

# Uncertainty Calculation for Complex Measurements

by

Pablo Vanwoerkom

An Honors Thesis Submitted to the  
Department of Mathematics, Engineering, and Computer Science  
in partial fulfillment of the requirements  
of graduating with honors.

Carroll College  
December 2005

**SIGNATURE PAGE**

This thesis for honors recognition has been approved for the  
Department of Mathematics, Engineering, and Computer Science.

Holly Zullo

Dr. Holly Zullo, Director

11/22/05

Date

Kelly Cline

Dr. Kelly Cline, Reader

11/28/05

Date

Joan Stottlemyer

Mrs. Joan Stottlemyer, Reader

11-22-05

Date

# Table of Contents

<b>Acknowledgments</b> .....	ii
<b>Abstract</b> .....	iii
<b>Chapter 1: Introduction to Measurement Uncertainty</b> .....	1
<b>Chapter 2: Models for Uncertainty Propagation</b> .....	1
2.1 Computation of Uncertainty Propagation Methods.....	2
2.2 Taylor Series Expansion.....	2
2.3 Limitations of Taylor Series Expansion.....	4
2.4 Generalized Inference.....	4
2.5 Sample Problem.....	5
2.6 Method Comparison.....	6
<b>Chapter 3: Implementation of Computational Tools for Uncertainty Propagation</b> ....	6
3.1 Basic Structure.....	7
3.2 Implementing the Taylor Series Expansion Method.....	7
3.3 Implementing the Generalized Inference Method.....	10
3.4 Challenges.....	14
<b>Chapter 4: Conclusions</b> .....	14
<b>References</b> .....	15
<b>Appendix A</b> .....	16
<b>Appendix B</b> .....	20

## Acknowledgements

I would like to thank all who have made the completion of this thesis possible. As this thesis is the product of work done at two different locations, individuals in both locations were crucial to this thesis. Although most of the actual work was done at the National Institute of Standards and Technology (NIST) at Boulder, Colorado, the thesis was written at Carroll College in Helena, Montana.

From Carroll College, I first want to thank my thesis director, Dr. Holly Zullo, for her encouragement and guidance in tackling this project. Thanks to her mentorship, completion of this thesis has flowed smoothly and efficiently.

I want to thank Dr. Kelly Cline for his thoughtful suggestions which have led to a more thorough thesis. Equally grateful I am to Mrs. Joan Stottlemeyer for helping me create a thesis free of grammatical errors.

I want to thank NIST for giving me the opportunity to participate in its summer REU program (SURF). From NIST, I thank Dr. Chih-Ming (Jack) Wang for his assistance in understanding the mathematical concepts involved in this work, as well as his warm-heartedness. Also from NIST, I am grateful to Dr. Brad Alpert for his help in learning  $R$  and Dr. Dominic F. Vecchia for helping me polish my summer presentation. Last but not least, I thank Ms. Lorna Buhse, the Statistical Engineering Division's secretary for her kindness and thoughtfulness. This work was funded in part by Federal Grant Number 70NANB5H1025.

Finally, I want to thank my family for all their support throughout my college career.

## Abstract

Propagations of errors, or uncertainties, can be computed in different ways. The ISO *Guide to the Expression of Uncertainty in Measurement* (GUM) recommends an estimate based on a first-order Taylor series expansion of the functional form of the measurement equation. However, other methods may be more appropriate depending, for example, on properties of the errors in the input quantities, or nonlinearities in the measuring function.

We have implemented a new method developed by C-M Wang and Hari Iyer for calculating the uncertainty of a measurement result using principles of *generalized inference*. Using Microsoft *Excel*, we developed macros for flexible uncertainty calculations based on the generalized inference method. These macros embed scripts for the open-source *R* statistical software system (through *R DCOM Server* and an Add-in for *Excel*). Also available are procedures for uncertainty calculations based on the common first-order Taylor series expansion method, as well as a more accurate second-order Taylor series expansion method.

## Chapter 1: Introduction to Measurement Uncertainty

Any measurement that is made always involves some uncertainty. Although known about, the study of measurement uncertainty is not always given the credit it deserves in science. There are three main reasons why an understanding of measurement uncertainty is essential. First, it quantifies reliability in our measurements. When using measurements for critical scientific research, scientists need to know how much to trust their results. Secondly, understanding of measurement uncertainty allows comparison between results. Comparison methods such as hypothesis testing would not be possible without some kind of uncertainty information. Finally, understanding of measurement uncertainty is needed for uncertainty propagation. Uncertainty propagation is explained below.

Whenever a measurand (any kind of measurement, either single or compound) involves more than one input quantity (some kind of measurement), the uncertainties of each input quantity must be combined in order to obtain the uncertainty of the measurand. This process of combining uncertainties is called uncertainty propagation. This paper reviews the general methods used for computing uncertainty propagation, presents new methods that are being used, and discusses how to implement all of these methods for computational simplicity.

## Chapter 2: Models for Uncertainty Propagation

Before specific methods are discussed, it is necessary to define uncertainty propagation in mathematical terms. Let

$$M = f(m_1, m_2, \dots, m_N) . \quad (1)$$

Here  $M$  represents the true value of our measurand.  $m_i$  represents the true value of each input quantity used to calculate  $M$ , and the  $m_i$  are related to  $M$  through the functional relationship  $f$ .  $f$ , although not restricted to a formula, can be a formula describing  $M$ , such as a physical process.

In practice, however, one does not know the true values of  $m_i$ . Therefore, we approximate them with quantities  $X_i$ , which can be obtained through various methods such as a population sample. The functional relationship does not change, but now the  $X_i$  values are used instead of the  $m_i$ . As a result we get an estimated value for  $M$ , which we will call  $Y$ .

$$Y = f(X_1, X_2, \dots, X_N) \quad (2)$$

The combined uncertainty is defined as the difference between the estimated value of the measurand  $Y$  and the true value  $M$ , which we will denote as  $R$ .

$$R = Y - M = f(X_1, \dots, X_N) - f(m_1, \dots, m_N) \quad (3)$$

We now present an example. Say we want to find the volume of a box. The formula for this measurement would be

$$V = f(L, W, H) = L \times W \times H$$

where  $L$ ,  $W$ , and  $H$  stand for length, width, and height, respectively. When we take the actual measurements for input quantities  $L$ ,  $W$ , and  $H$ , we obtain the following values:

$$L \approx \hat{l} \pm u_l$$

$$W \approx \hat{w} \pm u_w$$

$$H \approx \hat{h} \pm u_h$$

where the lowercase letters with the circumflex refer to the observed values of our input quantities and  $u_i$  refers to the uncertainty associated with the  $i^{\text{th}}$  input quantity. Thus, our new formula would be

$$v = f(l, w, h) = \hat{l} \times \hat{w} \times \hat{h}$$

and the combined uncertainty would be defined as

$$R_v = v - V = f(\hat{l}, \hat{w}, \hat{h}) - f(L, W, H).$$

Now we will review the most common methods of calculating R, also known as uncertainty propagation methods.

