Index

Session 1

Discussion Paper

Numerical Techniques for Uncertainty Modelling

Michael Boyd, Kelton Engineering Ltd

Numerical Techniques For Uncertainty Modelling

Mr Michael Boyd, Kelton Engineering Ltd

1 INTRODUCTION

This paper demonstrates two numerical techniques which can be used to calculate uncertainties more efficiently than using traditional analytical methods especially where the system being investigated is complicated.

The techniques described are a 'Finite Difference' method proposed in the ISO '*Guide to the Expression of Uncertainty in Measurement*' [1], followed by an introduction to Monte Carlo modelling. Algorithms for converting conventional computer random number generators into those required for other distribution types are provided in the Appendix.

A simple spreadsheet macro is described that will allow users to build an interface for creating dynamic uncertainty models for even the most complicated systems. The method described allows users to calculate uncertainties for any system using a simple point and click interface that requires little statistical expertise.

2 PROBABILITY DISTRIBUTIONS

In the development of any uncertainty model, much consideration must be given to the probability distribution associated with each input uncertainty. The following describes some common scenarios, and how to reduce the value to a standard uncertainty - the uncertainty associated with a one standard deviation confidence level. The analytical and finite difference methods require each input uncertainty to be expressed at the same confidence level so that they can be combined.

2.1 Normal Distribution

Uncertainties quoted with normal distributions can typically only be provided when a series of test points have been taken and the standard deviation of the resulting data has been calculated. This standard deviation is then multiplied by a coverage factor Student's 't' which is a function of the number of test points available and the confidence level required - t =1.96 for an infinite number of test points and 95% confidence level). More recently, national standards agencies have been recommending the use of a coverage factor of '2' which is described as having 'a confidence level approximately equal to 95%'.

Thus the standard uncertainty is typically calculated using either

$$u(x_i) = \frac{U_{95\%}}{1.96} \tag{1}$$

or

$$u(x_i) = \frac{U_{k=2}}{2} \tag{2}$$

2.2 Rectangular Distribution

In many cases, it is incorrect to assume the data has a normal distribution: the most common mistake is where the uncertainty is described using language such as 'the maximum error is…' or 'the tolerance of the instrument is….'. In this case the uncertainty has a rectangular probability distribution. The only information that is known is the error range within which the

instrument is likely to be reading - and is equally likely to be reading anywhere within this range. The standard uncertainty is calculated using

$$u(x_i) = \frac{U_{Rect}}{\sqrt{3}} \tag{3}$$

2.3 Triangular Distribution

If an uncertainty can be described using limits such as 'minimum expected value-most likely value-maximum expected value' then a triangular distribution is the relevant distribution to use, and

$$u(x_i) = \frac{U_{Triangular}}{\sqrt{6}} \tag{4}$$

3 COMBINING UNCERTAINTIES

When each input uncertainty is known and converted to a reference condition (standard uncertainty) then the inputs may be combined to give an overall uncertainty.

3.1 Analytical

The traditional equation used for calculating uncertainties is the 'root-sum-square' method, which is expanded to account for any (often neglected) correlations between the inputs

$$u_c^2(y) = \sum_{i=1}^{N}\left(\frac{\partial f}{\partial x_i}\right)^2 u^2(x_i) + 2\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\frac{\partial f}{\partial x_i}\frac{\partial f}{\partial x_j}u(x_i)u(x_j)r\left(x_i, x_j\right) \tag{5}$$

where the partial derivatives are used to account for the sensitivity of the output to each input. It is the calculation of these partial derivatives that are often complicated or laborious.

Note the correlation coefficient $r(x_i,x_j)$ is

'1' where the inputs are fully correlated
'0' where the inputs have no correlation

If i = j the inputs are the same thus the correlation coefficient is by definition '1'. Also, where no other correlations exist then (5) reduces to a simple 'Root-Sum-Square' calculation.

3.2 Finite Difference Method

The ISO '*Guide to the Expression of Uncertainty in Measurement'* [1] describes the following simple, but powerful method for solving even the most complicated uncertainty model using just two simple equations:-

If a mathematical equation is calculated from a series of inputs, that is

$$y = f(x_1, x_2, x_3 \ldots x_i) \tag{6}$$

and a temporary variable, Z_i, is introduced

$$Z_i = \frac{1}{2}\left[f\left(x_i + u(x_i)\right) - f\left(x_i - u(x_i)\right)\right] \tag{7}$$

where

$f(x_i + u(x_i))$ Add the uncertainty value to the input x_i and re-calculate y

$f(x_i - u(x_i))$ Subtract the uncertainty value from x_i and re-calculate y

and

$u(x_i)$ The standard uncertainty of input x_i

The combined uncertainty of the whole system is calculated using

$$u_c^2(y) = \sum_{i=1}^{n}\sum_{j=1}^{n} Z_i Z_j r(x_i, x_j) \quad (8)$$

A spreadsheet macro will be described that shows how (7) and (8) can be used for any uncertainty application and rivals or exceeds the Monte Carlo method in terms of simplicity, speed of execution and accuracy.

