LA-UR-15-23986
Approved for public release; distribution is unlimited.

Title: Whisper Source Code Inspection Report

Author(s): Sartor, Raymond Francis
Brown, Forrest B.

Intended for: Report

Issued: 2015-05-28

TABLE OF CONTENTS

1. Introduction

A method using MCNP6 with sensitivity and uncertainty data to determine the baseline Upper Subcritical Limits is discussed in:

- LA-UR-14-26558, *Whisper: Sensitivity/Uncertainty-Based Computational Methods and Software for Determining Baseline Upper Subcritical Limits*
- LA-UR-14-23202, *Methodology for Sensitivity and Uncertainty-Based Criticality Safety Validation*
- LA-UR-14-23352, *Validation of MCNP6.1 for Criticality Safety of Pu-Metal, -Solution, and –Oxide Systems*

The Whisper software (Version 1.0.0) was developed to implement the methodology described in these three documents (although LA-UR-14-23202 and LA-UR-14-23352 do not reference Whisper by name). The user instructions for the Whisper program and associated script files are given in:

- LA-UR-14-26436, *User Manual for Whisper (v1.0.0), Software for Sensitivity- and Uncertainty-Based Nuclear Criticality Safety Validation*

However, none of these documents explain the internal operation of the Whisper code. This report documents an inspection of the Whisper Fortran source code. As a result of this inspection, this report explains how (and confirms) the methodology was implemented in the source code. The following sections document the algorithms, data structures, and program structure in Whisper. When Whisper implements a specific equation in LA-UR-14-26558, a cross reference to the equation is made. Also, Appendix A provides a cross reference from the equations in LA-UR-14-26558 to the implementing subroutine or function.

2. Conclusions

During the inspection of the Whisper Fortran source code, several questions were raised. These questions are documented in Appendix B. Subsequent correspondence with XCP-3, the group that developed Whisper, resolved the majority of these issues. The following issues identified by the inspection require further action:

1. For benchmarks with a zero (or unrealistically low) experimental uncertainty, Whisper calculates an estimate of the experimental uncertainty. However, after changing the experimental uncertainty, Whisper does not update the bias uncertainty. The original input value is used in the calculations for calculational margin, application bias, and application bias uncertainty. This issue is identified in a problem report (NCS-SQM-WHISPER-PROBID07) in order to initiate corrective actions. (Appendix B, Item 9)

2. When performing benchmark rejection, the fractional difference between the calculated and experiment k_{eff} values is used in the test to determine whether any (additional) benchmarks should be rejected but the magnitude difference is used to select which benchmark to reject. This inconsistency should have a negligible influence on the Whisper results but a problem report (NCS-SQM-WHISPER-PROBID06) has been drafted in order to initiate corrective actions. (Appendix B, Item 10)

3. Software Purpose

The Whisper Fortran program has three primary purposes:

1. evaluate an application model and calculate the baseline USL,
2. reject inconsistent benchmark cases, and
3. generate adjusted covariance data.

Regarding the covariance data, Whisper is required in two circumstances, generating the initial covariance data (which is distributed with Whisper) and replacement covariance data. The following flowcharts illustrate the inputs to Whisper for these four processes. Other than Whisper, the programs and script files in the flowcharts are documented in the *Whisper Program Suite Validation and Verification Report*.

In the process flow charts, the computer programs are represented by black boxes and text. The program input arguments shown in square brackets [] are optional. The black arrows indicate that a program automatically initiates the execution of the next program. If necessary, the computer programs are numbered to illustrate the execution order. The red blocks, text, and arrows represent the user input files for this process. Other data files have blue blocks, text, and arrows.

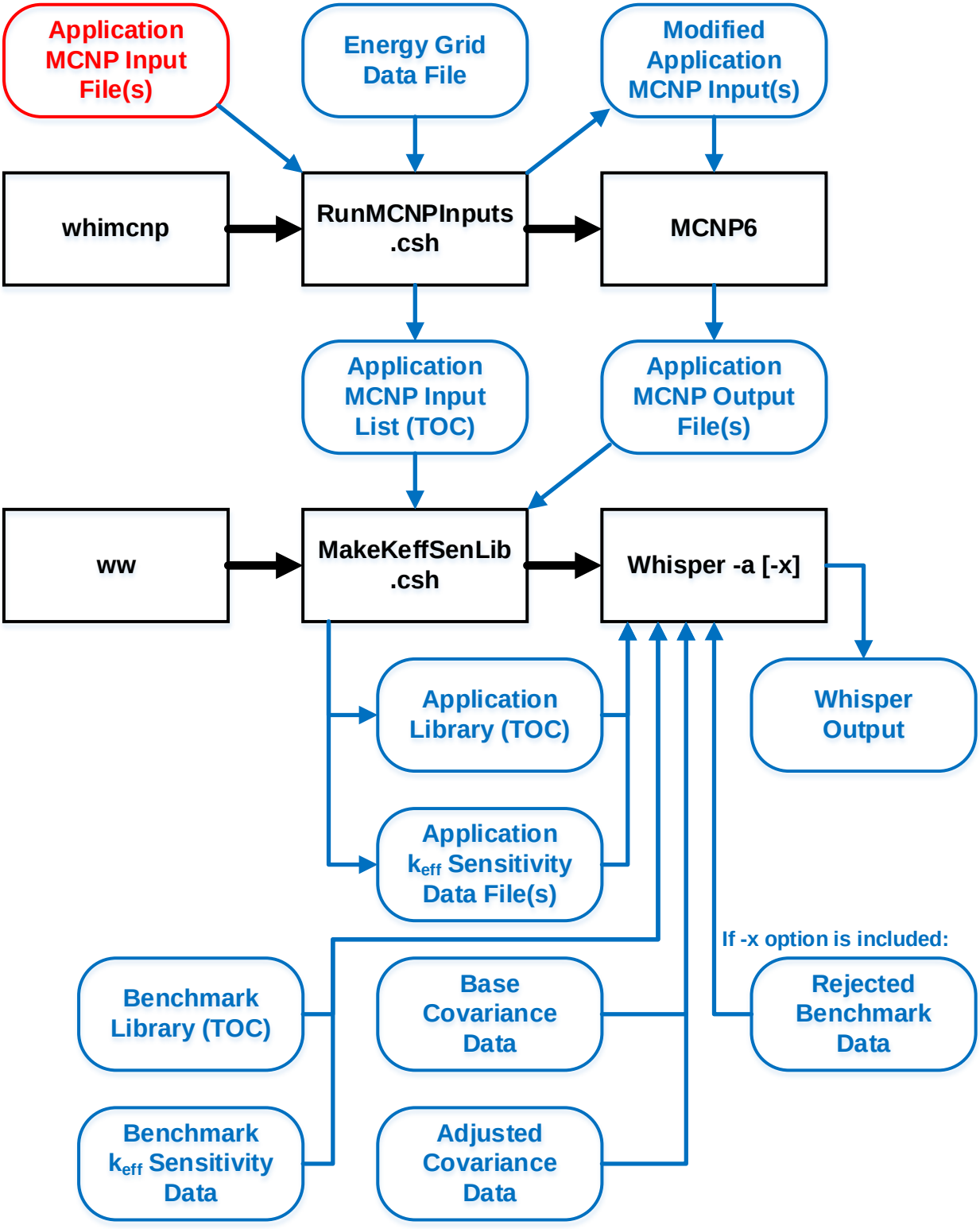

Figure 1
Application Model Evaluation Flowchart

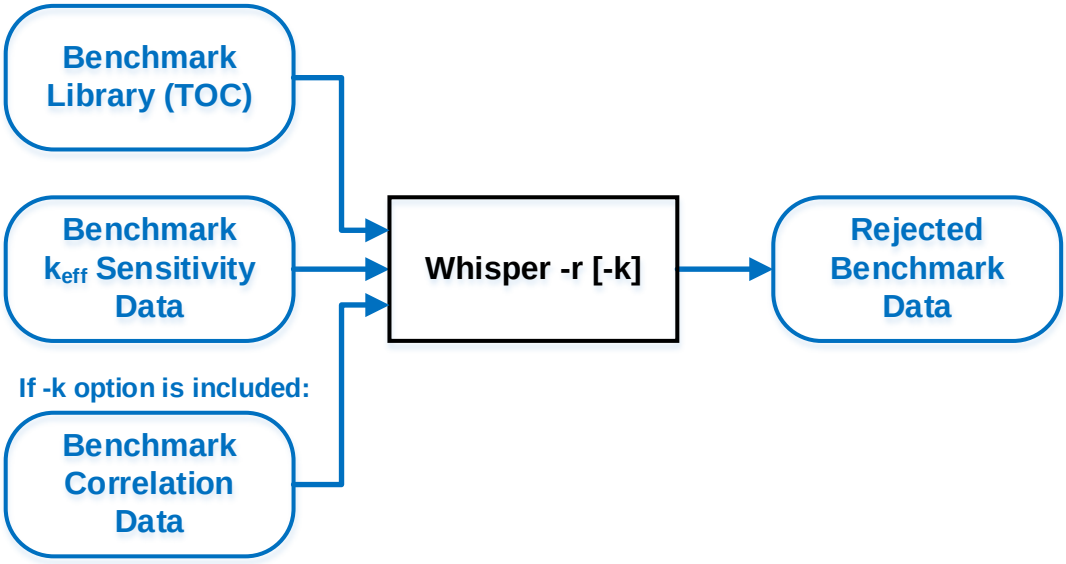

Figure 2
Benchmark Rejection Flowchart

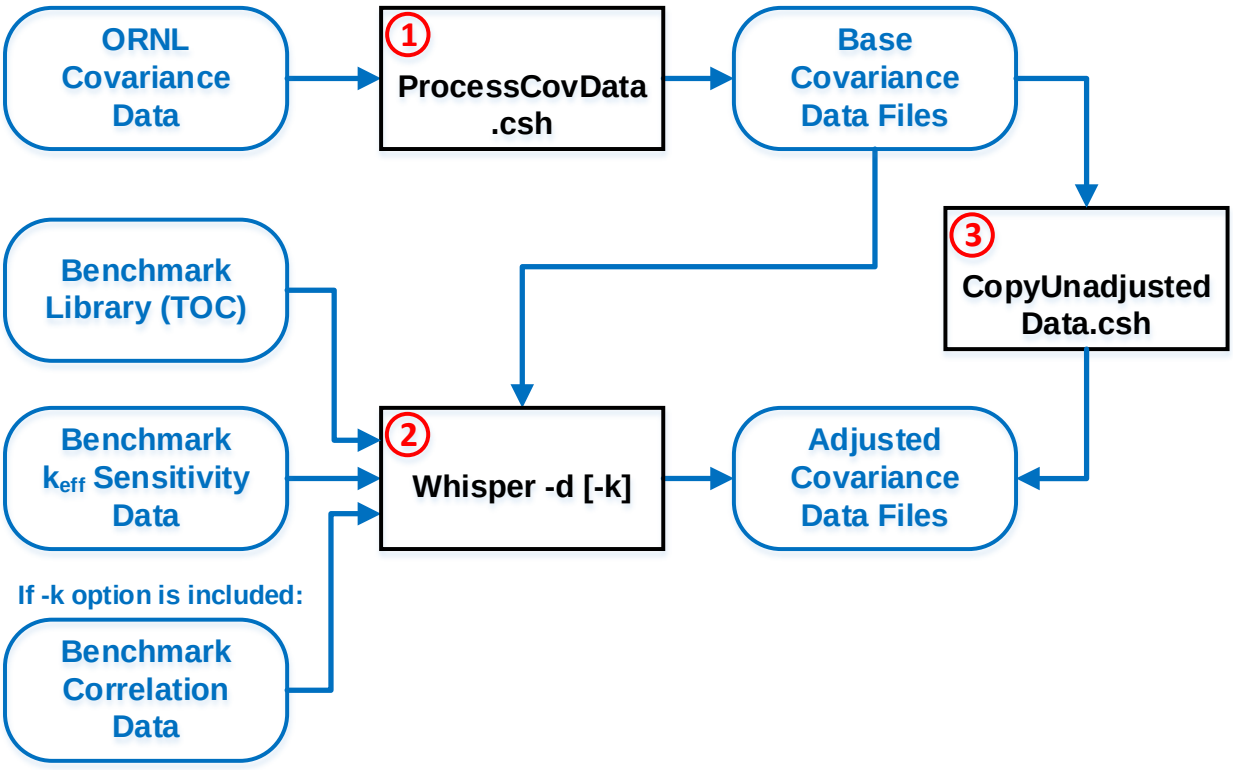

Figure 3
Initial Covariance Data Flowchart

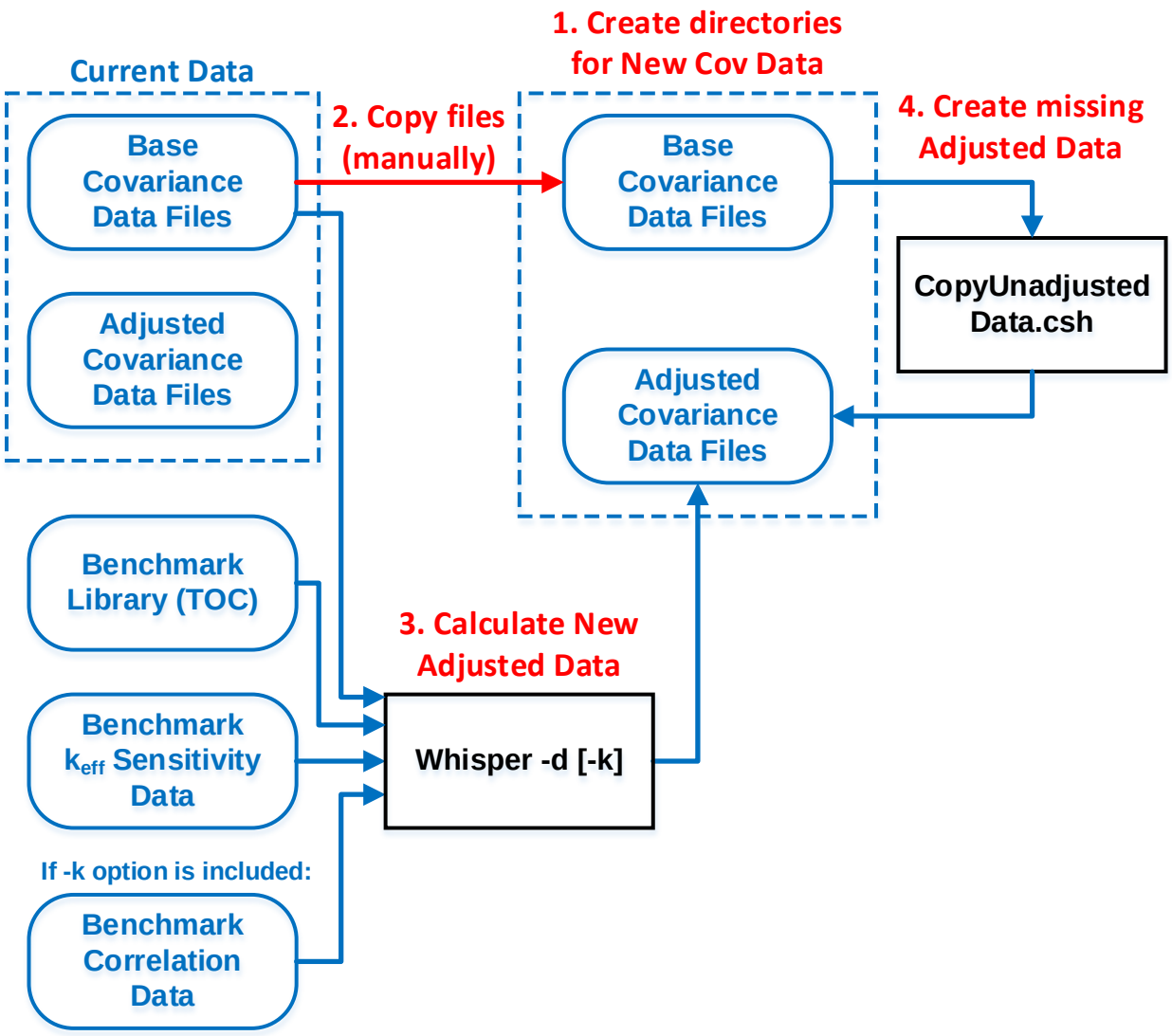

Figure 4
Replacement Covariance Data Flowchart

4. Algorithms

The following sections present the algorithms for the three purposes above. These algorithms are presented separately, although Whisper can execute a combination of these algorithms. To explain the sequence of steps, the algorithm in Section 4.1 includes references to Sections 4.2 and 4.3. LA-UR-14-26558 (and the Whisper source code) is the reference for these algorithms. B.T. Rearden, et. al., is also a good introduction to sensitivity and uncertainty analysis methods.

