Vol. **120** (2011) *ACTA PHYSICA POLONICA A* No. 4

Optical and Acoustical Methods in Science and Technology

Uncertainty Evaluation of Measurement Data Processing Algorithm Based on Its Matrix Form

K. Konopka* and T. Topór-Kamiński

Institute of Measurement Science, Electronics and Control, Silesian University of Technology

Akademicka 2, 44-100 Gliwice, Poland

Data processing algorithms are important parts of modern measurement systems. These algorithms are often delivered to the user as complex program and their numerical structure is not known. Therefore also influence of algorithm on processed data accuracy is not known. One of the methods to evaluate uncertainty propagation through algorithm is based on matrix form of algorithm. Coefficient matrix of algorithm represents its numerical operations and it is a basis to algorithm accuracy evaluation. The paper presents a method how to identify this coefficient matrix when algebraic form of the algorithm is known or is difficult to use. Identified matrix form of algorithm is then used to estimate uncertainty propagation through exemplary algorithms. Results are compared with experiments.

PACS: 52.80.−s, 77.22.Jp, 84.70.+p

1. Introduction

Measurement systems and instruments use many different algorithms for calculations on measured quantity values. Algorithms process measurement data in similar way other converters in measurement chain do. Therefore, as other converters, algorithms influence measurement uncertainty. They add their own uncertainties to final measurand uncertainty and as they process measurement data they change the part of final uncertainty caused by elements before algorithms. In the second case it is called uncertainty propagation.

One of the methods to evaluate uncertainty propagation through algorithm is based on matrix form of algorithm [1]. Matrix form of algorithm facilitates analysis of its metrological properties. Knowledge of algorithm coefficient matrix helps in determining how this algorithm transfers errors from input to the output. The coefficients of the matrix can be calculated using algebraic form of the algorithm if it is known. However in many cases algorithms are implemented in different software environments and their coefficients calculations are not overt. The paper describes an identification method of these coefficients, where the algorithm can be described just as a black box with inputs and outputs. This method enables quick, current calculations of algorithm coefficients in particular measurement situation and subsequently uses these coefficients to calculate current uncertainty of measurement values.

2. Matrix form of algorithm

Generally, it can be assumed that the algorithm converts quantity $x(t)$, continuous in t domain, to another quantity $X(\omega)$, continuous in ω domain. In practice, algorithm operates on series of discrete samples of input quantity. The output values of algorithm have also discrete form. Number of samples the algorithm operates on is limited (input window). Also output quantity has limited representation (output window). Considerations in the paper are limited to rectangular input window. It means that algorithm input data consist of input quantity samples $x(0), x(1), \ldots, x(K-1)$, where K means the number of input window samples. On the output of the algorithm one gets N element algorithm output data series $\{X(n)\} := \{X(0), X(1), \ldots, X(N-1)\}$ which can be interpreted as the discrete representation of algorithm output value X, continuous in ω domain.

In the situation presented one can describe algorithm data processing as a matrix equation [1, 2]:

$$\begin{bmatrix} X(0) \\ X(1) \\ . \\ . \\ X(N-1) \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,K-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,K-1} \\ & & . & \\ & & . & \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,K-1} \end{bmatrix} \times \begin{bmatrix} x(0) \\ x(1) \\ . \\ . \\ x(K-1) \end{bmatrix}, \tag{1}$$

where $a_{0,0}, a_{0,K-1}, a_{N-1,K-1}$ are the algorithm coefficients. Algorithm input and output data series are presented in (1) as vectors. When one denotes these vectors respectively as $\boldsymbol{x}$ and $\boldsymbol{X}$, the data processing algorithm represented by Eq. (1) can be written as