3.3 Monte Carlo Method

The Monte Carlo technique is a simple but powerful tool for modelling complex stochastic processes. It has gained acceptance as a means of establishing the uncertainty (or *risk*) of often complicated mathematical models and is used widely in the flow measurement community as well as other such diverse areas as drilling, reservoir appraisal and financial services.

The uncertainty model again is described as a mathematical equation with a series of inputs and an output. Each input is fully described by its

- Value
- Uncertainty
- Probability distribution (associated with the uncertainty)

The inputs are randomised according to its uncertainty range and probability distribution and the result is then calculated. If this process is repeated many times then the standard deviation of the result is a measure of the uncertainty. This method will converge to the true value after an infinite number of samples, but in practise the number of samples required is dependent on the accuracy required.

Note, the basic Monte Carlo method detailed above cannot account for any correlation in the input components and requires further modification if these are present. The author's solution is presented later in this paper.

3.3.1 Normal Distribution

The Polar Marsaglia method is used to generate standard normally distributed (Gaussian) random variables. It is based on the fact that X = 2U - 1 is U(-1, 1) uniformly distributed if U is U(0, 1) distributed and that for two such random variables X_1 and X_2 with $W = X^2_1 + X^2_2 <= 1$,

Then

$Y_1 = X_1$ Sqrt(-2 log(W)/W)

$Y_2 = X_2$ Sqrt(-2 log(W)/W)

The code actually used [2] is presented below and accepts the mean and standard deviation as inputs and the number returned Y is randomised around the mean.

3.3.2 Triangular Distribution

The triangular distribution function is calculated using the 'Inverse Transformation' method in which the mathematical equations describing the probability density functions for the uniform U(0,1) and triangular distributions are integrated, equated then inverted.

The code actually used is presented in the Appendix and accepts the mean, m, and function half width, a, where

a = Max Expected Value – Mean = Mean – Minimum Expected value.

Note the code could be simply adapted to allow the mean, maximum and minimum values to be accepted. This is a more powerful option as in some uncertainties the most likely expected value is not necessarily equidistant from the maximum and minimum value.

Note the actual code presented was found on a since discontinued website [3]

3.3.3 Rectangular Distribution

The rectangular distribution is usually available in standard software packages in the range U(0,1), this is modified to shift this range to a mean expected value plus or minus the expected tolerance.

The function accepts the mean, m, and the function half width (tolerance), a, where

a = Max Expected Value – Mean = Mean – Minimum Expected value.

3.3.4 Verification

The best way to verify that each of the described probability distribution functions described are valid is to generate many random numbers and to analyse the standard deviation of the numbers generated, as stated previously, the standard deviation of the normal, rectangular and triangular distributions are expected to be 1, $\sqrt{3}$ = 1.732 and $\sqrt{6}$ = 2.449 respectively. Figs 1-3 show the data collected after 1×10^6 trials.

Fig. 4 shows the effect of repeated trials and demonstrates the data falling within a band corresponding to $\pm 1/\sqrt{N} \times 100\%$ which is consistent with probabilistic theory both of which demonstrate that the Monte Carlo method requires a minimum of 1×10^6 trials to provide uncertainty estimates consistently within 0.1% of the analytically derived (exact) value.

Fig. 1. Probability Distribution - Normal

Inputs	
Mean	100
a	1
Dist. Type	Normal
No Samples	1.E+06
ChartWidth	10
Outputs	
Std. Dev'n	1.000171188
Theoretical Std. Dev'n	1.000000
Δ%	0.02

	Interval	N
	95.00	0
1	95.10	0
2	95.20	1
3	95.30	2
4	95.40	4
5	95.50	0
6	95.60	7
7	95.70	4
8	95.80	10
9	95.90	14
10	96.00	18
12	96.20	38
13	96.30	55
14	96.40	88
15	96.50	119
26	97.60	2481
27	97.70	3151
61	101.10	20747
62	101.20	18366
63	101.30	15986
64	101.40	14125
65	101.50	11917
66	101.60	10215
67	101.70	8773
68	101.80	7272
69	101.90	5955
70	102.00	4762
71	102.10	3948
74	102.40	2008
75	102.50	1591
76	102.60	1146
97	104.70	1
98	104.80	0
99	104.90	0
100	105.00	0

