Object-oriented software for evaluating measurement uncertainty

B. D. Hall
Measurement Standards Laboratory of New Zealand,
Callaghan Innovation, Lower Hutt 5040,
New Zealand.

Abstract

An earlier publication (B. D. Hall, *Metrologia*, **43** (2006) L56–L61) introduced the notion of an *uncertain number* that can be used in data processing to represent quantity estimates with associated uncertainty. The approach can be automated, allowing data processing algorithms to be decomposed into convenient steps, so that complicated measurement procedures can be handled. This article illustrates the uncertain-number approach using several simple measurement scenarios and two different software tools. One is an extension library for Microsoft Excel®. The other is a special-purpose calculator using the Python programming language.

1 Introduction

The *Guide to the expression of uncertainty in measurement* (GUM), first published in 1993, has profoundly influenced the way that measurement uncertainty is evaluated and reported [1]. The GUM has helped to standardize the way that measurement results are communicated, which has improved the traceability of results to the SI. Nevertheless, wider uptake of the GUM is hampered by the difficulty of formulating some types of problem and carrying out the associated computations. Greater use of software tools may offer a solution to this problem. This paper shows how the GUM recommendations can be applied using an algorithmic approach that allows more complicated problems to be handled.

A recent publication introduced the notion of an *uncertain number*, an abstract entity that encapsulates information about the value and uncertainty components of a quantity estimate, or measurement result [2]. Mathematical operations involving uncertain numbers can be automated. So uncertain numbers can be used to express measurement data processing at a higher conceptual level, leaving much of the analysis to software (e.g., partial derivatives can be inferred [3]).[1]

This article uses several simple examples to illustrate the uncertain-number approach. In each case, some of the effects that contribute to uncertainty are considered fixed (systematic errors). That is, the error is considered to endure for more than just one phase in the measurement procedure. Systematic errors couple different steps of a procedure, introducing correlation among intermediate results. When this happens, a conventional uncertainty calculation can become unmanageable. However, the uncertain-number approach is inherently multi-step, so it is able to deal with uncertainty due to systematic errors. The examples also show how the modularity of measurement systems can be exploited when developing data processing algorithms.

Two implementations of the uncertain numbers method are presented. One is an add-in for Microsoft Excel®, called `SGUM`. The other is a stand-alone program called the *GUM*

[1] Abstraction is a commonly used computational device. For example, special commands are often available complex-number maths and matrix operations that hide lower-level operations and make it easier to describe calculations.

Tree Calculator (`GTC`) [4]. In `SGUM`, a unique spreadsheet cell is associated with each uncertain number, which leads to intuitive spreadsheet layouts. `GTC` is a more powerful and versatile tool. It uses the Python [5] programming language, which has object-oriented programming features, and the calculator includes a library of functions and class definitions that can be used to simplify the expression of complicated calculations (A gives some further explanation about the Python programming structures used in the text). `GTC` can be used as an interactive calculator, or as a batch processor tailored to specific types of measurement problem. The examples in this paper give the reader a glimpse of how the combination of uncertain numbers and object-oriented design can be combined to develop effective data processing routines.

Section 2 briefly reviews the GUM approach to uncertainty calculation. Then sections 3, 4 and 5 describe three simple measurement scenarios and present uncertain-number data processing for each. Section 6 discusses some implications of the new approach. In particular, the possibility of providing full *transferability* of measurement results from one link in a traceability chain to the next, which is one of the defining principles of the GUM [1, §0.4]. Greater use of the method may therefore facilitate GUM-compliant uncertainty calculations along a traceability chain, from national measurement institutes though calibration laboratories to the end-users of measurements.

1.1 Notation

In this article the 'hat' symbol is used to denote an estimate, or measurement result. For example, $\widehat{R}_{\rm std}$ is an estimate of the quantity $R_{\rm std}$; the standard uncertainty of $\widehat{R}_{\rm std}$ is written as $u(\widehat{R}_{\rm std})$. Note, however, that there is no need to distinguish between unknown quantity values and estimates in data processing algorithms, because data are always numerical estimates. So, the hat annotation is dropped when naming software variables. For example, `R_std` would be used to refer to an uncertain number associated with $\widehat{R}_{\rm std}$, rather than `R_std_hat`.

The letter e, with an appropriate subscript, like $e_{\rm std}$, is used exclusively for residual measurement errors.

2 Measurement models and GUM uncertainty calculations

The GUM approach to evaluating measurement uncertainty is based on the idea of a *measurement model*, which is a mathematical relationship between the quantity intended to be measured Y and *all* the quantities $X_1, X_2, \cdots, X_N$ that influence the outcome of the measurement procedure. Some influence quantities may be undesirable effects that cannot be completely eliminated by the procedure. These residual errors are often ignored when calculating the measurement result, because their estimates are zero (or unity, when the error is a multiplicative factor) and therefore do not affect the result. However, for GUM uncertainty calculations, the measurement model must explicitly refer to all sources of error.

In the GUM, a measurement model is represented as a single function

$$Y = f(X_1, X_2, \cdots, X_N) \; . \tag{1}$$

The X_i will not be known, but an estimate of Y can be obtained from the information available. So,

$$\widehat{Y} = f(\widehat{X}_1, \widehat{X}_2, \cdots, \widehat{X}_N) \; . \tag{2}$$

represents the data processing required to obtain a measurement result.

The uncertainty of $\widehat{Y}$, as an estimate of the quantity of interest, can be evaluated using the *Law of Propagation of Uncertainty* (LPU) described in the GUM. The sensitivity of a result to each influence quantity is multiplied by the standard uncertainty of the influence-estimate to obtain a component of uncertainty in the result due to the i^{th} influence

$$u_i(\widehat{Y}) = \left.\frac{\partial Y}{\partial X_i}\right|_{X_i=\widehat{X}_i} u(\widehat{X}_i) \; . \tag{3}$$

These components can be combined using the LPU formula to obtain the standard uncertainty

$$u(\widehat{Y}) = \left[\sum_i \sum_j u_i(\widehat{Y})\, r_{ij}\, u_j(\widehat{Y})\right]^{1/2} \tag{4}$$

where r_{ij} is the correlation coefficient between estimates $\widehat{X}_i$ and $\widehat{X}_j$.

Components of uncertainty are needed when calculating the correlation coefficient between two results $\widehat{Y}_l$ and $\widehat{Y}_m$ when some influences are shared [1, §H.2.3],

$$r(\widehat{Y}_l, \widehat{Y}_m) = \frac{\sum_i \sum_j u_i(\widehat{Y}_l)\, r_{ij}\, u_j(\widehat{Y}_m)}{u(\widehat{Y}_l) u(\widehat{Y}_m)} \; . \tag{5}$$

They are also used in the calculation of effective degrees of freedom using the Welch-Satterthwaite formula [1, §G.4.1].

In practice, however, data-processing is often carried out as a sequence of distinct steps. Intermediate results can have meaning to the metrologist and may be noted as part of careful measurement practice. So, multi-step data processing will often be preferred to the use of a single measurement function that maps all influence estimates to a result.

In a multi-step procedure, the measurement model can be thought of as the composition of a set of intermediate measurement functions, which are evaluated in a particular order to obtain the measurement result. This is mathematically equivalent to the single-function approach. Each intermediate result could be expressed as

$$\widehat{X}_k = f_k(\Lambda_k) \; , \tag{6}$$

where 'k' identifies a particular step, $\widehat{X}_k$ is the value calculated at the step and Λ_k is a set of direct inputs to the function f_k, which may include results obtained from earlier steps. There is no real distinction to be made between an intermediate result and the final result in the multi-step formulation, because additional steps can always be added. Each step-function evaluates an estimate. An associated set of uncertainty components can also be obtained, from which a standard uncertainty of the estimate could be calculated. The chain rule for partial differentiation can be used to evaluate the partial derivatives needed in the uncertainty components. The details are discussed further in references [2], [6] and [7].

