XX IMEKO World Congress
Metrology for Green Growth
September 9−14, 2012, Busan, Republic of Korea

METROLOGICAL SELF-ASSURANCE OF DATA PROCESSING SOFTWARE

Kreinovich V. Ya.[1], *Reznik L. K.*[2], *Semenov K. K.*[3], *Solopchenko G. N.*[3]

[1] Department of Computer Science, University of Texas, El-Paso, USA, vladik@cs.utep.edu
[2] Department of Computer Science, Rochester Institute of Technology, New York, USA, lr@cs.rit.edu
[3] Department of Computer Science, Saint-Petersburg State Polytechnic University, Saint-Petersburg, Russia, semenov.k.k@gmail.com, g.n.solopchenko@mail.ru

Abstract: The metrological self-assurance for data processing software is discussed. The way to achieve this property for software is presented.

Keywords: metrological self-assurance, data processing, software metrological control.

1. INTRODUCTION

At present time, almost all measurement systems and units include digital processors and a specialized software usually employed in measurement data processing procedures like digital filtering, scaling, indirect measurement processing, etc. It is safe to state that no measurement result can be obtained without a calculation.

The important requirement of international metrological standards is that all software using in metrology and for measurements must be calibrated. One should determine metrological characteristics for each data processing result that one provides. We can analyze programs code and determine such characteristics manually. This way is very time-expensive. It might be contemplated for research purposes but is definitely not suitable for industrial applications. Much better is to perform this analysis automatically.

We will say that computer program for measurement data processing has metrological self-assurance if each calculation result has its interval characteristics of uncertainty and these characteristics are calculated automatically, and provided along with the result itself. This property is also called "self-validation", "self-verification" of computer programs in some publications. The main goal of this paper is to discuss ways to achieve the metrological self-assurance for data processing software.

This paper is organized as follows. After the Introduction, in section 2, mathematical prerequisites for metrological self-assurance problem formulization are presented. In section 3, the motivation for representing uncertainty as fuzzy variable is described. In section 4, the automatic differentiation of program code is chosen for derivatives estimation. And, in section 5, the application examples of researched approach are presented.

2. BACKGROUND INFORMATION

Any measuring procedure, which involves data processing, can be described as a function calculation $y = f(x_1, x_2, ..., x_n)$, where $x_1, x_2, ..., x_n$ are measurement results taken directly from the sensors and entered as inputs into computer procedures while y represents the final measurement result produced by this procedure. Measurement science postulates that all values of $x_1, x_2, ..., x_n$ are imprecise and we don't know their true values $\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n$. In practice pretty often all information that we know is that $|x_1 - \tilde{x}_1| \le \Delta x_1$, $|x_2 - \tilde{x}_2| \le \Delta x_2$, …, $|x_n - \tilde{x}_n| \le \Delta x_n$, where $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$ represent some measurement uncertainty characteristics, which can be obtained from technical data sheets and other documentation. We might or might not know the error distribution estimates inside intervals bounded by these inequalities, but we often can know the error type (systematic or random).

According to the international and many national standards all measurement results $x_1, x_2, ..., x_n$ should be represented not by simple numbers but as one of the possibilities by the interval set $I[x_1] = [x_1 - \Delta x_1, x_1 + \Delta x_1]$, $I[x_2] = [x_2 - \Delta x_2, x_2 + \Delta x_2]$, … $I[x_n] = [x_n - \Delta x_n, x_n + \Delta x_n]$, such, that $\tilde{x}_1 \in I[x_1]$, $\tilde{x}_2 \in I[x_2]$, ..., $\tilde{x}_n \in I[x_n]$.

Let $\tilde{y} = f(\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$ be the true value of data processing result. Our main goal is to determine such interval $I[y]$, that $\tilde{y} \in I[y]$, and to propose a computationally intensive fast algorithm, that would allow to determine this interval and implement it in the resource constrained environment that is quite typical in many industrial measurement devices. Then we will have the method to create metrological software with metrological self-assurance. There are several approaches to this problem solution.

1. Optimal interval analysis approach. We can try to get the precise boundaries $\underline{y}$ and $\overline{y}$ of $I[y]$, i. e. $\underline{y} = \inf_{\substack{x_1 \in I[x_1], ..., \\ x_n \in I[x_n]}} f(x_1, x_2, ..., x_n)$ and $\overline{y} = \sup_{\substack{x_1 \in I[x_1], ..., \\ x_n \in I[x_n]}} f(x_1, x_2, ..., x_n)$.

