dependence is increasing and with respect to which the dependence is decreasing.

XVI. General monotonic situation: analysis

In this case, for each i, we can find the corresponding direction of monotonicity ε_i by changing the value of the i-th input and tracing how this change will affect the resulting value f:
$$\varepsilon_i = \text{sign}(f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y})$$
for some small value $h_i > 0$; see, e.g., [6], [12], [13] and references therein. Thus, we arrive at the following algorithm.

XVII. General monotonic situation: algorithm

- First, we compute $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.
- Second, for all i from 1 to n, we compute $f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n)$ for some small value $h_i > 0$ and then find

$$\varepsilon_i = \mathrm{sign}(f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}).$$

- Finally, we compute the range $[\underline{y}, \overline{y}]$ by using the formulas

$$\underline{y} = f(\widetilde{x}_1 - \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n - \varepsilon_n \cdot \Delta_n);$$

$$\overline{y} = f(\widetilde{x}_1 + \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n + \varepsilon_n \cdot \Delta_n).$$

XVIII. General monotonic situation: computational complexity

In this case, in addition to the original call to f for computing $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$, we need n calls to find n signs ε_i, and then 2 more calls to compute $\underline{y}$ and $\overline{y}$. Overall, we thus need $n + 3$ calls to f.

XIX. Situation when the algorithm $f(x_1, \ldots, x_n)$ is a solution to a system of equations: description

We have mentioned that in many cases, the complex algorithm $f(x_1, \ldots, x_n)$ comes from the need to solve a difficult-to-solve non-linear system of equations. In such situations, the desired quantity y is one of several unknowns $y_1 = y$, y_2, $\ldots$, y_m which are related to the easier-to-estimate values by a system of non-linear equations

$$F_1(x_1, \ldots, x_n, y_1, \ldots, y_m) = 0;$$

$$\ldots$$

$$F_m(x_1, \ldots, x_n, y_1, \ldots, y_m) = 0.$$

Often, the functions $F_1, \ldots, F_m$ are reasonably easy to compute, it is the solution which requires a large amount of time.

We start with the values $\widetilde{y}_1, \ldots, \widetilde{y}_m$ which correspond to the (approximate) estimated values $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of the inputs. For these values, the following system of equations holds:

$$F_1(\widetilde{x}_1, \ldots, \widetilde{x}_n, \widetilde{y}_1, \ldots, \widetilde{y}_m) = 0;$$

$$\ldots$$

$$F_m(\widetilde{x}_1, \ldots, \widetilde{x}_n, \widetilde{y}_1, \ldots, \widetilde{y}_m) = 0.$$

We want to know how the uncertainty Δx_i in the inputs affects the uncertainty Δy_j in the outputs, i.e., how the inputs Δx_i affect the values Δy_j for which

$$F_1(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n, \widetilde{y}_1 - \Delta y_1, \ldots, \widetilde{y}_m - \Delta y_m) = 0;$$

$$\ldots$$

$$F_m(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n, \widetilde{y}_1 - \Delta y_1, \ldots, \widetilde{y}_m - \Delta y_m) = 0.$$

In this case, e.g., the above linearization algorithm means that we need to call f $n + 1$ times means that we need to solve difficult-to-solve non-linear system of equations $n + 3$ times. A natural question is: can we estimate the range $[\underline{y}, \overline{y}]$ faster, without actually having to solve that many systems of non-linear equations?

In this paper, we will show that such a speed-up is indeed possible. Preliminary speed-up results were described in [12], [13].