Uncertain-number software uses the multi-step approach just described. So calculations yield a GUM-compliant result, while offering the flexibility of step-by-step data processing. The propagation of uncertainty components is fully automated, so the job of keeping track of common influences that give rise to correlation among intermediate results is inherently solved by design. The following sections give some examples.

3 Example 1: resistors in series

We consider ten similar 1 kΩ resistors that are to be connected in series to realize a 10 kΩ resistor. Before joining them, the resistances are measured with an instrument that compares each resistor to a calibrated 1 kΩ reference.

A calibration certificate reports the value of the reference resistor as 1 kΩ, but this is only an estimate, so we write

$$\widehat{R}_{\text{std}} = R_{\text{std}} + e_{\text{std}} \tag{7}$$

where $\widehat{R}_{\text{std}}$ is the reported value, R_{std} is the actual (unknown) value and e_{std} is the unknown error. A standard uncertainty of 100 mΩ is also reported in the certificate, which concerns the possible magnitude of e_{std} that should be taken into account.

In a simple model of the comparator, an independent error e_i will be generated each time a resistor is compared with a reference. For the i^{th} resistance the comparator measures $d_i = R_i - R_{\text{std}}$ and indicates

$$\widehat{d}_i = d_i + e_i \ . \tag{8}$$

The uncertainty in $\widehat{d}_i$ as an estimate of d_i depends on the possible magnitude of e_i. The associated uncertainty will be specified in a calibration report for the instrument. Our estimate of R_i is

$$\widehat{R}_i = \widehat{d}_i + \widehat{R}_{\text{std}} \tag{9}$$

and, if negligible additional resistance is introduced when the resistors are connected together, an estimate of the total series resistance is

$$\widehat{R}_{\text{sum}} = \sum_{i=1}^{10} \widehat{R}_i \ . \tag{10}$$

The uncertainty of $\widehat{R}_{\text{sum}}$ depends on the possible magnitudes of the ten independent errors $e_1, e_2, \cdots, e_{10}$, as well as common the error e_{std}.

Data processing can be based on equations (7) – (10). Equations (7) and (8) define the input estimates: $\widehat{R}_{\text{std}}$, from the calibration of the standard resistor, and $\widehat{d}_i$ from the comparator readings. Information about the associated errors will be used to describe the input uncertainties. Equation (9), on the other hand, is an intermediate step that defines an estimate of each resistance. The final result (10) is obtained by summing the ten resistance estimates.

Note, when describing data processing software in the following sections, the terms *elementary* and *intermediate* will be used to distinguish between uncertain numbers representing inputs and intermediate results, respectively. Uncertain numbers always represent quantity estimates, but inputs must be defined using numerical data whereas intermediate results are defined by functional relationships. So, for example, the estimates defined in equations (7) and (8) will be associated with elementary uncertain numbers, while the estimates defined in (9) and (10) will be represented by intermediate uncertain numbers.

3.1 `SGUM` calculation

Fig. 1 shows a worksheet that calculates $\widehat{R}_{\text{sum}}$ and the standard uncertainty $u\left(\widehat{R}_{\text{sum}}\right)$. The cells containing uncertain numbers in columns B and C are highlighted in grey. The results of the calculation appear in the box at the bottom, which includes the correlation coefficient between estimates of two different resistances (corresponding here to rows 5 and 14).

Numerical input data appear in worksheet columns D and E. Column D contains estimates: D3 has the reported value of $\widehat{R}_{\text{std}}$ and cells D5 to D14 contain the comparator indications for the ten resistors (these are all zero, so each estimate is the same as the standard). Column E contains uncertainties: E3 has the value of $u\left(\widehat{R}_{\text{std}}\right)$ from the

calibration certificate and E5 has the standard uncertainty associated with comparator errors. The result, $\widehat{R}_{\text{sum}} = 10\,\text{k}\Omega$, has an uncertainty $u\left(\widehat{R}_{\text{sum}}\right) \approx 1\,\Omega$.[2]

	A	B	C	D	E
1	**Expression**	**UN**	**UN**	**x**	**u**
2					
3			**R_std**	1000	0.1
4					
5	R_i = d_i + R_std	**R_i**	**d_i**	0	0.001
6	R_i = d_i + R_std	**R_i**	**d_i**	0	
7	R_i = d_i + R_std	**R_i**	**d_i**	0	
8	R_i = d_i + R_std	**R_i**	**d_i**	0	
9	R_i = d_i + R_std	**R_i**	**d_i**	0	
10	R_i = d_i + R_std	**R_i**	**d_i**	0	
11	R_i = d_i + R_std	**R_i**	**d_i**	0	
12	R_i = d_i + R_std	**R_i**	**d_i**	0	
13	R_i = d_i + R_std	**R_i**	**d_i**	0	
14	R_i = d_i + R_std	**R_i**	**d_i**	0	
15					
16	R_sum += R_i	**R_sum**			
17					
18	**Results : R_sum**				
19					
20	value	10000			
21	uncertainty	1.000005			
22					
23	correlation	0.99990001			

Figure 1: Spreadsheet evaluating $\widehat{R}_{\text{std}}$ and $u\left(\widehat{R}_{\text{std}}\right)$.

`SGUM` provides a library of functions for uncertain-number data processing that can be entered into spreadsheet cells and executed when the worksheet is evaluated. Fig. 2 shows the worksheet functions used in this calculation. The function `unGaussian` defines elementary uncertain numbers. It is used in cell C3 to create an uncertain number representing the calibrated standard resistor; the numbers required for this definition come from cells D3 and E3. Elementary uncertain numbers are also defined for the comparator readings in C5 to C14, using the comparator indication, in the adjacent cell of column D (all values are zero here), and the value of uncertainty, in E5 (the same for all ten measurements).

Intermediate uncertain numbers representing the $\widehat{R}_i$ are obtained by adding the comparator readings to the reference resistance. The worksheet function `unEquation` is used to create these in B5 to B14. This function requires an equation-string and cell references to the equation's arguments. In Fig. 2, the equation string is in a cell to the left of the function, in column A, while an uncertain number representing $\widehat{d}_i$ is in the adjacent cell of column C and the uncertain number for $\widehat{R}_{\text{std}}$ is in C3.

Finally, an intermediate uncertain number for the resistance sum is defined by `unSum` in B16. An expression string for this is given in A16 and the range of uncertain-number

[2] A similar scenario is analyzed in the GUM (see §5.2.2, note 1, and §F.1.2.3, Example 2, [1]). In that case, the comparator uncertainty is considered negligible, so the correlation between resistance estimates is exactly 1.

cells to be summed is from B5 to B14.