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{x}. \tag{2}$$

* corresponding author; e-mail: krzysztof.konopka@polsl.pl

2.1. Complex output data

Output data of some algorithms, for example density functional theory (DFT), are presented in complex form. In such case data processing can be described by two algorithms, where one determines real values and the second imaginary ones. Such algorithms can be presented as two matrix equations

$$\boldsymbol{X}_{\mathrm{Re}} = \boldsymbol{A}_{\mathrm{Re}}\boldsymbol{x}, \tag{3}$$

$$\boldsymbol{X}_{\mathrm{Im}} = \boldsymbol{A}_{\mathrm{Im}}\boldsymbol{x}. \tag{4}$$

Output data of these algorithms form complex number series, which can be presented as vector

$$\boldsymbol{X} = \boldsymbol{X}_{\mathrm{Re}} + \mathrm{j}\,\boldsymbol{X}_{\mathrm{Im}}\,. \tag{5}$$

2.2. Single and multipoint algorithms

One can indicate a group of algorithms which produce a single output value — for example filtration algorithms, average or rms calculating algorithms etc. This kind of algorithm is called *single point* algorithm.

In another case, when one run of algorithm produces a set of values, such algorithm is called *multipoint* algorithm [1, 2].

Multipoint algorithm can be presented as a set of parallel single point algorithms which operate on the same input values series [1]. Every single point algorithm is represented by corresponding coefficient vector ${}^{0}\boldsymbol{A}, {}^{1}\boldsymbol{A} \ldots {}^{N-1}\boldsymbol{A}$ which is a single row of coefficient matrix $\boldsymbol{A}$ in (1). Execution of every single point algorithm generates one value, which can be described for n-th row of coefficient matrix $\boldsymbol{A}$ as

$$\begin{aligned} X(n) &= [a_{n,0}a_{n,1}\ldots a_{n,K-1}][x(0)x(1)\ldots x(K-1)]^{\mathrm{T}} \\ &= {}^{n}\boldsymbol{A}\boldsymbol{x}^{\mathrm{T}}, \end{aligned} \tag{6}$$

where ${}^{n}\boldsymbol{A} = [a_{n,0}a_{n,1}\ldots a_{n,K-1}]$ is a coefficient vector of single point algorithm represented by n-th row of matrix $\boldsymbol{A}$, $a_{n,0}a_{n,1}\ldots a_{n,K-1}$ are constant coefficients of single point algorithm, $\boldsymbol{x}^{\mathrm{T}} = [x(0)x(1)\ldots x(K-1)]^{\mathrm{T}}$ is the algorithm input values vector.

Dependence (6) can be also written in other form

$$\begin{aligned} X(n) &= a_{n,0}x(0) + a_{n,1}x(1) + \ldots + a_{n,K-1}x(K-1) \\ &= \sum_{k=0}^{K-1} a_{n,k}x(k)\,. \end{aligned} \tag{7}$$

Dependence (7) is a general single point algorithm processing equation, which form the base for further metrological analysis, where it is assumed that multipoint algorithm is considered as set of independent single point algorithms [1]. It means that output data from multipoint algorithm is considered as a set of independent numbers.

Considering multipoint algorithm as a set of single point ones considerably simplifies its metrological analysis with no influence on the generality of considerations. That is because usually analysis of single point algorithm gives information about structure and many properties of the whole multipoint algorithm [1, 2].

In further considerations it is assumed that algorithm input values form a series of instantaneous measurement results of changing in time quantity. Moreover, it is assumed that algorithm coefficients values are constant and independent of both input values series $\{x(k)\}$ and output values series $\{X(n)\}$.

3. Identification of algorithm coefficients

As mentioned in introduction, to determine metrological properties of algorithm one needs to identify its coefficient matrix values. This identification can be achieved by giving to the algorithm input appropriate test series. Algorithms operate completely on digital values therefore it is easy to generate any test series needed.