XX. Systems of equations, linearization case: analysis

First, we will consider the case when the functions F_k can be safely linearized, i.e., when the terms quadratic in Δx_i and in Δy_j can be safely ignored. In this case, the above equations lead to

$$\sum_{i=1}^{n} X_{ki} \cdot \Delta x_i + \sum_{j=1}^{m} Y_{kj} \cdot \Delta y_j = 0,$$

where we denoted

$$X_{ki} \stackrel{\text{def}}{=} \frac{\partial F_k}{\partial x_i}, \quad Y_{kj} \stackrel{\text{def}}{=} \frac{\partial F_k}{\partial y_j}.$$

Since the functions F_k are easy-to-compute, the computation of these partial derivatives does not require the time-consuming step of actually solving the non-linear system of equations. The above equations can be described in a matrix form: $X\Delta x = -Y\Delta y$. Thus, we can find the vector Δy as $\Delta y = -Y^{-1}X\Delta x$, i.e., as $\Delta y = -M\Delta x$, where the matrix M with elements m_{ji} has the form $M = Y^{-1}X$.

In this case,

$$\Delta y_j = \sum_{i=1}^{n} m_{ji} \cdot \Delta x_i.$$

In particular, for the desired quantity $y = y_1$, we get

$$\Delta y_1 = \sum_{i=1}^{n} m_{1i} \cdot \Delta x_i,$$

and thus, the interval of possible values of Δy is $[-\Delta, \Delta]$, where

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

Thus, we arrive at the following algorithm.

XXI. Systems of equations, linearization case: algorithm

- First, we use the estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$ to solve the original system of non-linear equations

$$F_1(\widetilde{x}_1, \ldots, \widetilde{x}_n, \widetilde{y}_1, \ldots, \widetilde{y}_m) = 0;$$

$$\ldots$$

$$F_m(\widetilde{x}_1, \ldots, \widetilde{x}_n, \widetilde{y}_1, \ldots, \widetilde{y}_m) = 0;$$

  and find the values $\widetilde{y}_1 = \widetilde{y}, \widetilde{y}_2 \ldots, \widetilde{y}_m$.
- Then, we compute the derivative matrices with elements

$$X_{ki} \stackrel{\text{def}}{=} \frac{\partial F_k}{\partial x_i}, \quad Y_{kj} \stackrel{\text{def}}{=} \frac{\partial F_k}{\partial y_j},$$

  by either analytically differentiating or by using numerical differentiation.

- Third, we compute the matrix $M = Y^{-1}X$ with elements m_{ji}, the bound

$$\Delta = \sum_{i=1}^{n} |m_{1i}| \cdot \Delta_i,$$

and the desired range $[\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.

XXII. Systems of equations, linearization case: computational complexity

According to the above algorithm, we can compute the desired bound Δ by solving the original system of non-linear equations only once: to find the original values $\widetilde{y}_j$.

Comment. This is clearly much faster than solving this system $n+1$ times. There is a computational advantage of using this technique even if the original system of equations is linear: in this case, instead of solving the system of equations $n+1$ times, we compute the inverse matrix once. It is known that asymptotically, the computation time needed to invert a matrix A is the same as the time needed to solve a general linear equation $Ax = b$ (see, e.g., [1]), so this idea indeed leads to a drastic decrease in computation time.

XXIII. System of equations, monotonic case: analysis

If we know that the dependence of y on each x_i is monotonic, but we do not know the direction of monotonicity, then we can find this direction by computing the sign ε_i of the partial derivative

$$m_{1i} = \frac{\partial y_1}{\partial x_i} :$$

$\varepsilon_i = \mathrm{sign}(m_{1i})$. Similarly to the linearized case, we can conclude that the matrix M with elements $m_{ki} = \dfrac{\partial y_k}{\partial x_i}$ can be computed as $M = Y^{-1}X$. Computing the partial derivatives X and Y does not require solving any system of non-linear equations. Thus, we arrive at the following algorithm.