## 2.1 Computation of Uncertainty Propagation Methods

To better keep track of the different methods for calculating uncertainty propagation, we first must state some definitions.

Standard Uncertainty: A value which specifies the uncertainty of a measurement, e.g., standard deviation.

Types of Measurements: Classification of *how* a measurement is obtained. There are only two types, type A and type B.

Type A Measurement: Measurements that are obtained through statistical analysis, thus having a value and a standard uncertainty associated with it (e.g., random samples).

Type B Measurement: Measurements that are not obtained through statistical analysis. These include measurements obtained through empirical information or measurements which may be defined through a probability density function (e.g., a normal curve or a uniform distribution).

## 2.2 Taylor Series Expansion

The most widely used method for calculating uncertainty propagation is the Taylor series expansion method and is also recommended in the ISO *Guide to the Expression of Uncertainty in Measurement* [2]. This method can only be used when an explicit formula describes the final measurand. Furthermore, this method is aimed at Type A uncertainties. The expansion is defined as follows:

$$R = \sum_{i=1}^N e_i \frac{\partial f}{\partial X_i} + \frac{1}{2} \left( \sum_{i=1}^N e_i \frac{\partial}{\partial X_i} \right)^2 f + \dots + \frac{1}{k!} \left( \sum_{i=1}^N e_i \frac{\partial}{\partial X_i} \right)^k f + \dots \quad (4)$$

where  $\frac{\partial f}{\partial X_i}$  represents the partial derivative of the formula  $f$  with respect to input quantity

$X_i$ , and  $e_i$  represents the standard uncertainty of input quantity  $X_i$ . The  $\frac{\partial f}{\partial X_i}$  are also known as sensitivity coefficients, since they act as weights to the uncertainties of each input quantity.

Although one can expand the series to infinity, this would be impractical and unnecessary since higher order terms have negligible impact on  $R$ . Usually a first-order expansion is sufficient, so Eqn. (4) simplifies to:

$$R = \sum_{i=1}^N e_i \frac{\partial f}{\partial X_i}. \quad (5)$$

Eqn. (5) is often squared, which represents the variance of  $Y$ . By combining like terms, we obtain:

$$R^2 = \sum_{i=1}^N \left(\frac{\partial f}{\partial X_i}\right)^2 e_i^2 + 2 \sum_{i=1}^{N-1} \sum_{j=i+1}^N \frac{\partial f}{\partial X_i} \frac{\partial f}{\partial X_j} e_i e_j \quad (6)$$

Next we manipulate (6) to get the expected value of the variance of  $Y$ , or the squared combined uncertainty of  $Y$ .

$$u_c^2(Y) = E(R^2) = \sum_{i=1}^N \left(\frac{\partial f}{\partial X_i}\right)^2 u^2(X_i) + 2 \sum_{i=1}^{N-1} \sum_{j=i+1}^N \frac{\partial f}{\partial X_i} \frac{\partial f}{\partial X_j} u(X_i, X_j) \quad (7)$$

Notice that the standard uncertainties  $e_i$  change into  $u(X_i)$ , which represents the expected uncertainty of  $X_i$ . Also notice that this formula allows us to include any covariance that might exist among input quantities, which is notated as  $u(X_i, X_j)$ . If no covariance exists, then that whole second term would cancel out. This formula is called the *law of propagation of uncertainty* and is what actually gets used to calculate the combined uncertainty.

Equation (7) will work for most formulas. However, if formula  $f$  is severely nonlinear, equation (7) will not weigh the uncertainties appropriately. In this case, we must expand the Taylor series to a higher order. The derivation of the second-order Taylor series expansion is tedious and complicated. Fortunately, C.M. Wang and Hari K Iyer [5] have done the work for us:

$$u_c^2(y) = E(R^2) = \sum_{i=1}^N \left(\frac{\partial f}{\partial x_i}\right)^2 u^2(x_i) + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \left(\frac{\partial^2 f}{\partial X_i \partial X_j}\right)^2 u^2(x_i) u^2(x_j) + \sum_{i=1}^N \frac{\partial f}{\partial X_i} \frac{\partial^3 f}{\partial X_i^3} u^4(x_i) + \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N \frac{\partial^2 f}{\partial X_i^2} \left(\frac{\partial f}{\partial X_j}\right)^2 u^2(x_i) u^2(x_j) + \frac{1}{4} \left(\sum_{i=1}^N \frac{\partial^2 f}{\partial X_i^2} u^2(x_i)\right)^2 \quad (8)$$

We can see that calculation of measurement uncertainty is very involved. Fortunately, computers and software can quickly apply this method and calculate combined uncertainty. Computer implementation will be discussed in Chapter 3.

### 2.3 Limitations of Taylor Series Expansion

Although the Taylor series expansion method is automatic, it has limitations. First, this method is only appropriate when non-linearities in  $f$  are not too severe. It is true that one could always use a higher-order Taylor series expansion. However, higher-order Taylor series expansions are complicated to derive, and sometimes derivatives are not easily obtainable. Finally, this method is mostly aimed at Type A uncertainties.

The next technique discussed overcomes these obstacles, but at the cost of simplicity.

### 2.4 Generalized Inference

The generalized inference method for computing uncertainty propagation is based on the Monte-Carlo method. While the Taylor series expansion deals with propagation of uncertainties, the generalized inference method deals with propagation of distributions. Instead of each input quantity being a fixed value and having an associated uncertainty, each input quantity becomes a distribution. Since each input quantity is not fixed, they are given the name *generalized pivotal quantities* (GPQ). Thus, the Monte Carlo method is used to simulate the calculation of formula  $f$  many times by randomly generating each input quantity. Random number generation is defined by the distribution of each input quantity. Through the simulation results, we gain intelligence on  $f$ .

For a practical understanding of this method, it must be explained in more detail (in relation to calculating uncertainty propagation). See Wang and its resources ([4]) for more details. Let us say we have a GPQ called  $Q$ . In our case,  $Q$  is a GPQ if it is defined by both its minimal sufficient statistics (which are obtained through observed random samples) and a random variable that is free of unknown variables. By minimal sufficient, we loosely refer to statistics that summarize all the information in our observed samples and do it in the most compact method, without losing information of our random samples (see [1]). For example,  $\bar{x}$  and  $s$  would be minimal sufficient statistics (assuming our GPQ follows a normal distribution). A random variable is a distribution and, as mentioned before, must be definable by known variables.

Following is a simple method for building a GPQ (partly originating from [4], p. 147):

- (1) Obtain a random variable that can be defined in two ways. First, by our minimal sufficient statistics and our unknown model parameter (our GPQ). Second, our random variable must also be defined by known parameter(s) (such as the degrees of freedom, or  $\mu=0$  and  $\sigma=1$  when dealing with a standard normal distribution).
- (2) Express (solve for) the model parameter as a function of the minimal sufficient statistics and the random variable.
- (3) Replace the minimal sufficient statistics with their observed values.