	A	B	C	D	E
1	Expression	UN	UN	x	u
2					
3			=unGaussian("R_std",D3,E3)	1000	0.1
4					
5	R_i = d_i + R_std	=unEquation(A5,C3,C5)	=unGaussian("d_i",D5,E5)	0	0.001
6	R_i = d_i + R_std	=unEquation(A6,C3,C6)	=unGaussian("d_i",D6,E5)	0	
7	R_i = d_i + R_std	=unEquation(A7,C3,C7)	=unGaussian("d_i",D7,E5)	0	
8	R_i = d_i + R_std	=unEquation(A8,C3,C8)	=unGaussian("d_i",D8,E5)	0	
9	R_i = d_i + R_std	=unEquation(A9,C3,C9)	=unGaussian("d_i",D9,E5)	0	
10	R_i = d_i + R_std	=unEquation(A10,C3,C10)	=unGaussian("d_i",D10,E5)	0	
11	R_i = d_i + R_std	=unEquation(A11,C3,C11)	=unGaussian("d_i",D11,E5)	0	
12	R_i = d_i + R_std	=unEquation(A12,C3,C12)	=unGaussian("d_i",D12,E5)	0	
13	R_i = d_i + R_std	=unEquation(A13,C3,C13)	=unGaussian("d_i",D13,E5)	0	
14	R_i = d_i + R_std	=unEquation(A14,C3,C14)	=unGaussian("d_i",D14,E5)	0	
15					
16	R_sum += R_i	=unSum(A16,B5:B14)			

Figure 2: This alternate view of the sheet in Fig. 1 shows the worksheet functions used to create uncertain numbers.

The value of the series-resistance estimate and the associated standard uncertainty are obtained by applying functions to the result, as shown in Fig, 3: `unValue` returns the estimate and `unUncertainty` calculates the standard uncertainty of the estimate. The correlation coefficient $r \approx 1$, in this case between the first and last resistance measurements, is evaluated in B23. The value of uncertainty $u\left(\widehat{R}_{\text{sum}}\right) = 1.000005\,\Omega \approx 10\,u(\widehat{R}_{\text{std}})$ is a result of the strong correlation ($r = 0.9999$) between resistance estimates. The systematic error e_{std} is a common influence of each R_i estimate and the associated uncertainty $u(\widehat{R}_{\text{std}})$ is much greater than $u(\widehat{d}_i)$. As a consequence, the $\widehat{R}_i$ are highly correlated and the uncertainties combine almost linearly.

To investigate several other scenarios, the numbers can be changed. For example, by setting the comparator uncertainty in E5 to the same value as $u(\widehat{R}_{\text{std}})$, $u(\widehat{d}_i) = 0.10\,\Omega$, the correlation coefficient between the resistance estimates drops to $r \approx 0.5$ and we obtain $u\left(\widehat{R}_{\text{sum}}\right) = 1.0488\,\Omega$. If we then reduce the reference-resistor uncertainty to $u(\widehat{R}_{\text{std}}) = 0.001\,\Omega$, the uncertainties associated with the independent errors e_i dominate and the correlation coefficient between the resistance estimates is negligible. The uncertainty components due to $u(\widehat{d}_i)$ then effectively add in quadrature and the standard uncertainty $u\left(\widehat{R}_{\text{sum}}\right) = 0.3164\,\Omega$, which is approximately $\sqrt{10}\,u(\widehat{d}_i)$.

18	Results : R_sum	
19		
20	value	=unValue(B16)
21	uncertainty	=unUncertainty(B16)
22		
23	correlation	=unGetCorrelation(B5,B14)

Figure 3: Worksheet functions `unValue` and `unUncertainty`, in cells B20 and B21, obtain $\widehat{R}_{\text{std}}$ and $u\left(\widehat{R}_{\text{std}}\right)$. Correlation between estimates of individual resistors can also be evaluated by `unGetCorrelation`.

A slightly different spreadsheet layout would be used to investigate the scenario in which there is a common (systematic) comparator error in all measurements. Distinct uncertain numbers can be defined for the comparator readings and the comparator error. Uncertain numbers with zero uncertainty would be defined for each comparator reading

(an SGUM function called unConstant creates uncertain numbers with no uncertainty) and a single uncertain number would be defined for the zero-valued estimate of the common comparator error. An uncertain number associated with $\widehat{d}_i$ could then be obtained by summing the uncertain number for the i^{th} reading and the uncertain number asssociated with the comparator error estimate. This addition could be conveniently combined the calculation of an uncertain number for $\widehat{R}_i$, by adding three terms instead of two.

3.2 GTC calculation

Listing 1 shows a sequence of instructions that calculate $\widehat{R}_{\text{sum}}$ and the standard uncertainty $u\left(\widehat{R}_{\text{sum}}\right)$.[3] Lines 1–8 associate names with numerical data. x_R_std and u_R_std refer, respectively, to the reported value of the reference resistor and the standard uncertainty. The ten comparator readings are entered in a one-dimensional data structure named x_d; the associated uncertainty u_d_i is the same for each measurement. The numbers used are the same as in §3.1.

Uncertain numbers are created in lines 10–16 of listing 1. The GTC function ureal defines elementary uncertain numbers, while intermediate uncertain numbers are defined implicitly by mathematical expressions. In line 11, the elementary uncertain number R_std is associated with $\widehat{R}_{\text{std}}$. Then, in lines 13–14, a sequence of intermediate uncertain numbers representing the ten $\widehat{R}_i$ are stored in R, another one-dimensional data structure. The semantics of lines 13–14 can be explained as follows. An elementary uncertain number representing the comparator reading is created in line 13 by the function call ureal(x_d_i,u_d_i). This is immediately added to R_std, creating an intermediate uncertain number that is stored as an element in R. The expression for x_d_i in x_d, in the line below, repeats this process for each element of the sequence x_d. The final step associates the name R_sum with an intermediate uncertain number that is the sum of the elements stored in R.

```
1  # standard resistor
2  x_R_std = 1E3 # Ohm
3  u_R_std = 0.1 # Ohm
4
5  # comparator readings
6  x_d = [ 0.0,0.0,0.0,0.0,0.0,
7          0.0,0.0,0.0,0.0,0.0]
8  u_d_i = 0.001
9
10 # uncertain numbers
11 R_std = ureal(x_R_std,u_R_std)
12
13 R = [ ureal(x_d_i,u_d_i) + R_std
14         for x_d_i in x_d ]
15
16 R_sum = sum(R)
17
18 # results
19 print value(R_sum)
20 print uncertainty(R_sum)
21 print get_correlation(R[0],R[9])
```

Listing 1: The calculation of $\widehat{R}_{\text{std}}$ and $u\left(\widehat{R}_{\text{std}}\right)$ using GTC

[3] Data processing procedures like this are stored in files that can be evaluated by GTC.

Results are displayed by the `print` statements in lines 19–21. In a similar way to §3.1, the estimate of R_{sum} and the associated standard uncertainty are obtained by applying the functions `value` and `uncertainty`, respectively, to the result `R_sum`. The correlation coefficient between the first and last resistor measurements is calculated by `get_correlation`. The numerical results are the same as obtained in §3.1.

To investigate the alternative scenario where a common (systematic) comparator error affects all readings, the code can be modified as shown in listing 2. As explained in §3.1, a single uncertain number should now be defined for the comparator error estimate (here, `e_comp`). Uncertain numbers associated with the $\widehat{R}_i$ are now obtained by adding the comparator readings to the uncertain numbers associated with the standard resistor and the comparator error estimate (`GTC` calculates the sum of an ordinary number with an uncertain number assuming that the number has no uncertainty).

```
10 # uncertain numbers
11 R_std = ureal(x_R_std,u_R_std)
12 e_comp = ureal(0,u_d)
13 R = [ x_d_i + e_comp + R_std
14         for x_d_i in x_d ]
```

Listing 2: Modifications to listing 1 when the comparator error is systematic

3.3 Comparison between `SGUM` and `GTC`

The spreadsheet layout in `SGUM` emphasizes the association between quantity estimates and the uncertain numbers representing them. An uncertain number defined in a spreadsheet cell is associated with one, and only one, estimate. So, when estimates are independent, each is associated with a different uncertain number (e.g., cells C5 to C14). However, when a single estimate is used repeatedly, multiple references to a single cell are used. In §3.1, for example, cell C3 contains the uncertain number associated with $\widehat{R}_{\text{std}}$ and is referenced when calculating the ten intermediate resistance estimates in cells B5–B14. The repeated reference to C3 shows that the error in $\widehat{R}_{\text{std}}$, as an estimate of R_{std}, is a common influence in all the resistance measurements.