In general case, algorithm processes series of N input values into series of N output values. Considering matrix form of algorithm (1) one can notice that if the input vector has form of unit impulse $[x(0)x(1)\ldots x(N-1)]^{\mathrm{T}} = [1\;0\;0\ldots 0]^{\mathrm{T}}$, then output vector will have values equal to values of coefficients from the first column of matrix $\boldsymbol{A}$. If one shifts “1” in input vector to the successive positions, one gets successive columns of coefficient matrix $\boldsymbol{A}$. Generally, test series for λ column determining has a form

$$v_{\lambda} = \begin{cases} 1 & \text{for} \quad k = \lambda, \\ 0 & \text{for} \quad k \neq \lambda, \end{cases} \tag{8}$$

where $\lambda = 0, 1, \ldots, N-1$ is a number of test series and $k = 0, 1, \ldots, N-1$ is a number of term in series.

Repeating the operation N times one obtains the whole coefficient matrix $\boldsymbol{A}$.

As a first example of using coefficient identification method fast Fourier transform (FFT) algorithm is chosen. FFT is a generally known transformation algorithm. It transforms data from time domain to frequency domain. General equation of discrete Fourier transform has a form

$$X(n) = \sum_{k=0}^{N-1} x(k)\,\mathrm{e}^{-\mathrm{j}\,2\pi nk/N}, \quad \text{for} \quad k = 0, 1, \ldots, N-1\,. \tag{9}$$

Values of the matrix $\boldsymbol{A}$ elements of FFT algorithm change with the length of the input vector. But identification method is easy to implement as a subroutine which calculates coefficients values currently as required. So in measurement system control program one can insert a module, which calculates algorithm matrix coefficients and run it when measurement conditions change. For example matrix $\boldsymbol{A}$ for $N = 10$ is

$$\boldsymbol{A}=\begin{bmatrix}
0.100+0.000\,i & 0.100+0.000\,i & 0.100+0.000\,i & 0.100+0.000\,i & 0.100+0.000\,i & 0.100+0.000\,i & 0.100+0.000\,i & 0.100+0.000\,i & 0.100+0.000\,i & 0.100+0.000\,i \\
0.100+0.000\,i & 0.081-0.059\,i & 0.031-0.095\,i & -0.031-0.095\,i & -0.081-0.059\,i & -0.100+0.000\,i & -0.081+0.059\,i & -0.031+0.095\,i & 0.031+0.095\,i & 0.081+0.059\,i \\
0.100+0.000\,i & 0.031-0.095\,i & -0.081-0.059\,i & -0.081+0.059\,i & 0.031+0.095\,i & 0.100+0.000\,i & 0.031-0.095\,i & -0.081-0.059\,i & -0.081+0.059\,i & 0.031+0.095\,i \\
0.100+0.000\,i & -0.031-0.095\,i & -0.081+0.059\,i & 0.081+0.059\,i & 0.031-0.095\,i & -0.100+0.000\,i & 0.031+0.095\,i & 0.081-0.059\,i & -0.081-0.059\,i & -0.031+0.095\,i \\
0.100+0.000\,i & -0.081-0.059\,i & 0.031+0.095\,i & 0.031-0.095\,i & -0.081+0.059\,i & 0.100+0.000\,i & -0.081-0.059\,i & 0.031+0.095\,i & 0.031-0.095\,i & -0.081+0.059\,i \\
0.100+0.000\,i & -0.100+0.000\,i & 0.100+0.000\,i & -0.100+0.000\,i & 0.100+0.000\,i & -0.100+0.000\,i & 0.100+0.000\,i & -0.100+0.000\,i & 0.100+0.000\,i & -0.100+0.000\,i \\
0.100+0.000\,i & -0.081+0.059\,i & 0.031-0.095\,i & 0.031+0.095\,i & -0.081-0.059\,i & 0.100+0.000\,i & -0.081+0.059\,i & 0.031-0.095\,i & 0.031+0.095\,i & -0.081-0.059\,i \\
0.100+0.000\,i & -0.031+0.095\,i & -0.081-0.059\,i & 0.081-0.059\,i & 0.031+0.095\,i & -0.100+0.000\,i & 0.031-0.095\,i & 0.081+0.059\,i & -0.081+0.059\,i & -0.031-0.095\,i \\
0.100+0.000\,i & 0.031+0.095\,i & -0.081+0.059\,i & -0.081-0.059\,i & 0.031-0.095\,i & 0.100+0.000\,i & 0.031+0.095\,i & -0.081+0.059\,i & -0.081-0.059\,i & 0.031-0.095\,i \\
0.100+0.000\,i & 0.081+0.059\,i & 0.031+0.095\,i & -0.031+0.095\,i & -0.081+0.059\,i & -0.100+0.000\,i & -0.081-0.059\,i & -0.031-0.095\,i & 0.031-0.095\,i & 0.081-0.059\,i
\end{bmatrix}. \tag{10}$$

