A TEST OF LINEARITY USING COVERING ARRAYS FOR EVALUATING UNCERTAINTY IN MEASUREMENT

RÜDIGER KESSEL, RAGHU KACKER

NIST, Mathematical and Computational Sciences Division,
100 Bureau Drive, Gaithersburg, MD 20899-8910, USA

Since the Guide to the Expression of Uncertainty in Measurement (GUM) was published in 1993 it has changed the evaluation of physical and chemical measurements. Nowadays almost all high level measurements include a detailed evaluation of uncertainty. This allows the scientific community to do the next step and evaluate uncertainty for derived evaluations like parameter fittings. The evaluation of the uncertainty for complicated parameters like the results from non-linear fitting procedures can be carried out in two steps. The first step is a sensitivity analysis of the evaluation algorithm and a test of mutual independence of the parameters. If the fitting algorithm is sufficiently robust a linear model is derived from the fitting algorithm which is then used in a second step to evaluate the uncertainty of the fitting parameters. This paper discusses the sensitivity analysis in detail with the emphasis on possibilities to check for robustness and linearity. An efficient method based on covering arrays is presented to test for hidden couplings between the input parameters inside the evaluation model.

1. Introduction

In science and metrology results are often calculated based on different data and given values. A wide variety of techniques and procedures are employed including linear and non-linear fitting, optimization and simulation. Traditionally these techniques provide little support for calculating and propagating uncertainty. Also the calculations may require complicated computer calculations which may be expensive to execute. The approach which we propose here is to treat the existing algorithms and calculation schemes as black boxes, where the relation between input and output is defined by some unknown or partly unknowable function. For the calculation of the uncertainty associated with the results we follow the Guide to the Expression of Uncertainty in Measurement (GUM) [1]. The GUM propagates estimates and standard uncertainties for the input quantities using a linear approximation of the measurement function. Figure 1 illustrates a general measurement function from the GUM [1]. The GUM approach does not require complete knowledge of probability density functions (pdfs) for the input quantities. Generally, the estimates and uncertainties for the input quantities can be more reliably determined than complete pdfs. Therefore the GUM approach is more widely applicable than other approaches such as the Monte Carlo Simulation [2] which

require knowledge of pdfs. In the discussion of the efficiency of the different computational approaches it is generally assumed that the evaluation of the black box model consumes by far the most resources.

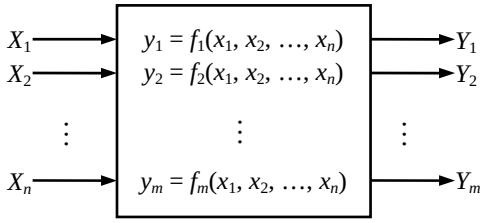

Figure 1. General measurement function

The GUM method can be applied to any kind of calculation scheme where we assume that we calculate some number m of results Y_k based on n input quantities X_i and therefore m functions f_k exist which link the input to the output (see Figure 1). Fortunately we do not need to know the functions in detail. To employ the GUM method it is enough if we know a linear equivalent of the functions around the value of the input quantities.

If the partial derivatives of the functions f_k are known, the linear equivalent of the functions can be calculated by a first order Taylor series expansion as described in the GUM. A numerical alternative to use the GUM approach which does not require evaluation of partial derivatives is described in [3]. This numerical approach, often referred to as spread-sheet method is popular among chemists.

The mainstream GUM method works well only if the unknown functions do not deviate too much from their linear approximations. Therefore we describe methods to evaluate the difference between the black box model and the linear approximation to test whether the linearized model is adequate. We have implemented the methods described here in a Python [4] script. In the last section we discuss some aspects of the implementation.

2. Calculation Of The Linear Model

The propagation of uncertainty according to the GUM is based on an equivalent linear model for the measurement function. The GUM proposes a first order Taylor series expansion to calculate the linear model. Alternatively numerical methods can be used. We use here the spread-sheet method [3].