The Python code of `GTC` may be harder to follow for some readers, because it requires familiarity with programming language structures [5]. For example, the Python idiom used to create uncertain numbers representing resistance estimates in lines 13–14 is more concise than the ten spreadsheet rows required in `SGUM`, but needed some explanation to interpret. However, the same principle applies: an uncertain-number instance is associated with one, and only one, estimate. The more advanced programming features available in `GTC` allow data processing to be tailored to particular types of problem. Later sections will show that `GTC` can be used to develop succinct and flexible data processing routines.

4 Example 2: a voltmeter model

In this section we consider a simple resistance ratio measurement. Fig. 4 shows an electrical circuit with a current source and two resistors connected in series. The resistor R_{s} has been calibrated and R_{x} is to be measured. A model of the voltmeter will be used to calculate the uncertainty of individual voltage measurements and the ratio of voltages can be used to estimate R_{x}.

When the voltmeter is operated on a particular range, we consider three errors: a gain error e_{gain}, an offset error e_{off}, and a differential linearity error e_{dif}. The possible

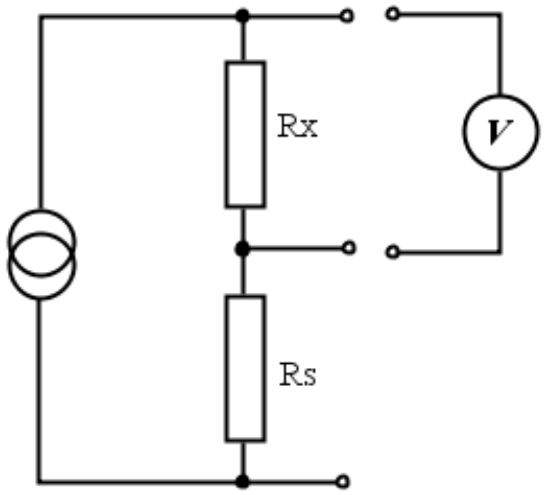

Figure 4: A simple resistance ratio measurement set-up. The voltmeter can be connected across either R_{s} or R_{x}.

magnitude of the gain error is proportional to the voltage being measured. The offset error is the actual response when zero voltage is applied. The differential linearity error is modeled as a random error from one measurement to the next. Differential linearity errors are typically an order of magnitude smaller than gain errors.

The meter measures an applied voltage V_i but indicates

$$x_i = V_i \left(1 + e_{\mathrm{gain}} + e_{\mathrm{dif}\cdot i}\right) + e_{\mathrm{off}} \ , \tag{11}$$

where the index i distinguishes between different measurements. An estimate of V_i is found by re-arranging (11) and replacing unknown quantities with estimates

$$\widehat{V}_i = \frac{x_i - \widehat{e}_{\mathrm{off}}}{1 + \widehat{e}_{\mathrm{gain}} + \widehat{e}_{\mathrm{dif}\cdot i}} \ . \tag{12}$$

Usually, estimates of the meter's residual errors reduce the model to an ideal instrument, so $\widehat{e}_{\mathrm{gain}} = 0$, $\widehat{e}_{\mathrm{dif}\cdot i} = 0$ and $\widehat{e}_{\mathrm{off}} = 0$.

If the current is constant, R_{x} can be obtained from the ratio of voltages measured across R_{s} and R_{x}. Then, using the calibrated value of R_{s},

$$\widehat{R}_{\mathrm{x}} \ = \widehat{R}_{\mathrm{s}} \frac{\widehat{V}_{\mathrm{x}}}{\widehat{V}_{\mathrm{s}}} \ . \tag{13}$$

For this type of measurement, it is well-known that the uncertainty in the ratio due to gain and offset errors can be reduced when one meter is used for both measurements (i.e., e_{gain} and e_{off} are the same during the voltage measurements across R_{s} and R_{x}). However, we will develop data processing using the model of (12) and show later that uncertain-number data processing can evaluate the measurement uncertainty in either case.

4.1 SGUM calculation

Equations (12) and (13) can be used to develop data processing. Fig. 5 shows a worksheet that calculates $\widehat{R}_{\mathrm{x}}$ and the standard uncertainty $u\left(\widehat{R}_{\mathrm{x}}\right)$. At the top of the sheet, in rows 3–6, information relating to the voltmeter error model is entered. Cells E4–E6 contain the standard uncertainty associated with estimates of the gain error, the offset error and the differential linearity error, respectively. To the left, uncertain numbers in

	A	B	C	D	E	F
1	Expression	UN	UN	x	u	Comment
2						
3	=(x_i - e_off)/(1+e_gain+e_dif_i)					voltmeter error model
4			e_gain		3.00E-06	uncertainty in gain is a few ppm
5			e_off		1.00E-06	uncertainty in offset is about 1 microvolt
6					1.00E-07	uncertainty in differential linearity
7						
8			R_s	1000	1.00E-03	standard 1000-ohm resistor
9						
10			x_i	5.01		meter indication for V_s
11			x_i	4.9885		meter indication for V_x
12						
13		V_s	e_dif_i			
14		V_x	e_dif_i			
15						
16	R_x = R_s * (V_x/V_s)	R_x				
17						
18	Results					
19						
20	value	995.7085828				
21	uncertainty	0.001005617				
22						
23	components of uncertainty:					
24	R_s	9.9571E-04				
25	e_dif_i (for V_s)	9.9571E-05				
26	e_dif_i (for V_x)	9.9571E-05				
27	e_off	8.5657E-07				
28	e_gain	0.0000E+00				

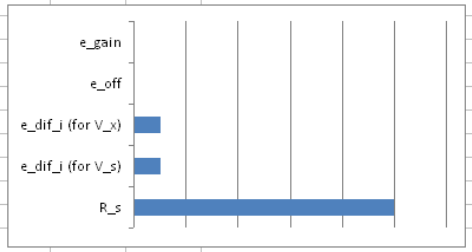

Figure 5: A worksheet to calculate the value of a resistance R_x from a voltage ratio measurement. A single voltmeter is used to measure the potential difference across each resistor.

C4 and C5 are associated with estimates of the gain and offset errors. Below, in C8, an uncertain number for $\widehat{R}_\mathrm{s}$ is defined.

Uncertain numbers associated with $\widehat{V}_\mathrm{x}$ and $\widehat{V}_\mathrm{s}$ are defined in B13 and B14. In Fig. 6, the Excel® dependency-tracing feature shows how the definition in B13 depends on uncertain numbers in other cells. B13 uses `unEquation` to evaluate the expression in A3 with uncertain-number arguments from cells C4, C5, C10 and C13. This uses equation (12) to estimate V_s from estimates of the meter errors and the meter indication. Cells C4, C5 and C13 define elementary uncertain numbers and C10 is an uncertain-number constant (a 'constant' has no uncertainty). The corresponding dependency tree for B14 would use the same expression in A3 and the same meter error terms in C4 and C5 (common influences), but link to the constant uncertain number in C11 (direct reading) and the elementary uncertain number for differential linearity in C14 (independent influence).