XXIV. System of equations, monotonic case: algorithm

- First, we use the estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$ to solve the original system of non-linear equations

$$F_1(\widetilde{x}_1, \ldots, \widetilde{x}_n, \widetilde{y}_1, \ldots, \widetilde{y}_m) = 0;$$

$$\ldots$$

$$F_m(\widetilde{x}_1, \ldots, \widetilde{x}_n, \widetilde{y}_1, \ldots, \widetilde{y}_m) = 0;$$

and find the values $\widetilde{y}_1 = \widetilde{y}, \widetilde{y}_2 \ldots, \widetilde{y}_m$.
- Then, we compute the derivative matrices with elements

$$X_{ki} \stackrel{\mathrm{def}}{=} \frac{\partial F_k}{\partial x_i}, \quad Y_{kj} \stackrel{\mathrm{def}}{=} \frac{\partial F_k}{\partial y_j},$$

by either analytically differentiating or by using numerical differentiation.
- Third, we compute the matrix $M = Y^{-1}X$ with elements m_{ji}, and the signs $\varepsilon_i = \mathrm{sign}(m_{1i})$.
- To find $\underline{y}$, we solve the system

$$F_1(\widetilde{x}_1 - \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n - \varepsilon_n \cdot \Delta_n, y_1, \ldots, y_m) = 0;$$

$$\ldots$$

$$F_m(\widetilde{x}_1 - \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n - \varepsilon_n \cdot \Delta_n, y_1, \ldots, y_m) = 0;$$

and take $\underline{y} = y_1$.
- To find $\overline{y}$, we solve the system

$$F_1(\widetilde{x}_1 + \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n + \varepsilon_n \cdot \Delta_n, y_1, \ldots, y_m) = 0;$$

$$\ldots$$

$$F_m(\widetilde{x}_1 + \varepsilon_1 \cdot \Delta_1, \ldots, \widetilde{x}_n + \varepsilon_n \cdot \Delta_n, y_1, \ldots, y_m) = 0;$$

and take $\overline{y} = y_1$. The desired range is now $[\underline{y}, \overline{y}]$.

XXV. System of equations, monotonic case: computational complexity

Thus, in the monotonic case, we can find the range of y by solving only 3 systems:

- first, we solve the system corresponding to the original estimates $\widetilde{x}_i$ and to find the original values $\widetilde{y}_j$; after this, we compute X, Y, M, and ε_i;
- second, we solve the system corresponding to the values $x_i = \widetilde{x}_i + \varepsilon_i \cdot \Delta_i$ and thus, compute $\overline{y}$ as the corresponding y_1;
- finally, we solve the system corresponding to the values $x_i = \widetilde{x}_i - \varepsilon_i \cdot \Delta_i$ and thus, compute $\underline{y}$ as the corresponding y_1.

Solving 3 systems takes longer than solving a single system (as in the linearized case), but it is much faster than solving $n+3$ systems according to the general monotonicity case.

XXVI. Case of unknown functional variables

In some practical situations, instead of the values $x_1, \ldots, x_n$ known with interval uncertainty, we have a function $x(t)$ known with interval uncertainty, i.e., we know the approximate function $\widetilde{x}(t)$ and we know the bounds $\Delta(t)$ on the approximation error. In other words, we know that for each t, the (unknown) actual value $x(t)$ belongs to the interval $[\widetilde{x}(t) - \Delta(t), \widetilde{x}(t) + \Delta(t)]$.

The desired value y is related to the function y by a known functional dependence $y = F(x)$: e.g., we may have

$$y = y_0 + \int a(t) \cdot x(t)\, dt,$$

or

$$y = y_0 + \int a(t) \cdot x(t)\, dt + \int a(t,s) \cdot x(t) \cdot x(s)\, dt\, ds.$$

In general, we have $\widetilde{y} = F(\widetilde{x})$ and $y = F(x) = F(\widetilde{x} - \Delta x)$, where $\Delta x(t) \stackrel{\mathrm{def}}{=} \widetilde{x}(t) - x(t)$.