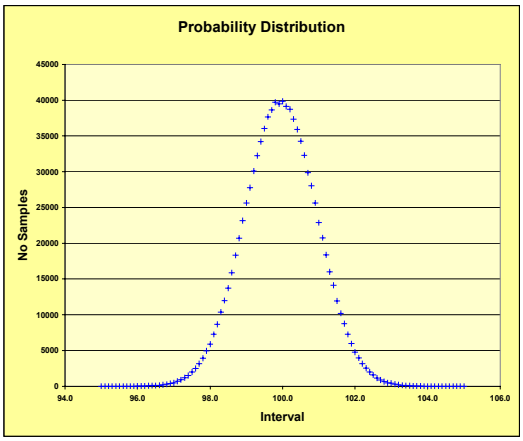


Fig. 2. Probability Distribution - Rectangular

Inputs	
Mean	100
a	1
Dist. Type	Rectangular
No Samples	1.E+06
ChartWidth	2
Outputs	
Std. Dev'n	0.577088791
Theoretical Std. Dev'n	0.577350
Δ%	-0.05

	Interval	N
	99.00	0
1	99.02	9900
2	99.04	9988
3	99.06	10066
4	99.08	10138
5	99.10	9810
6	99.12	9841
7	99.14	9993
8	99.16	10036
9	99.18	10089
10	99.20	9952
12	99.24	10059
13	99.26	9919
14	99.28	9918
15	99.30	9972
26	99.52	10114
27	99.54	9903
61	100.22	10056
62	100.24	10115
63	100.26	9992
64	100.28	9998
65	100.30	9910
66	100.32	10365
67	100.34	9986
68	100.36	9984
69	100.38	9988
70	100.40	10030
71	100.42	9931
74	100.48	9942
75	100.50	10038
76	100.52	9933
97	100.94	10007
98	100.96	9840
99	100.98	10115
100	101.00	0

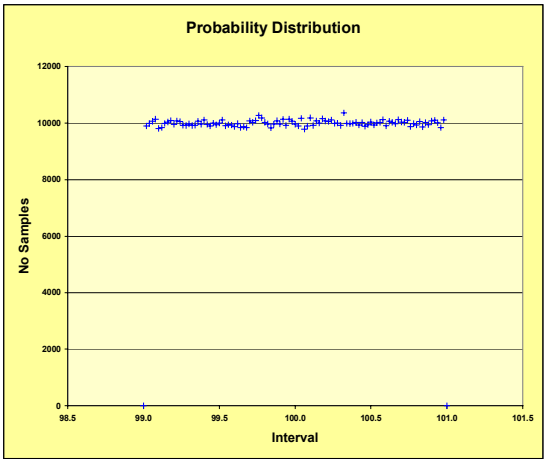


Fig. 3. Probability Distribution - Triangular

Inputs	
Mean	100
a	1
Dist. Type	Triangular
No Samples	1.E+06
ChartWidth	2
Outputs	
Std. Dev'n	0.408350291
Theoretical Std. Dev'n	0.408248
Δ%	0.02

	Interval	N
	99.00	0
1	99.02	626
2	99.04	1020
3	99.06	1365
4	99.08	1755
5	99.10	2264
6	99.12	2619
7	99.14	3021
8	99.16	3472
9	99.18	3730
10	99.20	4341
12	99.24	5015
13	99.26	5445
14	99.28	5852
15	99.30	6197
26	99.52	10647
27	99.54	11033
61	100.22	15435
62	100.24	14832
63	100.26	14643
64	100.28	14408
65	100.30	13814
66	100.32	13303
67	100.34	12862
68	100.36	12598
69	100.38	12065
70	100.40	11708
71	100.42	11501
74	100.48	9992
75	100.50	9673
76	100.52	9385
97	100.94	1002
98	100.96	665
99	100.98	202
100	101.00	0