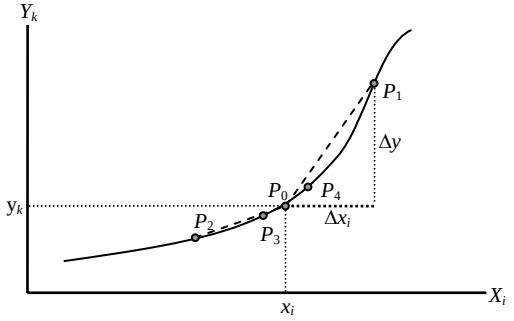

Figure 2. Linearization of function $f_k(\ldots, x_i, \ldots)$ with 5-point linearity test

The idea is simply to substitute the partial derivative in the Taylor expansion by the quotient of differences

$$\frac{\partial y_k}{\partial x_i} \cong \frac{\Delta y_k}{\Delta x_i} = \frac{y_{k,1} - y_{k,0}}{x_{i,1} - x_{i,0}} = c_{k,i} \text{ with } k = 1\ldots m,\ i = 1\ldots n \tag{1}$$

assuming a function $y_k = f_k(x_1, x_2, \ldots, x_n)$ is given and the relation between the result y_k and an input x_i is similar to the illustration in Figure 2. P_0 is given by the value of x_i and result of $f_k(\ldots, x_i, \ldots)$. Now we choose another point with coordinates $P_1 : [x_i + \Delta x_i, f_k(\ldots, x_i + \Delta x_i, \ldots)]$. We can estimate the sensitivity by the slope of a straight line through P_0 and P_1. If we would be interested in the best estimate of the slope in P_0 we would decrease Δx_i to some minimal value based on the numerical resolution of the computer calculation. Since we are not interested in the slope at the exact point P_0 but rather in an uncertainty interval around P_0 we can choose Δx_i to be equal to the given (or assumed) uncertainty $u(x_i)$. The number of model evaluations to calculate the linear model is $n + 1$, so the method scales with n.

In practice it is useful to calculate not only the sensitivities (Eq. 1) but also the relative sensitivities

$$S_{k,i} = c_{k,i} \cdot \frac{x_i}{y_k}. \tag{2}$$

With the relative sensitivities it is possible to judge if the sensitivity analysis is reasonable. The relative uncertainty multiplied by the relative sensitivity leads to the relative uncertainty contribution. A value for the relative sensitivity greater than one indicates that the black box model magnifies the uncertainty of the particular input quantity and further if the value is much larger than one it indicates that the black box model may not be numerically robust.

3. Linearity Checks

In the preceding section we presented the well-known spread-sheet approach to extract a linear model from a given model which may be non-linear. Now we discuss several possible linearity checks. All linearity checks calculate quotients of differences similar to Eq. 1 and compare the result with some combination of the sensitivities found in the previous section. The model will be considered sufficiently linear if the sensitivities agree within 5%.

3.1. *Three And Five Point Calculation*

The calculation of the linear model can be extended by choosing additional points and calculating the slope of the straight lines through them. Figure 2 illustrates this for 5 points. With P_0, P_1 and P_2 we can calculate two slopes for plus and minus $u(x_i)$. By comparing the two slopes we can detect whether P_0 is close to a local extreme, or whether $f_k(\ldots, x_i, \ldots)$ has a significant curvature around x_i. By adding two more points with a significantly smaller Δx_i we can test whether the curvature is significantly changing in an interval around x_i. In general the sensitivity can be calculated for the different points with the following equation:

$$c_{k,i}(w) = w \cdot \frac{f_k(\ldots, x_i + \dfrac{u(x_i)}{w}, \ldots) - f_k(\ldots, x_i, \ldots)}{u(x_i)} . \tag{3}$$