Of course N value is selected for presentation purposes, in real applications it would be much higher.

If one considered for the second example finite impulse response filter algorithms, identification procedure would be less complex. Far infrared (FIR) filter is a singlepoint algorithm, so its impulse response will reveal algorithm coefficients. For example the impulse response of a low pass filter with 11 weights, a sampling rate of 10000 Hz, and cut off frequency at 1000 Hz is presented in Fig. 1. Therefore matrix $\boldsymbol{A}$ being now rather vector $\boldsymbol{A}$ is

$$\boldsymbol{A}=[9.13781\times10^{-25}\quad 2.78515\times10^{-5}\quad 0.00332\quad 0.05381\quad 0.24454\quad 0.39661\quad 0.24454\quad 0.05381\quad 0.00332\quad 2.78515\times10^{-5}\quad 9.13781\times10^{-25}]. \tag{11}$$

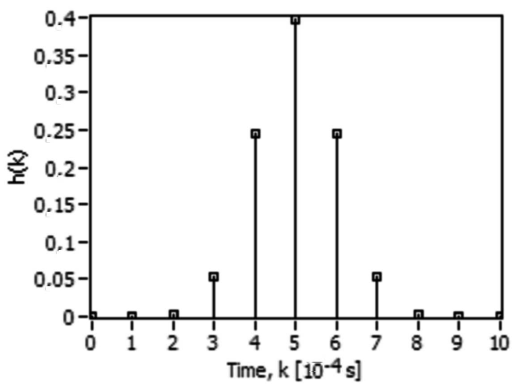

Fig. 1. The impulse response of a low pass filter with 11 weights, a sampling rate of 10000 Hz, and cut off frequency at 1000 Hz.

4. Uncertainty evaluation based on algorithm coefficients

Specific data processing in algorithms causes that it is convenient to consider uncertainty as a parameter of error values set, even if this error is only theoretical and impossible to determine, but it facilitates processing analysis and later transition to description by uncertainties [1, 3, 4]. Knowledge about error propagation enables to determine algorithm output uncertainty basing on input uncertainties and on input errors distribution shapes.

Therefore algorithm uncertainty model is based on error model presented in Fig. 2. In this model algorithm transfers errors from the input to the output with appropriate coefficient and also introduces its own errors.

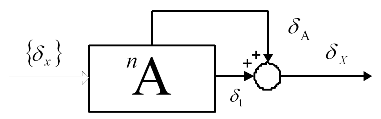

Fig. 2. General algorithm error model. $\{\delta_x\}$ is an algorithm input data error sequence, δ_t is an error resulting from data errors transfer from input to output, δ_A is an error added by algorithm.

Errors can be divided into three categories regarding the way they are processed by algorithms — static, dynamic and random errors [1, 3–5]. Uncertainty can be considered as a measure characterizing error set [1, 4, 6], therefore also uncertainties can be categorized as static, dynamic and random.