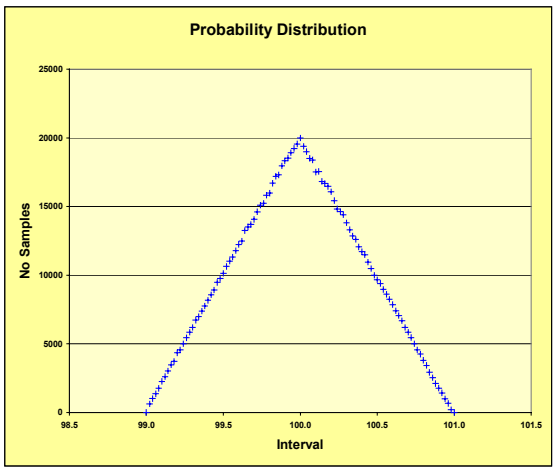
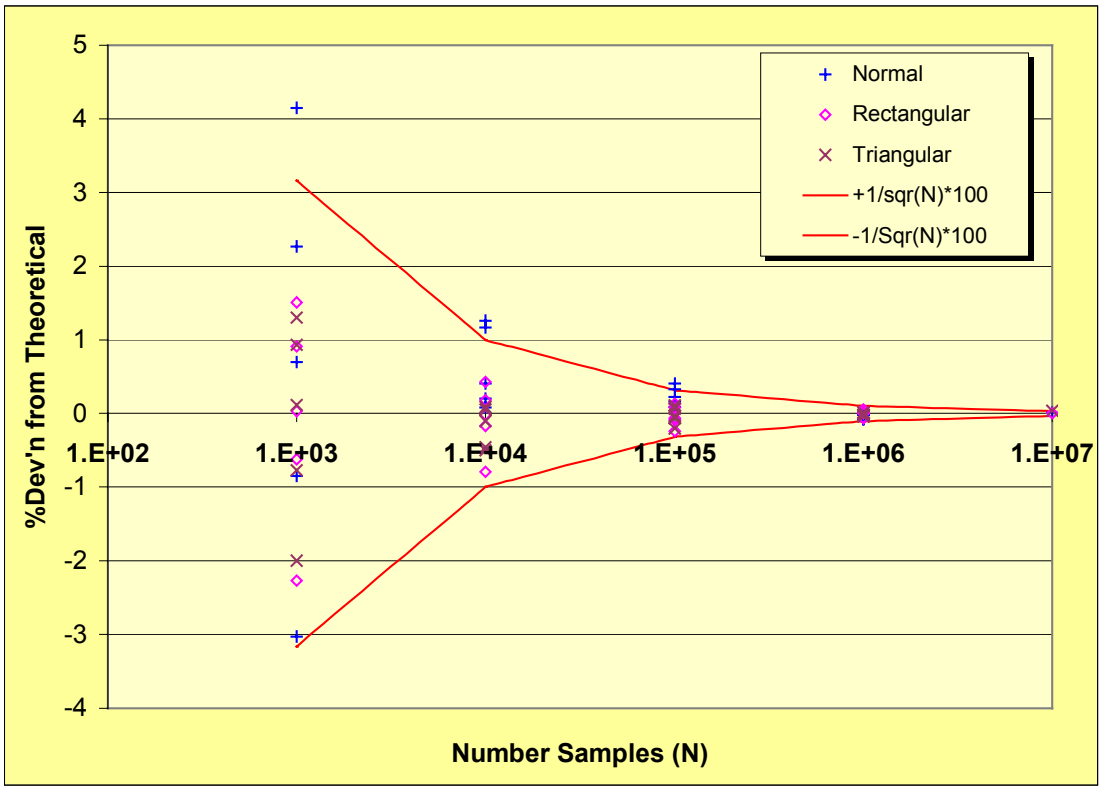


Fig. 4. Random Variation

Distribution Type	No Samples	ΔSD	Expected Deviation
Normal	1.E+03	-3.03	3.16
	1.E+04	0.08	1.00
	1.E+05	0.33	0.32
	1.E+06	-0.07	0.10
	1.E+07	0.01	0.03
Rectangular	1.E+03	0.03	
	1.E+04	0.18	
	1.E+05	-0.06	
	1.E+06	0	
	1.E+07	0.006	
Triangular	1.E+03	0.12	
	1.E+04	0.09	
	1.E+05	0.11	
	1.E+07	0.04	



3.3.5 Correlated Inputs

The traditional Monte Carlo method does not take into account any correlations between any input components, hence is effectively calculating the following component of Eq. (1).

$$u_c^2(y) = \sum_{i=1}^{N} \left(\frac{\partial f}{\partial x_i} \right)^2 u^2(x_i) \tag{9}$$

In order to calculate the effect due to correlations we must further estimate

$$2 \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} u(x_i) u(x_j) r(x_i, x_j) \tag{10}$$

which can be done using a double For-Next type loop and the sensitivity coefficients may be calculated using

$$\frac{\partial f}{\partial x} = \frac{f_2 - f_1}{x_2 - x_1} \tag{11}$$

where f_2 is the recalculated function where x_2 is varied from its initial value x_1 by a small amount, say 0.001%. As this is a modifier for the Monte Carlo method, this is calculated for each iteration (randomised value) of x_i and x_j.