When the approximation errors $\Delta x(t)$ are small, we can expand the dependence $y = F(\widetilde{x} - \Delta x)$ or Δx in Taylor series and ignore quadratic and higher order terms in this dependence. In this case, we conclude that

$$\Delta y = \int \frac{\delta y}{\delta x(t)} \cdot \Delta x(t)\, dt,$$

where $\dfrac{\delta y}{\delta x(t)}$ is a functional derivative. For example,

- For a linear functional $y = y_0 + \int a(t) \cdot (\widetilde{x}(t) - \Delta x(t))\, dt$, we have $\dfrac{\delta y}{\delta x(t)} = a(t)$.
- For a quadratic functional

$$y = y_0 + \int a(t) \cdot (\widetilde{x}(t) - \Delta x(t))\, dt +$$

$$\int a(t,s) \cdot (\widetilde{x}(t) - \Delta x(t)) \cdot (\widetilde{x}(s) - \Delta x(s))\, dt\, ds =$$

$$\left(y_0 + \int a(t) \cdot \widetilde{x}(t)\, dt + \int a(t,s) \cdot \widetilde{x}(t) \cdot \widetilde{x}(s)\, dt\, ds\right) -$$

$$\int a(t) \cdot \Delta x(t))\, dt - 2 \int a(t,s) \cdot \widetilde{x}(s) \cdot \Delta x(t)\, dt\, ds,$$

we have $\dfrac{\delta y}{\delta x(t)} = a(t) + 2 \int a(t,s) \cdot \widetilde{x}(s)\, ds$.

In this case, the largest possible value Δ of Δt is attained when

- $\Delta x(t) = \Delta(t)$ for all t with $\dfrac{\delta y}{\delta x(t)} \geq 0$, and
- $\Delta x(t) = -\Delta(t)$ for all t with $\dfrac{\delta y}{\delta x(t)} < 0$.

In other words, the range of possible values of y is

$$[\widetilde{y} - \Delta, \widetilde{y} + \Delta], \text{ where } \Delta = \int \left|\frac{\delta y}{\delta x(t)}\right| \cdot \Delta(t)\, dt.$$

XXVII. Case of uncertain boundary

In some practical problems, e.g., in many problems related to structural integrity, the desired quantity y is related to the solution u of a linear partial differential equation $Lu = f$ with a known right-hand side f and boundary conditions such as $u_{|\partial\Omega} = 0$.

In numerical mathematics, it is usually assumed that we know the domain Ω and its boundary $\partial\Omega$. However, in practice, we often only know the boundary $\partial\Omega$ with uncertainty. How does this uncertainty affect the solution u to the corresponding partial differential equation?

In this paper, we will consider this effect only for the linearized case, when the uncertainty in $\partial\Omega$ is small, and we can safely ignore terms which are quadratic and of higher order in terms of this uncertainty.

Our main idea is to transform the boundary condition corresponding into the actual (unknown) domain Ω to the boundary condition corresponding to the nominal (approximate) domain $\widetilde{\Omega}$. Let $\widetilde{u}$ be the solution corresponding to $\widetilde{\Omega}$. We are interested in the difference $\Delta u = \widetilde{u} - u$. For each point $y \in \partial\widetilde{\Omega}$, let $\rho(y)$ denote the distance from this point to the nearest point on the boundary $\partial\Omega$. In these terms, the closeness between the sets means, e.g., that we have an upper bound $\Delta_\Omega(y)$ on this distance.

Let us describe this closest point. Let $\varepsilon(y) = 1$ if the nearest point is outside $\widetilde{\Omega}$ and $\varepsilon(y) = -1$ if the nearest point is inside $\widetilde{\Omega}$. Let $\vec{n}$ be a unit vector orthogonal to $\partial\widetilde{\Omega}$ and directed outside $\widetilde{\Omega}$. Then, in the linear approximation, the closest point is equal to $y + \varepsilon(y) \cdot \rho(y) \cdot \vec{n}$, and the value of u at this closest point is equal to $u(y + \varepsilon(y) \cdot \rho(y) \cdot \vec{n}) = u(y) + \varepsilon(y) \cdot \rho(y) \cdot \dfrac{du}{dn}$. By definition, $u(y) = \widetilde{u}(y) + \Delta u(y)$. For $y \in \delta\widetilde{\Omega}$, we have $\widetilde{u}(y) = 0$, and in the term proportional to $\dfrac{du}{dn}$, the part proportional to Δu can be ignored. As a result, we get the boundary condition $\Delta u_{|\delta\widetilde{\Omega}} = g$, where $g(y) \stackrel{\text{def}}{=} \varepsilon(y) \cdot \rho(y) \cdot \dfrac{d\widetilde{u}}{dn}(y)$.

