The GUM Tree Calculator: A Python Package for Measurement Modelling and Data Processing with Automatic Evaluation of Uncertainty

Blair D. Hall

Abstract: There is currently interest in the digitalisation of metrology because technologies that can measure, analyse, and make critical decisions autonomously are beginning to emerge. The notions of metrological traceability and measurement uncertainty should be supported, following the recommendations in the Guide to the Expression of Uncertainty in Measurement (GUM). However, GUM offers no specific guidance. Here, we report on a Python package that implements algorithmic data processing using ‘uncertain numbers’, which satisfy the general criteria in GUM for an ideal format to express uncertainty. An uncertain number can represent a physical quantity that has not been determined exactly. Using uncertain numbers, measurement models can be expressed clearly and succinctly in terms of the quantities involved. The algorithms and simple data structures we use provide an example of how metrological traceability can be supported in digital systems. In particular, uncertain numbers provide a format to capture and propagate detailed information about quantities that influence a measurement along the various stages of a traceability chain. More detailed information about influence quantities can be exploited to extract more value from results for users at the end of a traceability chain.

Keywords: measurement uncertainty; guide to the expression of uncertainty in measurement; measurement modelling; uncertainty propagation; metrological traceability; uncertain number

1. Introduction

The worldwide dissemination of Système International (SI) units is a person-oriented paper-based process that is carefully managed by national bodies and coordinated by international organisations; however, that is about to change. The emergence of technologies that can measure and make critical decisions autonomously requires more of our measurement infrastructure to be implemented by digital systems. A growing number of initiatives are replacing paper-based, expert-oriented processes with automated digital ones (e.g., machine-readable formats for calibration reports [1] and a secure cloud-based platform for the legal metrology infrastructure in Europe [2]). Dissemination of SI provides what the metrology community calls traceability. Metrological traceability ensures that measurements are accompanied by information that can be used to determine the fitness-for-purpose of results in different situations. Traceability may be thought of as support for interoperability with measurement data but at present the expertise of skilled individuals is needed to interpret data and supporting information correctly. One outcome of digitalisation will be an ability to produce traceable measurement results in machine-actionable formats.

Measurement accuracy is fundamental to traceability. Traceable measurements must report information about the likely magnitude of the difference (error) between a measured value and the quantity intended to be measured. Metrologists refer to this as measurement uncertainty. During the 1980s, considerable effort went into harmonising the manner that measurement uncertainty is evaluated and communicated and this resulted in the
publication of the *Guide to the Expression of Uncertainty in Measurement* (GUM) [3], which remains today the primary reference for dealing with measurement uncertainty (the GUM was produced by a group of experts representing eight international scientific and technical organisations: the BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML).

Digitalisation of metrological infrastructure will inevitably need algorithmic implementations for GUM methods. However, the GUM itself offers no specific guidance, as it was written long before this need could have been anticipated. One approach, called uncertain numbers, provides an abstract representation for physical quantities that satisfies general criteria in the GUM for an ideal format to express uncertainty [4]. Quantities in a problem, such as specific lengths, masses, etc., are always considered to have definite values that can only be estimated with limited accuracy by measurement. Therefore, an uncertain number is designed to encapsulate information about the measured value and measurement accuracy. Using this abstraction allows data processing to be expressed algorithmically in terms of the quantities involved, leaving the associated uncertainty calculations to be handled automatically in the background. The uncertain-number approach can capture the effects of influence quantities at different stages of a traceable measurement and propagate this information along the chain to an end user. Detailed information about the uncertainty budget can sometimes be used to enhance the value of results. This report describes a Python package, called the *GUM Tree Calculator* (GTC), that uses uncertain numbers [5].

GTC is a very flexible tool that has been used in two quite different applications, which have been reported on recently [6,7]. We discuss the tool’s design and comment on aspects that support traceability in digital systems.

The next section provides an overview of GUM uncertainty calculations, metrological traceability, and GTC software. GTC is tested on a variety of platforms and different versions of Python 2 and 3 (refer to the github repository [5]). All code snippets use GTC version 1.3.6. The GUM presents its approach to evaluating measurement uncertainty in mathematical terms; thus, Section 2.1 summarises the key equations. However, an alternative formulation of these equations is more practical. This is described in Section 2.2, which motivates the development of the uncertain-number data type. The notion of metrological traceability is discussed in Section 2.3, and we explain why the uncertain-number format is useful to support traceability. Section 2.4 presents an example of GTC data processing applied to an electrical circuit. Section 3 looks in more detail at aspects of GTC design. The method used to automate uncertainty calculation is presented in Section 3.1. The data structures that support uncertainty propagation in uncertain-number objects are described in Sections 3.2–3.5; then, Section 3.6 describes how uncertain numbers can be saved and restored. More general considerations are discussed in Section 4. Appendix A briefly describes support for complex quantities, extensions for handling degrees of freedom, and some additional implementation details.

**Notation**

Upper and lower case letters are used to distinguish between quantities, which will never be determined exactly, and values that will be known (such as a numerical indication on a measuring instrument), respectively. For example, a measurement result \( y \) is written in lower-case because a result always has a definite numerical value: The value \( y \) is an estimate of \( Y \), which is the quantity intended to be measured. Upper-case is used for \( Y \) to indicate that the quantity cannot be known exactly. It is helpful to make this distinction because uncertain-number objects are used to represent quantities; the associated numerical estimates and other known values, such as uncertainty, will appear as attributes of uncertain numbers.

**2. Overview**

**2.1. GUM Method**

Measurements are always influenced by unpredictable factors, so a result can only ever approximate the quantity of interest. Influence factors can, however, be identified and
described in probabilistic terms. In this manner, a measurement process can be represented by a mathematical model. This is the approach taken in the GUM.

The first step is to identify a function called the measurement model that contains every quantity, including all corrections and correction factors that can contribute a significant component of uncertainty to the measurement result [3] [Section 4.1.2]. In the GUM, this is expressed as an explicit function:

\[ Y = f(X_1, X_2, \cdots, X_l), \]  

where \( Y \) is the quantity intended to be measured (the measurand) and the input arguments \( X_1, X_2, \cdots, X_l \) are quantities that influence the measurement outcome. All arguments of \( f(\cdots) \) are treated in the same way when evaluating uncertainty. Some of the input terms may represent other measured quantities. For example, electrical resistance could be measured by first measuring potential difference \( V \) and current \( I \) and then evaluating ratio \( R = V / I \). Other input terms may represent nuisance factors that perturb the measurement, such as Johnson noise in a resistor. The measurement model can be thought of as a recipe for evaluating the measurand: if \( X_1, X_2, \cdots, X_l \) were all known exactly, then \( Y \) could be determined. However, with only approximate values available for input quantities, an approximate value for \( Y \) will be obtained.

\[ y = f(x_1, x_2, \cdots, x_l). \]  

GUM uses a specific term, standard uncertainty, in relation to the unpredictability of measurement outcomes (i.e., the fact that \( y \) and \( Y \) differ by an unpredictable amount). A standard uncertainty is an estimate of the standard deviation of a probability distribution for the difference (error) between \( y \) and \( Y \). There is a formula in the GUM to propagate standard uncertainties through a measurement model and obtain the standard uncertainty in a result. For a model in the form of Equation (1), the standard uncertainty of \( y \), as an estimate of \( Y \), is calculated as follows:

\[ u(y) = \left[ \sum_{i=1}^{l} \sum_{j=1}^{l} u_i(y) r(x_i, x_j) u_j(y) \right]^{1/2}, \]  

where \( u_i(y) \) and \( u_j(y) \) are components of uncertainty that relate small changes in input values to corresponding changes in \( y \).

\[ u_i(y) = \frac{\partial Y}{\partial x_i} \bigg|_{x_i=x_i} u(x_i), \quad \text{and} \quad u_j(y) = \frac{\partial Y}{\partial x_j} \bigg|_{x_j=x_j} u(x_j). \]  

The terms \( u(x_i) \) and \( u(x_j) \) are the standard uncertainties in the input values \( x_i \) and \( x_j \), respectively. The correlation coefficient attributed to a pair of input estimates is \( r(x_i, x_j) \), where \( r(x_i, x_j) = 1 \) when \( i = j \).