The result, for $\widehat{R}_\mathrm{x}$, is defined in B16 by evaluating the expression in A16. Numerical results are shown in the box at the bottom of Fig. 5. These include the various components of uncertainty, which are also shown in a graph on the right. It is clear that the gain and offset errors make a negligible contribution to the measurement uncertainty in this case. However, if two different meters are used instead of one, the data processing can be changed as shown in Fig. 7. The last step of the calculation, in B21, is unchanged, but now a distinct set of uncertain numbers is defined for each meter's errors. The calculation of $\widehat{V}_\mathrm{s}$, in B18, refers to the first set in C4 and C5, whereas the calculation of $\widehat{V}_\mathrm{x}$ in B19 refers to the second set in C9 and C10. The components of uncertainty associated with each meter's e_gain error now dominate the uncertainty budget.

The lower uncertainty in the one-meter scenario when the ratio is calculated is due to 'cancelation' of the uncertainty contributions from gain and offset errors in the indi-

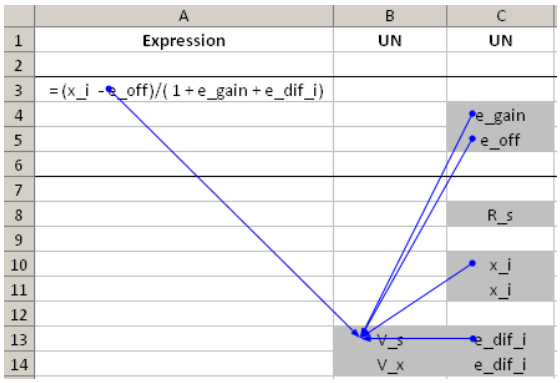

Figure 6: The calculation of an uncertain number for the voltage estimate in B13 depends on the four uncertain numbers defined in cells C4, C5, C10 and C13. The expression that is evaluated is in A3.

vidual voltage measurements. `SGUM` performs a complete sensitivity calculation, which obtains very low sensitivities for these influences because the errors are common influences (systematic). However, when the errors are independent (the two-meter case) this cancelation no longer occurs.

4.2 `GTC` calculation

A sequence of `GTC` instructions that calculate $\widehat{R}_x$ and $u\left(\widehat{R}_x\right)$ is shown in Listing 3. The instructions handle the one-meter case, but an extension to the two-meter measurement is explained below. Numerical data are associated with names in lines 1–13, then lines 16-21 define the uncertain numbers used in the calculation.[4]

The listing refers to an *ad hoc* class of objects called `Meter` that implements the voltmeter error model. A class is an object-oriented programming construct that allows specific behaviour to be modeled. It defines an association between a set of functions and a set of data that is specific to each class instance. In this case, `Meter` object data includes the two uncertain number instances associated with systematic errors in the meter model (the details are discussed below).

To adapt Listing 3 to the two-meter measurement, we would add a second `Meter` definition at line 17

```
B = Meter(u_e_gain,u_e_off,u_e_dif,'B')
```

This creates an independent set of uncertain numbers associated with the second meter's errors. Subsequently, line 19 would be changed to use meter `B` to obtain an uncertain number representing the second voltage measurement, i.e.,

```
V_x = B.voltage(x_x,'x')
```

No other changes are needed.

The class definition for `Meter` is shown in Listing 4. Each `Meter` object has a unique set of data (`self.u_e_dif`, `self.e_gain` and `self.e_off`) and two functions (`__init__` and `voltage`). The `__init__` function is called automatically by Python to initialise new instances. It creates uncertain numbers associated with estimates of the systematic (fixed) gain and offset errors (lines 6 and 9) and stores the standard uncertainty of the

[4]The `label` argument to `ureal` provides a string to identify different influences when the uncertainty budget is displayed.

	A	B	C	D	E	F
1	Expression	UN	UN	x	u	Comment
2						
3	=(x_i - e_off)/(1+e_gain+e_dif_i)					voltmeter error model A
4			e_gain		3.00E-06	uncertainty in gain is a few ppm
5			e_off		1.00E-06	uncertainty in offset is about 1 microvolt
6					1.00E-07	uncertainty in differential linearity
7						
8	=(x_i - e_off)/(1+e_gain+e_dif_i)					voltmeter error model B
9			e_gain		3.00E-06	uncertainty in gain is a few ppm
10			e_off		1.00E-06	uncertainty in offset is about 1 microvolt
11					1.00E-07	uncertainty in differential linearity
12						
13			R_s	1000	1.00E-03	standard 1000-ohm resistor
14						
15			x_i	5.01		meter indication for V_s (meter A)
16			x_i	4.9885		meter indication for V_x (meter B)
17						
18		V_s	e_dif_i			
19		V_x	e_dif_i			
20						
21	R_x = R_s * (V_x/V_s)	R_x				
22						
23	Results					
24						
25	value	995.7085828				
26	uncertainty	0.004351603				
27						
28	components of uncertainty:					
29	e_gain (meter A)	2.9871E-03				
30	e_gain (meter B)	2.9871E-03				
31	R_s	9.9571E-04				
32	e_off (meter A)	1.9874E-04				
33	e_off (meter B)	1.9960E-04				
34	e_dif_i (for V_s)	9.9571E-05				
35	e_dif_i (for V_x)	9.9571E-05				

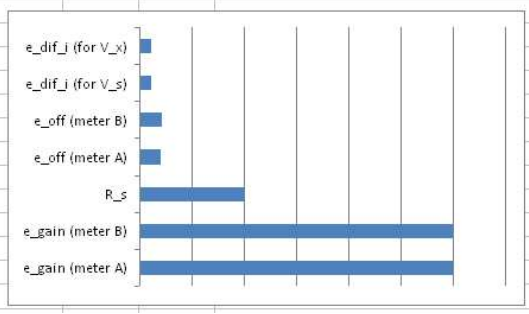

Figure 7: When two meters are used, the worksheet to calculate the value of a resistance R_x must define uncertain numbers associated with each meters' errors.

differential linearity estimate in `self.u_e_dif`, which will be used to create uncertain numbers associated with differential linearity errors in the function `voltage`.

The function `voltage` returns an intermediate uncertain number for a voltage measurement, as defined by equation (12). Each time `voltage` is called (e.g., lines 18-19 of Listing 3), the uncertain numbers associated for the systematic gain and offset errors estimates are combined with a new elementary uncertain number representing $\widehat{e}_{\text{dif·i}}$. Creating a new uncertain number for $\widehat{e}_{\text{dif·i}}$ each time ensures that $e_{\text{dif·i}}$ is treated as an independent random error.

The `Meter` class illustrates of how the lifetime of uncertain-number objects can be managed in `GTC`. In `SGUM`, there was a one-to-one association between uncertain numbers and cells. So, a systematic error is associated with a single spreadsheet cell and referenced multiple times. However, in bigger problems spreadsheet layouts can become unwieldy. The object classes in `GTC`, with specific member data and functionality, are better able to handle this complexity.