The following algorithms and equations use different subscripts to indicate the different parameters (for example, the isotope, energy bin, and reaction).

Subscript	Parameter
A	Application
a	Index on application
B	Benchmark
C	Combined (application and benchmark) data
i	• Index on isotope • Summation index
j	• Index on energy • Summation index
k	• Index on reactions • Variable related to k_{eff}
kk	Correlation data
m	Benchmark index
u	Index to benchmark with unknown uncertainty
x	Independent parameter for sensitivity coefficient (e.g., the cross section of a specific isotope, reaction, and incident energy)
xx	Covariance data
x'x'	Adjusted covariance data

This convention is used only in this document, not the reference documents or the Whisper program itself.

4.1 Evaluate Application Model and Calculate Baseline USL

The following steps are performed to evaluate an application model and calculate the corresponding baseline Upper Subcritical Limit.

Note: Report LA-UR-14-26558 uses row vectors and the corresponding matrix organization. Whisper uses a column organization. The following algorithm follows the Whisper notation; the following equations differ from LA-UR-14-26558 by the transposition of certain vectors and matrices.

1. Expert judgment has been used to determine the appropriate margin for unknowns that remain undetected in the transport code:

$$MOS_{software} = 0.005 \quad (1)$$

This value has been hard-coded in Whisper (parameter CodeAndMethodMargin).

2. For each benchmark (m):

a. Read the input data: the experimental k_{eff} ($k_{i,exp}$) and uncertainty ($\sigma_{i,exp}$), the calculational k_{eff} ($k_{i,calc}$) and uncertainty ($\sigma_{i,calc}$), and sensitivity coefficients ($S_{k,x}$) as isotope sensitivity matrices. A isotope sensitivity matrix arranges the sensitivity coefficients by row for each energy level and by column for each reaction, as shown in Figure 5.

	Reaction index				
Energy index	1	2	…	11	12
1	$S_k(1,1)$	$S_k(1,2)$	…	$S_k(1,11)$	$S_k(1,12)$
2	$S_k(2,1)$	$S_k(2,2)$	…	$S_k(2,11)$	$S_k(2,12)$
⋮	⋮	⋮	…	⋮	⋮
43	$S_k(43,1)$	$S_k(43,2)$	…	$S_k(43,11)$	$S_k(43,12)$
44	$S_k(44,1)$	$S_k(44,2)$	…	$S_k(44,11)$	$S_k(44,12)$

Figure 5 – Isotope Sensitivity Matrix Organization

1) The definition of a sensitivity coefficient for k_{eff} is [LA-UR-14-26558; eq. 28]:

$$S_{k,x} \equiv \frac{x}{k}\frac{\partial k}{\partial x} \quad (2)$$

2) Sensitivity coefficients are specific to (and indexed by) the isotope, reaction, and neutron energy. (This is the order used in Whisper.)

3) The nuclear reactions considered are elastic scattering, inelastic scattering, fission, capture [(n,2n), (n,γ), (n,p), (n,d), (n,t), (n,3He), (n,α)], fission total ν, and fission χ [LA-UR-14-23352; Sec. 3.1] [LA-UR-14-26436; Sec. 3.4]. All other reactions are going to be minor to criticality safety and are ignored [LA-UR-14-26436; Sec. 3.4].

b. Calculate the bias in k_{eff} [LA-UR-14-26558; eq. 21]:

$$\beta_m = k_{exp,m} - k_{calc,m} \quad (3)$$

c. Calculate the uncertainty in the k_{eff} bias [LA-UR-14-26558; eq. 22]:

$$\sigma_{\beta,m} = \sqrt{\sigma_{exp,m}^2 + \sigma_{calc,m}^2} \quad (4)$$

3. Read the benchmark correlation data. Populate the correlation matrix (as shown in Figure 6) with the correlation values between two benchmarks.

Benchmark index	Benchmark index				
	1	2	…	N-1	N
1	1	$r_{1,2}$	…	$r_{1,N-1}$	$r_{1,N}$
2	$r_{2,1}$	1	…	$r_{2,N-1}$	$r_{2,N}$
⋮	⋮	⋮	…	⋮	⋮
N-1	$r_{N-1,1}$	$r_{N-1,2}$	…	1	$r_{N-1,N}$
N	$r_{N,1}$	$r_{N,2}$	…	$r_{N,N-1}$	1

Figure 6 – Correlation Matrix Organization

a. The correlation matrix is initialized to the identity matrix with a row and column for each benchmark.

b. Read the correlation data file for the benchmark names and correlation value, i.e.,

$$name_{m1} \quad name_{m2} \quad r_{m1,m2}$$

The correlation value ($r_{m1,m2}$) is also placed in matrix position (m2, m1), i.e., is $r_{m2,m1}$.

- Any pair of benchmark names can be specified only once in the input file.
- If $name_{m1} = name_{m2}$, the input correlation value must be one.
- If $name_{m1} \neq name_{m2}$, the correlation value must be within the range of (-1,1).
- If a pair of benchmarks is not listed in the correlation file, the correlation value defaults to zero.

4. Read the application data: the calculated k_{eff} ($k_{calc,a}$) and uncertainty ($\sigma_{calc,a}$), and the sensitivity coefficients ($S_{k,x}$) as isotope sensitivity matrices.

5. For sensitivity coefficients on thermal scattering {S(α,β)} cross sections in the benchmark and application data, add the inelastic scattering sensitivity coefficients to the elastic scattering sensitivity coefficients. See Appendix C for further discussion of the covariance and sensitivity data for thermal scattering {S(α,β)} cross sections. Set all sensitivity factors for reactions other than scattering to zero. For the isotope sensitivity matrix shown above, add column 2 to column 1 and set columns 2 to 12 to zero.

6. Read the covariance data as individual covariance matrices. Each covariance value is for an isotope, energy, and reaction (e.g, i1, j1, k1) with another isotope, energy, and reaction (e.g., i2, j2, k2). Each individual covariance matrix is specific to one isotope and reaction pair (e.g, i1, k1) and another isotope and reaction pair (e.g., i2, k2). As shown in Figure 7, the rows and columns of the individual covariance matrix are for the energies (e.g., j1 and j2). See Sections 5.9 and 5.10 below for more information on the covariance data and individual covariance matrix.

a. The individual covariance matrix for the reverse order of isotope and reaction (e.g, i2, k2 and i1, k1) is the transpose of the matrix for i1, k1 and i2, k2.

	Energy index				
Energy index	1	2	…	43	44
1	$cov_{1,1}$	$cov_{1,2}$	…	$cov_{1,43}$	$cov_{1,44}$
2	$cov_{2,1}$	$cov_{2,2}$	…	$cov_{2,43}$	$cov_{2,44}$
⋮	⋮	⋮	…	⋮	⋮
43	$cov_{43,1}$	$cov_{43,2}$	…	$cov_{43,43}$	$cov_{43,44}$
44	$cov_{44,1}$	$cov_{44,2}$	…	$cov_{44,43}$	$cov_{44,44}$

Figure 7 – Individual Covariance Matrix Organization

7. For the benchmarks with an unknown (or unrealistically low) experimental uncertainty, estimate the experimental uncertainty ($\sigma_{exp,u}$). For each benchmark with an unknown (or unrealistically low) experimental uncertainty (B_u):
   a. For each benchmark, expand the isotope sensitivity matrices to create the benchmark sensitivity vector ($\vec{S}_m$).
      1) Each sensitivity vector contains all of the sensitivity coefficients for the corresponding benchmark (or application) organized by isotope, reaction, and energy as shown in Figure 8. The vector must include the isotopes for all of the benchmarks, as well as the specified reactions and energies. The sensitivity vector is all of the columns of all of the benchmark isotope sensitivity matrices (see Figure 5) stacked together.

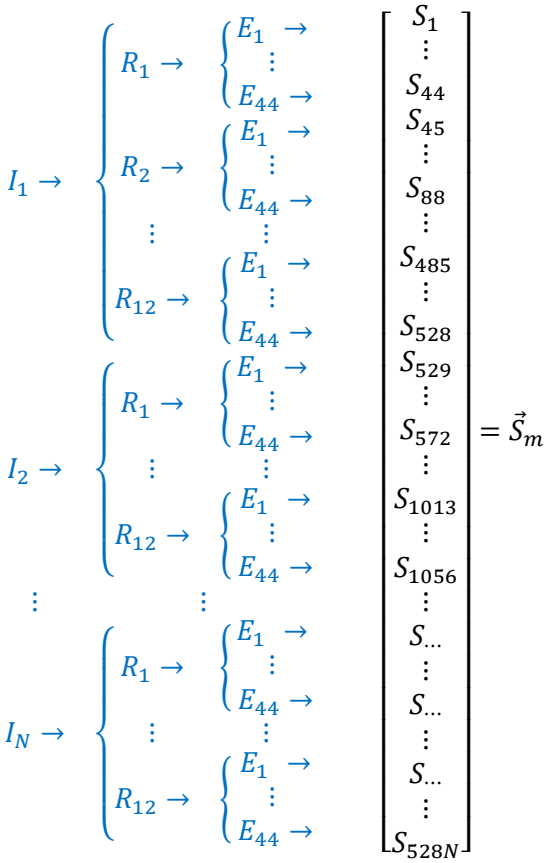

Figure 8 – Sensitivity Vector Organization

b. Create the expanded covariance data matrix ($\mathbf{C}_{xx}$) from the individual covariance matrices. Like the sensitivity vectors, the columns and rows of the matrix are arranged by isotope, reaction, and energy, as shown in Figure 9.

c. Calculate the variance of B_u [LA-UR-14-26558, eq. 31]:

$$Var_k(B_u) = \vec{S}_u^T \, \overline{\overline{C}}_{xx} \, \vec{S}_u \tag{5}$$

d. For each benchmark with a known (realistic) experimental uncertainty (B_m):

1) Calculate the variance of B_m [LA-UR-14-26558, eq. 31].

2) Calculate the covariance of B_u and B_m [LA-UR-14-26558, eq. 30]:

$$Cov_k(B_u, B_m) = \vec{S}_u^T \, \overline{\overline{C}}_{xx} \, \vec{S}_m \tag{6}$$

Isotope			#1			#1			…	Last isotope		
	Reaction		1			2			…	12		
		E Bin	1	…	44	1	…	44	…	1	…	44
#1	1	1 ⋮ 44	Covariance Data for isotope #1, reaction 1 & isotope #1, reaction 1			Covariance Data for isotope #1, reaction 1 & isotope #1, reaction 2			…	Covariance Data for isotope #1, reaction 1 & last isotope, last reaction		
#1	2	1 ⋮ 44	Covariance Data for isotope #1, reaction 2 & isotope #1, reaction 1			Covariance Data for isotope #1, reaction 2 & isotope #1, reaction 2			…	Covariance Data for isotope #1, reaction 2 & last isotope, last reaction		
⋮	⋮	⋮	⋮			⋮			⋮	⋮		
#1	12	1 ⋮ 44	Covariance Data for isotope #1, reaction 12 & isotope #1, reaction 1			Covariance Data for isotope #1, reaction 12 & isotope #1, reaction 2			…	Covariance Data for isotope #1, reaction 12 & last isotope, last reaction		
#2	1	1 ⋮ 44	Covariance Data for isotope #2, reaction 1 & isotope #1, reaction 1			Covariance Data for isotope #2, reaction 1 & isotope #1, reaction 2			…	Covariance Data for isotope #2, reaction 1 & last isotope, last reaction		
⋮	⋮	⋮	⋮			⋮			⋮	⋮		
Last isotope	12	1 ⋮ 44	Covariance Data for last isotope, last reaction & isotope #1, reaction 1			Covariance Data for last isotope, last reaction & isotope #1, reaction 2			…	Covariance Data for last isotope, last reaction & last isotope, last reaction		

Figure 9 - Expanded Covariance Matrix (C_{xx}) Organization

3) Calculate the correlation coefficient ($c_{k,i}$) between B_u and B_m [LA-UR-14-26558, eq. 32]:

$$c_{k,m} = c_k(B_u, B_m) = \frac{Cov_k(B_u, B_m)}{\sqrt{Var_k(B_u)}\ \sqrt{Var_k(B_m)}} \tag{7}$$

Negative correlation coefficients (c_k) are set to zero [LA-UR-14-26558; Sec. III.A.2].

e. Estimate the experimental uncertainty ($\sigma_{u,exp}$) for B_u [LA-UR-14-26558, eq. 44[1]]:

$$\sigma^2_{exp,u} = \frac{\sum_{i=1}^{N} c_{k,m}\sigma^2_{exp,m}}{\sum_{i=1}^{N} c_{k,m}} \tag{8}$$

where the summation is performed over all of the benchmarks with known (i.e., greater than zero) uncertainties (σ_{exp}).

Note: Whisper is missing a step to update the uncertainty in the bias ($\sigma_{\beta,m}$) with the new benchmark uncertainty.