Standard uncertainties are associated with a number called the degrees of freedom, usually denoted \( v \). The interpretation given to \( v \), \( u(x) \), and \( v \) in the GUM is analogous to familiar sample statistics: the sample mean, the standard error in the sample mean, and the degrees of freedom [8] [Chapter 9]. However, degrees of freedom are interpreted more broadly in the GUM, because uncertainty evaluation is not always based on a sample of data. When a standard uncertainty is considered to be known very accurately, the degrees of freedom is large (up to infinity), but a small number of degrees of freedom (as low as unity) signifies a very rough estimate of the underlying standard deviation. Again, the GUM provides an equation for propagating degrees of freedom, called the Welch–Satterthwaite formula. The number of degrees of freedom associated with a standard uncertainty \( u(y) \) is as follows.

\[ v_y = \frac{u^4(y)}{\sum_{i=1}^{l} \frac{u_i^4(y)}{v_i}}. \]
However, there is an important restriction on the use of this equation. It is not valid when input estimates that have finite degrees of freedom are correlated with each other. This is not a rare occurrence. For example, estimates obtained from linear least-squares regressions are often correlated and have finite degrees of freedom. Fortunately, an extended form of (5) can be used in some important special cases (see Appendix A.1) [9].

Equations (1)–(5) describe a methodology for evaluating measurement uncertainty that any GUM-compliant data processing should adhere to. However, in many situations, it is inconvenient, if not impossible, to formulate a single, complete, measurement model such as Equation (1). Usually, a traceable measurement is perceived as a staged process, and it is difficult to describe when approached at more than one stage at a time. However, there is a mathematically equivalent formulation of these calculations that allows staged models to be handled. This formulation leads to a new abstract data type called an uncertain number, which can represent inexactly known quantities [4]. The uncertain-number format satisfies the requirements for information exchange identified in the GUM [3] [Section 0.4].

2.2. The Uncertain-Number Methodology

A mathematical expression may often be decomposed into stages and evaluated algorithmically as a sequence of basic operations. For instance, the following is the case:

\[ V = v(1 - E_{\text{rel}}) - E_{\text{off}} - E_{\text{rnd}} \]

and can be broken into four stages (also shown in Figure 1):

\[ \begin{align*}
    y_1 &= 1 - E_{\text{rel}} \\
    y_2 &= v \times y_1 \\
    y_3 &= y_2 - E_{\text{off}} \\
    V &= y_4 = y_3 - E_{\text{rnd}}.
\end{align*} \]

![Figure 1. Decomposition of \( V = v(1 - E_{\text{rel}}) - E_{\text{off}} - E_{\text{rnd}} \) into a sequence of arithmetic operations.](image)

This approach can be applied to measurement models. The evaluation of some arbitrary function \( f(x_1, \cdots, x_l) \) can be decomposed into \( h = 1, \cdots, m \) stages, each producing an intermediate result:

\[ y_h = f_h(\Lambda_h), \]

with the final stage yielding the result \( y = y_m \). The set of inputs to the \( h \)th stage function, denoted here as \( \Lambda_h \), may include previous stage results \( y_1, \cdots, y_{h-1} \) and model inputs \( x_1, \cdots, x_l \). Using the chain rule for partial differentiation, the components of uncertainty
defined in Equation (4) can be evaluated at each stage. The component of uncertainty in $y_h$ due to uncertainty in the $j$th model input is as follows.

$$u_j(y_h) = \sum_{z_k \in \Lambda_h} \frac{\partial f_h}{\partial z_k} u_j(z_k).$$ (7)

Thus, the set of components of uncertainty $\{u_1(y_h), u_2(y_h), \cdots, u_l(y_h)\}$, corresponding to $\{x_1, x_2, \cdots, x_l\}$, can be evaluated stage-by-stage to finally obtain the set of components of uncertainty in the result, $\{u_1(y), u_2(y), \cdots, u_l(y)\}$ (when $j = k$, the notation $u_j(z_j)$ may be simplified to $u(z_j)$, which is the standard uncertainty of model input $x_j$).

In GTC, an uncertain number is used to encapsulate results at each stage ($y_h$, and the associated components of uncertainty, $\{u_1(y_h), u_2(y_h), \cdots\}$). Uncertain numbers provide a convenient and succinct representation for quantities. Their algebraic properties essentially match those of ordinary number types. Thus, data processing algorithms can be expressed with familiar mathematical operations applied to uncertain-number terms representing quantities in a model. There is no need to derive the expressions for components of uncertainty; this is handled algorithmically.

The results of uncertain-number calculations are also transferable: the result of one calculation may be used as an argument in further calculations (as is performed routinely in numerical calculation). This is an open-ended process that can, in principle, continue indefinitely. The transferability of results is needed to support metrological traceability in staged measurement models. This will be discussed further in the next section and in Section 3.6. The open-ended nature of uncertain-number computations is also illustrated in the example shown in Section 2.4.

2.3. Traceability Chains and Uncertainty

Traceability provides accurate and reliable information about physical quantities that can be used to inform decisions. Because a quantity of interest can never be determined exactly, a decision based on the information available may not be correct; there will be some uncertainty—in a colloquial sense—about the correctness of a decision informed by data subject to measurement error. However, the risks associated with poor decision outcomes can be managed if the unpredictability of measurement results can be described in probabilistic terms (i.e., if the accuracy can be quantified). In this sense, the metrologist’s use of the term measurement uncertainty is associated with a likely magnitude of measurement error. To address the need for results that can be relied upon, metrological traceability requires the careful evaluation of measurement uncertainty.

Traceable measurement can be thought of as a collaborative process that is carried out in stages. Ultimately, a traceable measurement is of benefit to a nominal ‘end user’ at the last stage of a traceability chain, who needs information about a quantity to inform a decision (e.g., measuring the weights of shipping containers to inform the loading distribution of a container ship). The accuracy of a final result depends on all the stages; thus, the sources of uncertainty must be traced as far back as the units of measurement realised at the beginning of the process. This ensures that the result is meaningful and comparable with other traceable measurements of the same quantity.