Once all model parameters have been transformed into GPQs, we replace them in our formula  $f$  and as a result obtain a GPQ for a combined measurement. The final GPQ is useless unless simulations are done on the formula (sometimes the distribution of the final GPQ may be derived analytically). With a large number of simulations, we can

obtain information on our measurand, such as figuring out what kind of distribution it follows or obtain the simulation's mean and standard deviation, which can be used to create confidence intervals and for other statistical purposes.

Notice also that Type B measurements can easily be incorporated into this method. The most common type B input quantities are those that are defined by normal and uniform distributions. If an input quantity were defined by a normal distribution, then the GPQ for  $\mu$  it would look something like this:

$$\hat{\mu} = \mu - \sigma Z_{0,1}$$

where  $Z_{0,1}$  is a standard normal random variable,  $\mu$  is the mean, and  $\sigma$  is the known standard deviation. If an input quantity were defined by a uniform distribution, then it would look like this:

$$\begin{aligned} \hat{\mu} &= \mu + U_{\pm r} \\ &\text{or} \\ \hat{\mu} &= U_{\pm r} \end{aligned}$$

Here  $\hat{\mu}$  can be defined as a value with a uniformly distributed uncertainty or just as a uniformly distributed uncertainty.  $\mu$  refers to our input quantity's value, and  $U_{\pm r}$  refers to uniform random variable with range  $\pm r$ .

## 2.5 Sample Problem

Referring to the volume of the box example, let us find the GPQ for the volume of the box. First, we must find the GPQ for each of the input variables. As an example, we will use the height of the box. Let us say we took  $n_h$  random samples of the height of the box and we obtained  $\hat{h}$  as the sample mean and  $u_h$  as its standard deviation. Thus, the most suitable random variable is a Student's t-distribution:

$$T = \frac{\bar{X} - \mu}{S / \sqrt{n}} \quad (9)$$

which can be defined entirely by the degrees of freedom,  $(n-1)$ . Now that we have random variable chosen, we continue with step 2 and express  $\mu$  as a function of  $\bar{X}$ ,  $S$ , and  $n$ . Thus we get this equation:

$$\mu = \bar{X} - \frac{S}{\sqrt{n}} T_{n-1} \quad (10)$$

Finally, as step 3 indicates, we replace the minimal sufficient statistics with our observed values and denote our GPQ with a tilde:

$$\tilde{h} = \hat{h} - \frac{u_h}{\sqrt{n_h}} T_{n_h-1} \quad (11)$$

We would do the same process for the other variables. We would obtain the following:

$$\tilde{l} = \hat{l} - \frac{u_l}{\sqrt{n_l}} T_{n_l-1} \quad (12)$$

$$\tilde{w} = \hat{w} - \frac{u_w}{\sqrt{n_w}} T_{n_w-1} \quad (13)$$

Finally, we would replace (11), (12), and (13) in our equation for  $f$ , which in this case describes the volume of the box. Thus, instead of being

$$V = \hat{l} \times \hat{w} \times \hat{h}, \quad (14)$$

now it will read as

$$\tilde{V} = \left( \hat{l} - \frac{u_l}{\sqrt{n_l}} T_{n_l-1} \right) \times \left( \hat{w} - \frac{u_w}{\sqrt{n_w}} T_{n_w-1} \right) \times \left( \hat{h} - \frac{u_h}{\sqrt{n_h}} T_{n_h-1} \right). \quad (15)$$

Once we have this equation, we can run as many simulations as needed and as a result obtain statistical information on the simulations. That is, we would simulate the distribution of  $\tilde{V}$  by randomly generating several thousand values from the  $T$  distribution. From this simulated distribution we can obtain a confidence interval for the mean of  $\tilde{V}$  as well as other useful statistics.

## 2.6 Method Comparison

One of the main advantages of the generalized approach is that an explicit formula is not required.  $f$  may be, for example, the median of simulated values of the combined measurand. Also, this method accepts various types of input quantities such as those defined by distributions. Nevertheless, it has drawbacks. The main drawback is that this method is not automatic, i.e., decisions must be made about each input quantity, and this often requires more information than the Taylor series expansion method.

Finally, through various examples, Wang and Iyer [4] show that confidence intervals obtained through the generalized approach tend to be wider than those obtained through the Taylor series expansion. Consequently, these intervals have a larger coverage.

## Chapter 3: Implementation of Computational Tools for Uncertainty Propagation

The many computations needed for most methods of calculation of uncertainty propagation make computers a valuable tool. Our software solution for applying the methods above considers ease-of-use. Mainly, it takes into account that *Microsoft Excel* is the most widely used data-gathering software. Thus, we want our software solution to work within *Excel*. We realize however, that *Excel* will not allow the implementation of the Taylor series expansion method because of its lack of ability to calculate partial derivatives analytically. To solve this problem, we make use of a separate program called *R* (see [3]), an open source statistical language and environment. We specifically chose *R* because it can compute partial derivatives, it provides tools allowing it to communicate with *Excel* (transparent to the user), its statistical ability is greater than *Excel*, it is more stable than *Excel*, and finally because it is free. *R* is able to

communicate with *Excel* through the Distributed Component Object Module (DCOM) provided by the *R* open-source community. The COM technology is an architecture developed by Microsoft which allows a program to set up an interface for other programs to interact with it. The distributed part means that communication can occur across networks. Also included with the server is an add-in for *Excel* called “RExcel” that allows the user to run *R* code within *Excel* in three different modes. These modes are scratchpad mode, macro mode, and worksheet functions. We make use of the macro mode for accessing *R*.

We implemented our uncertainty propagation methods in *Excel* spreadsheets by using macros and forms. These macros and forms make use of Visual Basic 6.3 for all the interface aspects such as functionality of the buttons and *R* scripts for all the computational aspects. Through *Excel* and the *R (D)COM Server* we are able to implement the Taylor series expansion method, as well as the generalized inference method, which are contained in the files “gen\_inference.xls” and “high\_order\_corrections.xls,” respectively. For a detailed instruction manual on the set-up of *R* and the *(D)COM Server*, and on how to use the two spreadsheets, see Appendix A.

### 3.1 Basic Structure

The structure of the various modules used for both uncertainty propagation methods is quite simple. A form calls various modules that form part of the method. We have basically four types of modules, as shown in Figure 1.