8. If also rejecting inconsistent benchmarks from the data, perform the benchmark rejection; see Section 4.2 below for this algorithm.
9. Read the adjusted covariance data (as individual covariance matrices).
   a. Alternatively calculate the adjusted covariance data; see Section 4.3 below for this algorithm.
10. Determine the uncertainty for the application model due to nuclear data uncertainty and variability.
    a. Create the expanded matrices for the covariance data ($\mathbf{C}_{xx}$) and the adjusted covariance data ($\mathbf{C}_{x'x'}$) from the individual covariance matrices.
    b. For each application, expand the isotope sensitivity matrices to create the application sensitivity vector ($\vec{\boldsymbol{S}}_A$).
    c. Calculate the square of adjusted relative uncertainties of *k* (from the adjusted nuclear data covariances).

$$C_{k'k'} = \vec{\boldsymbol{S}}_A^T\ \overline{\overline{\boldsymbol{C}}}_{x'x'}\ \vec{\boldsymbol{S}}_A \tag{9}$$

1) Note: LA-UR-14-26558 presents the equations for evaluating multiple application models simultaneously, i.e., calculate the covariance matrix for k [LA-UR-14-26558; eq. 42]:

$$\overline{\overline{\boldsymbol{C}}}_{k'k'} = \overline{\overline{\boldsymbol{S}}}_A^T\ \overline{\overline{\boldsymbol{C}}}_{x'x'}\ \overline{\overline{\boldsymbol{S}}}_A \tag{10}$$

Each diagonal element of the matrix $\boldsymbol{C}_{k'k'}$ is the adjusted relative uncertainty squared for the corresponding benchmark.

d. The uncertainty due to nuclear data uncertainty and variability is [LA-UR-14-26558; Sec. III.B.2]:

$$\sigma_{data} = \sqrt{C_{k'k'}} \tag{11}$$

[1] Note: Although LA-UR-14-26558, eq. 44 uses the symbol for bias uncertainty (σ_i as given in LA-UR-14-26558 ,eq. 22), the text (Section III.C) states this is a method for calculating the benchmark (experimental) uncertainty.

11. Calculate the calculational margin for the application model.
    a. Calculate the correlation coefficients ($c_{k,i}$) for the application model (A) with respect to each benchmark case (B_i) [LA-UR-14-26558, eq. 32]:

$$c_{k,m} = c_k(A, B_m) = \frac{Cov_k(A, B_m)}{\sqrt{Var_k(A)}\ \sqrt{Var_k(B_m)}} \tag{12}$$

    b. Determine the maximum values of the correlation coefficients for this application model ($c_{k,max}$).
    c. Determine the required weight [LA-UR-14-26558, eq. 35]:

$$w_{req} = w_{min} + w_{penalty}\left(1 - c_{k,max}\right) \tag{13}$$

    Where w_{min} is the minimum sample weight alowed for the validation (default value is 25) and $w_{penalty}$ is the coefficient for determining the penalty for not having a benchmark that is identically similar to the application (default value is 100).
    d. Calculate the benchmark weighting factors for the application model [LA-UR-14-26558; Sec. III.A.2].
        1) Initialize the acceptance correlation coefficient ($c_{k,acc}$) as 99.999% of $c_{k,max}$.
        2) Calculate the individual benchmark weights [LA-UR-14-26558, eq. 34]:

$$w_m = max\left\{0, \frac{c_{k,m} - c_{k,acc}}{c_{k,max} - c_{k,acc}}\right\} \tag{14}$$

        3) Perform the test [LA-UR-14-23202, eq. 8][2]:

$$\sum_i w_m \geq w_{req} \tag{15}$$

        4) If the test fails and $c_{k,acc} > 0$, decrement $c_{k,acc}$ by 0.001% of $c_{k,max}$ and return to the individual benchmark weight calculation {Step d.2) above}.
        5) When the iteration on $c_{k,acc}$ is completed, the relevant benchmarks are those with $c_{k,i} \geq c_{k,acc}$. The benchmarks with $c_{k,i} < c_{k,acc}$ have a zero weighting factor and will not influence the calculational margin.
    e. Determine the unadjusted calculational margin[3] for the application model.
        1) The computational or calculational margin is found by finding the value of x where the Extreme Value Theory (EVT) cumulative distribution function (CDF) equals (or exceeds) the desired confidence [LA-UR-14-23202, Sec. 3, last para.]:

$$F\left(m_{unadj}\right) \geq q \tag{16}$$

        where m_{unadj} is the unadjusted calculational margin and q is the confidence level (default value is 99%) [LA-UR-14-26558, eq. 24].

[2] LA-UR-14-23202, eq. 8 is a variation of LA-UR-14-26558, eq. 33.

[3] Unadjusted because this calculational margin value includes any non-conservative biases ($\beta_m < 0$).

2) The EVT CDF is [LA-UR-14-26558, eq. 19]:

$$F(x) = \prod_{m=1}^{N} F_m(x) \tag{17}$$

where N is the number of benchmarks.

3) Where $F_m(x)$ is the normal distribution CDF with weight w_m [LA-UR-14-26558, eq. 23]:

$$F_m(x) = (1 - w_m) + \frac{w_m}{2}\left[1 + erf\left(\frac{x - \beta_m}{\sqrt{2\sigma_{\beta,m}^2}}\right)\right] \tag{18}$$

4) The corresponding probability density functions (PDFs) are:

$$f(x) = \frac{d}{dx}F(x) = \sum_{m=1}^{N} \frac{dF_m(x)}{dx} \prod_{\substack{n=1 \\ n \neq m}}^{N} F_n(x) = F(x) \sum_{m=1}^{N} \frac{f_m(x)}{F_m(x)} \tag{19}$$

$$f_m(x) = \frac{d}{dx}F_m(x) = \frac{w_m}{\sqrt{2\pi}\sigma_{\beta,m}} exp\left[-\frac{1}{2}\left(\frac{x - \beta_m}{\sigma_{\beta,m}}\right)^2\right] \tag{20}$$

from the identity:

$$\frac{d}{dz}erf(z) = \frac{2}{\sqrt{\pi}} exp(-z^2) \tag{21}$$

f. Determine the unadjusted calculational margin for the application model.

1) Theoretically, the bias of the application is determined by [LA-UR-14-26558; eq. 25]:

$$\beta = \int_{-\infty}^{\infty} xf(x)dx = \int_{-\infty}^{\infty} xF(x) \sum_{m=1}^{N} \frac{f_m(x)}{F_m(x)} dx \tag{22}$$

2) This integration is performed with the trapezoid rule [LA-UR-14-26558; Sec. III.A.1]. Because the integration is performed numerically, the integral cannot be performed from negative infinity to positive infinity. Fortunately, this is not necessary because $f(x) \approx 0$ for small and large x values and does not contribute significantly to the integral. Therefore, in practice, the bias of the application is determined by:

$$\beta \approx \int_{xmin}^{xmax} xF(x) \sum_{m=1}^{N} \frac{f_m(x)}{F_m(x)} dx \tag{23}$$

The integration boundary *xmin* is a point at which the CDF is below a tolerance value (ε_x) and the CDF of *xmax* is within the tolerance to one, i.e.,

$$F(xmin) < \varepsilon_x \tag{24}$$

$$F(xmax) > 1 - \varepsilon_x \tag{25}$$

The default value of the tolerance is 10^{-9}.

3) Determine the non-conservative bias adjustment [LA-UR-14-26558; eq. 7]:

$$\Delta m = max(0, -\beta) \tag{26}$$

4) The adjusted calculational margin is determined by adding the non-conservative adjustment parameter to the unadjusted calculational margin [LA-UR-14-26558; eq. 27]:

$$m_{adj} = m_{unadj} + \Delta m \tag{27}$$

g. Determine the final calculational margin for the application model.

1) If the residual weight fraction is less than (or equal to) zero $\{(1 - \mathrm{w_{sum}}/\mathrm{w_{req}}) \leq 0\}$, the adjusted calculational margin is the final calculational margin:

$$m_{final} = m_{adj} \tag{28}$$

2) If the residual weight fraction is greater than zero $\{(1 - \mathrm{w_{sum}}/\mathrm{w_{req}}) > 0\}$, all of the benchmarks are too dissimilar to the application to satisfy the test $\sum_{\mathrm{i}} \mathrm{w_i} \geq \mathrm{w_{req}}$. To address this condition, an interpolation with the unweighted calculational margin is added [LA-UR-14-16558; eq. 36 [4]]

$$m_{final} = \frac{w_{sum}}{w_{req}} m_{adj} + \left(1 - \frac{w_{sum}}{w_{req}}\right) m_{unweighted} \tag{29}$$

The unweighted calculational margin is initially calculated in the same manner as the adjusted calculational margin (eq. 27 above) but with all benchmark weights (w_m) equal to one. Equation 29 relies on the fact that the unweighted calculational margin is always greater than the weighted calculational margin[5]. However, the minimum non-coverage penalty (default value of 0.05) is the lower limit for the unweighted calculational margin.

$$m_{unweighted} = \max\left(m_{adj}\{w_m = 1\}, minimum\ penalty\right) \tag{30}$$

12. Calculate the baseline Upper Subcritical Limit (USL) for the application model [LA-UR-14-26558; eq. 2 and 37]:

$$USL_{baseline} = 1.0 - m_{final} - MOS_{data} - MOS_{software} \tag{31}$$

a. The margin of subcriticality due to nuclear data uncertainty and variability is [LA-UR-14-26558; eq. 43]:

$$MOS_{data} = n_\sigma\ \sigma_{data} \tag{32}$$

[4] The LA-UR-14-16558 text with eq. 36 states the interpolation is performed with the calculational margin from eq. 24, i.e., the unadjusted calculational margin. However, the interpolation implemented in Whisper (i.e., the Whisper source code) uses the adjusted calculational margin from eq. 27.

[5] From equation 18, the weighted and unweighted CDFs are related by:

$$F_m(x) = (1 - w_m) + w_m G_m(x) = (G_m(x) - 1)w_m + 1$$

where $G_m(x)$ is the unweighted ($w_m = 1$) CDF. For $0 \leq w_m \leq 1$, $F_m(x)$ is a linear function of w_m from 1 to $G_m(x)$. Since CDF values (e.g., $G_m(x)$) are always less than one, $F_m(x)$ is a decreasing linear function of w_m. Therefore, for $w_m < 1$, the weighted CDF is always greater than the unweighted CDF ($F_m(x) > G_m(x)$). Since the EVT CDF is the product of the individual CDFs (eq. 17), the weighted EVT CDF ($F(x)$) is always greater than the unweighted EVT CDF ($G(x)$). The (unadjusted) calculation margins for the weighted and unweighted CDFs is determined by (from eq. 16):

$$F(m_F) = G(m_G) = q$$

Therefore, since $F(x)$ and $G(x)$ are increasing functions and $F(x) \geq G(x)$, the unweighted calculation margin (m_G) must be greater than the weighted calculation margin (m_F).

where n_σ is the standard deviation multiplier corresponding to the confidence level (for the default confidence level of 99%, $n_\sigma = 2.6$).

4.2 Reject Inconsistent Benchmark Cases

The generalized linear least squares (GLLS) method is used to adjust the nuclear data to minimize the chi-squared statistic (see LA-UR-14-23202, eq. 10). Benchmarks are iteratively rejected until the chi-squared value divided by the number of benchmarks is less than the maximum limit (default value is 1.2). [LA-UR-14-23352; Sec. 3.1]

The following steps are performed to reject inconsistent benchmarks.

1. Read the benchmark input data (as discussed in Section 4.1 above).
2. Read the benchmark correlation data (as discussed in Section 4.1 above).
3. Read the covariance data as individual covariance matrices (as discussed in Section 4.1 above).
4. For the benchmarks with an unknown (or unrealistically low) experimental uncertainty, estimate the experimental uncertainty (as discussed in Section 4.1 above).
5. Determine if any benchmarks should be rejected.
   a. Create the benchmark sensitivity matrix ($\mathbf{S}_{B,kx}$) from the previously calculated sensitivity factors. In LA-UR-14-26558, each row of the benchmark sensitivity matrix is the sensitivity vector for a benchmark experiment [LA-UR-14-26558, Sec. III.B.2, pg. 28]. In Whisper, each column of the benchmark sensitivity matrix is a sensitivity vector.
   b. Create the relative covariance matrix of the benchmark experiments including benchmark correlations ($\mathbf{C}_{kk}$). The elements of this matrix are calculated from the correlation ($r_{i,j}$) input data:

$$\left[\bar{\bar{C}}_{kk}\right]_{m1,m2} = r_{m1,m2} \times \frac{\sigma_{exp,m1}}{k_{exp,m1}} \times \frac{\sigma_{exp,m2}}{k_{exp,m2}} \tag{33}$$

   Because the correlation data is symmetric ($r_{m1,m2} = r_{m2,m1}$), matrix $\mathbf{C}_{kk}$ is symmetric.

   c. Create a diagonal matrix (**B**) containing the ratio of the benchmark experimental k_{eff} to the calculated k_{eff}:

$$B_{m,m} = \frac{k_{exp,m}}{k_{calc,m}} \tag{34}$$

   d. An alternative to creating $\mathbf{C}_{kk}$ and **B** separately (used in Whisper), is calculate the product (which is the sole purpose of $\mathbf{C}_{kk}$ and **B**):

$$\left[\bar{\bar{\boldsymbol{B}}}\ \bar{\bar{C}}_{kk}\ \bar{\bar{\boldsymbol{B}}}\right]_{m1,m2} = r_{m1,m2} \times \frac{\sigma_{exp,m1}}{k_{calc,m1}} \times \frac{\sigma_{exp,m2}}{k_{calc,m2}} \tag{35}$$

   Because the correlation data is symmetric, $r_{m1,m2} = r_{m2,m1}$, matrix [**B** $\mathbf{C}_{kk}$ **B**] is symmetric.

   e. Calculate the covariance matrix of the relative difference vector [LA-UR-14-26558, eq. 40]:

$$\bar{\bar{\boldsymbol{C}}}_{dd} = \bar{\bar{\boldsymbol{B}}}\,\bar{\bar{\boldsymbol{C}}}_{kk}\,\bar{\bar{\boldsymbol{B}}} + \bar{\bar{\boldsymbol{S}}}_{B,kx}^{T}\,\bar{\bar{\boldsymbol{C}}}_{xx}\,\bar{\bar{\boldsymbol{S}}}_{B,kx} \tag{36}$$

   f. Invert the covariance matrix of the discrepancy vector.
   g. Calculate the benchmark discrepancy vector (the differences between the benchmarks' calculated k_{eff} and the reference k_{eff} [LA-UR-14-26558, Sec. III.D]).

$$\vec{D}_m = \frac{k_{calc,m} - k_{exp,m}}{k_{calc,m}} \tag{37}$$

> Note: This discrepancy is the *fractional* difference between k_{calc} and k_{exp}.

h. Calculate the value of chi-squared per benchmark.

$$\chi^2 = \frac{1}{N}\,\vec{D}^T\,\bar{\bar{C}}_{dd}^{-1}\,\vec{D} \tag{38}$$

i. If χ^2 is less than (or equal to) the threshold (default value is 1.2), then benchmark rejection is not necessary.

6. If χ^2 is greater than the threshold (default value is 1.2), determine which benchmarks are inconsistent and should be rejected.

   a. For the benchmarks which have not been rejected [LA-UR-14-26558, eq. 45]:

$$[\Delta\chi^2]_m = \left[\bar{\bar{C}}_{dd}^{-1}\right]_{m,m} \times \left(k_{calc,m} - k_{exp,m}\right)^2 \tag{39}$$

> Note: The discrepancy used in this calculation of chi-squared uses the magnitude difference between k_{calc} and k_{exp}. Whereas, the discrepancy vector above and below uses the *fractional* difference.

   b. Reject the benchmark with the maximum $[\Delta\chi^2]_m$.

   c. Generate the inverted covariance matrix of the discrepancy vector and the benchmark discrepancy vector.

   d. Calculate the value of chi-squared per benchmark.

   e. If χ^2 is greater the threshold (default value is 1.2), repeat step 6.