While the GUM’s expression of a measurement model takes the form of a single all-encompassing Equation (1), the staged formulation in Section 2.2 handles the fact that people involved at one stage generally do not have detailed knowledge about processes carried out at other stages. For example, Figure 2 shows a traceable measurement in four parts (e.g., stages 1 and 2 correspond to the realisation of reference standards, stage 3 to the calibration of a measuring instrument using those standards, and stage 4 to an end-user measurement using the calibrated instrument). The staged model is described as follows;
where unspecified arguments ‘···’ represent some subset of the influence quantities $X_1, X_2, \cdots, X_l$. The end user can probably only formulate a model for stage 4, $f_4(Y_3, \cdots)$; thus, information about earlier stages must be summarised and reported down the chain in a suitable format. If the model was expressed as a single function, the composition of the stages would provide the following.

$$f(X_1, X_2, \cdots, X_l) = f_4(f_3(f_1(\cdots), f_2(\cdots), \cdots), \cdots).$$

Now, the outcome of data processing should not be affected by the expression of the model as a single function or a series of functions. This has a bearing on how information should be communicated along a traceability chain [10]. By reporting uncertain numbers between stages, final results can be obtained that are the same as would be found for a single model. Uncertain numbers realise the GUM’s ideal method for evaluating and expressing the uncertainty of a result [3] [Section 0.4].

![Figure 2](https://example.com/figure2.png)

**Figure 2.** A measurement carried out in four stages. Arrows indicate the transfer of information about intermediate results. The unspecified function arguments ‘···’ represent external quantities that influence the procedures. This figure does not represent a particular measurement, but the four stages may be regarded as follows: realisation of unit reference standards (stages 1 and 2), calibration of an instrument using the standards (stage 3), and a measurement made with the calibrated instrument (stage 4).

### 2.4. A Simple Example

This section presents an example of uncertain-number data processing applied to a simple electrical network. Figure 3 shows an electrical network with three resistors in series. A voltmeter can be connected between the lower terminal and any of the three terminals above, allowing the potential difference between terminals 0 and 1, 0 and 2, or 0 and 3 to be measured ($V_{10}$, $V_{20}$, or $V_{30}$, respectively).
Figure 3. Three resistors in an electrical network. A voltage measurement can be performed between the bottom terminal and any terminal above.

We adopt a simple model for an imperfect voltmeter. The model has three sources of error (influence quantities) that affect the response of a meter (reading) to an input voltage $V$. A random error (noise), represented as $E_{\text{rnd}}$, affects every reading; a systematic error, $E_{\text{off}}$, contributes a fixed offset to every reading; a systematic relative error, $E_{\text{rel}}$, contributes an error proportional to the reading itself (representing imperfect scaling or non-linearity of the instrument). The relationship between the input voltage, $V$, and a voltmeter reading, $v$, is expressed by the model (already shown in Section 2.2).

$$V = v(1 - E_{\text{rel}}) - E_{\text{off}} - E_{\text{rnd}}. \quad (8)$$

The influence quantities $E_{\text{rel}}$, $E_{\text{off}}$, and $E_{\text{rnd}}$ are unknown, and so their effects cannot be corrected. However, the displayed value $v$ is used as an approximation for $V$, because we assume that the instrument is properly adjusted. This is the same as assuming that the residual errors are small enough to be considered approximately zero. The uncertainty in the value of $v$, due to the estimates $e_{\text{rel}} = e_{\text{off}} = e_{\text{rnd}} = 0$, can be found if the uncertainties $u(e_{\text{rel}})$, $u(e_{\text{off}})$, and $u(e_{\text{rnd}})$ are known.

For uncertain number objects representing inputs to a measurement model, we find it helpful to adopt the term elementary uncertain number. Elementary uncertain numbers represent influence quantities. Numeric data must be provided when defining elementary uncertain numbers during the problem initialisation phase; this includes the following: a value (the estimate), a standard uncertainty, and a number of degrees of freedom.

We can use GTC to evaluate properties of the circuit, given measured values and some information about the voltmeter’s characteristics. Objects of the class `Voltmeter`, shown below, are used for data processing. During initialisation of a new `Voltmeter` (execution of `__init__()`, elementary uncertain numbers representing the two systematic errors are created and stored as instance variables (`ureal()` creates the uncertain numbers).

```python
from GTC import ureal, rp, result

class Voltmeter(object):
    def __init__(self,
        # Default characteristics for 1 V scale
        u_off=5E-3,
        u_rel=8E-4,
        u_rnd=1E-4
    ):
        self.u_rnd = u_rnd
        self.E_off = ureal(0.0, u_off, label="E_off")
```

The `applied_voltage()` method implements the model Equation (8). It returns an uncertain number for the applied voltage corresponding to a displayed value, \( v \) (the second argument, `index`, is used to create a label for the elementary uncertain number, \( E_{\text{rnd}} \), which is associated with random noise). In the code below, the uncertain numbers \( V_{10}, V_{20} \) and \( V_{30} \) are obtained from measurements of \( V_{10}, V_{20} \) and \( V_{30} \).

```python
# Displayed values, in volt
v_10 = 0.125841
v_20 = 0.385569
v_30 = 0.981950

# Voltmeter instance, using default specifications
dvm = Voltmeter()

# Uncertain numbers from displayed values
V_10 = dvm.applied_voltage(v_10,1)
V_20 = dvm.applied_voltage(v_20,2)
V_30 = dvm.applied_voltage(v_30,3)
```

We can infer circuit properties from these uncertain-number results. For example, the code below shows how accurately the voltage \( V_{10} \) was measured and the most important contributions to the uncertainty in that measured value.

```python
# Utility function to display a result
def display(v,label):
    print( "{}: {}\(\pm\){}").format(label,v )
    for l,u in rp.budget(v,trim=0):
        print( "{}: {}\(\pm\){}".format(l,u )

display(V_10,"V_{10}"
```

The function `display()` prints the measured value and a standard uncertainty in parentheses, followed by a list of components of uncertainty in order of magnitude. The label for each component is shown on the left and the magnitude of the component of uncertainty on the right. The two most significant figures of standard uncertainty are shown. Here, \( V_{10} \) has a measured value of 0.1258 volts and a standard uncertainty of 0.0050 volts.

```
V_{10}: 0.1258(50)
E_{off}: 0.005
E_{rel}: 0.00010067280000000002
E_{rnd_1}: 0.0001
```

Note that the dominant component of uncertainty is associated with the approximation made for systematic offset error \( E_{\text{off}} \). Very similar results are obtained for \( V_{20} \) and \( V_{30} \).

Other circuit properties can be calculated too. For example, the potential difference across resistor 2 can be found by taking the difference between \( V_{20} \) and \( V_{10} \). Subtracting those uncertain numbers in the argument, `display(V_{20}-V_{10},"V_{20}-V_{10}"`) yields the following.

```
V_{20}-V_{10}: 0.25973(25)
E_{rel}: 0.00020778240000000003
E_{rnd_1}: 0.0001
E_{rnd_2}: 0.0001
E_{off}: 0.0
```
This is an interesting result, which illustrates the detailed underlying calculation of uncertainty that is performed automatically. The standard uncertainty in the difference here is only 0.000 25 V—significantly less than the standard uncertainty in the individual measurements (both 0.0050 V). The uncertainty in this voltage difference is lower because it is insensitive to the offset $E_{\text{off}}$ (the offset is exactly the same in both readings). The display of components of uncertainty shows that the sensitivity to $E_{\text{off}}$ has been reduced to zero and that the influence of $E_{\text{rel}}$ is now dominant. We might also expect a smaller contribution to uncertainty to come from relative systematic error $E_{\text{rel}}$. However, that component varies in proportion to the applied voltage (it is a systematic relative error), and since $v_{10}$ is about three times larger than $v_{10}$, the contribution to uncertainty from $E_{\text{rel}}$ is still about two times larger than it was in the direct measurement of $V_{10}$.