We should try to find two optimums to find the problem solution for concrete function *f*. The computational cost of this optimization algorithm could be very high for the metrological self-assurance: we will have to calculate values of *f* many times for different arguments values. So we will have no result for one calculation.

The stochastic variant of interval analysis is Monte-Carlo techniques and that's why the statistical modeling isn't the way to achieve metrological self-assurance.

2. We can try to get interval $I'[y]$ that $I[y] \subseteq I'[y]$. We can calculate such interval automatically using different interval arithmetics. The main idea of this approach is to replace calculations with numbers $x_1, x_2, ..., x_n$ by calculations with intervals $I[x_1], I[x_2], ..., I[x_n]$: $I'[y] = f(I[x_1], I[x_2], ..., I[x_n])$.

Let $k[f] = \frac{L[I'[y]]}{L[I[y]]}$, where $L[I[y]] = \overline{y} - \underline{y}$ is the width of interval $I[y]$ and $L[I'[y]]$ is the width of interval $I'[y]$. Then the following heuristic proportion holds: the faster and easier the interval arithmetic technique that we have chosen is, the bigger the value of $k[f]$ will be.

For metrology the value $k[f] \leq 1.5$ is acceptable because of rounding of final calculations result to one or maximum two significant digits. Bigger values aren't allowable. Unfortunately, when we use easy and fast interval arithmetic techniques for complex function *f* the value of $k[f]$ is bigger than 1.5 and when we use complex interval algebras techniques the computational cost of calculations is unacceptable. That's why we have to abandon this approach.

3. We can try to get interval $I''[y]$ that $I''[y] \approx I[y]$, i. e. the boundaries of $I''[y]$ are close to boundaries of $I[y]$, but it isn't guaranteed that $I[y] \subseteq I''[y]$. We have to round the final results and that's why it is acceptable if the boundaries of $I[y]$ and $I''[y]$ are close enough.

Requirements for solution become weaker along approach list and that's why algorithms for solution may become simpler. In addition to this only if we chose approach # 3 as prerequisite for further analysis we can take into account such suggestion. If all Δx_1, Δx_2, ..., Δx_n are small enough as it is usually in metrology then we can replace function $y = f(x_1, x_2, ..., x_n)$ with first-order terms from its Taylor series:

$$\Delta y \approx \sum_{i=1}^{n} \frac{\partial f(x_1, ..., x_n)}{\partial x_i} \cdot \Delta x_n . \qquad (1)$$

This linear model is much more simple object for analysis.

3. APPROACHES FOR UNCERTAINTY REPRESENTATION

There have been quite a few different approaches for uncertainty representation in published literature. Some of them better suit for systematic error processing, some – for random errors processing, but there are not many that can be universal enough for both types. One of them is the representation of uncertainty as a fuzzy variable.

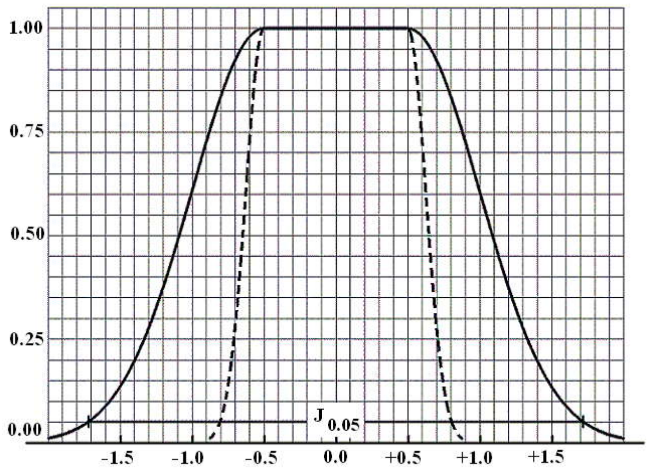
The simple test for different approaches is mean value calculating for the set of direct measurements results $x_1, x_2, ..., x_n$, errors of which are inside the known intervals $I[x_1], I[x_2], ..., I[x_n]$ with the probability $P \leq 1$.