7. Write list of rejected (and excluded) benchmarks to the rejection file.

4.3 Covariance Data Processing

For the adjusted covariance data, the algorithm is to calculate the adjusted or residual covariance matrix according to LA-UR-14-26558, eq. (41). The following steps are performed in Whisper to process the covariance data.

1. Read the benchmark input data (as discussed in Section 4.1 above).
2. Read the benchmark correlation data (as discussed in Section 4.1 above).
3. If evaluating application models, read the application data: the calculated k_{eff} ($k_{calc,a}$) and uncertainty ($\sigma_{calc,a}$), and the sensitivity coefficients ($S_{k,x}$).
4. Read the covariance data as individual covariance matrices (as discussed in Section 4.1 above).
5. For the benchmarks with an unknown (or unrealistically low) experimental uncertainty, estimate the experimental uncertainty (as discussed in Section 4.1 above).
6. For the combined benchmark and application data:

   a. Create the benchmark sensitivity matrix ($\mathbf{S}_{B,kx}$) from the previously calculated sensitivity factors. In LA-UR-14-26558, each row of the benchmark sensitivity matrix is the sensitivity

vector for a benchmark experiment [LA-UR-14-26558, Sec. III.B.2, pg. 28]. In Whisper, each column of the benchmark sensitivity matrix is a sensitivity vector.

b. Create the relative covariance matrix of the benchmark experiments including benchmark correlations ($\mathbf{C}_{kk}$). The elements of this matrix are calculated from the correlation ($r_{i,j}$) input data:

$$\left[\bar{\bar{C}}_{kk}\right]_{m1,m2} = r_{m1,m2} \times \frac{\sigma_{exp,m1}}{k_{exp,m1}} \times \frac{\sigma_{exp,m2}}{k_{exp,m2}} \tag{40}$$

Because the correlation data is symmetric ($r_{m1,m2} = r_{m2,m1}$), matrix $\mathbf{C}_{kk}$ is symmetric.

c. Create a diagonal matrix (**B**) containing the ratio of the benchmark experimental k_{eff} to the calculated k_{eff}:

$$B_{m,m} = \frac{k_{exp,m}}{k_{calc,m}} \tag{41}$$

d. An alternative to creating $\mathbf{C}_{kk}$ and **B** separately (used in Whisper), is calculate the product (which is the sole purpose of $\mathbf{C}_{kk}$ and **B**):

$$\left[\bar{\bar{\boldsymbol{B}}}\, \bar{\bar{C}}_{kk}\, \bar{\bar{\boldsymbol{B}}}\right]_{m1,m2} = r_{m1,m2} \times \frac{\sigma_{exp,m1}}{k_{calc,m1}} \times \frac{\sigma_{exp,m2}}{k_{calc,m2}} \tag{42}$$

Because the correlation data is symmetric, $r_{m1,m2} = r_{m2,m1}$, matrix [**B** $\mathbf{C}_{kk}$ **B**] is symmetric.

e. Calculate the covariance matrix of the relative difference vector [LA-UR-14-26558, eq. 40]:

$$\bar{\bar{\boldsymbol{C}}}_{dd} = \bar{\bar{\boldsymbol{B}}}\, \bar{\bar{\boldsymbol{C}}}_{kk}\, \bar{\bar{\boldsymbol{B}}} + \bar{\bar{\boldsymbol{S}}}_{B,kx}^{T}\, \bar{\bar{\boldsymbol{C}}}_{xx}\, \bar{\bar{\boldsymbol{S}}}_{B,kx} \tag{43}$$

f. Calculate the residual (or adjusted) covariance matrix [LA-UR-14-26558; eq. 41].

$$\bar{\bar{\boldsymbol{C}}}_{x'x'} = \bar{\bar{\boldsymbol{C}}}_{xx} - \left[\bar{\bar{\boldsymbol{C}}}_{xx}\, \bar{\bar{\boldsymbol{S}}}_{B,kx}\, \bar{\bar{\boldsymbol{C}}}_{dd}^{-1}\, \bar{\bar{\boldsymbol{S}}}_{B,kx}^{T}\, \bar{\bar{\boldsymbol{C}}}_{xx}\right] \tag{44}$$

7. The compressed form of $C_{x'x'}$ is saved as the new or replacement adjusted covariance data.

5. Variables

The following sections explain the variables used in Whisper.

Whisper uses the derived data type introduced with Fortran 90, i.e., data types with defined components of the intrinsic and previously defined data types. For the derived data type variables explained below, the components that are not for data storage, i.e., references to other derived variables and pointers to procedures, are in light grey highlighting. The following is not a comprehensive list of all of the derived variables used, however, Appendix D does list all of the derived variable *types* defined in Whisper. In the text of this document, component names are underlined, for example Data(n) % M(j1,j2), to indicate the association of the component names.

5.1 Index Variables

The data in Whisper is dependent on several variables, for example, the isotope, energy bin, and reaction. In order to explain the data structures and operations of Whisper, it was decided that a standard set of index variables (rather than the variables used in Whisper) would improve the explanations. This document uses the following convention for indices to aid the reader. This convention is used only in this document, not the Whisper program itself.

Index Variable	Associated with
i	Isotope
j	Energy bin
k	Reaction
m	Benchmark
n	Numerical

5.2 Energy Bins and Reactions

Whisper V1.0.0 uses fixed parameter variable names and the associated values in the following tables to define the energy bins and reactions used in the program. These parameters are used to allocate other variables, such as the sensitivity matrices.

Parameter	Variable Name	Value
Number of energy bins	NErg	44
Number of reactions (or MT numbers)	NRxn	12

Variable EBins contains the forty-five boundaries for the energy bins from low (10^{-11} MeV) to high (20 MeV). The energy bins are specified in parameter EBins, which is listed below.

```
real(8), parameter :: EBins(1:NErg+1) = &
  [ 1.0000e-11, 3.0000e-09, 7.5000e-09, 1.0000e-08, 2.5300e-08, 3.0000e-08, &
  & 4.0000e-08, 5.0000e-08, 7.0000e-08, 1.0000e-07, 1.5000e-07, 2.0000e-07, &
  & 2.2500e-07, 2.5000e-07, 2.7500e-07, 3.2500e-07, 3.5000e-07, 3.7500e-07, &
  & 4.0000e-07, 6.2500e-07, 1.0000e-06, 1.7700e-06, 3.0000e-06, 4.7500e-06, &
  & 6.0000e-06, 8.1000e-06, 1.0000e-05, 3.0000e-05, 1.0000e-04, 5.5000e-04, &
  & 3.0000e-03, 1.7000e-02, 2.5000e-02, 1.0000e-01, 4.0000e-01, 9.0000e-01, &
  & 1.4000e+00, 1.8500e+00, 2.3540e+00, 2.4790e+00, 3.0000e+00, 4.8000e+00, &
  & 6.4340e+00, 8.1873e+00, 2.0000e+01 ]
```

The variable names in Whisper use the terms *reactions* and *MT numbers* interchangeably. This document does so as well. The MT numbers for the reactions of concern are listed in parameter RelevantMTs. Module ParametersMod sets array RelevantMTs to:

2, 4, 18, 16, 102, 103, 104, 105, 106, 107, 452, 1018

The reaction associated with each MT number is:

Relevant Reaction (MT) Number	MCNP6 Definition [LA-CP-13-00634; Table 3-104]
2	Elastic
4	Total Inelastic
18	Total Fission
16	(n,2n)
102	(n,γ)

Relevant Reaction (MT) Number	MCNP6 Definition [LA-CP-13-00634; Table 3-104]
103	(n,p)
104	(n,d)
105	(n,t)
106	(n,^{3}He)
107	(n,α)
452	Total Fission ν
1018	Total Fission Chi

5.3 Benchmark Data

Variables BenchmarkData and ExcludedBenchmarks are one-dimension arrays of type BenchmarkDataType. Each element of the array has the following data structure.

Component	Data Type	Init. Value
NIso	integer	
MaxIso	integer	
ZA(i)	character(len=20)	allocatable
Init	procedure	
AddIso	procedure	
WriteIso	procedure	
IsoExists	procedure	
FindIso	procedure	
FileName	character(len=40)	
KeffCalc	real(8)	
KeffCalcUnc	real(8)	
IsoSen(:)	type(IsoSenType)	allocatable
IsoSen(i) % Sk(j,k)	real(8)	allocatable
UnionizeKeffSen	procedure	
UnionizeKeffSenVector	procedure	
Union[6]	generic => UnionizeKeffSen, UnionizeKeffSenVector	
KeffBench	real(8)	
KeffBenchUnc	real(8)	
KeffBias	real(8)	
SigmaBias	real(8)	

[6] Component Union selects subroutine UnionizeKeffSen or UnionizeKeffSenVector depending on the arguments supplied.

- If there are two arguments that are type KeffSenDataType, then Union calls subroutine UnionizeKeffSen.
- If there is one or two arguments that are arrays of type KeffSenDataType, then Union calls subroutine UnionizeKeffSenVector.

Where

i = isotope index,
j = energy group index (from 1 to 44), and
k = reaction index (from 1 to 12).

5.4 Application Data

Variable ApplicationData is a one-dimension array of type ApplicationDataType. Each element of the array has the following data structure.

Component	Data Type	Init. Value
NIso	integer	
MaxIso	integer	
ZA(i)	character(len=20)	allocatable
Init	procedure	
AddIso	procedure	
WriteIso	procedure	
IsoExists	procedure	
FindIso	procedure	
FileName	character(len=40)	
KeffCalc	real(8)	
KeffCalcUnc	real(8)	
IsoSen(:)	type(IsoSenType)	allocatable
IsoSen(i) % Sk(j,k)	real(8)	allocatable
UnionizeKeffSen	procedure	
UnionizeKeffSenVector	procedure	
Union[6]	generic => UnionizeKeffSen, UnionizeKeffSenVector	
DataUnc	real(8)	
CalcMargin	real(8)	
Bias	real(8)	
BiasUnc	real(8)	
USL	real(8)	
KeffOverUSL	real(8)	

5.5 Isotope Sensitivity Matrix

Embedded within the benchmark and application data, there is a separate sensitivity matrix for each individual isotope. This document calls each matrix within the benchmark and application data an *isotope sensitivity matrix* to distinguish them from the sensitivity matrix discussed below. Each isotope sensitivity matrix (variable Sk(j,k)) has 44 rows for the energy bins and 12 columns for the reactions (MT numbers)[7]. Figure 10 illustrates the relationship between the isotope sensitivity matrix locations, energy bins, and reactions.

[7] This document follows the convention that the indices for a two-dimensional array are in the order of row, column.

	MT:	2	4	…	452	1018
Energy		k=1	2	…	11	12
10^{-11} to 3×10^{-9}	j=1	Sk(1,1)	Sk(1,2)	…	Sk(1,11)	Sk(1,12)
3×10^{-9} to 7.5×10^{-9}	2	Sk(2,1)	Sk(2,2)	…	Sk(2,11)	Sk(2,12)
…	…	…	…	…	…	…
6.4340 to 8.1873	43	Sk(43,1)	Sk(43,2)	…	Sk(43,11)	Sk(43,12)
8.1873 to 20	44	Sk(44,1)	Sk(44,2)	…	Sk(44,11)	Sk(44,12)

Figure 10 – Whisper Isotope Sensitivity Matrix

5.6 Sensitivity Vector

Whisper requires vectors of the sensitivity factors for the benchmarks and the applications to perform its calculations. Conceptually, the sensitivity vector is a sequential list of the sensitivity factors for each isotope, each reaction, and each energy bin in the covariance data. However, Whisper performs the following operations in order to compress the sensitivity vector.

1. When all of the sensitivity factors for an isotope and reaction are zero, the isotope and reaction are omitted from the sensitivity vector.

2. When the first sensitivity factor and first diagonal covariance value are both zero, the first (minimum) energy bin is omitted from the sensitivity vector. This omission continues for the subsequent consecutive energy bins that also satisfy the condition of zero sensitivity and zero diagonal covariance.

Figure 11 illustrates the organization of the sensitivity vector for one benchmark or application case.

Isotope	Reaction	E Bin	Sensitivity Vector (variable S)
#1	#1	*MinEBin*	IsoSen(1) % Sk(1,*MinEBin*)
		…	…
		44	IsoSen(1) % Sk(1,44)
#1	#2	*MinEBin*	IsoSen(1) % Sk(2,*MinEBin*)
		…	…
		44	IsoSen(1) % Sk(2,44)
…	…	…	…
#2	#1	*MinEBin*	IsoSen(2) % Sk(1,*MinEBin*)
		…	…
		44	IsoSen(2) % Sk(1,44)
…	…	…	…
imax	jmax	*MinEBin*	IsoSen(imax) % Sk(jmax,*MinEBin*)
		…	…
		44	IsoSen(imax) % Sk(jmax,44)

Where

imax = number of isotopes with non-zero sensitivity factors in the benchmark or application data,

jmax = number of reactions with non-zero sensitivity factors for the last isotope in the benchmark or application data, and

MinEBin = the minimum energy bin value *for each isotope and reaction*.

Figure 11 – Whisper Sensitivity Vector

5.7 Sensitivity Matrices

5.7.1 Benchmark Data

Whisper requires a matrix of the benchmark sensitivity factors to perform its calculations. In Whisper, each column of the benchmark sensitivity matrix is a sensitivity vector as described above, and shown in Figure 12 below. In LA-UR-14-26558, each row of the benchmark sensitivity matrix ($\mathbf{S}_{B,kx}$) is the sensitivity vector for a benchmark experiment [LA-UR-14-26558, Sec. III.B.2, pg. 28]. Therefore, Whisper does not transpose the sensitivity matrix when LA-UR-14-26558 does, and vice versa.

5.7.2 Application Data

LA-UR-14-26558, Sec. III.B.2, discusses a sensitivity matrix for the applications ($\mathbf{S}_{A,kx}$); see equations 42 and 43. In Whisper, the USLs for the application models are calculated individually within a loop. Therefore, the sensitivity vector for each application is used, rather than a matrix for all applications, in the USL calculations. That is, in LA-UR-14-26558 eq. 42, $\mathbf{S}_{A,kx}$ is a vector and $\mathbf{C}_{k'k'}$ is a single value (i.e., $C_{k'k'}$). Also, the input to eq. 43 is the single value ($C_{k'k'}$) rather than a diagonal element of the matrix $\mathbf{C}_{k'k'}$.