This paper considers propagation of random errors as they are typically the most important source of uncertainty. As random errors are meant errors which can be considered in probabilistic categories. Basing on [1], the dependence between random error variances on the input and the output of the algorithm is

$$\sigma_X^2 = A^2\sigma_x^2, \tag{12}$$

where A is a square root of the sum of squared singlepoint algorithm coefficients

$$A=\sqrt{\sum_{k=0}^{K-1} a_k^2}\,. \tag{13}$$

Therefore when data contains random errors, uncertainty propagation can be calculated basing on appropriate variances.

When random errors are concerned one meets usually one of two situations. First — when quantization errors dominate and then input error distribution shape can be approximated by uniform distribution [7]. Second one is when noise dominates and error distribution shape is normal.

Assuming that algorithm has more than three coefficients and the coefficient values do not differ from each other excessively, one can assume that probability density function of algorithm output error $g(\delta_X)$ has approximately normal distribution, no matter if input errors have normal or uniform distribution. This results from central limit theorem [8]. Uncertainty with confidence level $\alpha = 0.95$ calculated using definition [1, 6] for normal distribution, and presented as multiplicity of standard deviation, is for the output quantity [8]:

$$U(X) = 1.96\sigma_X \,. \tag{14}$$

For input errors which have normal distribution this uncertainty is

$$U_r(x) = 1.96\sigma_x \,.$$

Taking into consideration expression (12), Eq. (14) can be noted as

$$U(X) = 1.96A\sigma_x \,. \tag{15}$$

Uncertainty with confidence level $\alpha = 0.95$ calculated for random error which probability density function is uniform is [8]

$$U_q(x) = 1.65\sigma_x \,. \tag{16}$$

When quantization errors dominate one can determine proportion of output uncertainty (15) and input uncertainty (16) and obtain coefficient k_q which specifies propagation of uncertainty caused by quantization error from algorithm input to the output

$$k_q = \frac{U(X)}{U_q(x)} = \frac{1.96A\sigma_x}{1.65\sigma_x} = 1.19A \,. \tag{17}$$

Therefore coefficient k_q value is directly proportional to value A, which is an important parameter of every algorithm. Basing on A one can specify quantitatively uncertainty propagation through algorithm.

In case when noise dominates, propagation coefficient can be written as

$$k_r = \frac{U(X)}{U_r(x)} = \frac{1.96A\sigma_x}{1.96\sigma_x} = A \,. \tag{18}$$

Presented considerations show that when $\boldsymbol{A}$ matrix coefficients are identified and the parameters of input uncertainty sources are known, then output uncertainty can be calculated with little cost.

5. Examples of uncertainty calculations

Example I: Input values of algorithm contain quantization error of 12-bit AD converter. Converter input voltage range is $U_{\mathrm{IN}} = -1, \ldots + 1$ V, so quantum value is q $= \frac{1-(-1)}{2^{12}} = 4.883 \times 10^{-4}$ V. Uncertainty on the input of algorithm is $U_q(x) = 0.95 \times \frac{1}{2} \times 4.883 \times 10^{-4}$ V $= 2.32 \times 10^{-4}$ V, and the standard deviation $\sigma_{q,x} = U_q(x)/1.65 = 1.41 \times 10^{-4}$ V.

Considering FFT algorithm described in Sect. 3 one must define, which output value is actual measurand to which uncertainty is going to be calculated. For the further calculations it is assumed that considered measurement result is the real part of the first element of the output vector $X(1)_{\mathrm{Re}}$.

Assuming that input window of FFT algorithm has 1024 values one can identify coefficient matrix $\boldsymbol{A}$ as described in Sect. 3 and then calculate value $A_{\mathrm{Re},1}$ for its first row using (13). In presented example it is $A_{\mathrm{Re},1} = 0.0220971$. Finally one can evaluate uncertainty of the first element of the output vector when quantization error dominates as follows:

$$U_q(X(1)_{\mathrm{Re}}) = 1.96A_{\mathrm{Re},1}\sigma_{q,x} = 6.1 \times 10^{-6} \text{ V}. \tag{19}$$

The same calculations can be made for other algorithm output values.