As example we can examine any interval arithmetic. Let $\Delta x_1 = \Delta x_2 = \Delta x_n = \Delta x$ and $P < 1$. Then the error interval width Δm for mean value $m = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$ doesn't decrease corresponding to Δx as it should be. That's why interval arithmetic can sufficiently process only the systematic errors.

Because of using linear model (1) we can process different components of uncertainty separately: systematic errors can be processed independently and random errors can be processed independently. That's why we can easily combine two different approaches together into a complex one. For example we can represent random errors with probabilistic arithmetic [1] and represent systematic errors with classic interval arithmetic, process these uncertainty components through (1) and combine results into one final error interval characteristic. The only question is which approaches we should choose. Because of very weak requirements for interval $I''[y]$ we shouldn't choose complex methods, we can restrict with very simple estimations for Δy.

We can get such simple estimates using only one approach instead of two. We can replace external joining of two different methods by internal joining of systematic and random errors representation inside one well-known mathematical apparatus as it follows from Occam's razor.

The separate representation of systematic and random error components inside one mathematical object can be achieved by fuzzy interval introduced in [2]. Fuzzy interval membership function can be determined as curvilinear trapezium, which is presented on picture 1 by the solid line.



Picture 1. Fuzzy interval membership function.

The trapezium latter sides are halves of Gaussian curve $g(x) = \exp\left(-\frac{x^2}{2 \cdot \sigma^2}\right)$, whose parameter σ is proportional to the standard deviation of random error. Nested interval J_α is interval that contains not less than $P = 1-\alpha$ part of error, the upper side boundaries are the limit values of systematic error component.

From continuum of different rules for arithmetic operations for fuzzy variables we choose the following. Let r_1 and r_2 be fuzzy intervals, $S[r_1]$ and $S[r_2]$ be their supports, r_3 is a result of an arithmetical operation, $\otimes$ is symbol of arithmetical operation. Then membership function (MF) of number r_3 can be determined by well-known algebraic rule:

$$\mu_{r_3}(x_3) = \sup_{\substack{x_3 = x_1 \otimes x_2 \\ r_1 \in S[r_1],\, r_2 \in S[r_2]}} \left[\mu_{r_1}(x_1) \cdot \mu_{r_2}(x_2) \right] \qquad (2).$$

If we now return to the mean value calculating test then the mean of m = 16 fuzzy intervals with membership function as solid line on picture 1 will have MF as dashed line on picture. The MF trapezium upper side for the mean value will be the same with MF trapezium upper side of averaging fuzzy intervals as it should be for systematic error component and the projection of latter sides will reduce proportionally $\frac{1}{\sqrt{m}}$ as it should be for random error component.

The membership function of fuzzy interval can be described only with two numbers $\{\Delta, \sigma^2\}$: with the boundaries $[-\Delta, +\Delta]$ of upper side of trapezium and with the value of parameter σ of latter sides Gaussian curve. Then if uncertainty for all direct measurement results $x_1, x_2, ..., x_n$ will be represented by fuzzy intervals $\{\Delta_1, \sigma_1^2\}$, $\{\Delta_2, \sigma_2^2\}$, …, $\{\Delta_n, \sigma_n^2\}$ then it can be shown that uncertainty for y from formula (1) will be represented by fuzzy interval:

$$\left\{ \sum_{i=1}^{n} \left| \frac{\partial f(x_1, ..., x_n)}{\partial x_i} \right| \cdot \Delta_i, \sum_{i=1}^{n} \left(\frac{\partial f(x_1, ..., x_n)}{\partial x_i} \cdot \sigma_i \right)^2 \right\}.$$

Its components are well-known formulas for uncertainty systematic and random components estimates, which are used in classic metrology. The separate processing of systematic and random error through formula (1) can be easily organized with fuzzy interval technique and can be calculated fast.

4. DERIVATIVES ESTIMATES

Statement (1) contains the derivatives $\frac{\partial f(x_1, ..., x_n)}{\partial x_i}$ of calculating function f. We should estimate them automatically, fast and accurate. The best technique for this is well-known automatic differentiation of functions, which are presented by their program code [3]. This approach is based on Clifford algebra of dual numbers and its main idea is to replace calculations with real numbers $x_1, x_2, ..., x_n$ by calculations with dual numbers $D_1, D_2, ..., D_n$. Any dual number $D_i = (a_i, b_i)$ for i = 1, 2, … n, where $a_i = a_i(t)$ is the real part of dual number and $b_i = a_i'(t)$ is infinitesimal part.