5 Example 3: a pressure measurement

We consider an instrument for measuring gas pressure in this section. The system has a transducer, that produces a voltage in response to an applied pressure, and a meter that displays the magnitude of the response. Adopting a simple model for each component, it is assumed that the transducer response is proportional to pressure and that the meter's accuracy is limited by the finite resolution of a digital display. The system

```
# meter parameters
u_e_gain = 3E-6
u_e_off = 1E-6
u_e_dif = 1E-7

# standard 1k-ohm resistor
x_R_s = 1000
u_R_s = 1E-3
R_s = ureal(x_R_s,u_R_s,label='R_s')

# meter indications
x_s = 5.01
x_x = 4.9885

# meter instances
A = Meter(u_e_gain,u_e_off,u_e_dif,'A')

V_s = A.voltage(x_s,'s')
V_x = A.voltage(x_x,'x')

R_x = R_s * V_x/V_s

# Display results
print value(R_x)
print uncertainty(R_x)

print """
uncertainty budget:"""

for cpt in reporting.budget(R_x,trim=0):
    print " %s: %G" % cpt
```

Listing 3: Calculation of $\widehat{R}_{\mathrm{x}}$ and $u\left(\widehat{R}_{\mathrm{x}}\right)$ using `GTC`. See §A.1 for an explanation of how the uncertain budget is obtained in the last lines.

can be calibrated by taking readings at different pressures from a reference pressure generator.

A model of the transducer output as a function of pressure p_i is

$$v_{\mathrm{t}\cdot i} = \frac{p_i - a + e_{\mathrm{rep}\cdot i}}{b} \ , \tag{14}$$

where a and b are the unknown parameters of the linear transducer response and $e_{\mathrm{rep}\cdot i}$ represents the variability in transducer output from one measurement to the next. A model of the meter indication is

$$x_i = v_{\mathrm{t}\cdot i} + e_{\mathrm{res}\cdot i} \ , \tag{15}$$

where x_i differs from the transducer output because of a finite number of digits displayed. The error in the i^{th} reading is $e_{\mathrm{res}\cdot i}$.

A detailed calibration report provides estimates of the intercept and slope, $\widehat{a}$ and $\widehat{b}$, of a linear calibration function that converts a meter indication x into an estimate of pressure. Standard uncertainties $u(\widehat{a})$ and $u(\widehat{b})$ are reported, as is the correlation coefficient r between $\widehat{a}$ and $\widehat{b}$. Also reported is the uncertainty associated with repeatability and the uncertainty associated with systematic error in the calibration itself. The repeatability uncertainty is given in pressure units, while the calibration uncertainty is relative to the indicated pressure.

```
1  class Meter(object):
2      def __init__(self,u_e_gain,u_e_off,u_e_dif,tag):
3          self.u_e_dif = u_e_dif
4
5          label = 'e_gain_%s' % tag
6          self.e_gain = ureal(0,u_e_gain,label=label)
7
8          label = 'e_off_%s' % tag
9          self.e_off = ureal(0,u_e_off,label=label)
10
11     def voltage(self,x,tag):
12         label = 'e_diff_%s' % tag
13         e_dif = ureal(0,u_e_dif,label=label)
14         return (x - self.e_off)/(1 + self.e_gain + e_dif)
```

Listing 4: The `Meter` class definition. `__init__` is called when a new `Meter` object is created. `voltage` is called to generate an uncertain number representing a measurement.

Equations (14) and (15) can be used to estimate p_i, in two steps[5]

$$\widehat{v}_{\mathrm{t}\cdot i} = x_i - \widehat{e}_{\mathrm{res}\cdot i} \tag{16}$$

$$\widehat{p}_i = \frac{\widehat{b}\widehat{v}_{\mathrm{t}\cdot i} + \widehat{a} - \widehat{e}_{\mathrm{rep}\cdot i}}{1 - \widehat{e}_{\mathrm{cal}}} \; . \tag{17}$$

These equations can be used to develop data processing. Estimates of the residual errors $\widehat{e}_{\mathrm{cal}}$, $\widehat{e}_{\mathrm{rep}\cdot i}$ and $\widehat{e}_{\mathrm{res}\cdot i}$ are all zero, corresponding to an ideal system.

5.1 `GTC` calculation

In this section, as in section 4.2, classes of objects are associated with components of the measurement system. There is a `Transducer` class, representing the pressure-to-voltage transducer, and a `Meter` class, representing the display unit. We will consider the calculation of a pressure ratio from measurements of two different pressures.

Listing 5 shows a sequence of instructions that calculates the pressure ratio. Lines 1 to 15 collect information about system uncertainties and the calibration equation. Lines 17 and 19 define objects representing the meter and transducer. Data processing is done in lines 21 to 29 and results are displayed by calling the function `results` in lines 31 to 33.

The inputs to the calculation are: two readings, $x_1 = 0.956$ and $x_2 = 1.1123$, the calibration uncertainty, reported as 30 ppm, the repeatability uncertainty reported as 3 Pa and the parameter estimates for the calibration equation. The intercept a has been estimated as 0.9152 kPa, with an uncertainty 0.1165 kPa, and the slope b has been estimated as 99.9634 kPa/V, with an uncertainty 0.0559 kPa/V. The correlation coefficient between the estimates of a and b is $r = -0.84$.

The results for the first pressure measurement, printed out by `GTC`, are

```
p=96.4802, u=0.077424

Components of uncertainty:
 a_t1: 0.1165
```

[5]Note the denominator in (17) accounts for an additional calibration error term that is not in the models. The calibration uncertainty was reported because a small difference may exist between the pressure reference values used for calibration and the actual pressures.

```
 b_t1: 0.0534404
 e_rep_1: 0.003
 e_cal: 0.00289441
 e_res_1: 0.00288569
```

In this format the pressure estimate and standard uncertainty are reported in units of kPa, followed by a list of the more important components of uncertainty in decreasing order of magnitude (units are kPa). The dominant contribution to the uncertainty budget is the uncertainty of $\widehat{a}$, the offset term in the calibration equation.[6]

For the second pressure measurement, the results are

```
p=112.104, u=0.0727842

Components of uncertainty:
 a_t1: 0.1165
 b_t1: 0.0621776
 e_cal: 0.00336313
 e_rep_2: 0.003
 e_res_2: 0.00288569
```

Note that, because this pressure is higher, the influence of the calibration uncertainty has become more important than the uncertainty due to repeatability.

The results for the pressure ratio are

```
p=0.860628, u=0.000153405

Components of uncertainty:
 a_t1: 0.000144837
 e_rep_1: 2.67607E-05
 e_res_1: 2.57411E-05
 e_rep_2: 2.3031E-05
 e_res_2: 2.21535E-05
 b_t1: 6.36269E-07
 e_cal: 3.38813E-21
```

Note that the slope no longer makes a significant contribution to the uncertainty budget of the ratio and the calibration uncertainty component is effectively zero.[7] The sensitivity to those errors is reduced because they are common to both measurements.

If different transducers are used to measure p_1 and p_2, the data processing can be modified simply by adding the following at line 20[8]

```
t2 = Transducer(tag='t2',**t_args)
```

and changing line 29 to use `t2` instead of `t1`. The pressure ratio results are now

```
p=0.860627, u=0.000892498

Components of uncertainty:
 a_t1: 0.00103913
 a_t2: 0.000894304
 b_t2: 0.000477145
```

[6] That this component of uncertainty can actually exceed the combined standard uncertainty of the pressure is due to a strong correlation coefficient between $\widehat{a}$ and $\widehat{b}$.

[7] The ratio of two pressure measurements described by (17) does not depend on the common calibration error e_{cal} at all. So the uncertainty component due to calibration error is exactly zero. The value of `3.38813E-21` obtained here is due to the finite accuracy of the floating-point number representation used in numerical calculations.