			Sensitivity Matrix			
Isotope	Reaction	E Bin	Benchmark 1	Benchmark 2	…	Last Benchmark
#1	#1	*MinEBin*	IsoSen(1) % Sk(1,*MinEBin*)	IsoSen(1) % Sk(1,*MinEBin*)	…	IsoSen(1) % Sk(1,*MinEBin*)
		…	…	…	…	…
		44	IsoSen(1) % Sk(1,44)	IsoSen(1) % Sk(1,44)	…	IsoSen(1) % Sk(1,44)
#1	#2	*MinEBin*	IsoSen(1) % Sk(2,*MinEBin*)	IsoSen(1) % Sk(2,*MinEBin*)	…	IsoSen(1) % Sk(2,*MinEBin*)
		…	…	…	…	…
		44	IsoSen(1) % Sk(2,44)	IsoSen(1) % Sk(2,44)	…	IsoSen(1) % Sk(2,44)
…	…	…	…	…	…	…
#2	#1	*MinEBin*	IsoSen(2) % Sk(1,*MinEBin*)	IsoSen(2) % Sk(1,*MinEBin*)	…	IsoSen(2) % Sk(1,*MinEBin*)
		…	…	…	…	…
		44	IsoSen(2) % Sk(1,44)	IsoSen(2) % Sk(1,44)	…	IsoSen(2) % Sk(1,44)
…	…	…	…	…	…	…
imax	jmax	*MinEBin*	IsoSen(imax) % Sk(jmax,*MinEBin*)	IsoSen(imax) % Sk(jmax,*MinEBin*)	…	IsoSen(imax) % Sk(jmax,*MinEBin*)
		…	…	…	…	…
		44	IsoSen(imax) % Sk(jmax,44)	IsoSen(imax) % Sk(jmax,44)	…	IsoSen(imax) % Sk(jmax,44)

Figure 12 – Whisper Benchmark Sensitivity Matrix

5.8 Benchmark Correlation Data

Variable BenchmarkCorrel is type BenchmarkCorrelationMatrixType and has the following data structure.

Component	Data Type	Init. Value
Name(m)	character(len=40)	allocatable
Mat(m,m)	real(8)	allocatable

Where

m = benchmark index.

5.8.1 Benchmark Correlation Matrix

Variable BenchmarkCorrel % Mat is the correlation matrix for the benchmarks. Because each benchmark always has perfect correlation with itself, the diagonal elements must always be one (1). The rest of the matrix is symmetric about the diagonal.

Benchmark index	Benchmark index				
	1	2	…	N-1	N
1	1	$r_{1,2}$	…	$r_{1,N-1}$	$r_{1,N}$
2	$r_{2,1}$	1	…	$r_{2,N-1}$	$r_{2,N}$
⋮	⋮	⋮	…	⋮	⋮
N-1	$r_{N-1,1}$	$r_{N-1,2}$	…	1	$r_{N-1,N}$
N	$r_{N,1}$	$r_{N,2}$	…	$r_{N,N-1}$	1

Figure 13 – Benchmark Correlation Matrix (BenchmarkCorrel %Mat)

5.9 Covariance Data

Variables CovarianceData and AdjustedCovarianceData are type CovarianceMatrixType and have the following data structure.

Component	Data Type	Init. Value
Data(:)	type(MatrixType)	allocatable
Data(n) % M(j1,j2)	real(8)	allocatable
Iso(:,:)	type(IsotopeLevelCovarianceMatrixType)	allocatable
Iso(:,:) % Rxn(:,:)	type(ReactionLevelCovarianceMatrixType)	allocatable
Iso(i1,i2) % Rxn(k1,k2) % Data	type(MatrixType), pointer	=> null()
Iso(i1,i2) % Rxn(k1,k2) % doTranspose	logical	= .false.
Iso(i1,i2) % mt1(k1)	integer	allocatable
Iso(i1,i2) % mt2(k2)	integer	allocatable
ZA(i)	character(len=6)	allocatable
Read	procedure, public => ReadCovarianceData	
ReadFile	procedure, private => ReadCovarianceFile	
Write	procedure, public => WriteCovarianceData	
FindZAIndex	procedure, public => CovarianceMatrixFindZAIndex	
FindMTIndex	procedure, public => CovarianceMatrixFindMTIndex	
FindNextFreeMatrix	procedure, private => CovarianceMatrixFindNextFreeMatrix	
ConstructSUMap	procedure, public => CovarianceMatrixConstructSUMapVector	
ExpandCovarianceMatrixSensitivity Vectors	procedure, public	
ExpandCovarianceMatrixSensitivity Matrix	procedure, public	
Expand[8]	generic, public => ExpandCovarianceMatrixSensitivityVectors, & ExpandCovarianceMatrixSensitivityMatrix	

Where

i, i1, i2 = isotope index,
j1, j2 = energy group index (from 1 to 44),
k1, k2 = reaction index, and
n = numerical index.

The relationships of the covariance data are illustrated in Figure 14.

[8] Component Expand selects subroutine ExpandCovarianceMatrixSensitivityVectors or ExpandCovarianceMatrixSensitivityMatrix depending on the arguments supplied.

- If the first argument types are KeffSenDataType, KeffSenDataType, and ExpandedCovarianceMatrixSensitivityVectorType, then Expand calls subroutine ExpandCovarianceMatrixSensitivityVectors.
- If the first argument types are an array KeffSenDataType, and ExpandedCovarianceMatrixSensitivityMatrixType, then Expand calls subroutine ExpandCovarianceMatrixSensitivityMatrix.

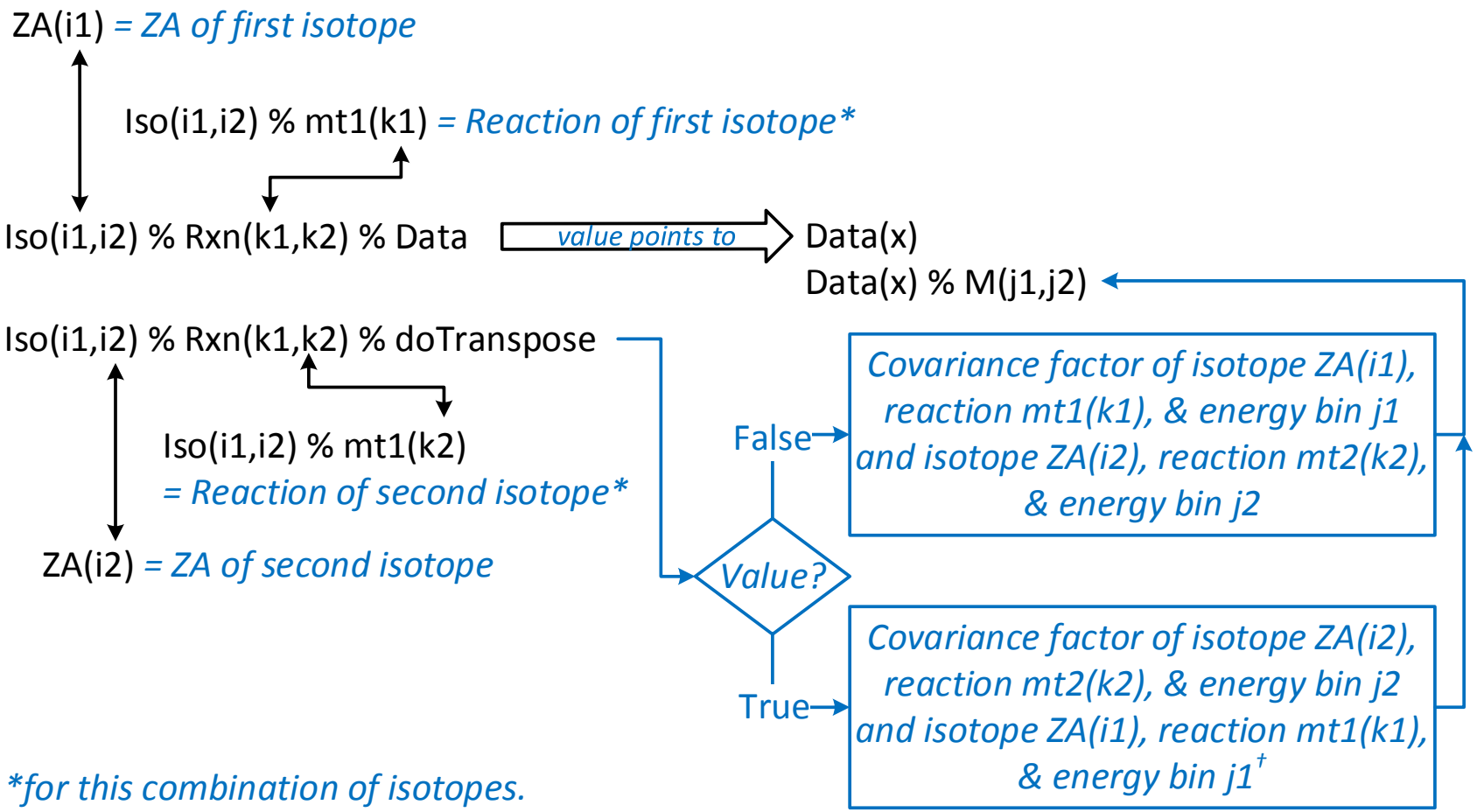

Figure 14 – Whisper Covariance Data Relationships

5.10 Individual Covariance Matrix

Each (individual) covariance matrix is placed in component Data(n) % M(j1,j2). The array index on Data is only a sequential placement/count of the covariance matrices and has no identifying information. Each covariance matrix is a 44 × 44 array (where 44 is the number of energy groups in Whisper V1.0.0).

Component Iso(i1,i2) % Rxn(k1,k2) % Data points to the covariance data for isotope ZA(i1) & reaction mt1(k1) and isotope ZA(i2) & reaction mt2(k2). If component Iso(i1,i2) % Rxn(k1,k2) % doTranspose is false, matrix M is the covariance matrix for isotope ZA(i1) & reaction mt1(k1) and isotope ZA(i2) & reaction mt2(k2). If component Iso(i1,i2) % Rxn(k1,k2) % doTranspose is true, matrix M must be transposed to get the covariance matrix for isotope ZA(i1) & reaction mt1(k1) and isotope ZA(i2) & reaction mt2(k2).

Components Iso(i1,i2) % Rxn(k1,k2) % Data and Iso(i2,i1) % Rxn(k2,k1) % Data both point to the same covariance data. However, the covariance matrix must be transposed for one of these isotope-reaction ordered pair. That is, for components Iso(i1,i2) % Rxn(k1,k2) % doTranspose and Iso(i2,i1) % Rxn(k2,k1) % doTranspose, one must be true (and the other must be false).

	Energy index				
Energy index	1	2	…	43	44
1	$cov_{1,1}$	$cov_{1,2}$	…	$cov_{1,43}$	$cov_{1,44}$
2	$cov_{2,1}$	$cov_{2,2}$	…	$cov_{2,43}$	$cov_{2,44}$
⋮	⋮	⋮	…	⋮	⋮
43	$cov_{43,1}$	$cov_{43,2}$	…	$cov_{43,43}$	$cov_{43,44}$
44	$cov_{44,1}$	$cov_{44,2}$	…	$cov_{44,43}$	$cov_{44,44}$

Figure 15 – Individual Covariance Matrix Organization

5.11 Expanded Covariance Matrix

A single matrix of the covariance values for the isotopes and reactions in the covariance data is required for calculations. This is called the expanded covariance matrix in Whisper and is denoted by component C in the derived type ExpandedCovarianceMatrixType. The expanded covariance matrix also uses the compression operations that the sensitivity vector uses (see Section 5.6 above). Figure 16 illustrates the organization of the expanded covariance matrix.

As discussed in Section 0 above, components Iso(i1,i2) % Rxn(k1,k2) % Data and Iso(i2,i1) % Rxn(k2,k1) % Data point to the same individual covariance matrix. Components Iso(i1,i2) % Rxn(k1,k2) % doTranspose and Iso(i2,i1) % Rxn(k2,k1) % doTranspose determine whether matrix is transposed or not before being placed in the expanded covariance matrix.

Isotope			#1			#1			…	Last isotope		
	Reaction		1			2			…	12		
		E Bin	*MinEBin*	…	44	*MinEBin*	…	44	…	*MinEBin*	…	44
#1	1	*MinEBin*	Covariance Data for isotope #1, reaction 1 & isotope #1, reaction 1			Covariance Data for isotope #1, reaction 1 & isotope #1, reaction 2			…	Covariance Data for isotope #1, reaction 1 & last isotope, reaction 12		
		⋮										
		44										
#1	2	*MinEBin*	Covariance Data for isotope #1, reaction 2 & isotope #1, reaction 1			Covariance Data for isotope #1, reaction 2 & isotope #1, reaction 2			…	Covariance Data for isotope #1, reaction 2 & last isotope, reaction 12		
		⋮										
		44										
⋮	⋮	⋮	⋮			⋮			…	⋮		
#1	12	*MinEBin*	Covariance Data for isotope #1, reaction 12 & isotope #1, reaction 1			Covariance Data for isotope #1, reaction 12 & isotope #1, reaction 2			…	Covariance Data for isotope #1, reaction 12 & last isotope, reaction 12		
		⋮										
		44										
#2	1	*MinEBin*	Covariance Data for isotope #2, reaction 1 & isotope #1, reaction 1			Covariance Data for isotope #2, reaction 1 & isotope #1, reaction 2			…	Covariance Data for isotope #2, reaction 1 & last isotope, reaction 12		
		⋮										
		44										
⋮	⋮	⋮	⋮			⋮			…	⋮		
Last isotope	12	*MinEBin*	Covariance Data for last isotope, last reaction & isotope #1, reaction 1			Covariance Data for last isotope, last reaction & isotope #1, reaction 2			…	Covariance Data for last isotope, reaction 12 & last isotope, reaction 12		
		⋮										
		44										

Where the value *MinEBin* is specific to each isotope and reaction.

Figure 16 – Whisper Expanded Covariance Matrix Organization

6. Initial Values

6.1 File Parameters

The file parameters and initial values are set in module FilesMod.

6.2 Command Line Defaults

The defaults for the options selected on the command line are set in module FilesMod.

6.3 User Options

The default values for the user options are set in module OptionsMod. See Section 7.4 below for a list of the user options and the default values.

Module OptionsMod also sets the rejection method to the iterative diagonal method. Because the iterative diagonal method is the only method available in Whisper V1.0.0, the user cannot change the rejection method.

6.4 Technical Parameters

The technical parameters are set in module ParametersMod. Examples of the technical parameters are listed below.

Parameter	Variable Name	Value
k_{eff} for critical system	KeffCritical	1.0
Calculational margin due to the code and method	CodeAndMethodMargin	0.005
Number of energy bins	NErg	44
Number of reactions (or MT numbers)	NRxn	12
Data rejection method name and case value	IterativeDiagonal	0
Energy bin boundaries	EBins(1:NErg+1)	See file.
Number of relevant MT numbers	NumRelevantMTs	12

The MT numbers for the reactions of concern are listed in parameter RelevantMTs. Module ParametersMod sets array RelevantMTs to:

2, 4, 18, 16, 102, 103, 104, 105, 106, 107, 452, 1018

See Section 5.2 above for the reaction associated with each MT number.

7. Program Structure

The main program of Whisper calls 14 subroutines; these subroutines in turn call additional subroutines and functions. Appendix E lists all of the Whisper subroutines and functions. The complete call tree logic for Whisper is given in Appendix F. However, this report does not describe every subroutine and function in Appendices E and F in detail, i.e., has a dedicated section for every subroutine and function. When appropriate, a small summary of a subroutine (and all its subordinates) is given when the call statement within the higher tier subroutine is discussed. The subroutines and functions that are covered in detail are listed in the following abbreviated call tree.