3. Aspects of GTC Design

Using the mathematics described in the previous section applied to a given measurement model, GTC is required to evaluate a measured value, a standard uncertainty, and a number of degrees of freedom. This data processing can involve many computational stages and hundreds of influence factors. In addition, GTC can report the components of uncertainty in a result due to the uncertainty of individual influence (input) quantities and the components of uncertainty due to uncertainty in particular intermediate results, as required. Furthermore, it can store and retrieve uncertain numbers, allowing stages along a traceability chain to be appropriately handled.

This section describes how GTC has been designed to meet these challenges. The GTC package was first released four years ago, but our experience with the uncertain-number approach reaches back more than twenty years. We have used different programming languages and changed our thinking about how to implement the technique. For instance, early versions encountered difficulties when the size and variety of the measurement problems grew, and when additional software features were requested. Some programming languages were found to be better suited than others; larger problems exposed scaling weaknesses in our designs; and additional features place strain on some of the data structures and algorithms. GTC implements what we now consider to be our ‘best’ approach.

3.1. Simultaneous Calculation of Value and Uncertainty

Section 2.2 explained that the calculation of components of uncertainty can be handled using the chain rule for partial differentiation when a measurement model is expressed in stages. GTC extends this further by decomposing stage model expressions into very basic operations, such as $\times, \div, \sin(), \exp()$, etc. This is effectively using a computational technique called automatic differentiation [11]. Arithmetic operator overloading and a library of mathematical functions for uncertain numbers are used to automate decomposition of mathematical expressions into simple steps and then to evaluate the value and components of uncertainty at each step.

All the basic uncertain-number functions and arithmetic operations defined in GTC are either univariate or bivariate. For a univariate function, $f_h(z)$, Equation (7) reduces to the following:

\[
\frac{\partial f_h}{\partial z} u_j(z) ,
\]

and for a bivariate function, $f_h(z_1, z_2)$, Equation (7) becomes the following.

\[
\frac{\partial f_h}{\partial z_1} u_j(z_1) + \frac{\partial f_h}{\partial z_2} u_j(z_2) .
\]

Thus, for example, the uncertain-number trigonometric sine function can be handled as follows. If the value of an uncertain-number input is $z$, then the value of the uncertain-number result is $y_h = \sin(z)$. Furthermore, if there are two components of uncertainty associated with the input, \( u_1(z) \) and \( u_2(z) \), then the two corresponding components of
uncertainty associated with the result are \( u_1(y_h) = c u_1(z) \) and \( u_2(y_h) = c u_2(z) \), where \( c = \frac{\partial f_h}{\partial z} = \cos(z) \) is the derivative in (9).

3.2. Unique Identifiers

GTC algorithms track the identity of elementary uncertain numbers representing influence factors. The subscript \( i \) that appears on the terms for components of uncertainty, \( u_i(y) \), is the same as the subscript appearing on the influence quantities, \( X_i \), in model Equation (1).

\[
Y = f(X_1, X_2, \ldots).
\]

Software and digital records must somehow keep track of these \( i \)'s, even when measurements are carried out in stages, at different locations and at different times. GTC uses a simple tuple of integers as an identifier format. The first integer is kept fixed for a given session while the second integer takes the value of a counter that is incremented each time an elementary uncertain number is created. To ensure that these identifiers are unique in time and space, the first integer is a Universally Unique Identifier (UUID) formatted as a 128-bit integer.

This identifier format reveals nothing about the influence quantity, although identifiers can be arranged in order, which improves the performance of some algorithms. Would a more sophisticated type of identifier be useful? For the purposes of data processing, the only requirement is uniqueness. Nevertheless, GTC already allows text labels to be associated with nodes (used as labels for influence quantities in uncertainty budgets) and a planned enhancement to GTC will allow information about influence quantities to be held in a manifest and indexed by unique identifiers. Such a manifest could accompany a digital record of uncertain-number data. This could address any need for additional metadata about influence quantities, without the burden of minting and configuring more specialised digital objects designed to access information on the internet [12].

3.3. Node Classes

During a GTC calculation, most of the uncertain numbers that are created may be regarded as temporary objects and can be garbage-collected almost immediately. If this is not performed in large problems, the demands on memory can seriously limit performance. To address this, GTC only holds essential information about influence quantities, and information about certain intermediate results as required. This essential information is kept in small node objects, allowing the memory occupied by larger uncertain-number objects to be reclaimed when not required.

There are two classes of node: A Leaf is associated with elementary uncertain numbers and a Node is associated with uncertain numbers representing intermediate results; Leaf is a subclass of Node (Figure 4). A Leaf is created whenever an elementary uncertain number is declared. The information encapsulated includes the following: a value of standard uncertainty; a number of degrees of freedom; a Boolean flag, which identifies objects declared to be independent; a string label, for display purposes; a unique identifier; and two Python collection objects. One of these is the dictionary correlation, which holds values of \( r(x_i, x_j) \) for calculations such as Equation (11). The other is the ensemble set, which is used to identify other nodes associated with an ensemble of closely related elementary uncertain numbers. Ensembles are used in the calculation of degrees of freedom described in Section 2.1 (see also Appendix A.1).

While a Leaf is created for every elementary uncertain number, there is no need to create nodes at every stage of a calculation. When intermediate components of uncertainty are required (for an intermediate result of particular significance or when an uncertain number will be stored), the function `result()` is used to create a new Node (as shown below in Section 3.5 and later in Appendix A.3).
The uid attribute can be used to sort Nodes

Leaf
+ independent : Boolean
+ correlation : dict
+ ensemble : set

The uid attribute can be used to sort Nodes

3.4. Propagating Uncertainty

During uncertainty propagation, components of uncertainty must be evaluated at each step. The cumulative effect of these computations can dominate execution time and the demands on memory can be high. Moreover, the overhead of looking up correlation coefficient values for pairs of inputs in Equation (3) during the final calculation of a standard uncertainty is inefficient.

To address this, information about components of uncertainty is stored in several sequences in uncertain-number objects (u_components, d_components and i_components in Figure 5). The elements in these sequences consist of a component of uncertainty paired with a node that holds information about the corresponding influence quantity (see Figure 4).