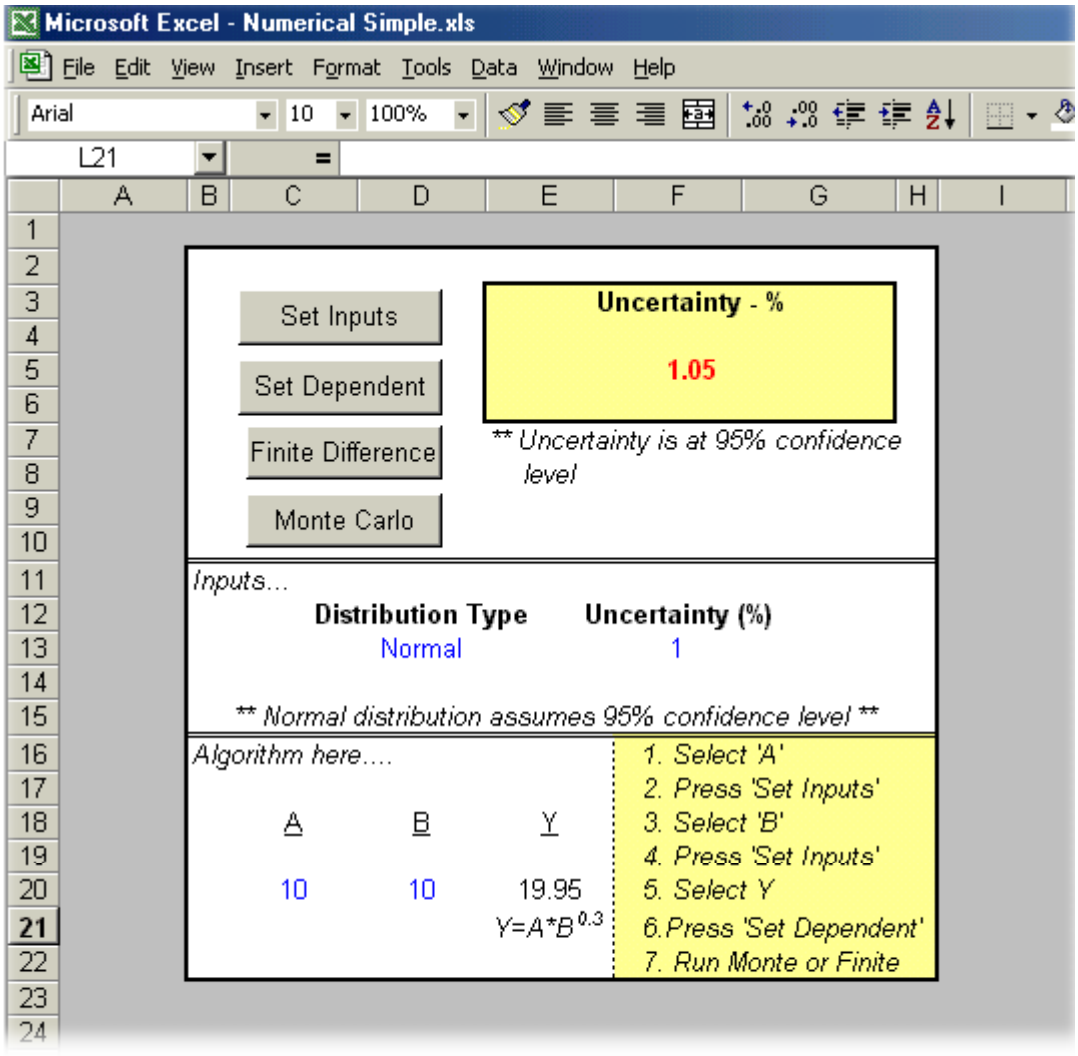
4 SPREADSHEET IMPLEMENTATION

The following describes a spreadsheet model, Fig. 5, that will allow the user to calculate even the most complex uncertainty model by simply pointing and clicking on the worksheet. The user can select to run either the Finite Difference or Monte Carlo methods.

The four control buttons must be assigned to the macro with the equivalent name, and the overall uncertainty, distribution type and associated uncertainty are expected to be in cells [F5], [D13] and [F13] respectively. These are the only limitations of the supplied software and the calculation area need not be limited to, or complexity of or the area shown on the example,

Note, the code supplied (at the end of the paper) has been deliberately simplified to ensure the code is as readable and understandable as possible, hence the omission of for example even the simplest error checking.

Fig. 5. Uncertainty Analysis Spreadsheet



5 SPREADSHEET EXAMPLE – HYDROCARBON ALLOCATION

Many new North Sea developments involve smaller satellite fields are evacuated through existing infrastructure and a multiphase flowmeter (MPM) is used to measure one of the field's production and the other is measured 'by-difference' saving on the cost of installing a new separator and dedicated fiscal metering. The fields are typically allocated fuel and flare prorated on field production rates which are then used to allocate the field's actual export rates.

In the model shown, see Table 1, the inputs - MPM, fuel, flare and export gas are measured with 10, 5, 5 and 1% respectively each considered to be a tolerance limit (rectangular distribution) and the export meter is considered to be normally distributed, k=2.