[8] For simplicity, exactly the same calibration data are used here for the second transducer. In practice, different numbers would be used to initialise `t2`, which is a different transducer.

```
# System parameters
resolution = 1E-4 # microvolt
u_cal = 3E-5 # kPa / kPa

e_cal = ureal(0,u_cal,label='e_cal')

t_args = dict(
    e_cal = e_cal,
    u_rep = 0.003, # kPa
    a_x = 0.9152, # kPa
    a_u = 0.1165, # kPa
    b_x = 99.9634, # kPa / V
    b_u = 0.0559, # kPa / V
    r = -0.84
)

m = Meter(resolution)

t1 = Transducer(tag='t1',**t_args)

# Data processing
x1 = 0.956
x2 = 1.1123

v1 = m.voltage(1,x1)
v2 = m.voltage(2,x2)

p1 = t1.pressure(1,v1)
p2 = t1.pressure(2,v2)

results(p1)
results(p2)
results(p1/p2)
```

Listing 5: Data processing for two pressure measurements and the calculation of the pressure ratio.

```
b_t1: 0.000476509
e_rep_1: 2.67608E-05
e_res_1: 2.57411E-05
e_rep_2: 2.3031E-05
e_res_2: 2.21535E-05
e_cal: 3.38813E-21
```

The standard uncertainty of the ratio has increased approximately six-fold, because the transducer errors are now independent, so the cancelation that occurred in the first scenario does not occur here. Nevertheless, the calibration error is common to both transducers (assuming it is an error arising from calibrations at the same laboratory) and therefore the associated uncertainty component is zero.

Listing 6 shows the `Meter` class definition. This models the behaviour associated with the finite number of digits in the display, following equation (16). Line 3 calculates the standard uncertainty assuming a uniform distribution of full-width `resolution` and saves this number for later use. Each time `voltage` is called, an elementary uncertain number is created and subtracted from the indication, representing this independent source of error.

Listing 7 shows the definition of the `Transducer` class, which models the transducer

```
class Meter(object):
    def __init__(self,resolution):
        self.u_res = resolution / math.sqrt(12)

    def voltage(self,tag,x):
        label = "e_res_%s" % tag
        e_res = ureal(0,self.u_res,label=label)

        return x - e_res
```

Listing 6: The class definition of the meter display.

behaviour. `Transducer` objects store data relating to the calibration curve and the calibration and transducer repeatability uncertainties. During initialization, elementary uncertain numbers are defined in lines 5 and 8, representing $\widehat{a}$ and $\widehat{b}$. The correlation coefficient between them is registered in line 10.

The function `pressure`, based on (17), returns an uncertain number associated with an estimate of the pressure. The argument `v` will be obtained from the `voltage` function of the `Meter` class (e.g., Listing 5 lines 25–29). The repeatability error in each measurement is independent, so, in line 19, a new elementary uncertain number is created each time `pressure` is called. An intermediate uncertain number for the pressure estimate is then obtained following equation (17).

```
class Transducer(object):
    def __init__(self,tag,a_x,a_u,b_x,b_u,r,e_cal,u_rep):

        label = "a_%s" % tag
        self.a = ureal(a_x,a_u,label=label)

        label = "b_%s" % tag
        self.b = ureal(b_x,b_u,label=label)

        set_correlation(r,self.a,self.b)

        self.e_cal = e_cal

        self.u_rep = u_rep
        self.tag = tag

    def pressure(self,tag,v):
        label = "e_rep_%s" % tag
        e_rep = ureal(0,self.u_rep,label=label)

        p = (v*self.b + self.a - e_rep)/(1 - self.e_cal)

        return p
```

Listing 7: The class definition of the transducer.

6 Discussion

Uncertain numbers are an example of *computational thinking* [8]. By presenting an abstract representation, they are able to emphasize a more intuitive computational model that can be used to solve many types of problem. One of the obvious advantages is that translation of model equations into software will be easier, because the natural step-by-step data processing often used to evaluate a measurement result is closely related

to the equations used in uncertain-number calculations. For instance, in the resistance-ratio measurement, it is natural to think in terms of the two voltage measurements (12) and then the calculation of resistance (13). A single equation for resistance, like

$$R_{\mathrm{x}} = R_{\mathrm{s}} \cdot \left[\frac{x_{\mathrm{x}} - e_{\mathrm{off}\cdot\mathrm{x}}}{x_{\mathrm{s}} - e_{\mathrm{off}\cdot\mathrm{s}}}\right] \cdot \left[\frac{1 + e_{\mathrm{gain}\cdot\mathrm{s}} + e_{\mathrm{dif}\cdot\mathrm{s}}}{1 + e_{\mathrm{gain}\cdot\mathrm{x}} + e_{\mathrm{dif}\cdot\mathrm{x}}}\right]$$

is less likely to relate closely to the metrologist's concept of the procedure.

Multi-step data processing and the possibility of translating mathematical models directly into uncertain-number software sets the stage for reuse of components in different problems. When decomposing complicated problems into simpler ones, some parts are likely to reoccur in different situations. There will be recurring data processing algorithms, as well as common components in measurement systems. The software associated with these could be accumulated in libraries of reusable code, which would simplify the development of subsequent data processing. Libraries would also ease the burden of testing and validation, because the library code could be subjected to more careful quality control.

One area that could benefit from the new approach is modern measurement systems, where 'plug-and-play' of modular components is popular. In that context, a collection of re-usable data processing elements might be very useful. Uncertain numbers would allow the problems of complexity in uncertainty calculations faced by the Test and Measurement industry to be managed [9], because the propagation of uncertainty components across component interfaces can be standardized and automated [10]. Hypothetically, a manufacturer could provide data processing for components without knowing the configuration of a client's system. Similarly, a client could assemble a measurement system from a range of different components with built-in uncertainty propagation, without detailed information about their internal operation, as long as the interfaces were standardised.

Another example of system complexity is the complex-valued impedance measurements performed at radio and microwave frequencies. Here, data processing must deal with a large number of influences, many of which are systematic errors. Typically, up to twelve complex-valued parameters are needed to correct for imperfect system hardware. Estimates of these parameters are obtained in an initial adjustment phase of the measurement, but the residual errors still make a significant contribution to the uncertainty of results. In addition, the imperfect cables and connectors that link the measuring instrument to the device under test contribute errors that endure for parts of a procedure and then change, when a cable is moved or a new connection is made. Furthermore, the metrologist is at liberty to operate the system in slightly different ways, thereby changing the specific influences. The flexible nature of uncertain-number algorithms is ideally suited to such problems [11].

A particularly interesting aspect of uncertain-number processing is that there is no final step: a result, as perceived by one observer, may be another's input. This is true of calibration laboratories, whose clients use the artifacts and instruments that have been sent to the laboratory for calibration. The pressure measurement example in §5 is a case in point. We showed that the uncertainty of a client's pressure ratio measurements can be reduced by linking the detailed results of a calibration report to the client's data processing. Currently, however, most calibration laboratories would not report so much information, because their clients are unlikely to use it. Instead, reports are likely to be simplified, even if doing so may limit the accuracy of subsequent measurements.

The task of calculating uncertainty at the end of a chain of traceable measurements can be a burden, because it is often difficult to construct an adequate measurement model

for the uncertainty analysis. The multi-step approach, however, allows the workload to be shared. Detailed information could be provided at every stage of the chain and incorporated in subsequent calculations. A standardized way of communicating detailed information about measurement results and their uncertainty components would need to be worked out, but software like `GTC` offers a glimpse of what could be achieved. Measurement results could be provided to a client in electronic form allowing the client to add the data processing associated with their own measurement procedure. For example, in pressure measurement of §5, the client's part of the pressure-ratio data-processing corresponds only to lines 21–33 of Listing 5. If the calibration laboratory could supply the rest of the data processing with the report, the client's task of evaluating the uncertainty of the pressure ratio becomes much easier.