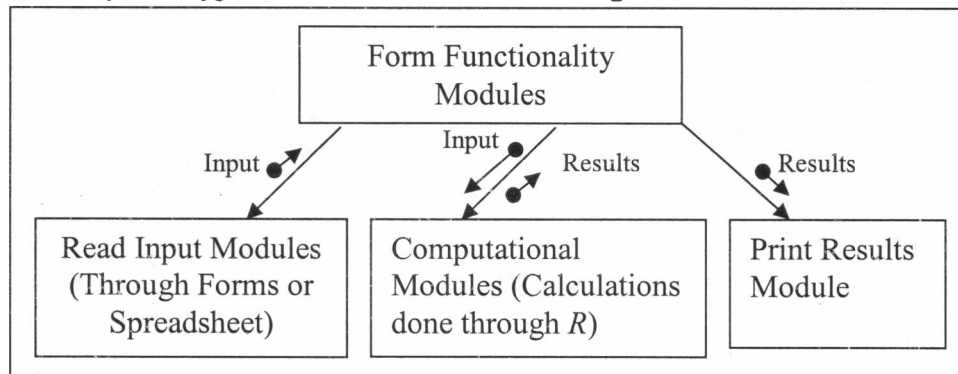


Figure 1: General Structure Chart

### 3.2 Implementing the Taylor Series Expansion Method

This method is implemented in the spreadsheet “high\_order\_corrections.xls.” The forms contained in this spreadsheet allow the user to enter all the necessary information to compute the combined uncertainty by applying either the first or second order Taylor series expansion to their data. The computation side of the method is contained in two *R* scripts, “poe1.r” (first order) and “poe2.r” (second order). These scripts can be found in Appendix B. “poe2.r” is extracted from Wang and Iyer (See [5]). If the first order Taylor series expansion is used, the user can also obtain the combined

degrees of freedom. Figure 2 shows a detailed diagram describing the structure of the Taylor series expansion method.

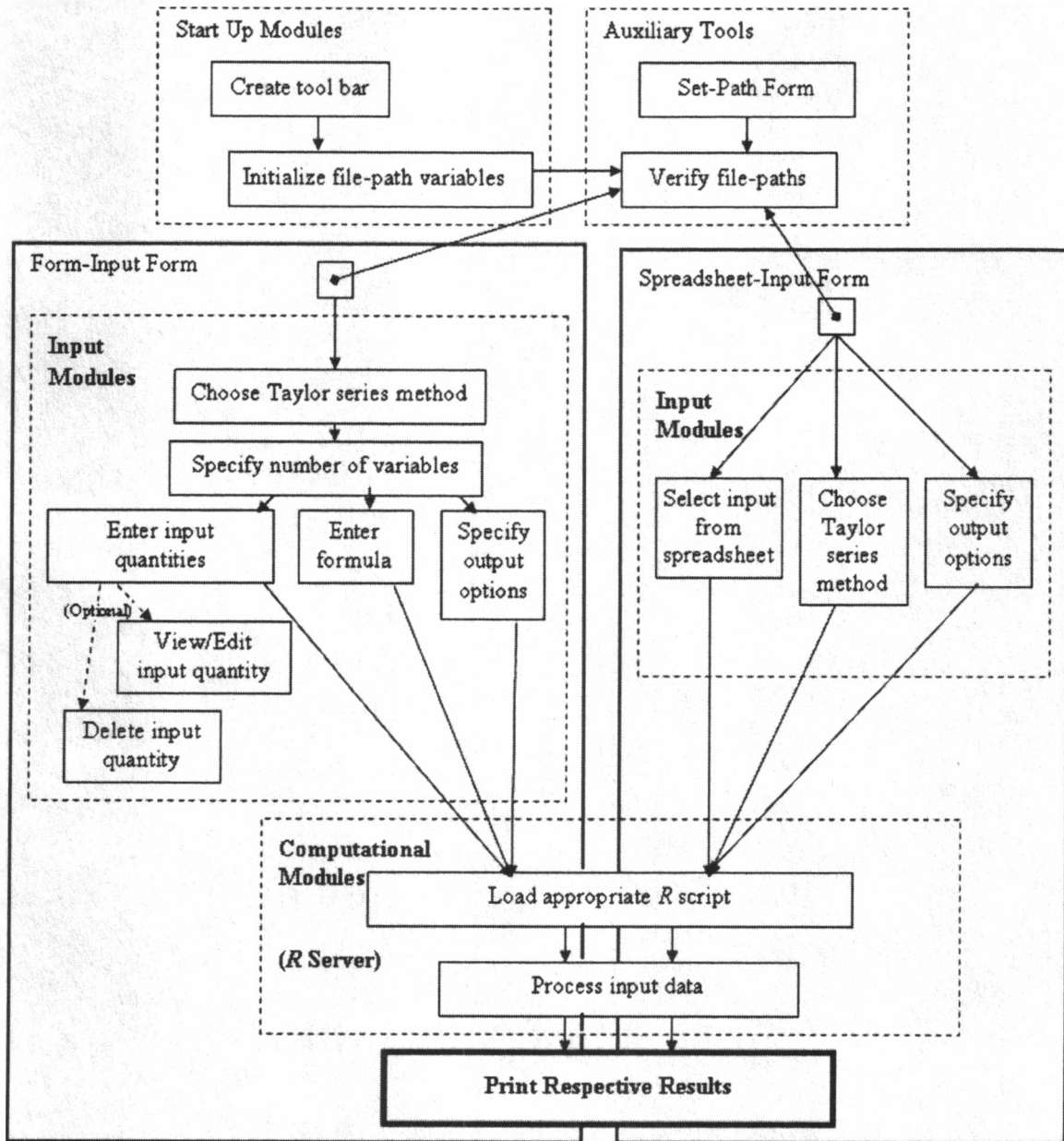


Figure 2: Structure chart for "Taylor series expansion" spreadsheet

One can start the "Form-input Form" from the Toolbar's button, "Taylor Expansion Uncertainty Propagation." Figure 3 shows the starting form when this method is used.

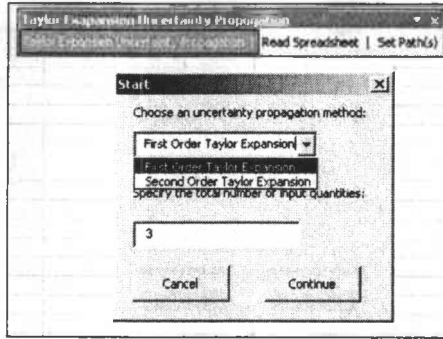


Figure 3: Form-Input Form's starting form

The "Form-input Form" method provides a simple interface that allows the user to enter all the necessary information: input quantities, the formula, and output options. Figure 4 shows an example screenshot. When entering input quantities, we are able to enter the variable name, the value, the standard uncertainty, and the degrees of freedom (degrees of freedom only works with the first order).

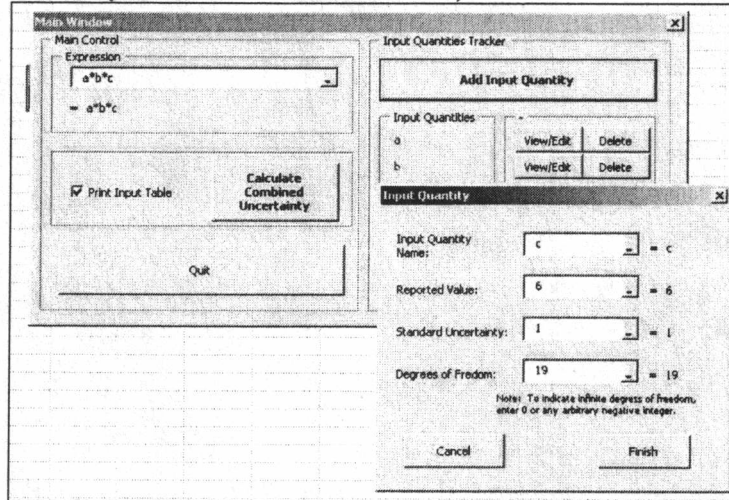


Figure 4: Form-Input Form's main forms

The spreadsheet-input form provides an easy way to input data into the Taylor series expansion method by reading a spreadsheet. This is shown in Figure 5.