The parameter w determines Δx_i expressed as a fraction of $u(x_i)$. Useful values for w are 1, −1, 10 and −10. This leads to the criteria for sufficient linearity

$$\left| \frac{c_{k,i}(1)}{c_{k,i}(-1)} - 1 \right| < \varepsilon_{\text{lin}} , \left| \frac{c_{k,i}(1)}{c_{k,i}(10)} - 1 \right| < \varepsilon_{\text{lin}} \text{ and } \left| \frac{c_{k,i}(-1)}{c_{k,i}(-10)} - 1 \right| < \varepsilon_{\text{lin}} , \tag{4}$$

with ε_{lin} being the selected linearity limit of 0.05. The number of model evaluations to calculate the linearity check is $(p-1)\cdot n + 1$ with p being the number of points used for the check, so the method scales with n.

3.2. *All Pairs Coupling Test*

The linearity check discussed in the last section is a robust method to check for sufficient linearity within one input parameter but it does not check for any coupling between parameters (input quantities). An example of such coupling is the product of two input quantities $Y = X_1 \cdot X_2$. This example is of some relevance in metrology. A significant deviation from the linear model can be observed if the value of one or both of the quantities is small compared to the uncertainty of the value. The three and five point linearity check will not detect this situation.

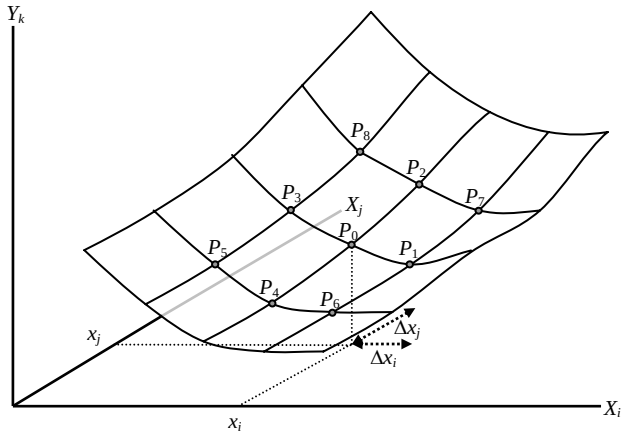

Figure 3. Evaluation of the function $f_k(\ldots,x_i,x_j,\ldots)$ around P_0

If we want to check for this kind of non-linearity in the black box model, we need to calculate additional points. We choose a pair of input quantities x_i and x_j for which we want to analyze the model. Figure 3 illustrates the method. During the three point linearity check we have already calculated P_0, P_1, P_2, P_3 and P_4. So we know the linear sensitivity in x_i and x_j to be $c_{k,i}(1)$ and $c_{k,j}(1)$.
Now we can evaluate the model at P_5, P_6, P_7 and P_8 and calculate the difference from the center point P_0:

$$d_k(w_i,w_j) = f_k(\ldots,x_i+\frac{u(x_i)}{w_i},x_j+\frac{u(x_j)}{w_j},\ldots)-f_k(\ldots,x_i,x_j,\ldots) \qquad (5)$$

with w_i and w_j being either plus or minus one. Table 1 shows the relation between the points and the w_i and w_j values.

Table 1. Combinations of w_i and w_j

Point	w_i	w_j
P_5	–1	–1
P_6	+1	–1
P_7	+1	+1
P_8	–1	+1

A similar difference can be calculated for the linear model

$$l_k(w_i,w_j) = c_{k,i}(1)\cdot\frac{u(x_i)}{w_i}+c_{k,j}(1)\cdot\frac{u(x_j)}{w_j}. \qquad (6)$$

The coupling can be checked by evaluating the following condition:

$$\left|\frac{d_k(w_i,w_j)-l_k(w_i,w_j)}{l_k(w_i,w_j)}\right| < \varepsilon_{\text{lin}}. \qquad (7)$$

Since it is not known in advance which of the input quantities might be coupled we need to repeat the check for all pairs. The number of model evaluations to

calculate the all-pairs coupling check is $(n^2-n)/2$, so this method scales as n^2 if n is large.