Table 1 Allocation Model

Allocation Model	mmscf	% Unc'ty	Distribution	Coverage	
Prod 'B' = MPM measured value	30	10	Rectangular		***Input*
Prod 'A' = Export + Fuel + Flare - Field 'B'	85				
Production = Prod 'A' + Prod 'B'	115				
Metered Platform Fuel	10	5	Rectangular		***Input*
Fuel 'A' = Prod 'A' / Production * Fuel	7.39				
Fuel 'B' = Fuel - Fuel B	2.61				
Metered Platform Flare	5	5	Rectangular		***Input*
Flare 'A' = Field 'A' / Production * Flare	3.70				
Flare 'B' = Flare - Flare 'B'	1.30				
Meter Platform Export	100	1	Normal	2	***Input*
Export 'A' = Prod 'A' - Fuel 'A' - Flare 'A'	73.9				***Dependent*
Export 'B' = Export - Export 'A'	26.1				

The allocation arithmetic is contained within a spreadsheet and the following procedure shows how simple it is to calculate the uncertainty using the spreadsheet macro provided. Note the actual macro used differs from that provided, but the changes are cosmetic, the core functionality is the same.

Step 1 Set each 'input' uncertainty, Fig. 6
Step 2 Set the 'dependent' cell, Fig. 7
Step 3 Set the target cell for the resulting uncertainty, Fig. 8
Step 4 Select 'run' too initiate the calculation
Step 5 The model was set for 'Finite Difference' hence answer immediately displayed, Fig. 9
Step 6 Try Monte Carlo, set option, Fig. 10
Step 7 Each input cell is randomised and the standard deviation displayed on the result for each iteration. One million spreadsheet updates to calc within 0.1%.

Note the by-difference method means that the uncertainties are dependent on the relative field production rates, hence prior to startup, the production scenario for field life should be run. Also the regulator may be interested in allocated fuel and flare uncertainties.