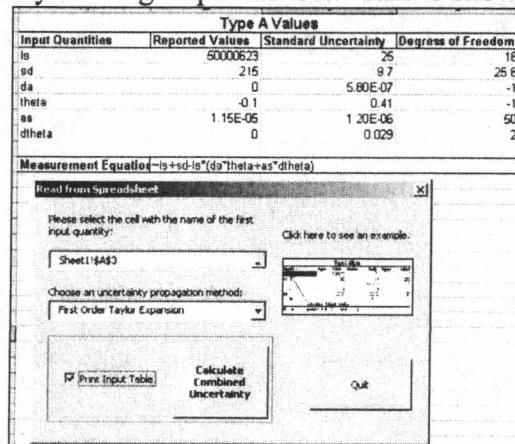


Figure 5: Spreadsheet-Input Form

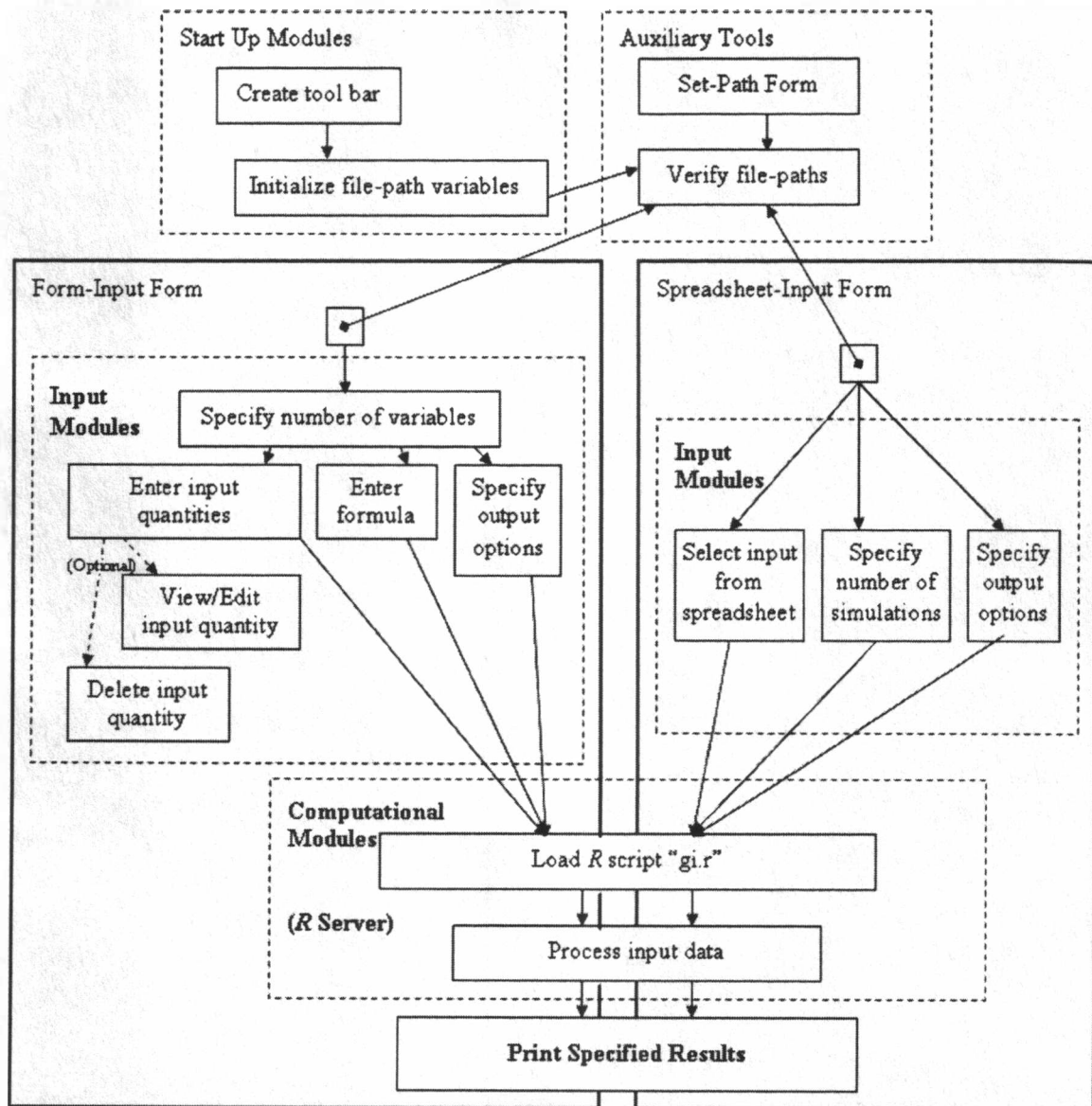
Finally, both the form-input form and the spreadsheet-input form will display results in an identical manner. Figure 6 shows example output for calculating the combined uncertainty and degrees of freedom when using a first-order Taylor series expansion.

Combined Uncertainty	Combined Degrees of Freedom	Input Quantities	Reported Values	Standard Uncertainty	Degree of Freedom
31.71	16.7	Is	5000623		18
		sd	215		25.6
		da	0	0.0000058	-1
		theta	-0.1	0.41	-1
		as	0.000115	0.000012	50
		dtheta	0	0.029	2
		Expression	$Is + sd \cdot Is \cdot (da \cdot theta + as \cdot dtheta)$		