Program Whisper

1. WriteHeader
2. ParseCommandLine
3. CheckFiles
4. ReadAndSetupUserOptions
5. ReadAndSetupBenchmarks
   - 5.1. ReadKeffSenData
6. ReadAndSetupBenchmarkCorrelations
7. ReadAndSetupApplications
   - 7.1. ReadKeffSenData
8. ReadAndSetupCovarianceData
   - 8.1. CovarianceData % Read => ReadCovarianceData
     - 8.1.1. This % ReadFile => ReadCovarianceFile
   - 8.2. MakeThermalScatterConsistent
9. EstimateUnknownBenchmarkUncertainties
   - 9.1. CovarianceData % Expand => ExpandCovarianceMatrixSensitivityVectors
     - 9.1.1. This % ConstructSUMap => CovarianceMatrixConstructSUMapVector
     - 9.1.2. EC % SUMap % GetSensitivityVector => SUMapGetSensitivityVector
     - 9.1.3. EC % SUMap % GetExpandedCovarianceMatrix => SUMapGetExpandedCovarianceMatrix
10. RejectBenchmarks
11. AdjustNuclearData
    - 11.1. CovarianceData % Expand => ExpandCovarianceMatrixSensitivityMatrix
12. CalculateNuclearDataUncertainties
    - 12.1. CovarianceData % Expand => ExpandCovarianceMatrixSensitivityVectors
    - 12.2. AdjustedData % Expand => ExpandCovarianceMatrixSensitivityVectors
13. CalculateUpperSubcriticalLimits
    - 13.1. CalculateCalculationalMargin
      - 13.1.1. CalculateSimilarityWeights
        - 13.1.1.1. CovarianceData % Expand => ExpandCovarianceMatrixSensitivityVectors
14. WriteUSLOutputSummaryTable

These subroutines are described in the following sections. The following sections follow the order given above (for the first occurrence of each subroutine). The subroutine descriptions include, as applicable, subsections on the input values, the results, and a summary of the source code. The summary is an outline of primary logic and calculations for the subroutine or function[9]. The outline is kept as conceptual as possible and use of the Fortran variable name is minimized. The following structures are used in the outline:

- For action statements, the line number in the module file is given first, followed by a description of the action performed.
- For conditional and loop control statements, the logic is described first, the line number in parentheses follows.

Appendix G lists six subroutines that are included in the Whisper source code but are not executed in the program logic. However, the routines may be used in definitions of type-bound procedures.

7.1 Subroutine WriteHeader

This subroutine writes the program version (e.g., "Whisper 1.0.0") to the terminal. This subroutine is listed in the InputProcessingMod file.

7.2 Subroutine ParseCommandLine

This subroutine gets the input parameters from the command line and sets the flag variables. The following table lists the command line arguments for setting program variables. There are two additional command line arguments, --help (or –h) and --version (or -v), which display the requested information and terminate execution. This subroutine is listed in the InputProcessingMod file.

Command		Argument Variable Name	Default Value	Flag Variable Name[1]
Short	Long Form			
-a	--applications	ApplicationLibraryFile	' '	wasApplicationFileSpecified
-b	--benchmarks	BenchmarkLibraryFile	[2]	wasBenchmarkFileSpecified
-c	--covlibpath	CovariancePath	[3]	wasCovariancePathSet
-d	--adjusted	AdjustedCovariancePath	' '	wasAdjustedCovariancePathSet
-k	--bench_correl	BenchmarkCorrelFile	' '	wasBenchmarkCorrelFileSpecified
-m	--iso_coverage	IsotopicCoverageFile	' '	wasIsotopicCoverageFileSpecified
-o	--output	OutputFile	Whisper.out	n/a
-r	--reject	RejectionOutputFile	' '	wasRejectFileSpecified
-t	--threads	NumOpenMPThreads	16	n/a
-u	--user_options	UserOptionsFile	' '	wasUserOptionsFileSpecified

[9] The source code summaries explain the primary logic and calculations for the subroutine or function. Other statements, although essential for program operation, have not been not documented in the summaries. Examples include input and output file open (and close), dynamic memory allocation (and deallocation), and most write statements. It is presumed that a reader familiar with Fortran 2008 will understand the statements not discussed in the summary.

Command		Argument Variable Name	Default Value	Flag Variable Name[1]
Short	Long Form			
-x	--exclude	BenchmarkExcludeFile	' '	wasExcludeFileSpecified

Table Notes

[1] All of the flag variables are initially set to false (in module FilesMod) and are set to true only by the command line.

[2] If the BenchmarkLibraryFile is not specified in the command line, the environment variable WHISPER_BENCHMARK_TOC is used. The default value for WHISPER_BENCHMARK_TOC is Whisper/Benchmarks/TOC/BenchmarkTOC.dat.

[3] If the path to the covariance data directory is not specified in the command line (CovariancePath), the environment variable WHISPER_COVDATA_PATH is used. The default value for WHISPER_COVDATA_PATH is Whisper/CovarianceData/SCALE6.1. For the given path, the base (unadjusted) covariance data is in subdirectory 'Data' and the adjusted covariance data is in subdirectory 'Adjusted'.

7.3 Subroutine CheckFiles

This subroutine checks whether the files are ready for input or output. This subroutine is listed in the InputProcessingMod file.

7.4 Subroutine ReadAndSetupUserOptions

The user can supersede Whisper default values by including a user's option file on the whisper command line.

7.4.1 Inputs

7.4.1.1 User Option File

If the command line specifies a user options file, the following table lists the variables that can be set with their default values. This subroutine is listed in the InputProcessingMod file.

Variable Name	Default Value
NumOpenMPThreads	16
ThresholdChiSquare	1.2d0
CalcMarginConfidenceLevel	0.990d0
dxCalcMargin	1.d-5
dxAcceptSimilarity	1.d-5
DataUncMultiplier	2.6d0
MinimumWeightSum	25.0d0
WeightSumPenalty	100.0d0
MinimumNonCoveragePenalty	0.05d0

Variable Name	Default Value
UnknownDataUncertainty	0.1d0
AdjustedCovarianceCutoff	1.d-6
IntegrationTolerance	1.d-9
IntegrationLimitTolerance	1.d-6

7.4.1.2 User Option File Format

See LA-UR-14-26436, Section 6.1, User Options.

7.5 Subroutine ReadAndSetupBenchmarks

This subroutine reads the benchmark library (or TOC) file. The name of the benchmark library was previously placed in variable BenchmarkLibraryFile. This subroutine is listed in the InputProcessingMod file.

7.5.1 Inputs

7.5.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
BenchmarkData	BenchmarkData	inout
ExcludedBenchmarks	ExcludeData	inout

7.5.1.2 Benchmark Library File

The benchmark library (or TOC) file has the following format. The first line is a file path where Whisper gets the sensitivity profiles. The subsequent lines consist of the MCNP input filename for the benchmark, the benchmark k_{eff}, the benchmark uncertainty in k_{eff}, the calculated k_{eff}, and the calculated uncertainty in k_{eff}. [LA-UR-14-26436, Sec. 3.4]

7.5.1.3 Exclude Benchmark File

The exclude benchmark file contains a list of benchmark names (one per line) to be excluded from the validation [LA-UR-14-26436, Sec. 2.3 and 2.4].

7.5.1.4 Benchmark Sensitivity Data File

Each benchmark sensitivity data file has the sensitivity data for one or more isotopes (ZAID) in the following format.

1. For each reaction (j):
    a. ZAID and reaction name, for example:

```
1001.80c elastic
```

b. For reactions other than fission chi, the table header is (including blank lines):

```
            energy range (MeV)          sensitivity   rel. unc.
```

c. If the reaction is fission chi, the table header is (including blank lines):

```
       incident energy range:   0.0000E+00  1.0000E+36 MeV

              outgoing
           energy range (MeV)          sensitivity   rel. unc.
```

d. The sensitivity values for the ZAID, the reaction, and each of the 44 energy bins is listed. Each line lists the lower energy bound (MeV), the upper energy bound (MeV), the sensitivity factor, and the relative uncertainty. For example, the first 2 (of the 44 lines) for 1001.80c elastic are:

```
        1.0000E-11  3.0000E-09          -2.4282E-06       0.7635
        3.0000E-09  7.5000E-09           2.6719E-06       1.9396
```

e. Two blank lines.

Whisper assumes the reactions (for each isotope) are in the following order.

1. The first reaction is elastic scattering.
2. The second reaction is inelastic scattering.
3. The last reaction is fission chi.

7.5.2 Results

The following BenchmarkData components in the Whisper main program (see Section 5.3) are set:

- Niso – by calls to ReadKeffSenData.
- MaxIso – by calls to ReadKeffSenData.
- ZA(i) – by calls to ReadKeffSenData.
- FileName – read from the benchmark library file.
- KeffCalc – read from the benchmark library file.
- KeffCalcUnc – read from the benchmark library file.
- IsoSen(i) % Sk(j,k) – by calls to ReadKeffSenData.
- KeffBench – read from the benchmark library file.
- KeffBenchUnc – read from the benchmark library file.
- KeffBias – calculated from the benchmark library file data.
- SigmaBias – calculated from the benchmark library file data.

Where

i = isotope index,
j = energy group index, and

k = reaction index.

The following ExcludedBenchmarks components in the Whisper main program are set:

- FileName

The other components of ExcludeData are not set (or used) by Whisper.

7.5.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The following line numbers are from the file for module InputProcessingMod.

1. Lines 376 to 383 – Count the number of lines in the benchmark library file.
2. Line 387 – Initialize the number of excluded benchmarks to zero.
3. If an exclude benchmark file was specified on the command line (line 389):
   a. Lines 393 to 397 – Count the number of excluded benchmarks.
   b. Lines 403 to 406 – Read the filenames for the excluded benchmarks.
   c. Line 409 – Calculate the number of benchmarks which were not excluded.
4. If an exclude benchmark file was not specified on the command line (line 389):
   a. Line 412 – Set the number of benchmarks to the number of benchmark filenames in the benchmark library file.
5. For each line in the benchmark library file (line 424):
   a. Line 425 – Read a line of the benchmark library file:
      - the name of the benchmark input file,
      - the benchmark k_{eff},
      - the benchmark uncertainty in k_{eff},
      - the calculated k_{eff}, and
      - the calculated uncertainty in k_{eff} from the benchmark library file.
   b. If the benchmark name is not on the exclusion list (line 426):
      1. Lines 434 to 438 – Save the collected benchmark data (variables FileName, KeffCalc, KeffCalcUnc, KeffBench, KeffBenchUnc).
      2. Line 439 – Calculate the bias in k_{eff} (variable KeffBias) according to eq. 3 [LA-UR-14-26558 eq. 21].
      3. Line 440 – Calculate the uncertainty in the k_{eff} bias (variable SigmaBias) according to eq. 4 [LA-UR-14-26558 eq. 22].

4. Line 442 – Read the k_{eff} sensitivity data from the benchmark data file using function ReadKeffSenData (see Section 7.6 below). See Section 7.5.2 above for additional crosswalk information between the benchmark data components and the Whisper subroutines.

    a. Note: If an isotope has more than one data set in the benchmark data file, then the subsequent sensitivity factors are added to the previous sensitivity factors.

7.6 Function ReadKeffSenData

7.6.1 Inputs

7.6.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
BenchmarkData(m)	KS	inout
BenchmarkLibraryPath	FilePath	in

7.6.2 Results

For the element of BenchmarkData that is passed to function ReadKeffSenData, the following components are set from data in the benchmark sensitivity file:

- Niso – by calls to AddIso.
- MaxIso – by calls to AddIso, which calls Init.
- ZA(i) – by calls to AddIso.
- IsoSen(i) % Sk(j,k)

Where

i = isotope index,
j = energy group index, and
k = reaction index.

Function ReadKeffSenData also returns a logical value (true or false) via variable Success.

7.6.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module InputProcessingMod.

1. For each line in the benchmark data file:

    Note: Lines 576 and 620 form an infinite loop that is exited when the end-of-file is reached (line 579).

    a. Line 578 – Read isotope ZA ID.

    b. Line 579 – If end-of-file is reached on the benchmark data file, exit the do loop (go to Step 2 below).

    c. Line 583 – Convert isotope ZA ID to ZA number.

d. Line 584 – Find the array index value that points to this isotope in the sensitivity data by calling function FindIso. If the isotope is not found in the sensitivity data, FindIso returns zero.

e. If the isotope is not in the sensitivity data (line 585):

1. Lines 586 to 592 – Add the isotope to the sensitivity data and initialize the sensitivity data to zero.

f. Lines 601 to 619 – Read the isotope sensitivity matrix and add the current values to any previous values.

2. Line 623 – Set function return value (variable Success) to "true".

7.7 Subroutine ReadAndSetupBenchmarkCorrelations

This subroutine reads the correlation data between different benchmarks as documented with the experimental data. See LA-UR-14-26436, Sec. 4.1 for more information on the benchmark experimental correlation data. This subroutine is listed in the InputProcessingMod file.

7.7.1 Inputs

7.7.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
BenchmarkData	BenchmarkData	in
BenchmarkCorrel	BenchmarkCorrel	inout

7.7.1.2 Benchmark Correlation File

The benchmark correlation file lists one correlation per line. Each line has the names of two benchmark input files followed by a correlation coefficient. [LA-UR-14-26436, Sec. 4.1].

7.7.2 Results

This subroutine sets all of the components of BenchmarkCorrel (see Section 5.5 above) for the Whisper main program.

7.7.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module InputProcessingMod.

1. Line 699 – Initialize the correlation matrix (all correlation factors) to zero.

2. For each benchmark that has been read by Whisper (line 702):

a. Line 703 – Save the benchmark name in the correlation data.

b. Line 704 – Set the diagonal element of the correlation matrix to 1.

Note: At this point, each benchmark is perfectly correlated to itself and to no other benchmarks.

3. If a benchmark correlation file was specified on the command line (line 707):

   a. For each line in the benchmark correlation file:

      Note: Lines 714 and 778 form an infinite loop that is exited when the end-of-file is reached (line 716).

      1. Line 715 – Read two benchmark names and the correlation value from the benchmark correlation file.

      2. Line 716 – If the end-of-file is reached on the benchmark correlation file, exit the do loop (and return to the main program).

      3. Lines 719 to 735 – For the first and second benchmark names, find the array indices that point to the matching benchmark names in the correlation data.

      4. Lines 737 to 765 and 768 to 773 – Error checking. For non-fatal errors, execution continues with step 3.a.1 above. For fatal errors, the execution of Whisper is terminated (stopped).

      5. Lines 766 and 767 – Save correlation value between the two benchmarks (both orders of the benchmark names/indices).

7.8 Subroutine ReadAndSetupApplications

This subroutine sets up the application (process model) data. This subroutine is listed in the InputProcessingMod file.

7.8.1 Inputs

7.8.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
ApplicationData	ApplicationData	inout

7.8.1.2 Application Library File Format

The application library (or TOC) file has the following format. The first line is a file path where Whisper gets the sensitivity profiles. The subsequent lines consist of the MCNP input filename for the application, the benchmark k_{eff} (this is normally zero for an application and is ignored if it is not), the benchmark uncertainty in k_{eff} (again, normally zero for an application and is ignored if it is not), the calculated k_{eff}, and the calculated uncertainty in k_{eff}. [LA-UR-14-26436, Sec. 3.4]

7.8.2 Results

The following ApplicationData components in the Whisper main program are set:

- Niso – by calls to ReadKeffSenData.
- MaxIso – by calls to ReadKeffSenData.

- ZA(i) – by calls to ReadKeffSenData.
- FileName – read from the application library file.
- KeffCalc – read from the application library file.
- KeffCalcUnc – read from the application library file.
- IsoSen(i) % Sk(j,k) – by calls to ReadKeffSenData.

Where

i = isotope index,
j = energy group index, and
k = reaction index.