During calculations, an uncertain number is created at every step. The components of uncertainty are evaluated by weighting the components of uncertainty for the inputs to the step and combining these weighted components when common influences are involved. By keeping the elements in sequences ordered, this process can be handled efficiently by stepping along the sequences and identifying any common influences. The ordering is established by the uid attribute of node objects.
For example, consider the multiplication of a pair of uncertain numbers, \( f_h(Z_1, Z_2) = Z_1 Z_2 \).
Suppose the component-of-uncertainty sequence for the first argument contains the following elements, where only the first four digits of the UUID integer are shown:

\[
\begin{align*}
(7953..., 1) &: 3, \\
(7953..., 2) &: -1, \\
(7953..., 4) &: 15, \\
(7953..., 6) &: -5 \\
\end{align*}
\]

and the component-of-uncertainty elements for the second argument are as follows.

\[
\begin{align*}
(7953..., 1) &: -1, \\
(7953..., 3) &: 2, \\
(7953..., 4) &: 2, \\
(7953..., 5) &: 12 \\
\end{align*}
\]

To obtain the component-of-uncertainty sequence for the product, components in the first sequence are weighted by the value of the second argument and components in the second sequence are weighted by the value of the first argument. The weighted components of common influences are added together (in this case, there is a pair of common influences identified by \( (7953...,1) \) and \( (7953...,4) \)). Thus, if the value of the first argument is 5 and the value of the second is 10, the components of uncertainty for the product are as follows.

\[
\begin{align*}
(7953..., 1) &: 25, \\
(7953..., 2) &: -10, \\
(7953..., 3) &: 10, \\
(7953..., 4) &: 160, \\
(7953..., 5) &: 60, \\
(7953..., 6) &: -50 \\
\end{align*}
\]

When evaluating Equation (3), the number of terms to be summed grows in proportion to the square of the number of input arguments. Moreover, a value of \( r(x_i, x_j) \) is needed for every pair of inputs. However, in practice, there are very few non-trivial correlation coefficients assigned (when \( i = j \), \( r(x_i, x_j) = 1 \) and usually \( r(x_i, x_j) = 0 \), when \( i \neq j \)). Therefore, not only is the overhead of looking up correlation coefficients unnecessary but many terms in the double sum are zero. To streamline this calculation, the data for independent and dependent influences are separated in two different component-of-uncertainty sequences (\textit{u\_components} and \textit{d\_components}, in Figure 5). When elementary uncertain numbers are defined, it will be known whether correlation coefficients will be associated with the inputs; thus, dependent and independent influences can be identified and separated. The evaluation of Equation (3) can then be handled more efficiently. For instance, if there are independent estimates \( X_1, \cdots, X_K \) and dependent estimates \( X_{K+1}, \cdots, X_l \), the calculation of Equation (3) can be expressed as follows:

\[
u(y) = \left[ \sum_{i=1}^{K} u_i^2(y) + 2 \sum_{i=K+1}^{l} \sum_{j=i+1}^{l} u_i(y) r(x_i, x_j) u_j(y) \right]^{1/2}, \tag{11}\]

where the double sum is now only over dependent terms.

### 3.5. Intermediate Results

In some problems, the interpretation of results is complicated by a large number of influences and, hence, a large number of components of uncertainty. A succinct and
often more intuitive presentation can sometimes be obtained, without sacrificing rigour, by reporting the sensitivity of a final result to uncertainty in intermediate results. This is implemented in GTC by using another sequence for components of uncertainty with respect to designated intermediate results (\(i\) _components in Figure 5).

To initialise the process of calculating an intermediate component of uncertainty, function result() must be applied to an uncertain number. This seeds a new element in \(i\) _components. Thereafter, propagation occurs, as before, by weighting the intermediate components of stage inputs by the partial derivatives of the stage function. The elements in the sequence \(i\) _components are also node-value pairs, but a Node rather than a Leaf is used (see Figure 4).

The electrical network example can be used to illustrate the use of intermediate results. Suppose the current through the series network is measured as 1.0000 mA, with a standard uncertainty of 0.0010 mA. The resistance of the resistor in the middle of the network, and a breakdown of the contributions to uncertainty in that value, is obtained simply from the following.

\[
I = ureal(1E-3,0.001E-3,label="I")
\]

\[
display(-(V_20-V_10)/I, "R2")
\]

The results are as follows.

\[
R2: 259.73(36)
I: 0.2597279999999999
E_{rel}: 0.20778240000000003
E_{rnd_1}: 0.1
E_{rnd_2}: 0.1
E_{off}: 0.0
\]

However, to see how resistance depends on voltage and current measurements, we declare the voltage difference to be an intermediate result and report an uncertainty budget in terms of only the current and voltage.

\[
V_{20-V_{10}} = result(V_{20-V_{10}},label="V_{20-V_{10}}")
\]

\[
for l,u in rp.budget(V_{20-V_{10}}/I,influences=[I,V_{20-V_{10}}]):
print("{}: \{\}").format(l,u)
\]

The results are as follows:

\[
I: 0.2597279999999999
V_{20-V_{10}:} 0.2513434418276316
\]

which shows that the contribution to uncertainty from the current measurement is comparable to the contribution from the voltage measured across the resistor. This was not so obvious from the complete list of uncertainty components obtained earlier.

3.6. Storage and Retrieval of Uncertain Numbers

Section 2.3 explained that traceable measurements can be preformed in stages and that, by reporting uncertain numbers when data are processed at each stage, the final uncertainty can be evaluated correctly. The results at one stage will be required at a later time and place. Thus, uncertain numbers must be stored somehow and their identities, which correspond to physical quantities in the actual measurement, must be retained. In GTC, an Archive object is used to manage storage and retrieval of uncertain numbers. The unique identifiers described in Section 3.2 keep track of uncertain-number identities in different Python sessions.

As an example, the code below saves an uncertain number for the voltage difference \(V_{20} - V_{10}\) in a text file using a JSON format. Note that the GTC function result() is applied to designate \(V_{20} - V_{10}\) as an intermediate result. This is a prerequisite for storage.

```python
from GTC import pr
a = pr.Archive()
```
a.add( V_20_V_10 = result(V_20 - V_10) )
with open("file_name.json", "w") as f:
    pr.dump_json(f,a,indent=4)

The uncertain number V_20_V_10 for the voltage difference is retrieved by the following:

with open("file_name.json", "r") as f:
a = pr.load_json(f)
display( a["V_20_V_10"],"V_20_V_10")

which produces the following output.

V_20_V_10: 0.25973(25)
E_rel: 0.00020778240000000003
E_rnd_1: 0.0001
E_rnd_2: 0.0001
E_off: 0.0

This shows that the necessary information about influence quantities has been retained. Alternatively, the measurements of V_10 and V_20 can be saved individually.

a = pr.Archive()
a.add( V_10 = result(V_10) )
a.add( V_20 = result(V_20) )
with open("file_name.json", "w") as f:
    pr.dump_json(f,a,indent=4)

Then, in a later session, the difference between the individual results can be evaluated by the following:

with open("file_name.json", "r") as f:
a = pr.load_json(f)
V_10 = a["V_10"]
V_20 = a["V_20"]
display(V_20 - V_10,"V_20_V_10")

which provides exactly the same results.

V_20_V_10: 0.25973(25)
E_rel: 0.00020778240000000003
E_rnd_1: 0.0001
E_rnd_2: 0.0001
E_off: 0.0

Although the code here suggests that uncertain-number objects are simply being saved and restored, there is more to it: information about related elementary and intermediate uncertain numbers is also included in the digital record. When an uncertain number is ‘added’ to an archive, objects that hold information about related influences are identified from the component-of-uncertainty sequences. For instance, we showed earlier that the calculation associated with V_10 could be decomposed into stages (Figure 1). Referring again to that figure and thinking about data processing, the error terms, E_{rel}, E_{off}, and E_{rnd}, correspond to elementary uncertain numbers, and each of the circled mathematical operations represents an intermediate stage in the calculation. If the intention is to store the uncertain-number V_10, then information is also required about the voltmeter errors. Figure 6 shows objects with information that would be saved. Later, when the contents of an Archive is loaded back into a different session, this contextual information is immediately restored (nodes are created with the appropriate identifiers). This ensures that when uncertain numbers are retrieved from an archive, they behave as they would have in the context of the original session, which maintains the integrity of information in a traceability chain.
Figure 6. A UML object diagram showing information that would be collected when the uncertain number \( V_{10} \) is stored. The three Leaf objects contain information about the error terms in the voltmeter measurement model: the fixed offset, the relative error and the random noise. Only one Leaf for the influence of noise is shown; however, if several readings had been taken there would be a different node for each reading.