Figure 6: Sample output

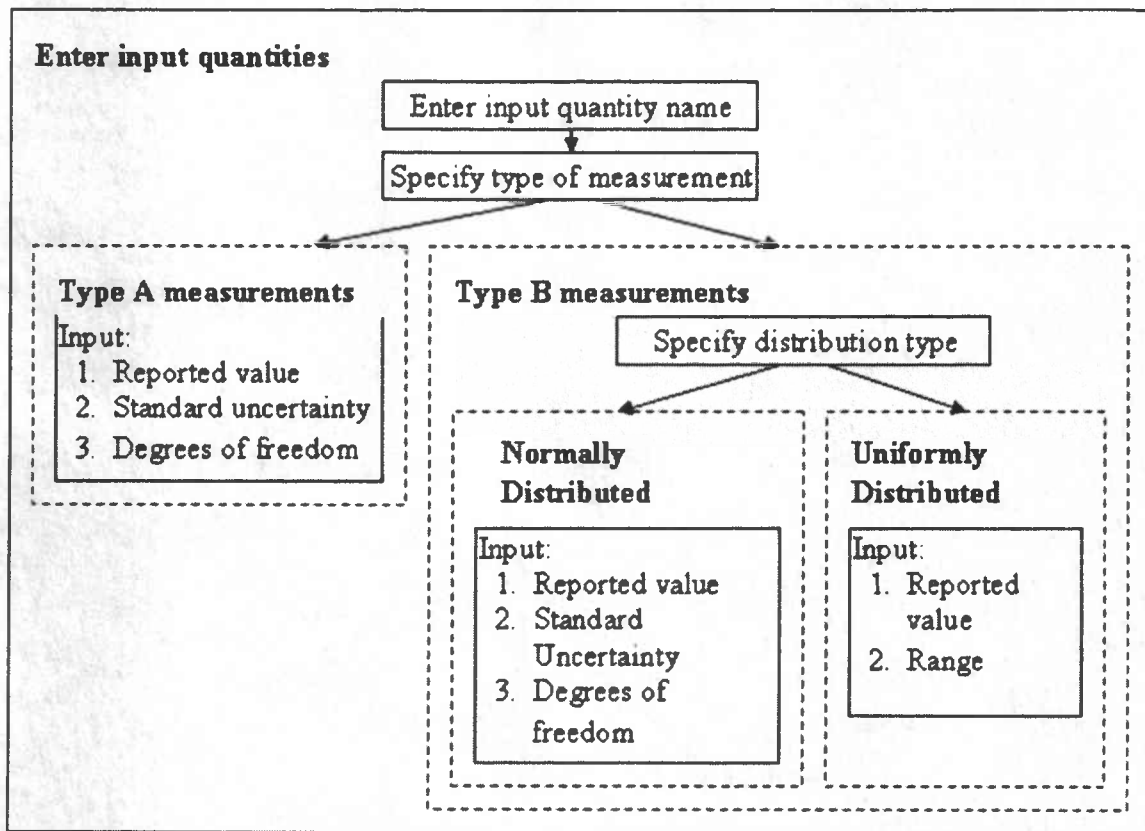
### 3.3 Implementing the Generalized Inference Method

This method is implemented in the spreadsheet “high\_order\_corrections.xls.” Due to the many decisions that must be made when using the generalized inference method, having a form that will allow the user to systematically enter all the data is useful. Our spreadsheet allows the user to run simulations on the measurand, as well as output the simulation’s mean and standard deviation. Also, it allows the user to save the input data by printing it out in a table format. Included in our spreadsheet is a macro which will read this input data. This is especially useful when one may want to change some parameters in the input data. The computation side of the method is contained in one *R* script, “gi.r,” which can be found in Appendix B. Figure 7 shows a detailed structure chart for this spreadsheet.



**Figure 7:** Generalized inference spreadsheet structure chart

The spreadsheet's "form-input form," as mentioned before is an organized way for users to enter all the necessary input. The method of entering the input quantities for this method is more involved and it requires another structure chart (see Figure 8) to demonstrate this process.



**Figure 8:** Entering input quantities for the generalized inference approach

As can be seen in Figure 9, the output options available are for showing the generated values, a histogram, and the input through a table. Also in Figure 9, in the background are shown the numerical results of a simulation. In this particular problem, we are simulating the length of a gauge block that is being calibrated. The results for this simulation, as can be seen, are a mean of 50 000 838 nm and a standard deviation of 35 nm, which can be used to obtain the 99% confidence interval of (50 000 746, 50 000 930).

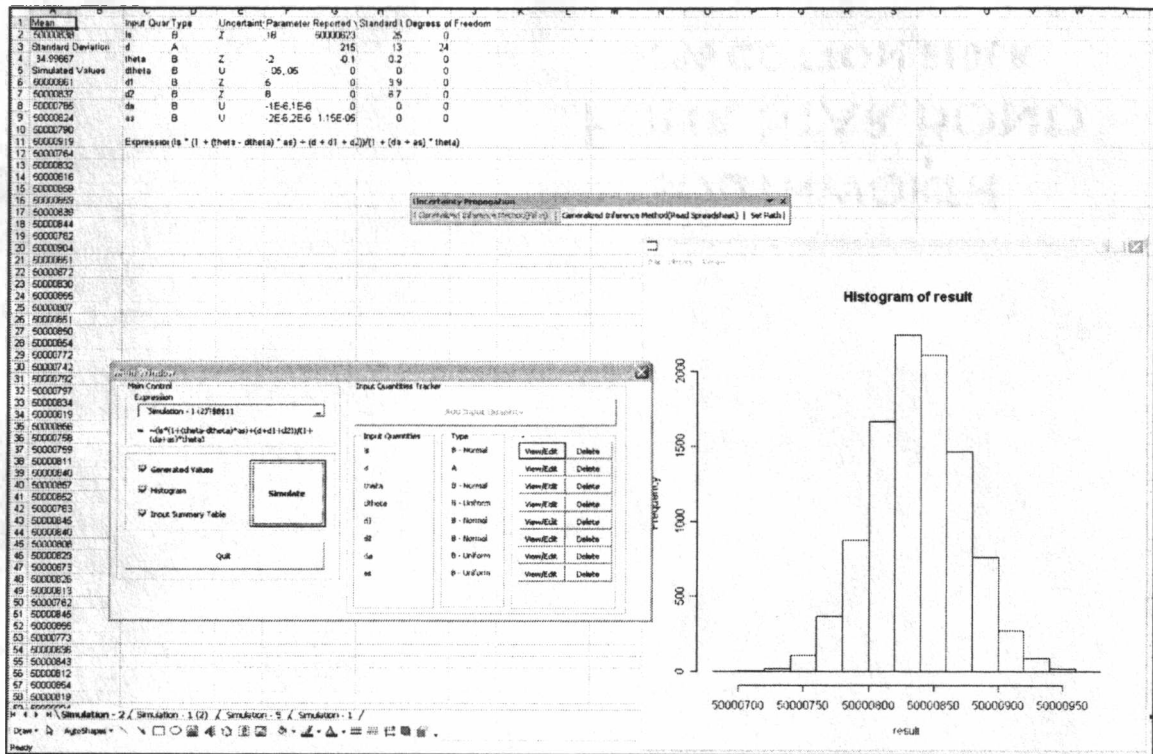


Figure 9: Form-input form for generalized inference

The spreadsheet “spreadsheet-input form” will read input from a spreadsheet. This is shown in Figure 10. Output is generated exactly the same as in the “form-input form.”