Example II: Errors in input values of algorithm are dominated by Gaussian noise. Standard deviation of the noise is $\sigma_{r,x} = 5 \times 10^{-3}$ V. Input uncertainty is then $U_r(x) = 1.96 \times 5 \times 10^{-3}$ V $= 9.8 \times 10^{-3}$ V. Uncertainty of the first element of FFT output vector is calculated as

$$U_r(X(1)_{\mathrm{Re}}) = 1.96A_{\mathrm{Re},1}\sigma_{r,x} = 2.17 \times 10^{-4} \text{ V}. \tag{20}$$

Example III: Measurement data is processed by low pass FIR filter algorithm with 11 weights, a sampling rate of 10000 Hz, and cut off frequency at 1000 Hz. Vector $\boldsymbol{A}$ is presented in dependence (11), so value A can be calculated using (13) as $A = 0.531703$.

Assuming that input data contains quantization error the same as in example I one can calculate estimation of output uncertainty caused by this error

$$U_q(X_{\mathrm{FIR}}) = 1.96A\sigma_{q,x} = 1.47 \times 10^{-4} \text{ V}. \tag{21}$$

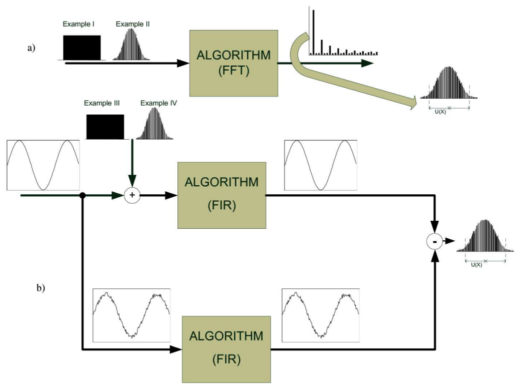

Fig. 3. Scheme of experimental determination of algorithm output uncertainty. (a) Pure noise on algorithm input. Output contains both transferred input errors and errors inserted by algorithm. (b) Input values contain random errors. Output values are compared with results without random errors. Histogram of their difference is a base for uncertainty evaluation. Algorithm own errors are eliminated. When algorithm own errors are small, both procedures give the same results.

Example IV: The same FIR filter as in example III and Gaussian noise as in example II. Output uncertainty caused by this noise is

$$U_r(X_{\mathrm{FIR}}) = 1.96A\sigma_{r,x} = 5.22 \times 10^{-3} \text{ V}. \tag{22}$$

Results (19)–(22) were checked experimentally. Experiments can be done in two ways presented in Fig. 3.

Procedure presented in Fig. 3b replicates normal functioning of measurement data processing. Standard signal is processed by algorithm twice — with added random errors having known distribution, and without these errors.

Difference of output values is an output error. Basing on set of these errors standard deviation and uncertainty can be calculated.

In second procedure presented in Fig. 3a the input quantity of algorithm is pure noise of desired distribution (uniform or normal). Then basing on the output value histogram one can determine the standard deviation and uncertainty.

The choice between procedures depends on the form of processing algorithm which is investigated, also on possibility to obtain pure signal without noise, or possibility to obtain pure noise.

Experiments presented in the paper were made using the second procedure (Fig. 3a) for examples I and II and the first one (Fig. 3b) for examples III and IV.

Results for system parameters, the same as in examples I–IV, are presented in Table. It can be observed that despite uncertainty calculations using matrix A are approximate [1, 9], they match with experimental values.

TABLE

Results of experimental uncertainty evaluation compared with calculations based on algorithm coefficients.