Fig. 6

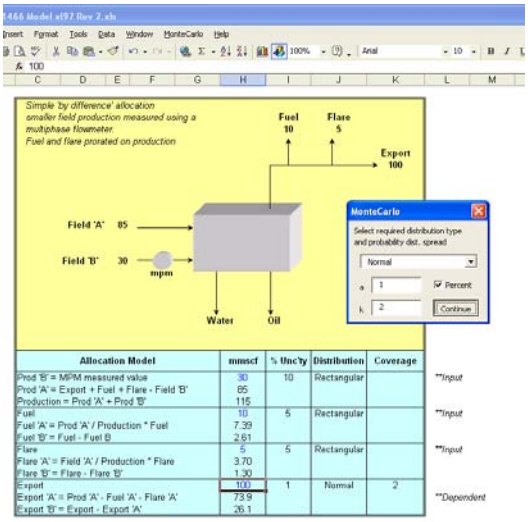


Fig. 7

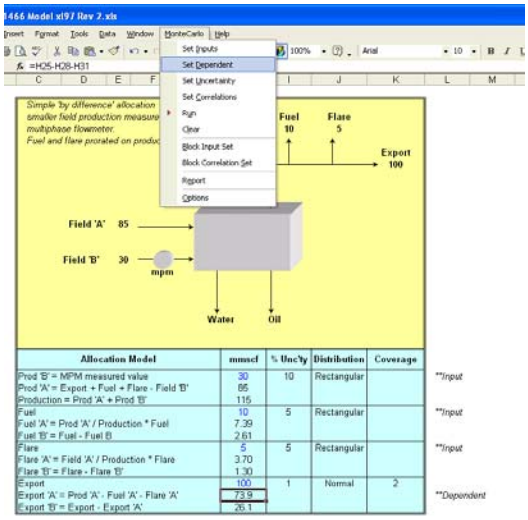


Fig. 8

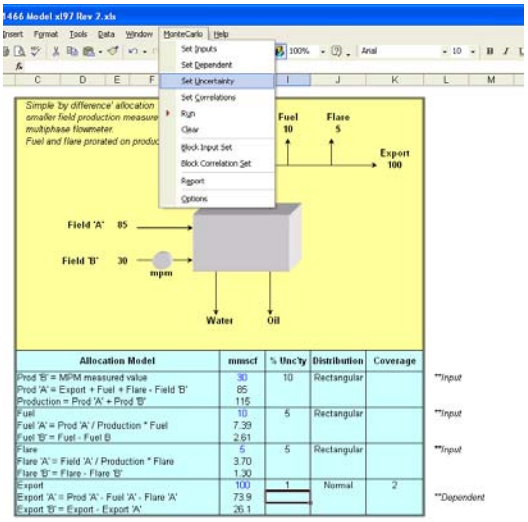


Fig. 9

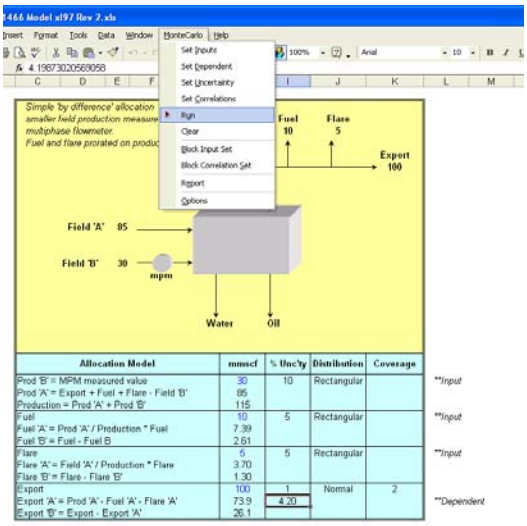


Fig. 10

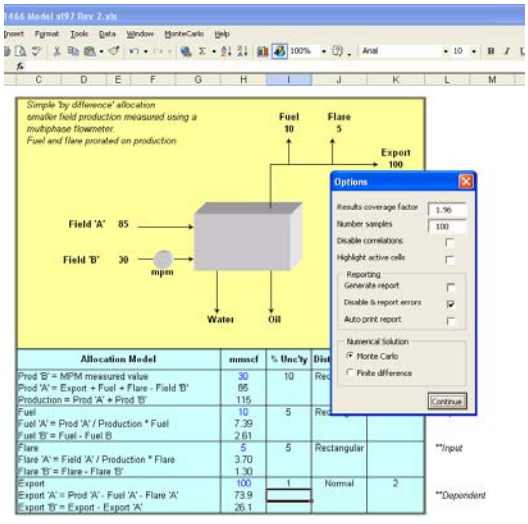
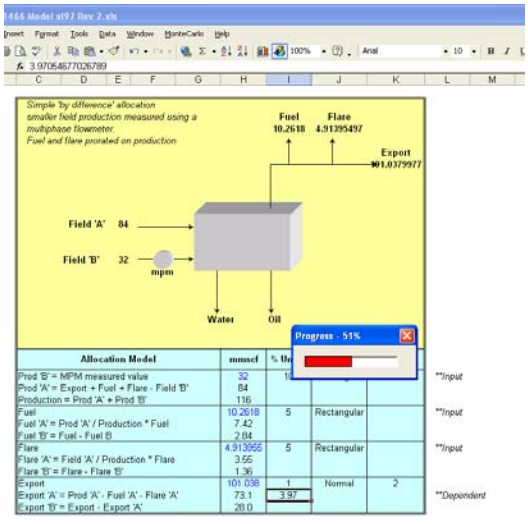


Fig. 11



6 SUMMARY AND CONCLUSIONS

Uncertainty models are often complicated or laborious to calculate, primarily due to the need to calculate the partial derivatives of the output with respect to each input. In many cases this leads to the sensitivity coefficients and the correlations between the inputs being ignored and the calculation is simplified to a simple Root-Sum-Square estimate. This can lead to uncertainty estimates which are greater or smaller than the actual value.

While the benefit of the Monte Carlo method has been explained in detail in previous papers presented to the NSFMW, this paper supplies the functions needed to generate the random numbers.

A Finite Difference method has been described that has all of the advantages of the Monte Carlo method and can be programmed such that it can be used as a 'black box' (in a similar fashion to commercial packages such as Crystal Ball) but requires no iterations. The Monte Carlo method requires over one million calculations to ensure convergence with the true value of 0.1%. Further. the Finite Difference method is traceable to published standards whereas no known standards exist for randomising functions. Finally the Finite Difference method is a 'deterministic' calculation (one correct answer) which can be verified/audited within the precision available on the respective computers being used to calculate the values.

The author would therefore like the readers to ask themselves…..'*do you really need Monte Carlo*?'.

7 REFERENCES

1. BIPM, IEC, IFCC, ISO, IUPAC, IUPAP, OIML, *Guide to the Expression if Uncertainty in Measurement*. International Organisation for Standardisation, Geneva, Switzerland.