4. Discussion

This paper provides some insight into the usefulness of uncertain numbers, which have the distinctive feature of providing an abstract representation for measured quantities, allowing uncertainty calculations to be automated. Recently, uncertain numbers have been applied to a goniometric measurement system for optical reflectance. The four-axis goniometric system has many configuration errors that must be considered in the measurement model to account for final measurement uncertainties [7]. The application of GTC was carefully compared with alternative computational methods, using Monte Carlo and direct mathematical analysis. GTC was found to be the preferred choice. Using the information provided by uncertain numbers, the authors were able to obtain a better understanding of the measurement system and the inherent correlations between significant measurement errors. This enabled them to significantly improve the accuracy of certain measurements.

The inherent support for metrological traceability is perhaps the most important quality of uncertain numbers. This aspect is implemented in data structures and storage formats used by GTC, which is a particular choice but other formats would be possible. One can easily imagine a more heterogeneous situation, where processing at various stages would be carried out using different software tools. To support this, the format for exchange of data between stages would need to be standardised. That is, there would need to be agreed formats for representing uncertain numbers, which would be used in digital reporting documents such as calibration reports [13].

Digitalisation should offer benefits that are not currently available. The GUM recommends that detailed information about influence quantities be reported at each stage. However, this rarely happens, because calibration certificates and other measurement reports are intended to be read by people; thus, handling the additional data would be difficult. As a consequence, information about common influence factors is rarely shared. A simple situation where this might arise is the scenario of a batch of sensors that are calibrated using a more accurate reference device. If the common reference is ignored, the accuracy of results obtainable from a survey of the sensors’ readings is compromised [14]. However, uncertain-number calibration factors can track common effects and account for them when comparing readings from different sensors. This was illustrated in Section 2.4, where \( E_{\text{off}} \) contributed a common offset to single voltage readings but nothing to uncertainty in the voltage difference. It is also worth noting that a ‘smart’ sensor capable of reporting
uncertain-number results would not need to process a lot of information. As was the case of the simple voltmeter, a model of the sensor measurement might only require a few influence factors and the calculations would be simple.

The various stages of a traceable measurement often occur in different locations (national metrology institute, second-tier calibration laboratory, etc.) but they may also happen at different times in the same location. For example, a working standard might be calibrated in-house against an externally calibrated transfer standard. The working standard would then be used repeatedly to calibrate different instruments at different times. Importantly, measurement errors realised when at the time the working standard is calibrated should be treated as systematic effects in subsequent instrument calibrations. Performing this would allow any bias, or correlation, in downstream measurement results using those instruments to be accounted for correctly. This could be easily handled by digitalisation if uncertain-number storage and retrieval mechanisms are used to save calibration data for the working standard and later retrieve it for reuse when instruments are calibrated.

During formal international measurement comparisons, national metrology institutes (NMIs) go to much greater pains when reporting measurement data than they do for regular calibration work. These international comparisons assess the competence of NMIs in performing specific types of measurement. The more detailed reporting requirements in comparisons align with the GUM’s recommendations in this case. A recent study, which explored a future scenario where an uncertain-number reporting format was used by all participants, showed that using uncertain numbers would not only provide the information required, but they would also simplify comparison analysis and comparison linking and provide additional insights into the results [6].

Measurement models are needed in order to use uncertain numbers effectively. The close correspondence between quantity terms in a model and uncertain numbers in data processing routines makes software development and testing more robust and reliable and avoids the need to explicitly derive expressions for the components of uncertainty from a model, which GTC handles automatically. However, although modelling lies at the heart of the GUM’s approach, skilled metrologists are often confident in their ability to assess measurement uncertainty heuristically and frequently elide the formal modelling step. This presents a problem for digitalisation, because digital systems need a rigorous formal problem definition for autonomous operation. Some tutorial guidance on developing measurement models has been provided in a recent booklet [15] and is also the subject of another paper [16]. There is also a new supplement to the GUM, which deals with modelling [17].

One common conceptual difficulty when modelling is the omission of influence quantities estimated as zero. These terms would not be needed in conventional data processing; however, they must be modelled, because the actual (unknown) values affect the final measurement result, and so they contribute to uncertainty. Influence quantities with trivial estimates are often called residual errors. The electrical network example, in Section 2.4, included three residual errors that were all estimated as zero. These terms were represented by uncertain numbers and modelled imperfect voltmeter behaviour.

5. Conclusions

GTC is a software tool for data processing with automatic evaluation of measurement uncertainty. It follows international best-practice, described in the GUM, and offers useful extensions to those methods for important special cases. The use of uncertain numbers is a distinctive feature of GTC. The uncertain-number data-type facilitates data processing, which can be performed in a piece-wise and open-ended manner. This allows calculations to be more easily matched to the models of a measurement performed in stages. The automation of uncertainty calculations allows measurement data processing to be made more rigorous, which can lead to accuracy enhancement in some cases. The uncertain-number format significantly exceeds current paper-based practices that support traceability.
Therefore, GTC and the data structures used to implement uncertain numbers are a useful example of software that meets the requirements of a fully functional digital infrastructure for metrological traceability.

**Funding:** This is work was funded by the New Zealand Government.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The author is grateful to Peter Saunders and Joseph Borbely for careful reviews of the manuscript.

**Conflicts of Interest:** The author declare no conflict of interest.

**Abbreviations**
The following abbreviations are used in this manuscript:

- **BIPM** International Bureau of Weights and Measures;
- **GUM** Guide to the Expression of uncertainty in measurement;
- **IEC** International Electrotechnical Commission;
- **IFCC** International Federation of Clinical Chemistry and Laboratory Medicine;
- **ILAC** International Laboratory Accreditation Cooperation;
- **ISO** International Organisation for Standardisation;
- **IUPAC** International Union of Pure and Applied Chemistry;
- **IUPAP** International Union of Pure and Applied Physics;
- **JSON** JavaScript Object Notation;
- **NIST** National Institute of Standards and Technology;
- **NMI** National Metrology Institute;
- **OIML** International Organization of Legal Metrology;
- **SI** International System of Units (Système International);
- **UUID** Universally Unique Identifier.

**Appendix A. Additional Details**

This Appendix includes details about a number of other aspects of GTC. Support for problems involving finite degrees of freedom with correlated inputs is discussed in Appendix A.1 and support for complex quantities is briefly covered in Appendix A.2. Appendix A.3 presents an unusual case, where counter-intuitive results are obtained due to the relationship between Python variables and underlying uncertain-number objects. This provides further insight into computational mechanisms. Appendix A.4 briefly describes the validation of GTC and Appendix A.5 compares GTC with some similar software projects.