7 Conclusions

Uncertain-number software can fundamentally change our approach to measurement data processing. Compared to the single measurement equation that is conventionally used in GUM uncertainty analysis, the multi-step approach supported by uncertain numbers offers a number of advantages. It allows complexity to be managed effectively, for complicated measurement procedures as well as instrumentation systems, and simplifies the treatment of systematic errors. The approach is likely to be easier to formulate than a single model equation, because data-processing steps are often closely related to identifiable elements of a measurement system or to actions in a measurement procedure. This means, in turn that uncertain-number data processing will be easier to maintain when changes to the measurement procedure occur.

Uncertain-number software offers increased support for traceable measurements. In particular, it could be used effectively to transfer detailed information along a chain of traceable measurement results, in keeping with one of the fundamental principles of the GUM. For calibration laboratories, detailed calibration results could be combined with software to deliver better support for traceability to a client. Equipment manufacturers, national metrology institutes and calibration laboratories could all contribute to the development of measurement models according to their respective roles in the measurement process and these models could be disseminated as software.

Acknowledgement

This article is dedicated to our colleague M. Fitzgerald, who passed away during the preparation of this work. His advice and comments about the examples, together with contributions from J. Lovell-Smith and M. Early, are gratefully acknowledged. The author would like to thank the referees for helpful remarks.

This work was funded by the New Zealand Government as part of a contract for the provision of national measurement standards.

References

References

[1] BIPM, IEC, IFCC, ISO, IUPAC, IUPAP, and OIML. *Evaluation of measurement data Guide to the expression of uncertainty in measurement JCGM 100:2008 (GUM 1995 with minor corrections)*. BIPM Joint Committee for Guides in Metrology, Paris, Sèvres, 1 edition, 2008.

[2] B. D. Hall. Computing uncertainty with uncertain numbers. *Metrologia*, 43:L56–L61, 2006.

[3] L B Rall. The arithmetic of differentiation. *Math. Mag.*, 59:275, 1986.

[4] The `GTC` and `SGUM` software is available without charge at `mst.irl.cri.nz`. There is also an uncertain-number module for the R data processing language.

[5] The official Python internet site is `www.python.org`.

[6] B. D. Hall. The GUM Tree design pattern for uncertainty software. In *Advanced Mathematical and Computational Tools in Metrology VI*, volume 66 of *Series on Advances in Mathematics for Applied Sciences*, pages 199–208. World Scientific, 2004.

[7] B. D. Hall. Propagating uncertainty in instrumentation systems. *IEEE Trans. Instrum. Meas.*, 54:2376–2380, 2005.

[8] J. M. Wing. Computational thinking and thinking about computing. *Philosphical Transactions of the Royal Society A*, 366:3717–3725, 2008.

[9] B Anderson. The New Economy: What Role Will Metrology Play. `http://www.agilent.com/metrology/pdf/NCSLI2001_keynote_transcript.pdf`, 2001. Keynote speech to 40$^{\text{th}}$ NCSL International Symposium, Washington, DC.

[10] B. D. Hall. Component interfaces that support measurement uncertainty. *Computer Standards and Interfaces*, 28:306–310, 2006.

[11] M Wollensack, J Hoffmann, J Ruefenacht, and M Zeier. VNA Tools II: S-parameter uncertainty calculation. In *Microwave Measurement Conference, 79$^{\text{th}}$ ARFTG*, 22 June 2012.

A Some elements of Python

This appendix briefly explains some of programming constructs used in this article. The reader may also be interested in a lot of introductory material about Python that is available on the internet [5].

A.1 Iteration

Python has a simple `for` statement that iterates over the elements of a sequential data structure. For example, the following will print odd numbers from 1 to 9

```
1 for i in (1,3,5,7,9):
2     print i
```

Note that the line following the `for` statement is indented. This is the way that Python delimits a block of instructions. In this case, all intended lines following the `for` statement form a block and are executed in sequence for each iteration.

Iteration was used at the end of listing 3 to display the uncertainty budget. The code was

```
1 for cpt in reporting.budget(R_x,trim=0):
2     print " %s: %G" % cpt
```

In this case a sequence is created by calling the function `reporting.budget(R_x,trim=0)`. The elements of this sequence are assigned successively to `cpt`. Each `cpt` is actually a string paired with the numerical value of the uncertainty component. The string identifies the elementary uncertain number associated with the component of uncertainty. The format string `"%s: %G"` inserts these values into one line of the uncertainty budget.

A.2 String formatting

String formatting was used in listing 3 to display the uncertainty budget in line 31. It is also used in `Meter` classes in listings 4 and 6 to label elementary uncertain numbers.

In Python it is possible to format strings of characters using a compact notation. This uses a 'format' string as a template with format specifiers for one or more 'values'. For example, to create and display the string `'pi = 3.141'` we could write

```
1 a = 3.141
2 n = 'pi'
3 print "%s = %G" % (n,a)
```

Here the format string is `"%s = %G"` and the values are `n,a`. The string format operator is the percentage symbol `%` between the format string and the sequence of values.

Format specifiers are replaced by the values of arguments on the right. These specifiers begin with a `%` symbol and are followed by conversion codes. The conversion specifier `%s`, for example, inserts a string value. So, the string '`pi`' is inserted to the left of the '`=`' sign. The conversion specifier `%G` generates a string representation of a floating point number, here the value of `a`.

A.3 Functions

Functions are defined using a very simple syntax. Listing 8 shows the definition of the `results` function used in listing 5. Note again the indentation of lines following the

```
def results(x):
    print
    print "p=%G, u=%G" % (x.x,x.u)
    print "\nComponents of uncertainty:"
    for cpt in rp.budget(x,trim=0):
        print " %s: %G" % cpt
```

Listing 8: The definition of `results`.

the `def` statement, which identifies the block of instructions comprising the function definition. Function arguments (here just `x`) are passed as a sequence following the function name. A `return` statement is used to send back a result to the calling context (nothing is returned by `results`).

A.4 Classes

There is a class mechanism in Python that supports the standard features of object-oriented programming. Class definitions describe software entities with common behaviour and individual data sets. Used in combination with uncertain numbers, classes are a very effective tool for modeling aspects of complicated measurement procedures.

In this paper, different types of `Meter` classes were defined (listings 4 and 6). In so doing, the model associated with each type of meter can be developed independently of the data processing in which objects of the `Meter` class are used.

The behaviour of a class of objects is determined by the functions (or methods) in the class definition. At the same time, each object of a particular class can have an independent set of data members.

In listing 6, shown again here, there are two functions: `__init__` and `voltage`, and one data member: `self.u_res`. Note that there is always an initial argument `self` in functions belonging to a class. When the functions are called, Python automatically supplies the `self` argument, which contains the object's data members.

```
class Meter(object):
    def __init__(self,resolution):
        self.u_res = resolution / math.sqrt(12)

    def voltage(self,tag,x):
        label = "e_res_%s" % tag
        e_res = ureal(0,self.u_res,label=label)

        return x - e_res
```

The name of the class is used to create new objects, for example `m = Meter(resolution)`. However, during the process of object creation the `__init__` function is called with the same arguments, to initialize a new object. For example, `resolution` will be passed as an argument to `__init__` when `m = Meter(resolution)` is executed and used to define the data member `self.u_res`. The `voltage` function uses `self.u_res` to create new elementary uncertain numbers for the estimated resolution error of each reading.