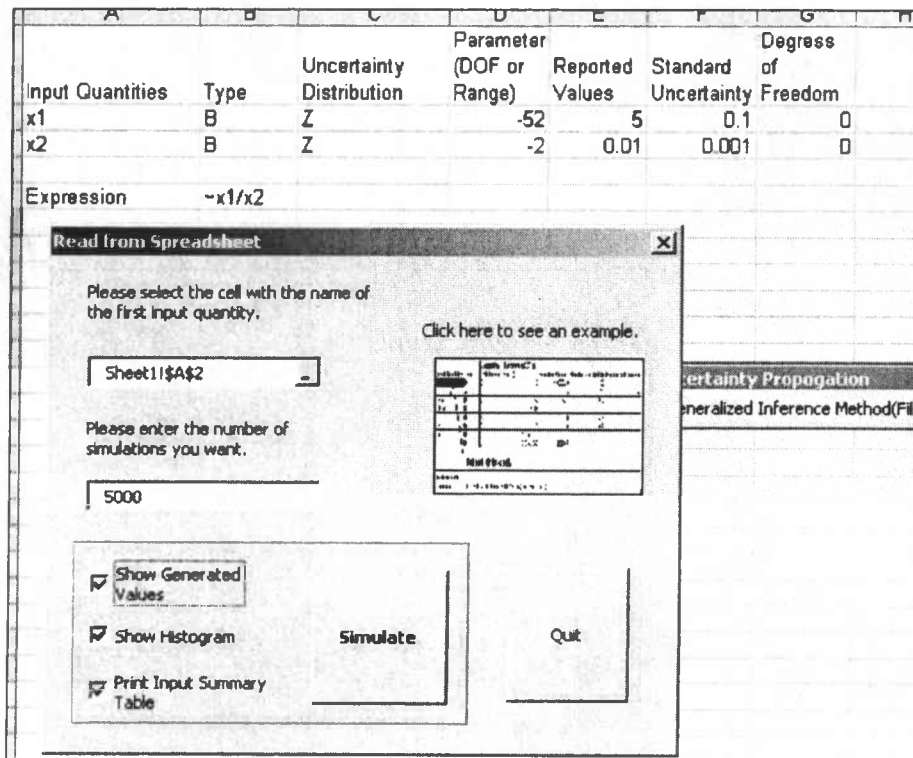


Figure 10: Spreadsheet-input form for generalized inference

### 3.4 Challenges

A major challenge was coding the functionality of the forms. Each spreadsheet's forms required about one thousand lines of code. A particular challenge was adding the "edit/view" and "delete" buttons that were associated with each input quantity (in the form-input form). Because these two buttons are dynamically created as input quantities are added, an obvious thing to try is to create an array of identical buttons, individually accessed through the array index. However, this solution failed. Buttons would show up on the form, but only the last button added would function. Apparently, an array of buttons share only one *handle* (a pointer that enables the program to access a resource) and that is why only the last button worked. This problem was solved by first creating a *class* containing both of the buttons as well as two dynamically generated labels. Then, we created an array of this class. As we found out, each element in this new array has a separate handle space for its objects (buttons and handles).

Another challenge encountered when coding the computational side of the methods was debugging *R* commands within *Excel*. Whenever an error occurred with an *R* command, *Excel* would show an empty dialogue box. Thus, one did not know what had happened. By using breakpoints with the application's debugging tool, it was possible to isolate invalid commands. Then through various tests such as trying the commands in *R* and making trial and error changes, these types of errors could be removed.

As a minor note, *R* functions are computed more slowly because *R* is an interpreted computer language, rather than a compiled computer language. An interpreted computer language means that instead of creating a binary program to run a procedure, *R* interprets commands on the spot, making it more flexible, but slower. On today's fast computers, this speed difference should be negligible.

## Chapter 4: Conclusions

The generalized inference approach is more flexible than the Taylor series expansion. Furthermore, for complex formulas that are highly non-linear, the generalized inference approach is more accurate and it usually gives wider confidence intervals than the Taylor series expansion.

Both uncertainty propagation methods benefit from computational tools. By fusing the mathematical and statistical power of *R* and the intuitive usability of *Excel*, we have created tools for uncertainty propagation that will address a vast majority of situations. The spreadsheets described in Chapter 3 present a simple and efficient interface.

Further work may involve expanding the existing tools to incorporate other uncertainty methods that address certain rare situations such as those described in [4].

## References

- [1] DeGroot M H 1989 *Probability and Statistics* 2<sup>nd</sup> edn (Menlo Park: Addison-Wesley Publishing Company).
- [2] International Organization for Standardization (ISO) 1995 *Guide to the Expression of Uncertainty in Measurement* (Geneva, Switzerland: International Organization for Standardization).
- [3] R Development Core Team 2005 *R: A Language and Environment for Statistical Computing* *R* Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.
- [4] Wang C M and Iyer H K 2005 *Metrologia* **42** 145-153.
- [5] Wang C M and Iyer H K 2005 *Metrologia* **42** 406-410.

# Appendix A

## Using *Excel* for Computing Uncertainty Propagation

This quick guide gives instructions for installing the *R (D)COM Server* and for using the *Excel* forms created for computing uncertainty propagation through the Taylor series expansion method and the generalized inference method.

### *R (D)COM Server* Set Up

If you don't have *R* installed already, read the following sub-section. If you already have *R* installed, but not the *R (D)COM Server*, jump to the "Installing the *R (D)COM Server*." If you have both of these components installed, go to the next section "Using the *Excel* Forms" to learn how to use the *Excel* form.

#### Installing *R*

1. You must install *R* in order to use the *R (D)COM Server* (unless you have a remote *R* server set up). To install *R*, first go to <http://cran.r-project.org/index.html> and choose the closest mirror to your location.
2. In the following page, click on the "Windows (95 and later)" link.
3. In the next page, click on the "base" link.
4. In the next page download the current version of the setup program (e.g. "rw2011.exe").
5. Once the download is finished, run the setup program and follow the directions.

**Note:** If you only plan on using *R* for the *Excel* form, you only need the "Minimal Installation" of *R*. If you plan to do more, doing the "Full Installation" is advisable.

#### Installing the *R-(D)COM Server*

1. Go to <http://cran.r-project.org/contrib/extra/dcom> and download the latest binary of the setup program for the server (e.g. "RSrv135.exe").
2. Once the download is finished, run the setup program and follow the directions.

**Note:** Unless you have a remote server set up for *R*, select the default installation "Local Server Installation."

# Appendix B

## **poe1.r – 1<sup>st</sup> order Taylor series expansion function**

---

```
poe.1 <- function(expr,arg,result,uncert, DOF){
#INPUT of the function:
#expr - the measurement equation in "expression"
#    mode, e.g., expression(x/y) or in "call"
#    mode, e.g., ~x/y
#arg - list of input quantities, e.g., c("x","y")
#result- reported values of input quantities
#uncert- uncertainties associated with reported values
#DOF - degrees of freedom of input quantity
#OUTPUT of the function
#root mean square error and combined degrees of freedom
#based on the first-order Taylor expansion
if(!is.expression(expr))expr<-as.expression(expr[[2]])
narg<-length(arg)
for(i in (1:narg))assign(arg[i],result[i])
dx1<-1:narg
for(i in (1:narg)){
  dx1[i]<-eval(D(expr,arg[i]))
}
delta.y <- sqrt(sum(dx1^2*uncert^2))
DOF.y <- 0
for(i in (1:narg)){
  if(DOF[i] < 0){
    add <- 0
  }else{
    add <- ((dx1[i]^4*uncert[i]^4)/DOF[i])
  }
  DOF.y <- DOF.y + add
}
DOF.y <- (delta.y^4)/DOF.y
list(rmse=delta.y,veff=DOF.y)
}
```