The remaining components of ApplicationData are calculated later by Whisper (if an application file was specified on the command line).

7.8.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module InputProcessingMod.

1. If an application file was specified on the command line (line 483):

    a. Lines 492 to 495 – Count the number of lines in the application library file.

    b. For each line in the application library file (line 506):

        1. Line 507 – Read the application filename, the calculated k_{eff}, and the calculated uncertainty in k_{eff}.

        2. Line 509 – Read the k_{eff} sensitivity data from the application data file using function ReadKeffSenData (also in the InputProcessingMod file). See Section 7.8.2 for additional crosswalk information between the application data components and the Whisper subroutines.

            a. Note: If an isotope has more than one data set in the application data file, then the subsequent sensitivity factors are added to the previous sensitivity factors.

2. If an application file was not specified on the command line:

    a. Allocate zero space to ApplicationData; this setting is used later to determine whether there is application data to process.

7.9 Subroutine ReadAndSetupCovarianceData

This subroutine sets up the base (unadjusted) covariance data [LA-UR-14-26436, Sec. 3.3]. This subroutine is listed in the InputProcessingMod file.

7.9.1 Inputs

7.9.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
CovarianceData	CovarianceData	inout
BenchmarkData	BenchmarkData	inout
ApplicationData	ApplicationData	inout

7.9.2 Results

The following components of CovarianceData in the Whisper main program are set:

- Data(n) % M(j1,j2) – by a call to ReadCovarianceData
- Iso(i1,i2) % Rxn(k1,k2) % Data – by a call to ReadCovarianceData
- Iso(i1,i2) % Rxn(k1,k2) % doTranspose – by a call to ReadCovarianceData
- Iso(i1,i2) % mt1(k1) – by a call to ReadCovarianceData
- Iso(i1,i2) % mt2(k2) – by a call to ReadCovarianceData
- ZA(i) – by a call to ReadCovarianceData

Where

i, i1, i2 = isotope index,
j1, j2 = energy group index (from 1 to 44),
k1, k2 = reaction index, and
n = numerical index.

7.9.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module InputProcessingMod.

1. Line 651 – List all of the isotopes in the benchmark data.

2. Line 652 – List all of the isotopes in the application data.

3. Line 653 – Merge the benchmark and application isotopes lists into a single list (UnionList).

4. Line 660 – Read the covariance data for the merged isotope list (UnionList) by calling subroutine ReadCovarianceData. Subroutine ReadCovarianceData is described in Section 7.10 below.

5. Lines 663 to 665 – Convert the sensitivity factors on the S(α,β) cross sections in the benchmark data such that the sensitivity on both inelastic and elastic scattering is treated as a sensitivity on elastic scattering by calling subroutine MakeThermalScatterConsistent.

6. Lines 666 to 668 – Convert the sensitivity factors on the S(α,β) cross sections in the application data such that the sensitivity on both inelastic and elastic scattering is treated as a sensitivity on elastic scattering by calling subroutine MakeThermalScatterConsistent.

7.10 Subroutine ReadCovarianceData

This subroutine reads the covariance data from the covariance data file. This subroutine is called by subroutines ReadAndSetupCovarianceData and AdjustNuclearData to read the base and adjusted covariance data, respectively. This subroutine is listed in the CovarianceMatrixMod file.

7.10.1 Inputs

7.10.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
CovarianceData	This	inout
AdjustedCovarianceData		
UnionList	IsoList	in
Path	Path	in

7.10.1.2 Covariance File

Each covariance data file (for a particular isotope) contains the following data:

1. A single line with:
   a. ZA number for the isotope
   b. Number of energy bins
   c. Number of correlations (pairs of ZA & MT numbers) in the file (NumZAMT)
2. The boundaries of the energy bins (number of bins plus 1 values)
3. NumZAMT lines of:
   a. The ZA and MT numbers for the two correlated isotopes-reactions (ZA1, MT1, ZA2, MT2)
4. NumZAMT blocks of:
   a. The ZA and MT numbers for the two correlated isotopes (ZA1, MT1, ZA2, MT2)
   b. (Map(j), NRow(j), j=1, NErg)
   c. (CovData(j), j=1,NErg*NErg)

7.10.2 Results

The following components of CovarianceData or AdjustedCovarianceData in the Whisper main program are set:

- Data(n) % M(j1,j2) – by calls to ReadCovarianceFile
- Iso(i1,i2) % Rxn(k1,k2) % Data – by calls to ReadCovarianceFile
- Iso(i1,i2) % Rxn(k1,k2) % doTranspose – by calls to ReadCovarianceFile
- Iso(i1,i2) % mt1(k1) – by calls to ReadCovarianceFileHeader
- Iso(i1,i2) % mt2(k2) – by calls to ReadCovarianceFileHeader
- ZA(i) – by calls to ReadCovarianceFileHeader

Where

i, i1, i2 = isotope index,
j1, j2 = energy group index (from 1 to 44),
k1, k2 = reaction index, and
n = numerical index.

7.10.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module CovarianceMatrixMod.

1. Lines 285 to 292 – From the list of unique isotopes (passed by argument to subroutine ReadCovarianceData), count the number of covariance data files that exist.

2. For each isotope in the list with a corresponding covariance data file (lines 300 to 303):

    a. Line 305 – For each covariance data file determine the following data using subroutine ReadCovarianceFileHeader. Subroutine ReadCovarianceData has a local array, CovHeaderData, for this data. The components of CovHeaderData(j) are given below in parentheses.

        1. ZA number for the isotope (FileZA)
        2. Number of energy bins (NErg)
        3. Number of pairs of ZA & MT numbers in the file (NumFileZAMT)
        4. Number of pairs of ZA & MT numbers in the file for which both MT numbers are relevant[10] and will be used (NumUsedZAMT)
        5. Number of unique MT numbers in the two MT lists below (NumUniqueMT)
        6. A list of the pairs of ZA & MT numbers for which both MT numbers are relevant (UsedZAMT(i=1,NumUsedZAMT) which has components ZA1, ZA2, MT1, MT2)
        7. A list of the unique (and relevant) MT numbers (UniqueMT(i=1,NumUniqueMT))

3. For each existing covariance data file (line 328):

    a. Line 329 – Record the ZA number in each block of covariance data (previously determined by line 305).

4. For each combination of two covariance files (lines 333 and 334):

    a. Line 345 – Record the list of unique (and relevant) MT numbers in first covariance data file in the corresponding block of covariance data.

    b. Line 346 – Record the list of unique (and relevant) MT numbers in second covariance data file in the corresponding block of covariance data.

    c. Lines 352 to 356 – Set the data pointers to null, i.e., CovarianceData % Iso(i,j) % Rxn(k,L) % Data to null (where i & j are indices to isotopes and covariance files, and k and L are indices to reactions).

5. For each isotope in the list with a corresponding covariance data file (lines 364 to 367):

    a. Read the covariance data by calling ReadCovarianceFile (see Section 7.11 below).

[10] An MT number is relevant when the the MT number is listed in the array RelevantMT (see Section 2.1).

7.11 Subroutine ReadCovarianceFile

This subroutine reads the covariance data from the file whose name is passed in. The read data may be either the base (unadjusted) or adjusted covariance values depending on the filename. This subroutine is listed in the CovarianceMatrixMod file.

7.11.1 Inputs

7.11.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
CovarianceData AdjustedCovarianceData	This	inout
Trim(FileName)	FileName	in
CovarianceHeaderData(n)	HeaderData	in

7.11.2 Results

The following components of CovarianceData or AdjustedCovarianceData in the Whisper main program are set:

- Data(n) % M(j1,j2)
- Iso(i1,i2) % Rxn(k1,k2) % Data
- Iso(i1,i2) % Rxn(k1,k2) % doTranspose

Where

i, i1, i2 = isotope index,
j1, j2 = energy group index (from 1 to 44),
k1, k2 = reaction index, and
n = numerical index.

7.11.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module CovarianceMatrixMod.

1. For each correlation (pair of ZA & MT numbers) in the file (line 475):

    a. Line 476 – Read the ZA and MT numbers for the two correlated isotopes-reactions (variables ZA1, MT1, ZA2, MT2) from the covariance data file.

    b. Line 482 – Read the covariance values into a one-dimensional array (variable CovData). This reads the covariance matrix column by column (see Appendix H) into the one-dimensional array. The covariance data (matrix rows and columns) are also in decreasing energy order (see Appendix H).

    c. If MT1 and MT2 are both relevant MTs (line 487):

        1. Lines 493 to 498 – Create a covariance matrix (variable M) from the covariance data (variable CovData) with the rows and columns in increasing energy order (see Sec. 5.2). The one-dimensional array is loaded into the matrix with the following arrangement:

CovData Index Value and M(i,j) Location							
M(i,j)	j = 1	j = 2	j = 3	…	j = n-2	j = n-1	j = n
i = 1	n^2	(n-1)n	(n-2)n	…	3n	2n	n
i = 2	n^2-1	(n-1)n-1	(n-2)n-1	…	3n-1	2n-1	n-1
i = 3	n^2-2	(n-1)n-2	(n-2)n-2	…	3n-2	2n-2	n-2
…	…	…	…	…	…	…	…
i = n-2	n^2-n+3	n^2-2n+3	n^2-3n+3	…	2n+3	n+3	3
i = n-1	n^2-n+2	n^2-2n+2	n^2-3n+2	…	2n+2	n+2	2
i = n	n^2-n+1	n^2-2n+1	n^2-3n+1	…	2n+1	n+1	1

Note: The covariance matrix can also be expressed as M(i,j) = CovData(n(n-j) +n-i+1), although Whisper does not use this formula.

2. Lines 500 to 501 – For ZA1 and ZA2, find the array index values (iZA and jZA) that point to the matching ZA numbers in the covariance list (component ZA). Index value is zero if isotope does not occur in the covariance data.

3. If both isotopes are in the covariance data (line 505):

    a. Line 506 – For MT1, find the array index value that points to the matching MT number in the MT list for the first isotope (component mt1).

    b. Line 507 – For MT2, find the array index value that points to the matching MT number in the MT list for the second isotope (component mt2).

    c. For the ZA indices *in original order* and the MT indices *in original order* {Iso(iZA,jZA) % Rxn(iMT,jMT)}:

        1. Line 510 – Set the data pointer to Data(iMat) which contains the M matrix just created.

        2. Line 511 – Set the doTranspose flag to false.

    d. If the covariance data was for two different isotopes or two different MT numbers (line 512):

        1. For the ZA indices *in reverse order* and the MT indices *in reverse order* {Iso(jZA,iZA) % Rxn(jMT,iMT)}:

            a. Line 513 – Set the data pointer to Data(iMat) which contains the M matrix just created.

            b. Line 514 – Set the doTranspose flag to true.

7.12 Subroutine MakeThermalScatterConsistent

Per the source code, this subroutine converts S(α,β) such that both inelastic and elastic are treated as elastic to maintain consistency with ORNL definitions. This subroutine is listed in the CovarianceMatrixMod file.

7.12.1 Inputs

7.12.1.1 Arguments

Whisper Variable*	Subroutine Variable	Intent
BenchmarkData(m)	kSen	inout
ApplicationData(n)		

*A vertically split cell indicates different variables used in separate call statements.

7.12.2 Results

When IsoSen(i) corresponds to S(α,β) cross sections, the following Whisper main program variables are recalculated (by separate subroutine calls):

- BenchmarkData (m) % IsoSen(i) % Sk(j,k)
- ApplicationData(n) % IsoSen(i) % Sk(j,k)

7.12.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module CovarianceMatrixMod.

1. For each isotope in the sensitivity data (line 645):
    a. If the isotope corresponds to S(α,β) cross sections, i.e., ZA number contains an alphabetical character, perform the following (line 647):
        1. Line 649 – Add the inelastic sensitivity factors (2nd column of Sk) to the elastic sensitivity factors (1st column of Sk).
        2. Line 650 – Set all sensitivity factors for reactions other than scattering (2nd and subsequent columns of Sk) to zero.

7.13 Subroutine EstimateUnknownBenchmarkUncertainties

For each benchmark that does not have a known uncertainty, the uncertainty is assumed to be the similarity coefficient weighted average of the known benchmark uncertainties. [LA-UR-14-26436, Sec. 3.3]

7.13.1 Inputs

7.13.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
BenchmarkData	BenchmarkData	inout
CovarianceData	CovarianceData	inout

7.13.2 Results

This subroutine calculates a value for:

- BenchmarkData(m) % KeffBenchUnc

when the input value of BenchmarkData(m) % KeffBenchUnc has an value less than 10^{-5}.

7.13.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module UncertaintyMod.

1. Lines 45 to 47 – Calculate variances of each benchmark data set (from variable KeffBenchUnc).
2. Perform an outer loop (m1) over each benchmark (line 50):
    a. If benchmark m1 has a variance $\leq 10^{-10}$ (line 51):
        1. Line 62 – Create the expanded covariance matrix and the sensitivity vector (called $\mathbf{C_{xx}}$ and $\mathbf{S_A}$ in LA-UR-14-26558) for benchmark m1 by calling ExpandCovarianceMatrixSensitivityVectors.
        2. Line 63 – Calculate the variance of benchmark m1 according to eq. 5 [LA-UR-14-26558, eq. 31].
        3. Perform an inner loop (m2) over each benchmark (line 67):
            a. If benchmark m2 has a variance $> 10^{-10}$ (line 68):
                1. Line 70 – Create the expanded covariance matrix and the sensitivity vectors (called $\mathbf{C_{xx}}$ and $\mathbf{S_B}$ in LA-UR-14-26558) for benchmark m1 and benchmark m2 by calling ExpandCovarianceMatrixSensitivityVectors.
                2. Line 71 – Calculate the variance of benchmark m2 according to eq. 5 [LA-UR-14-26558, eq. 31].
                3. Line 72 – Calculate the covariance between benchmark m1 and benchmark m2 according to eq. 6 [LA-UR-14-26558, eq. 30].
                4. Line 74 – Calculate the correlation coefficient according to eq. 7 [LA-UR-14-26558, eq. 32] and LA-UR-14-26558 (Section III.A.2) statement that "Whisper sets negative c_k to zero."

4. Line 78 – Calculate the benchmark uncertainty variance according to LA-UR-14-26558, eq. 44.

7.14 Subroutine ExpandCovarianceMatrixSensitivityVectors

This subroutine generates the expanded covariance matrix and two sensitivity vectors (for LA-UR-14-26558 eq. 31). This subroutine is in the CovarianceMatrixMod file. This subroutine is called by the following subroutines:

- EstimateUnknownBenchmarkUncertainties,
- CalculateNuclearDataUncertainties, and
- CalculateUpperSubcriticalLimits.

Subroutines ExpandCovarianceMatrixSensitivityVectors and ExpandCovarianceMatrixSensitivityMatrix are similar in that both produce the expanded covariance matrix. However, there are differences in the format of the input and output data. For the sensitivity data, subroutine ExpandCovarianceMatrixSensitivityVector requires two sets of benchmark data (i.e., two variables of type KeffSenDataType) for input and produces two vectors (two 1-d arrays) of sensitivity data. The output from ExpandCovarianceMatrixSensitivityVectors are components of an ExpandedCovarianceMatrixSensitivityVectorType variable.