Experiment parameters	Standard deviation of output quantity σ_X from experiment	Output uncertainty $U(X)$ from experiment	Output uncertainty $U(X)$ from calculations
as in example I	$\sigma_{q,X_{\mathrm{Re}}} = 3.11 \times 10^{-6}$	$U_q(X(1)_{\mathrm{Re}}) = 6.10 \times 10^{-6}$	$U_q(X(1)_{\mathrm{Re}}) = 6.10 \times 10^{-6}$
as in example II	$\sigma_{r,X_{\mathrm{Re}}} = 1.11 \times 10^{-4}$	$U_r(X(1)_{\mathrm{Re}}) = 2.17 \times 10^{-4}$	$U_r(X(1)_{\mathrm{Re}}) = 2.17 \times 10^{-4}$
as in example III	$\sigma_{q,X_{\mathrm{FIR}}} = 7.49 \times 10^{-5}$	$U_q(X_{\mathrm{FIR}}) = 1.469 \times 10^{-4}$	$U_q(X_{\mathrm{FIR}}) = 1.47 \times 10^{-4}$
as in example IV	$\sigma_{r,X_{\mathrm{FIR}}} = 2.65 \times 10^{-3}$	$U_r(X_{\mathrm{FIR}}) = 5.19 \times 10^{-3}$	$U_r(X_{\mathrm{FIR}}) = 5.22 \times 10^{-3}$

6. Conclusions

The paper presents one of possible ways to determine measurement uncertainty when processing algorithms are used in measurement system. It is based on algorithm matrix form. It is necessary in described procedure to find out algorithm coefficient matrix. The identification method of data processing algorithm coefficient matrix is proposed. This method enables easy determination of matrix coefficients, even when algorithm structure is not known. Knowledge about the length of input and output vector is sufficient. Presented method can be implemented as a subroutine which automatically calculates current values of coefficient matrix $\boldsymbol{A}$ depending on changing operating conditions.

Basing on determined matrix coefficients one can determine propagation of errors of different kinds through the algorithm. Further analysis determines also uncertainty propagation through algorithm. Nowadays measurement data processing algorithms are generally used, therefore analysis of uncertainty propagation through algorithms is important, although often omitted. When algorithms influence on error budget and on uncertainty budget is omitted, it can cause false estimation of measurement accuracy, as different kinds of errors and uncertainties can be considerably amplified or attenuated. Presented in the paper uncertainty evaluation method can considerably facilitate uncertainty determining process and the error estimation. The method is approximate but experiments show that results match with good accuracy. One should also take into account that experimental uncertainty evaluation is not always possible. On the other hand, using presented approximate method enables to calculate uncertainty even continuously, parallel to measurements.

References

[1] J. Jakubiec, *Application of Reductive Interval Arithmetic to Uncertainty Evaluation of Measurement Data Processing Algorithms*, Monograph, SUT, Gliwice 2002.

[2] J. Jakubiec, *Elektryka* **169**, 7 (2000) (in Polish).

[3] J. Jakubiec, K. Konopka, in: *IEEE IMTC 2003 — Instrumentation and Measurement Technology Conf. Vail*, IEEE, Piscataway 2003, p. 115.

[4] J. Jakubiec, K. Konopka, in: *IMEKO TC7 Symp. "Measurement Science of the Information Era"*, Cracow University of Technology, Kraków 2002, p. 76.

[5] J. Jakubiec, *Pomiary Automatyka Kontrola PAK* **6**, 08 (2006) (in Polish).

[6] J. Jakubiec, *Pomiary Automatyka Kontrola PAK* **2**, 04 (2007) (in Polish).

[7] J. Jakubiec, *Pomiary Automatyka Kontrola PAK* **01**, 08 (2008).

[8] *Guide to the Expression of Uncertainty in Measurement*, ISO/IEC/OIML/BIPM, 1992 (1995).

[9] K. Konopka, in: *Instrumentation and Measurement Technology Conf. — IMTC 2007,* Warsaw, IEEE Piscataway 2007, p. 100.