All arithmetic operations for the dual numbers are determined in such manner:

$$D_1 \pm D_2 = (a_1, b_1) \pm (a_2, b_2) = (a_1 \pm a_2, b_1 \pm b_2),$$
$$D_1 \cdot D_2 = (a_1, b_1) \cdot (a_2, b_2) = (a_1 \cdot a_2, a_1 \cdot b_2 + a_2 \cdot b_1),$$
$$\frac{D_1}{D_2} = \frac{(a_1, b_1)}{(a_2, b_2)} = \left(\frac{a_1}{a_2}, \frac{b_1 \cdot a_2 - b_2 \cdot a_1}{a_2^2} \right) \text{ and}$$

in addition the functional transformation

$$f(D) = (f(a), f'(a)).$$

The automatic differentiation realization doesn't request calculating finite differences. It produces accurate derivatives for each calculation fast. So if we join in one program uncertainty representation as fuzzy intervals and automatic differentiation then we will be able to achieve metrological self-assurance for it without serious modification in its code. The necessary code changing contains only connecting the program library, where a presented approach is realized, and replacing data type for input and intermediate variables from floating-points numbers to new type, which represents data and its uncertainty characteristics. An example for C++ language head file is given in Table 1. Code modifications are typed in bold, the meaningful program part has no changes.

Table 1. Code modifications example

Unmodified source code	Modified source code

#include <math.h>
#include <stdlib.h>
 	
void func (void)
{
double *x*=2, *z*;
z = *x* * *x* * exp(*x*)+log(*x*);
printf("%*f*", *z*);
}
	#include <math.h>
	#include <stdlib.h>
	#include "edouble.h"
	
	void func (void)
	{
	edouble *x*(2, 1);
	edouble *z*;
	z = *x** *x**exp(*x*)+log(*x*);
	printf("%*f*", *z*);
	}

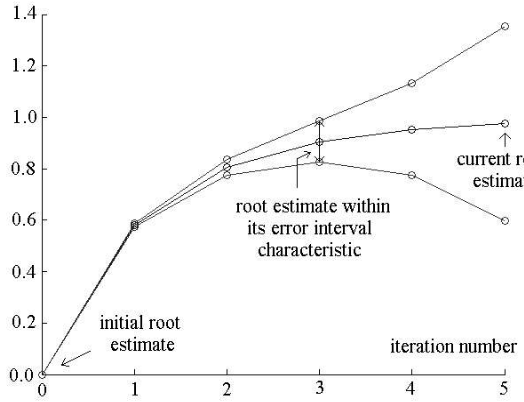
5. SOME EXAMPLES

The presented approach has been programmed and tested on some applications. Two of them are described in this section.

1. The presented approach has been used for finding root estimates for transcendental equation $\exp(k \cdot x) = \lambda \cdot x$, where k and λ are measured values and x is a variable. Root has been estimated by classic iterative Newton method. Let measured value of λ is 2.718 and measured value of k is 1.000. Then the equation has the only root that value is x = 1.000, which is corresponding to the point in which straight line $y(x) = \lambda \cdot x$ touches the exponent $y(x) = \exp(k \cdot x)$. Such conditions cause the ill-defined root-estimation problem. If we change the value of parameters k and λ with a very little deviation, the equation may have no roots.

Let values of k and λ were measured with the only systematic error, which values are not greater than 0.01. Let the initial root estimation for Newton method be x = 0.0. We modify the program for root estimating with a proposed approach and create its analogy with a metrological self-assurance property. Then on the different iterations we will have root estimations and their error interval characteristics, which are illustrated by picture 2.

Picture 2 shows that iterations converge to the equation root $x = 1.000$ but error interval increases with new iterations. It looks reasonable to determine the iteration process stop condition as follows: if the error interval width increasing is for the next iteration larger than the value of root estimation clarification for the current iteration then process should be stopped. So we should finish root estimation on the iteration number 4. The result root estimation is [0.775, 1.132]. This interval contains the true value $x = 1.000$.



Picture 2. Root estimation iterations.