2. Tompone, R J. Ohio State University
http://psweb.sbs.ohio-state.edu/faculty/rtimpone/computer_resources/polar.htm

3. Website (since discontinued): http://people.delphi.com/rrutt.triangle.txt

Listing (using Microsoft Excel VBA)

```
Dim Inputs() As Range
Dim Dependent As Range
Dim Uncertainty As Range
Dim u#()
Dim k#()
Dim CorrMatrix#()
Dim DistType$()
Dim n&

'Stores the input values collected from worksheet in variables
Sub SetInputs()
    Dim i&, ID&

    n = n + 1
    ID = n
    ReDim Preserve Inputs(ID), u(ID), DistType(ID), k(ID)

    Set Inputs(ID) = Application.ActiveCell

    'Assign the uncertainty and associated distribution type
    u(ID) = [F13] / 100 * Inputs(ID)
    DistType(ID) = [D13]
End Sub

'Set the dependent cell
Sub SetDependent()
    Set Dependent = Application.ActiveCell
End Sub

'Runs the Finite Difference method routine
Sub Finite()
    Dim Z#(), i&, j&, Y1#, Y2#, u2y#
    ReDim Z(n), CorrMatrix(n, n)

    'Collect the correct coverage factor and recalculate the
    'standard uncertainty for each input
    For i = 1 To n
        If DistType(i) = "Triangular" Then
            k(i) = Sqr(6)
        ElseIf DistType(i) = "Rectangular" Then
            k(i) = Sqr(3)
        ElseIf DistType(i) = "Normal" Then
            k(i) = 1.96
        End If
        u(i) = u(i) / k(i)
    Next i

    'Vary the input cells and calculate the answer to get Z
    For i = 1 To n

        Inputs(i) = Inputs(i) - u(i)
        Application.Calculate
        Y1 = Dependent

        Inputs(i) = Inputs(i) + u(i) * 2
        Application.Calculate
        Y2 = Dependent

        Inputs(i) = Inputs(i) - u(i)
        Application.Calculate

        Z(i) = 0.5 * (Y2 - Y1)

    Next i

    For i = 1 To n
        For j = 1 To n
            If i = j Then CorrMatrix(i, j) = 1

            u2y = u2y + Z(i) * Z(j) * CorrMatrix(i, j)
        Next j
```

```
    Next i

    [E5] = Sqr(u2y) / Dependent * 100 * 1.96

    'Reset all variables to 0 to prevent errors...
    End

End Sub


'Runs the Monte Carlo method routine
Sub Monte()
    Dim InitInputs() As Double, i&
    ReDim InitInputs(n)

    'Store the initial cell values
    For ID = 1 To n
        InitInputs(ID) = Inputs(ID)
        InitDependent = Dependent
    Next ID

    For i = 1 To 1000
        For ID = 1 To n
            'Select the distribution type and randomise accordingly
            If DistType(ID) = "Normal" Then
                Inputs(ID) = Gaussian(InitInputs(ID), u(ID) / 1.96)
            ElseIf DistType(ID) = "Rectangular" Then
                Inputs(ID) = Rectangular(InitInputs(ID), u(ID))
            ElseIf DistType(ID) = "Triangular" Then
                Inputs(ID) = Triangular(InitInputs(ID), u(ID))
            End If
        Next ID
            'Calculate and update the running uncertainty
            [E5] = SD(CDbl(Dependent), i) / InitDependent * 100 * 1.96

            'Update the screen every 100 calculations
            If i Mod 100 = 0 Then
                Application.ScreenUpdating = True
            End If
    Next i

    'Restore the original values
    For ID = 1 To n
        Inputs(ID) = InitInputs(ID)
    Next ID

    'Reset all variables to 0 to prevent errors...
    End

End Sub

'Calculates the standard Deviation
Function SD(x#, i&)
    Static SigmaX#, SigmaX2#

    SigmaX2 = SigmaX2 + x ^ 2
    SigmaX = SigmaX + x

    If i > 5 Then 'Prevent errors when too few samples
    SD = Sqr((i * SigmaX2 - SigmaX ^ 2) / (i * (i - 1)))
    End If
End Function

'Normal distribution function
Function Gaussian(m#, a#)
'Extracted from
'http://psweb.sbs.ohio-state.edu/faculty/rtimpone/computer_resources/polar.htm
    Dim X1#, X2#, w#, Y1#
    Dim Y2#
    Dim use_last%

    If a = 0 Then
        Gaussian = m
    End If

    If (use_last) Then
        Y1 = Y2
```

```
        use_last = 0
    Else
        Do
            X1 = 2 * Rnd() - 1
            X2 = 2 * Rnd() - 1
            w = X1 * X1 + X2 * X2
        Loop While (w >= 1)

        w = Sqr((-2# * Log(w)) / w)
        Y1 = X1 * w
        Y2 = X2 * w
        use_last = 1
    End If

    Gaussian = (m + Y1 * a)
End Function

'Triangular distribution function
Function Triangular(m#, a#)
    'extracted from
    'from http://people.delphi.com/rrutt.triangle.txt
    Dim Max#, Min#, Most#, ca#, ba#, cb#, r#

    If a = 0 Then
        Triangular = m
        Exit Function
    End If

    Max = m + a
    Min = m - a
    Most = m

    ca = Max - Min
    ba = Most - Min
    cb = Max - Most

    r = Rnd()

    If r <= (ba / ca) Then
        Triangular = Min + Sqr(r * ba * ca)
    Else
        Triangular = Max - Sqr((1 - r) * cb * ca)
    End If
End Function

'Rectangular distribution function
Function Rectangular(m#, a)
    Dim r#

    If a = 0 Then
        Rectangular = m
    End If

    r = Rnd
    Rectangular = (m - a) + (2 * a) * r

End Function
```