## **poe2.r – 2<sup>nd</sup> order Taylor series expansion function**

---

```
poe.2 <- function(expr,arg,result,uncert){
#INPUT of the function:
#expr - the measurement equation in "expression"
#    mode, e.g., expression(x/y) or in "call"
#    mode, e.g., ~x/y
#arg - list of input quantities, e.g., c("x","y")
#result- reported values of input quantities
#uncert- uncertainties associated with reported values
#OUTPUT of the function
#root mean square error and standard deviation of the
#estimate based on the second-order Taylor expansion
```

```

#
#"expression" mode or "call" mode
if(!is.expression(expr))expr<-as.expression(expr[[2]])
narg<-length(arg)
#the values that all derivatives are evaluated at
for(i in (1:narg))assign(arg[i],result[i])
#dx1 = df/dx, dx2 = d(df/dx)/dx, dx dy = d(df/dx)/dy
#dx3 = d(d(df/dx)/dx)/dx, dx1 dx2 = d(d(df/dy)/dy)/dx
dx1<-1:narg
dx2 <-1:narg
dx dy <-matrix(nrow=narg,ncol=narg)
dx3<-1:narg
if(narg>1)dx1 dx2<-matrix(nrow=narg,ncol=narg)
for(i in (1:narg)){
  dx1[i]<-eval(D(expr,arg[i]))
  dx2[i]<-eval(D(D(expr,arg[i]),arg[i]))
  dx3[i]<-eval(D(D(D(expr,arg[i]),arg[i]),arg[i]))
  for(j in(1:narg)){
    dx dy[i,j]<-eval(D(D(expr,arg[i]),arg[j]))
    if(narg>1)dx1 dx2[i,j]<-eval(D(D(D(expr,arg[i]),arg[j]),arg[j]))
  }
}
#common terms
c.term<-sum(dx1^2*uncert^2)+0.5*matrix(uncert^2,nrow=1) %*%
  dx dy^2 %*% matrix(uncert^2,ncol=1)+sum(dx1*dx3*uncert^4)
e.term1<-0
e.term2<-0
if(narg>1){
  for(i in(1:narg))
    for(j in (1:narg)){
      if(j!=i){
        e.term1<-e.term1+dx2[i]*dx1[j]^2*uncert[i]^2*uncert[j]^2
        e.term2<-e.term2+dx1[i]*dx1 dx2[i,j]*uncert[i]^2*uncert[j]^2
      }
    }
}
e.term3 <-(sum(dx2*uncert^2))^2/4
sigma.y<-sqrt(c.term+e.term1)
delta.y <- sqrt(c.term+e.term1+e.term3)
gum.y <-sqrt(c.term +e.term2)
list(rmse=delta.y[1,1],sd=sigma.y[1,1],gum2=gum.y[1,1])
}

```

## **gi.r – Generalized inference simulation function**

---

```
gi <- function(expr,type,arg,means,uncert, dof,dist_type,dist_par,iterations){
#INPUT of the function
# expr - the measurement equation in "expression"
#   mode, e.g., expression(x/y) or in "call"
#   mode, e.g., ~x/y
# type - refers to the method of evaluation ("A" or "B")
#   A: by statistical analysis through a series of observations
#   B: other than statistical analysis
# arg - list of input quantities, e.g., c("X","y")
# means - reported values of input quantities
# uncert- uncertainty associated with reported values(for type A quantities)
# dof - Degrees of Freedom(for type A distributions)
# dist_type - for type B quantities, specifies the assumed
#   distribution("Z" or "U" or empty if quantity is of type A)
#   Z: Normal
#   U: Uniform
# dist_par - parameters for assumed distributions(applys only for type B quantities)
#   for Z distributions, enter an integer that is 0 or less
#   to specify infinite degrees of freedom
# iterations - how many calculation simulations are needed

# OUTPUT of the function
#an array of all the simulated calculations
#"expression" mode or "call" mode
if(!is.expression(expr))expr<-as.expression(expr[[2]])
narg<-length(arg)

#array to keep track of the simulated values of each of the generalized pivotal quantities
gpq <- array(0,narg)

#list to record the correct random number generation command
#for each input quantity(Normal, T-dist, and Uniform)
rand <- list(1:narg)

#array to keep track of all the simulated calculations of the measurand
measurand_array<-array(0,iterations)

for(i in (1:narg)){
#if a comma is not found in dist_par, then either the input quantity
#is of type A or type B with a Z distribution
if(regexpr(",","toString(dist_par[i]))[1]==-1){
#only type B quantities with a Z-distribution can have 0 or negative values
#to specify infinite degrees of freedom. Thus if it's higher than 0,
#one is ASSUMING it's a Z-dist, so we must transform its distribution to a T-dist
#if it's 0 or lower, that input quantity truly follows a Z-dist, so we don't change it.
```

```

if(type.convert(toString(dist_par[i]))>0 || type[i] == "A" || type[i] == "a"){
  dist_type[i] <- "T"
  if(type[i] == "A" || type[i] == "a")dist_par[i] <- dof[i]
}
}

#uncert will only be 0 if "i" is a uniformly distributed input quantity.
#Thus it's uncertainty will be randomly generated
#Since the rand[i] is multiplied by the uncert[i],
  #uncert[i] must be set to 1 in order for it to count
if(uncert[i]==0)uncert[i]<-1

#This code determines the correct random generator distribution
  #to use in relation with the input quantity
rand[i]<-switch(EXPR=dist_type[i],
Z=list(call("rnorm",1,means[i],uncert[i])), z=list(call("rnorm",1,means[i],uncert[i])),
U=list(call("runif",1,
  type.convert(substr(toString(dist_par[i]),1,regexpr(",",toString(dist_par[i]))[1]-1)),
  type.convert(substr(toString(dist_par[i]),regexpr(",",toString(dist_par[i]))[1]+1,
  nchar(toString(dist_par[i]))) )),
u=list(call("runif",1,
  type.convert(substr(toString(dist_par[i]),1,regexpr(",",toString(dist_par[i]))[1]-1)),
  type.convert(substr(toString(dist_par[i]),regexpr(",",toString(dist_par[i]))[1]+1,
  nchar(toString(dist_par[i]))) )),
T=list(call("rt",1,type.convert(toString(dist_par[i])))),
t=list(call("rt",1,type.convert(toString(dist_par[i])))
)
}
for(j in (1:iterations)){
  #assigns an actual value to each input quantity
  for(i in (1:narg)){
    gpq[i] <- means[i]-uncert[i]*eval(rand[[i]])
  }
  #assign the values obtained above to the actual variables names passed in "arg"
  for(i in (1:narg))assign(arg[i],gpq[i])
  #evaluate the formula and store it
  measurand_array[j] <- eval(expr)
}
#return the array of calculated simulations
measurand_array
}

```