**Appendix A.1. Ensembles**

As noted in Section 2.1, the Welch-Satterthwaite formula cannot be used on correlated data with finite degrees of freedom. However, there is an extension that can be applied in situations where data are deemed to come from closely related quantities with fixed interdependencies [9]. To implement this, GTC algorithms must be able to identify sets of uncertain numbers declared as representing related quantities. The `ensemble` attribute of the `Leaf` node is used for this purpose (Figure 4). An ensemble is a set of `Leaf` nodes. There are some GTC functions that declare ensembles automatically, such as functions for linear regression; in other cases an ensemble can be explicitly defined by `multiple_ureal()` (the GTC online documentation for `multiple_ureal()` shows a calculation from GUM Appendix H2 [18]).
GTC includes regression functions that estimate the parameters of a straight line passing close to a sample of data. The finite sample size means that uncertainties in estimates for the slope and intercept have finite degrees of freedom and are usually correlated.

The code below shows a least-squares regression for nine data points. The GTC function `line_fit()` returns an object with an attribute that holds a pair of uncertain numbers for the slope and intercept `(a, b)`.

```python
from GTC import type_a, get_correlation

x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
y = [15.6, 17.5, 36.6, 43.8, 58.2, 61.6, 64.2, 70.4, 98.8]
result = type_a.line_fit(x, y)
a, b = result.a_b
print("a =", repr(a))
print("b =", repr(b))
print("r(a,b) =", get_correlation(a, b))
```

The results are as follows.

\begin{align*}
a &= \text{ureal}(4.813888888888881, 4.886206312183354, 7) \\
b &= \text{ureal}(9.408333333333335, 0.8683016476563609, 7) \\
r(a,b) &= -0.888523316639
\end{align*}

The slope and intercept are correlated and there are seven degrees of freedom associated with the uncertainties. However, these results may still be used to calculate the expected value `y` for `x = 5.5`:

\begin{align*}
y_p &= a + b \times 5.5 \\
\text{print("y_p =", repr(y_p))}
\end{align*}

which produces a result with seven degrees of freedom.

\begin{align*}
y_p &= \text{ureal}(56.55972222222223, 2.2835948151943155, 7.0)
\end{align*}

**Appendix A.2. Complex Quantities**

Section 2.2 described data processing for real-valued quantities, but very similar formulae apply to complex quantities. These are also implemented in GTC. A review of measurement uncertainty for complex quantities has been given by Hall [19].

GTC can handle mathematical expressions with a mixture of real-valued and complex-valued quantities and results may be either real or complex uncertain numbers, as is appropriate. An uncertain complex number is implemented as a pair of uncertain real numbers; thus, uncertainty is represented by uncertainties in the real and the imaginary components as well as the correlation coefficient between those components. A convenient format for specifying uncertainty in a complex value is a $2 \times 2$ variance-covariance matrix. The number of degrees of freedom associated with uncertainty in the real and imaginary components is the same.

Often a complex quantity is evaluated from a small sample of data. In that case, the real and imaginary component estimates are dependent, being evaluated from the same sample; they will also have a finite number of degrees of freedom. As already mentioned, the combination of finite degrees of freedom and correlation creates problems for data processing. However, when converting from a complex quantity to a real one, the modified form of the Welch–Satterthwaite formula can be useful [9]. For example, suppose a complex number $z = x + iy$ has been evaluated from a small sample, $x = 0.20$ and $y = 0.0$, the real and imaginary components each have a variance of 0.1, there is a covariance between the components of 0.05, and there are 10 degrees of freedom. If the real-valued magnitude of the following:

\begin{align*}
|z| &= \sqrt{x^2 + y^2}.
\end{align*}

is of interest, it can be evaluated as follows.
from GTC import ucomplex, magnitude

z = ucomplex(0.20,[0.1,.05,.05,0.1],10)
print("mag(z) = {!r}".format( magnitude(z) ))

The result
ureal(0.2,0.31622776601683794,10.0)
is an uncertain real number with 10 degrees of freedom.

On the other hand, if the result of a calculation is an uncertain complex number, there
is an alternative to the Welch–Satterthwaite formula that must be used [20]. Here is an
example (from [20]):

from GTC import ucomplex

x1 = ucomplex( 1, (0.96,-0.34,-0.34,0.27), 5 )
x2 = ucomplex( 1, (0.51,0.33,0.33,0.31), 3 )
x3 = ucomplex( 1, (0.45,0.28,0.28,1.65), 6 )

z = x1 + x2 + x3

print("z = {0.x}".format( z ))
print("u(z) = {0.u}".format( z ))
print("r(x,y) = {0.r}".format( z ))
print("df(z) = {0.df}".format( z ))

which displays the complex value, the standard uncertainties, the correlation coefficient,
and the degrees of freedom.

z = (3+0j)
u(z) = StandardUncertainty(real=1.3856406460551018, imag=1.493318452306808)
r(x,y) = 0.13048503857331625
df(z) = 11.340977790491408

Appendix A.3. Uncertain Number Objects and References

The combination of uncertain numbers and Python language features can provide
intuitive and meaningful representations of a problem domain. In particular, the distinction
between random and systematic effects can be elegantly captured in object-oriented designs.
However, on rare occasions, the behaviour of the Python variable names that refer to objects
in memory can result in confusion. It is interesting to see how this can happen, because it
provides insight into the computational processes. Here is a simple example.

Consider the following equations.

\[ u = x, \]
\[ v = x + y, \]
\[ w = u + v. \]

What is \( \partial w / \partial u \)? To find the answer using GTC, we may performing the following (note,
partial derivatives are evaluated when elementary uncertainty numbers are declared with
an uncertainty of unity):

from GTC import ureal, component, result

x = ureal(0,1,label="x")
y = ureal(0,1,label="y")
u = x
v = x + y
w = u + v

print("partial derivative wrt x =", component(w,x) )
print("partial derivative wrt u =", component(w,u) )
which displays the following.

```
partial derivative wrt x = 2
partial derivative wrt u = 2
```

If we had in mind that \( w = u + v \), this result may come as a surprise because \( \partial w / \partial u = 1 \) would be expected. However, it is important to remember that the terms in a calculation correspond to uncertain-number objects in memory and not the variable names in code. Both \( x \) and \( u \) refer here to the same elementary uncertain number. Therefore, equation \( w = u + v \) actually corresponds to \( w = 2x + y \) in terms of the underlying objects, and so \( \partial w / \partial x = 2 \) is correct.

Confusion is created by equating the Python variables \( u = x \) if it is (incorrectly) assumed that \( u \) and \( x \) are somehow different. If we intend to take the derivative of \( w = u + v \) with respect to \( u \), a distinct uncertain-number object must be created for \( u \) (and designated as an intermediate result to allow an intermediate component of uncertainty to be calculated).

To implement this calculation, we may use the unary ‘+’ operator to create an additional uncertain number representing \( u \) in memory. This operator copies the numerical attributes of its argument into a new uncertain number. As far as calculation is concerned, this object corresponds to a distinct term. The following code clones \( x \) and designates it as an intermediate result to allow the component of uncertainty to be evaluated.

```python
x = ureal(0,1,label="x")
y = ureal(0,1,label="y")
u = result(+x,label="u")
v = x + y
w = u + v
print( "partial derivative wrt x =", component(w,x) )
print( "partial derivative wrt u =", component(w,u) )
```

This displays the following.

```
partial derivative wrt x = 2
partial derivative wrt u = 1
```

This situation is unusual. Normally, `result()` would be applied to an object produced as the result of a calculation; thus, there is almost never a need to clone uncertain numbers as is shown here.