3.3. *Covering Array Based Coupling Test*

The all pairs coupling test discussed in the previous section is a useful method to judge whether a black box model has some significant pair-wise coupling between the input parameters. Unfortunately the evaluation may be costly because it scales with n^2. Especially for complicated data fitting and expensive simulation runs when the number of input parameters is large (greater 20) this analysis is expensive. Much more resources are needed for the coupling test than for the extraction of the linear model.

To overcome this problem we look for couplings between several pairs in one run. Instead of varying just one pair of input parameters we vary several pairs in one run and check for significant differences. With all runs we have to ensure that all combinations of input pair variations appear in the runs. This is an application of so called covering arrays [5]. Covering arrays are used in software testing as optimized test patterns for finding parameter interaction faults which are hidden couplings in software programs. So a covering array is an optimized test pattern for a coupling check. The covering are constructed in such a way that all necessary patterns are covered by the arrays. The covering arrays have as many columns as the black box model has input parameters. Every row is a separate test case. The number of rows depends on the number of parameters, the number of discrete values for each parameter and the strength of the array. The strength is a measure of the complexity of the interactions that can be detected. For the application of the coupling test we need a binary array with strength of two. A number of different algorithms exist to generate covering arrays in an efficient way [6].

The covering array (for an example see Table 2) contains a row with n values which are either −1 or +1. For every row of the covering array the difference

$$d_k^*(w_1,\ldots,w_n) = f_k(x_1 + \frac{u(x_1)}{w_1},\ldots,x_n + \frac{u(x_n)}{w_n}) - f_k(x_1,\ldots,x_n) \tag{8}$$

is calculated from one extra black box model evaluation. The $w_1 \ldots w_n$ are the pattern values from the rows of the covering array. We also need a similar value

$$l_k^*(w_1,\ldots,w_n) = \sum_{i=1}^{n} c_{k,i}(1) \cdot \frac{u(x_i)}{w_i} \tag{9}$$

based on the assumption that the model is perfectly linear. In Eq. 9 the $w_1 \ldots w_n$ are the same pattern values as in Eq. 8. Under the condition that

$$\left| \frac{d_k^*(w_1,\ldots,w_n) - l_k^*(w_1,\ldots,w_n)}{l_k^*(w_1,\ldots,w_n)} \right| < \varepsilon_{lin}\,, \tag{10}$$

we can judge whether or not any coupling is observable with this test pattern. If no significant coupling shows up during any of the model evaluations we can conclude that the linear model is representing the behavior of the black box model for small changes in the input values. It is possible that a coupling in one pair of parameters may be compensated by another coupling in another parameter and we are not observing the coupling with the combinatorial pattern. This can only be detected with the all pairs approach with a much higher cost.

Table 2. Example of a binary covering array of strength 2 for 10 parameters

run	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
1.	–1	–1	–1	–1	–1	–1	–1	–1	–1	–1
2.	–1	+1	+1	+1	+1	+1	+1	+1	+1	+1
3.	+1	–1	+1	–1	+1	–1	+1	–1	+1	–1
4.	+1	+1	–1	+1	–1	+1	–1	+1	–1	+1
5.	–1	+1	+1	–1	–1	–1	–1	+1	+1	–1
6.	–1	–1	–1	+1	+1	+1	+1	–1	–1	+1
7.	–1	+1	+1	+1	+1	–1	–1	–1	–1	–1
6.	–1	–1	–1	–1	–1	+1	+1	+1	+1	+1
9.	+1	–1	+1	–1	–1	+1	+1	+1	–1	–1
10.	+1	–1	–1	+1	+1	–1	–1	–1	–1	+1

The number of model evaluations to calculate the combinatorial coupling check is dependent on the number of rows in the covering array. For a large number of parameters the upper bound scales with $\log_2(n)$.

4. Implementation

The linear model approximation and the different linearity checks were implemented in a Python script [4], which can call an arbitrary command line program to execute the black box model calculations.