Since L is a linear operator, from $Lu = f$ and $L\widetilde{u} = f$, we conclude that $L\Delta u = 0$. In general, the solution to a linear problem $Lv = 0$, $v_{|\delta\widetilde{\Omega}} = g$ is linear in g, i.e., $v(x) = \int a(x,y) \cdot g(y)\, dy$ for some function $a(x,y)$. Substituting the above expression for $g(y)$ into this formula, and taking into account that $\rho(y) \leq \Delta_\Omega(y)$, we conclude that for every x, we have

$$|\Delta u(x)| \leq \Delta_u(x) \stackrel{\text{def}}{=} \int |a(x,y)| \cdot \Delta_\Omega(y) \cdot \left|\frac{d\widetilde{u}}{dn}(y)\right| dy.$$

Comments. If we have uncertainty both in f and in Ω, then we can simply add the resulting bounds on $\Delta u(x)$.

A similar idea can be also applied to the case of non-linear equations L, the only difference is that we will need to linearize the operator L.

Acknowledgments

The authors are thankful to the anonymous referees for valuable discussions.

References

[1] Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2001.

[2] D. Dubois and H. Prade, Operations on fuzzy numbers, *International Journal of Systems Science*, 1978, Vol. 9, pp. 613–626.

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

[4] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, London, 2001.

[5] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*, Prentice Hall, New Jersey, 1995.

[6] V. Kreinovich, J. Beck, C. Ferregut, A. Sanchez, G. R. Keller, M. Averill, and S. A. Starks, "Monte-Carlo-type techniques for processing interval uncertainty, and their potential engineering applications", *Reliable Computing*, 2007, Vol. 13, No. 1, pp. 25–69.

[7] V. Kreinovich and S. Ferson, "A New Cauchy-Based Black-Box Technique for Uncertainty in Risk Analysis", *Reliability Engineering and Systems Safety*, 2004, Vol. 85, No. 1–3, pp. 267–279.

[8] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

[9] R. E. Moore and W. Lodwick, Interval Analysis and Fuzzy Set Theory, *Fuzzy Sets and Systems*, 2003, Vol. 135, No. 1, pp. 5–9.

[10] H. T. Nguyen and V. Kreinovich, Nested Intervals and Sets: Concepts, Relations to Fuzzy Sets, and Applications, In: R. B. Kearfott and V. Kreinovich, eds., *Applications of Interval Computations*, Kluwer, Dordrecht, 1996, pp. 245–290.

[11] H. T. Nguyen and E. A. Walker, *A First Course in Fuzzy Logic*, CRC Press, Boca Raton, Florida, 2006.

[12] A. Pownuk, "General Interval FEM Program Based on Sensitivity Analysis", *Proceedings of the NSF workshop on Reliable Engineering Computing*, Savannah, Georgia, February 20–22, 2008.

[13] M. V. Rama Rao, A. Pownuk, and I. Skalna, "Stress Analysis of a Singly Reinforced Concrete Beam with Uncertain Structural Parameters", *Proceedings of the NSF workshop on Reliable Engineering Computing*, Savannah, Georgia, February 20–22, 2008.

[14] S. Rabinovich, *Measurement Errors and Uncertainties: Theory and Practice*, American Institute of Physics, New York, NY, 2005.