**Appendix A.4. Testing and Validation**

GTC has a modular structure. It uses Python arithmetic operator overloading and mathematical function definitions to decompose mathematical expressions into basic uncertain-number operations. This makes the code amenable to unit testing. The calculation of values uses standard Python mathematical operations and processing of components of uncertainty uses of automatic differentiation, which also makes use of built-in Python arithmetic and mathematical libraries. An extensive suite of test cases has been built up to verify implementation details [5]. Calculations are also checked against standard examples from appendices to the GUM and other published sources, including various forms of regression analyses with uncertainties. Moreover, GTC has been used for more than a decade at the Measurement Standards Laboratory, where it is closely scrutinised by different groups. Very few issues have been reported since the project was made publicly available on github in 2018.

**Appendix A.5. Similar Software**

Software to evaluate measurement uncertainty is often used alongside other data processing tools (a notable example is the web-based calculator called *The NIST Uncertainty Machine* [21]). However, the separation of data processing into different work streams is unnecessary with GTC, because uncertainty calculation is an integral part of all data
processing. GTC may be incorporated in projects to provide data processing and support for traceability. This is also the case for the C# library called UncLib [22], which is part of the data acquisition and data processing application called VNA Tools II [23], now used by many leading microwave metrology laboratories. Similarly to GTC, UncLib handles measurements of real-valued and complex-valued quantities and provides support for traceability by identifying input quantities and allowing them to be stored and retrieved. However, the data structures of UncLib have been designed to support a particular optimisation strategy, which results in some different behaviours (see Zeier et al. [22] [Section 3.3]). For instance, the evaluation of the intermediate components of uncertainty may fail unless certain preconditions are satisfied [22] [Section 3.4] and the extensions to the Welch–Satterthwaite formula and degrees of freedom support for complex quantities cannot be implemented.

A well-known Python package that calculates uncertainty is Uncertainties [24]. This package is intended for engineering error and sensitivity analyses, such as described by Bevington and Robinson [25]. Similarly to GTC and UncLib, uncertainties use automatic differentiation to evaluate partial derivatives during data processing. However, it does not calculate degrees of freedom, nor does it handle complex quantities or provide for storage and retrieval of results.

References

1. Hackel, S.; Härtig, F.; Hornig, J.; Wiedenhöfer, T. The Digital Calibration Certificate. PTB-Mitteilungen 2017, 127, 75–81. [CrossRef]
2. Thiel, F. Digital transformation of legal metrology-The European Metrology Cloud. OIML Bull. 2018, 59, 10–21.
3. BIPM; IEC; IFCC; ILAC; IUPAC; IUPAP; OIML. Evaluation of Measurement Data—Guide to the Expression of Uncertainty in Measurement JCGM 100:2008 (GUM 1995 with Minor Corrections), 1st ed.; BIPM Joint Committee for Guides in Metrology: Paris, France, 2008.
4. Hall, B.D. Computing with Uncertain Numbers. Metrologia 2006, 43, L56–L61. [CrossRef]
5. Hall, B.D.; Borbely, J.S. GUM Tree Calculator and v1.3.6. Available online: https://github.com/MSLNZ/GTC; doi:10.5281/zenodo.5459479 (accessed on 13 November 2021).
6. Hall, B.D.; Koo, A. Digital Representation of Measurement Uncertainty: A Case Study Linking an RMO Key Comparison with a CIPM Key Comparison. Metrology 2021, 1, 166–181. [CrossRef]
7. Molloy, E.; Saunders, P.; Koo, A. Effects of rotation errors on goniometric measurements. Metrologia 2021, in press. [CrossRef]
8. Walpole, R.E.; Myers, R.H.; Myers, S.L.; Ye, K. Probability & Statistics for Engineers and Scientists, 8th ed.; Pearson Education: Upper Saddle River, NJ, USA, 2007; p. 07458.
9. Willink, R. A generalization of the Welch–Satterthwaite formula for use with correlated uncertainty components. Metrologia 2007, 44, 340. [CrossRef]
10. Hall, B.D.; White, D.R. Digital representation of uncertainty for metrological traceability. In Advanced Mathematical and Computational Techniques in Metrology XII; Pavese, F., Forbes, A.B., Chunovkina, A.G., Zhang, N.F., Eds.; Series on Advances in Mathematics for Applied Sciences; World Scientific: Singapore, 2021; Volume 90, pp. 262–272.
11. Rall, L.B.; Corliss, G.F. An Introduction to Automatic Differentiation. In Computational Differentiation: Techniques Applications, and Tools; Berz, M., Bischof, C.H., Griewankpp, A., Eds.; Siam Proceedings in Applied Mathematics Series; SIAM: Philadelphia, PA, USA, 1996; Volume 89, pp. 1–17.
12. Klump, J.; Huber, R. 20 Years of Persistent Identifiers—which Systems are Here to Stay? Data Sci. J. 2017, 16, 9. [CrossRef]
13. Hackel, S.; Härtig, F.; Hornig, J.; Wiedenhöfer, T. Metrologie für die Digitalisierung von Wirtschaft und Gesellschaft. 2017.
14. Hall, B.D. An Opportunity to Enhance the Value of Metrological Traceability in Digital Systems, 2nd ed.; Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT); IEEE: Naples, Italy, 2019; pp. 16–21.
15. Hall, B.D.; White, D.R. An Introduction to Measurement Uncertainty; Measurement Standards Laboratory of New Zealand: New Zealand, 2020. [CrossRef]
16. Hall, B.D. Object-oriented software for evaluating measurement uncertainty. Meas. Sci. Technol. 2013, 24, 055004. [CrossRef]
17. BIPM; IEC; ILAC; IFCC; ISO; IUPAC; IUPAP; OIML. Guide to the Expression of Uncertainty in Measurement—Part 6: Developing and Using Measurement Models JCGM GUM-6:2020, 1st ed.; BIPM Joint Committee for Guides in Metrology: Paris, France, 2020.
18. Hall, B.D.; Borbely, J.S. GTC Documentation. Available online: https://gtc.readthedocs.io/en/stable/ (accessed on 13 November 2021).
19. Hall, B.D. Evaluating the measurement uncertainty of complex quantities: A selective review. Metrologia 2015, 53, S25–S31. [CrossRef]
20. Willink, R.; Hall, B.D. A classical method for uncertainty analysis with multidimensional data. Metrologia 2002, 39, 361 – 369. [CrossRef]
21. Lafarge, T.; Possolo, A. The NIST Uncertainty Machine. *Ncsli Meas. J. Meas. Sci.* 2015, 10, 20–27. [CrossRef]

22. Zeier, M.; Wollensack, M.; Hoffmann, J. METAS UncLib-a measurement uncertainty calculator for advanced problems. *Metrologia* 2012, 49, 809–815. [CrossRef]

23. Wollensack, M.; Hoffmann, J.; Ruefenacht, J.; Zeier, M. VNA Tools II: S-parameter uncertainty calculation. In Proceedings of the 79th ARFTG Microwave Measurement Conference Digest, Montreal, QC, Canada, 22–22 June 2012. [CrossRef]

24. Lebigot, E.O. Uncertainties: A Python Package for Calculations with Uncertainties. Available online: http://pythonhosted.org/uncertainties (accessed on 13 November 2021).

25. Bevington, P.R.; Robinson, D.K. *Data Reduction and Error Analysis*; McGraw-Hill: New York, NY, USA, 2003.