2. As the second example we have chosen the program that performs calculations corresponding to standard algorithm [4], which is described by the following functions:

$$U_\rho = 0.125 \cdot \left(a_1 \cdot \sum_{i=1}^{4} U_i + a_2 \cdot \sum_{i=5}^{8} U_i \right), \ \rho = \frac{\pi \cdot d \cdot U_\rho}{I_1 \cdot \ln 2},$$

$$U_y = 0.125 \cdot \sum_{i=9}^{16} U_i, \ R = \frac{10^4 \cdot d \cdot U_y}{I_2 \cdot B}, \ \eta = \frac{R}{\rho}.$$

Values a_1 and a_2 are the roots of equation $2 \cdot \cosh\left(\frac{A-1}{A+1} \cdot \frac{\ln 2}{a} \right) = \exp\left(\frac{\ln 2}{a} \right)$, which is solving for the variable a when parameter A values are equal to $A_1 = \frac{U_1 + U_2}{U_3 + U_4}$ and $A_2 = \frac{U_5 + U_6}{U_7 + U_8}$.

The program purpose is to calculate values of germanium single-crystal parameters: electrical resistivity ρ, Hall mobility η and Hall coefficient R. The initial data for calculations are direct measurements results for crystal plate thickness d, electric currents I_1 and I_2 through crystal, the induction B of magnetic field, in which the crystal is, and Hall potential differences U_1, …, U_{16} through 16 different directions along the crystal plate.

This program is taken from the real metrological practice. It is employed in the high-impedance germanium crystals control system. There are no standard germanium crystals, which parameters have a priori known values and which can be used for a system error estimation. That is why we need to use numeric methods for metrological control of this complex measuring part and there is real necessity in metrological self-assurance of this measured data processing software.

We tested several models of uncertainty. In the first version all initial data uncertainties had only systematic error component and no random component. The limit values for relative errors were $\delta_B = 3\,\%$, $\delta_{I_1} = \delta_{I_2} = 1\,\%$, $\delta_d = 1\,\%$, $\delta_{U_1} = \delta_{U_2} = ... = \delta_{U_{16}} = 1\,\%$. Other versions differ from first: all initial data have the random error as in the first version and has also random component with increasing value of a standard deviation limit value σ from version to version. These versions can be characterized by ratio $t = \frac{\sigma_B}{B \cdot \delta_B} = \frac{\sigma_{I_1}}{I_1 \cdot \delta_{I_1}} = \ldots = \frac{\sigma_{U_{16}}}{U_{16} \cdot \delta_{U_{16}}}$ which is equal for all initial data.

We calculated limit values of uncertainty for parameter η and compared them with Monte-Carlo modeling results. Results are identical as it is illustrated with table 2.

Table 2. Modeling results for parameter η.

Version	Ratio t	Monte-Carlo modeling	Presented approach
1	0.0	7.0 %	7.0 %
2	0.2	8.1 %	8.1 %
3	0.4	9.2 %	9.2 %
4	0.6	10.4 %	10.4 %

6. COCNLUSIONS

High efficient modern computers and their widespread application in metrology allow us to try to develop software, which performs its own metrological analysis automatically. The main idea of software metrological self-assurance is to make this analysis for any calculation and at the same time with this calculation inside the program without human participation.

The combined use of automatic differentiation and initial data uncertainty representation as fuzzy intervals provides metrological self-assurance even for discontinuous functions calculations. At the same time it allows to take into account not only the systematic uncertainty component but also the random component. The provided examples illustrate the approach described.

7. REFERENCES

[1] R. C. Williamson, T. Downs. Probabilistic Arithmetic. I. Numerical Methods for Calculating Convolutions and Dependency Bounds, in International Journal of Approximate Reasoning. Vol. 4, pp. 89–158, 1990.

[2] L. K. Reznik, W. C. Jonson, G. N. Solopchenko. Fuzzy interval as a Basis for Measurement Theory, in Proc. NASA Conf. NAFIPS'94. San-Antonio, Texas. Pp. 405–406, 1994.

[3] A. Griewank. On Automatic Differentiation, in: Mathematical Programming: Recent Developments and Applications. Kluwer Academic Publisher, pp.83–108, 1989.

[4] USSR Standard GOST 16153-80. Single-crystalloid germanium. Technical conditions. (In Russian).