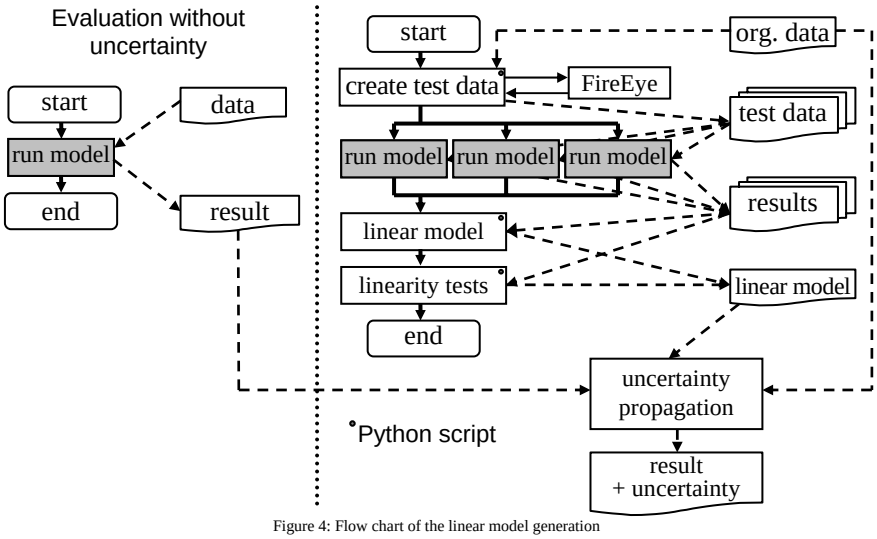

Figure 4: Flow chart of the linear model generation

The basic flow chart is shown in Figure 4. Based on the original data file a number of modified test data files are generated to calculate the linear model and to perform the linearity checks. All data is stored in text files (csv). The necessary covering arrays are generated with the command line version of the software tool FireEye [6].

The test data files are used by the black box model which generates the result files. Since the model calculations are independent from each other it is possible to execute several runs of the model in parallel. After all runs of the model are completed the linear model is generated and the linearity checks are run. For test purposes the black box model can be replaced by a white box model with a known algebraic equation system. This allows verifying the script by testing that the correct linear system is determined for well defined equations.

5. Conclusions

It is well-known that the standard GUM method can be applied to any kind of result evaluation if the evaluation model is sufficiently linear. The so-called spread-sheet method can be efficiently used to calculate the sensitivities of an equivalent linear model for the measurement. If the result evaluation is available in the form of a command line program which can be called by a script language it is possible to generate automatically an equivalent linear model which can then be used to propagate the uncertainty. In this way the propagation of uncertainty can be added to virtually any kind of computerized result evaluation without any change to the existing code.

Additionally we propose some useful linearity and coupling tests to verify that the basic assumption about the sufficient linearity of the model can be justified. We introduce covering arrays as a way to find efficient test cases. The combination of linear model generation, three-point linearity test and coupling test based on covering arrays is an efficient way to establish a linear model which allows us to use mainstream GUM uncertainty propagation. The proposed approach is not a mathematical proof of linearity; however, it is of practical use especially useful in cases when the calculation of the result evaluation requires a lot of computing resources.

References

[1] BIPM *et. al.* Guide to the Expression of Uncertainty in Measurement (GUM). ISBN 92-67-1011889, 1st Edition (1993) 2ed Edition (1995).
[2] BIPM *et. al.* Supplement 1 to the GUM: Propagation of distributions using a Monte Carlo method. 2008)
[3] Kragten J. *Analyst.* 119:2161–2166 (1994)
[4] Python Programming Language, `http://www.python.org`
[5] Lawrence, J. F. *et. al.* Binary Covering Arrays. submitted (2008)
[6] FireEye: Executable copies can be obtained from `kuhn@nist.gov`