7.14.1 Inputs

7.14.1.1 Arguments

Whisper Variable*	Subroutine Variable	Intent
CovarianceData AdjustedCovarianceData[+]	This	in
BenchmarkData(n1)	kSen1	in
BenchmarkData(n2)	kSen2	in
ExpCov ExpAdj[+]	EC	out
UnknownDataUncertainty	Unc	in (optional)

Where n2 may be equal to, or not equal to, n1.
*A vertically split cell indicates different variables used in separate call statements.
[+]When AdjustedCovarianceData is the input, ExpAdj is the output.

7.14.2 Results

The results for this subroutine are:

- the expanded covariance matrix
  (variable EC % C in subroutine ExpandCovarianceMatrixSensitivityVectors)

- two sensitivity vectors
  (variables EC % S1 and EC % S2 in subroutine ExpandCovarianceMatrixSensitivityVectors).

7.14.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module CovarianceMatrixMod.

1. Line 786 – Create a new matrix of sensitivity factors (UnionList) by merging the two matrices in the input benchmark data sets by calling subroutine UnionKeffSen.

   Note: For isotopes that are in both benchmarks, for each individual sensitivity coefficient, the input sensitivity coefficient with the largest magnitude (absolute value) is selected.

2. Line 787 – From the new matrix of sensitivity factors (UnionList), determine the *parameters* of the sensitivity/uncertainty map from the sensitivity and covariance data by calling function CovarianceMatrixConstructSUMapVector. (The components of EC % SUMap contain the parameters of the sensitivity/uncertainty map.)

3. Line 795 – Create a vector of sensitivity factors (EC % S1) from the sensitivity matrices in kSen1 for the isotopes in EC % SUMap by calling subroutine SUMapGetSensitivityVector.

4. Line 796 – Create a vector of sensitivity factors (EC % S2) from the sensitivity matrices in kSen2 for the isotopes in EC % SUMap by calling subroutine SUMapGetSensitivityVector.

5. Create the expanded covariance matrix (EC % C) by calling subroutine SUMapGetExpandedCovarianceMatrix.

   a. If the optional input for unknown data uncertainty is present (line797), call SUMapGetExpandedCovarianceMatrix with the unknown data uncertainty value (line 798). When there is no data for the combination of isotopes and reactions, the corresponding block of covariance data (see Section 5.11 above) is completed with the unknown data uncertainty value used as the diagonal values and zero as the non-diagonal values.

   b. If the optional input for unknown data uncertainty is not present (line797), call SUMapGetExpandedCovarianceMatrix without the unknown data uncertainty value (line 800). When there is no data for the combination of isotopes and reactions, the corresponding block of covariance data (see Section 5.11 above) is completed with all zeroes.

7.15 Function CovarianceMatrixConstructSUMapVector

This subroutine determines the *parameters* of the sensitivity/uncertainty map from the sensitivity and covariance data. This subroutine is in the CovarianceMatrixMod file.

7.15.1 Inputs

7.15.1.1 Arguments

Whisper Variable*	Subroutine Variable	Intent
CovarianceData	This	in
AdjustedCovarianceData		
UnionList+	kSen	in

*A vertically split cell indicates different variables used in separate call statements.
+Variable local to Subroutine ExpandCovarianceMatrixSensitivityVectors.

7.15.2 Results

Function CovarianceMatrixConstructSUMapVector returns variable Map (of derived type SUMapVectorType).

Function Variable	Type
Map % nBlocks	integer
Map % TotalSize	integer
Map % SU(:) % ZA	character(len=6)
Map % SU(:) % Rxn	integer
Map % SU(:) % MinEBin	integer
Map % Cov	pointer

Function CovarianceMatrixConstructSUMapVector ultimately sets the corresponding main program variables:

1. ExpCov % SUMap (the sensitivity/uncertainty map from the sensitivity and base covariance data) when called by subroutines:

    a. EstimateUnknownBenchmarkUncertainties,
    b. CalculateNuclearDataUncertainties,
    c. GetDiscrepancyCovarianceMatrix,
    d. AdjustNuclearData, and
    e. CalculateSimilarityWeights.

2. ExpAdj % SUMap (the sensitivity/uncertainty map from the sensitivity and adjusted covariance data) when called by subroutines:

    a. CalculateNuclearDataUncertainties.

7.15.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module CovarianceMatrixMod.

1. Line 193 – Save a pointer (variable Map % Cov) to the covariance data used to create the expanded covariance data (i.e., point to CovarianceData or AdjustedCovarianceData).

2. For all of the isotopes and reactions with a non-zero sensitivity vector (lines 198 to 200):

    a. Line 201 – Count the number of isotope & reaction pairs which have any non-zero sensitivity factors, i.e., the column or vector of the isotope sensitivity matrix has a non-zero value (variable Map % nBlocks).

3. For each isotope and reaction with a non-zero sensitivity vector (lines 212 to 214):

    a. Line 215 – Record the isotope ZA (variable Map % SU(iBlock) % ZA).

    b. Line 216 – Record the MT number of the reaction (variable Map % SU(iBlock) % Rxn).

c. Lines 219 to 224 – For the isotope sensitivity matrix (variable Sk) column for this isotope and reaction, find the first row (lowest energy bin) that is not zero.

   If a sequence of zero sensitivity values begins in the first (minimum) energy bin, the explicit zero values will be omitted from the sensitivity vector when possible.

d. If the first row with a non-zero sensitivity factor (energy index) is greater than 1 (line 227):

   Determine the minimum energy group (Map % SU(iBlock) % MinEBin) that has *either* a non-zero sensitivity factor *or* a non-zero diagonal element of M.

   1. Line 229 – Find (and record) the array index value that points to the isotope in the covariance data by calling function CovarianceMatrixFindZAIndex. If the isotope is not found in the sensitivity data, CovarianceMatrixFindZAIndex returns zero.

   2. If the isotope is in the covariance data (line 230):

      a. Line 231 – Find (and record) the array index value that points to the reaction (MT number) in the covariance data by calling function CovarianceMatrixFindMTIndex. If the reaction is not found in the sensitivity data, CovarianceMatrixFindMTIndex returns zero.

   3. If the isotope and MT number are in the covariance data (line 233):

      a. Lines 235 to 241 – Search the covariance matrix M for the lowest energy diagonal element that is non-zero. Record the minimum energy group (Map % SU(iBlock) % MinEBin) that has either a non-zero diagonal element of M or a non-zero sensitivity factor.

   4. If the isotope or reaction are not in the covariance data, i.e., there is not a matrix M for the isotope & reaction combination (line 233):

      a. Line 244 – Record the minimum energy group (Map % SU(iBlock) % MinEBin) based on the sensitivity factors alone (i.e., step 3.c above).

e. If the energy index value is not greater than 1 (lines 227):

   1. Record that the minimum energy group with non-zero sensitivity factors (Map % SU(iBlock) % MinEBin) is group 1.

f. Line 250 – Tally the total number of sensitivity factors that will be used (Map % TotalSize) because either the sensitivity factor or diagonal covariance value is not zero.

7.16 Subroutine SUMapGetSensitivityVector

This subroutine creates a vector of sensitivity factors (S) from the sensitivity matrices (Sen) for the isotopes and reactions in the covariance data. This subroutine is in the CovarianceMatrixMod file.

7.16.1 Inputs

7.16.1.1 Arguments

Whisper Variable*	Subroutine Variable	Intent
EC % SUMap	This	in
BenchmarkData(m)	Sen	in
ApplicationData(n)		
ExpCov % S1	S	out
ExpCov % S2		

*A vertically split cell indicates different variables used in separate call statements.

Component EC % SUMap is populated with the base covariance data when subroutine ExpandCovarianceMatrixSensitivityVectors calls SUMapGetSensitivityVector. Component EC % SUMap is populated with the adjusted covariance data when subroutine ExpandCovarianceMatrixSensitivityMatrix calls SUMapGetSensitivityVector.

7.16.2 Results

This subroutine creates a vector of sensitivity factors (S) from the sensitivity matrices (Sen) for the isotopes and reactions in the covariance data. See Section 5.6 above for the data organization.

7.16.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module CovarianceMatrixMod.

1. For each isotope and reaction which have any non-zero sensitivity factors (i.e., non-zero column of Sk) (line 670):

    a. Line 671 – Find the array index value that points to the matching isotope in the benchmark or application data (variable Sen) by calling function FindIso.

    b. Lines 672 to 674 – Find the array index value that points to the matching reaction in the relevant MT list (RelevantMT).

    c. For the minimum energy bin with a non-zero sensitivity factor or non-zero diagonal covariance value and above (line 675):

        1. If the isotope is in the covariance data (line 678):

            a. Line 679 – Copy the sensitivity factor from isotope sensitivity matrix.

        2. If the isotope is not in the covariance data (line 678):

            a. Line 681 – Set the sensitivity factor to zero.

7.17 Subroutine SUMapGetExpandedCovarianceMatrix

This subroutine creates a single matrix of the covariance values for the isotopes and reactions in the covariance data. This is called the expanded covariance matrix in Whisper. This subroutine is in the CovarianceMatrixMod file.

7.17.1 Inputs

7.17.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
EC % SUMap*	This	in
EC % C*	C	out
UnknownDataUncertainty	Unc	Optional in

*Local variables in subroutines ExpandCovarianceMatrixSensitivityVectors and ExpandCovarianceMatrixSensitivityMatrix.

7.17.2 Results

This subroutine creates a single matrix of the covariance values for the isotopes and reactions in the covariance data. This is called the expanded covariance matrix in Whisper. See Section 5.11 above for the data organization.

7.17.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module CovarianceMatrixMod.

1. For each isotope and reaction which have any non-zero sensitivity factors (i.e., non-zero column of Sk) perform the outer loop (line 707):

    a. Line 708 – For the isotope in this block of sensitivity/uncertainty data, find the array index value that points to the matching isotope in the covariance data by calling CovarianceMatrixFindZAIndex.

    b. Lines 709 to 713 – For the reaction in this block of sensitivity/uncertainty data, find the array index value that points to the matching reaction in the covariance data; when appropriate call CovarianceMattrixFindMTIndex.

    c. Perform an inner loop for each isotope and reaction which have any non-zero sensitivity factors (i.e., non-zero column of Sk) after the outer loop block (line 719):

        1. Line 720 – For the isotope in the inner loop block of sensitivity/uncertainty data, find the array index value that points to the matching isotope in the covariance data by calling CovarianceMatrixFindZAIndex.

        2. Lines 721 to 725 – For the reaction in the inner loop block of sensitivity/uncertainty data, find the array index value that points to the matching reaction in the covariance data; when appropriate call CovarianceMattrixFindMTIndex.

3. Lines 731 to 739 – Check if the combination of these two reactions exists in the covariance matrix.

4. If the combination of these two reactions exists in the covariance matrix (line 742):

   Copy the covariance matrix to the expanded covariance matrix.

   a. If the covariance data for the outer-inner combinations of isotopes and reactions *does* indicate that transposition is required (line 743):

      1. Line 744 – Transposed locations of the original matrix are copied to the inner loop row – outer loop column locations.

      2. Line 745 – The inner loop column – outer loop row matrix is transposed and copied to the outer loop row – inner loop column locations.

   b. If the covariance data for the outer-inner combinations of isotopes and reactions does *not* indicate that transposition is required (line 743):

      1. Line 747 – The covariance matrix (outer loop rows - inner loop columns) is copied (element by element) to the outer loop row – inner loop column locations.

      2. Line 748 – The outer loop row – inner loop column matrix is transposed and copied to the inner loop row – outer loop column locations.

5. If the combination of these two reactions does not exist in the covariance matrix (line 742):

   a. Lines 752 – Initialize inner loop row – outer loop column matrix to zero.

   b. If first location is on the matrix diagonal and optional uncertainty input (Unc) is present (line 753):

      Note: Optional uncertainty input is always present in Whisper 1.0.0.

      1. Lines 754 to 756 – Set diagonal elements to the variance (square of the uncertainty).

   c. If first location is *not* on the matrix diagonal or the optional uncertainty input (Unc) is *not* present (line 753):

      1. Line 758 – Set transposed locations (i.e., outer loop row – inner loop column) to zero.

7.18 Subroutine RejectBenchmarks

This subroutine performs the benchmark rejection which is described in LA-UR-14-26558 Sec. III.D (and LA-UR-14-23202 Sec. 4.1). This subroutine is in the DataAdjustmentMod file.

7.18.1 Inputs

7.18.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
BenchmarkData	BenchmarkData	inout
BenchmarkCorrel	BenchmarkCorrel	inout
CovarianceData	CovarianceData	in
ExcludedBenchmarks	ExcludedBenchmarks	inout

7.18.2 Results

The excluded benchmark data contains a list of benchmark filenames. These benchmarks were either excluded by the user via an input exclude benchmark file, or rejected by subroutine RejectBenchmarks. The other components of ExcludedBenchmarks are not set (or used) by Whisper.

7.18.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module DataAdjustmentMod.

1. If a benchmark rejection file was specified on the command line (line 329):

    a. Line 337 – For each benchmark, calculate the discrepancy between the calculated k_{eff} and the reference k_{eff} according to eq. 37 [LA-UR-14-26558, Sec. III.D].

    > Subroutine GetDiscrepancyVector calculates the *fractional* difference between KeffCalc and KeffBench. Section III.D of LA-UR-14-26558 defines discrepancy as the "difference of the calculated k_{eff} from its reference value."

    b. Line 338 – Generate the covariance matrix of the discrepancy vector according to eq. 36.

    c. Line 339 – Invert the covariance matrix of the discrepancy vector.

    d. Line 340 – Calculate the value of chi-squared per benchmark according to eq. 38.

    e. If the value of chi-squared per benchmark is less than or equal to the threshold value[11] (line 345):

        1. Line 355 – Return to the main program (no rejection is needed).

    f. While the value of chi-squared per benchmark is more than the threshold value (line 379):

        1. Lines 396 to 400 – Calculate the chi-squared diagonal terms according to eq. 39 [LA-UR-14-26558, eq. 45][12].

[11] The default threshold value in Whisper is 1.2 [LA-UR-14-26558, Sec. III.D].

[12] LA-UR-14-26558, Sec. III.D, refers to this as the diagonal χ^2 term, but χ^2 is a scalar and does nothave any diagonal terms. This calculation is for the diagonal terms of the inverse covariance matrix.

> Note: This calculation of chi-squared uses the difference between KeffCalc and KeffBench. Whereas, the fractional difference calculated by Subroutine GetDiscrepancyVector (called by lines 337 and 425) uses the *fractional* difference for the discrepancy and chi-squared values.

2. Line 409 – Identify the benchmark with the largest chi-squared diagonal (with variable j).
3. Lines 414 to 420 – Create new list of benchmarks excluding the benchmark with the largest chi-squared diagonal.
4. Line 422 – Create new set of benchmark correlations excluding the benchmark with the largest chi-squared diagonal.
5. Line 425 – For each benchmark remaining, calculate the discrepancy between the calculated k_{eff} and the reference k_{eff} [LA-UR-14-26558, Sec. III.D].
6. Line 426 – Generate the covariance matrix of the discrepancy vector.
7. Line 427 – Invert the covariance matrix of the discrepancy vector.
8. Line 429 – Calculate the value of chi-squared per remaining benchmarks.

g. Lines 452 to 454 – The excluded benchmark data is updated to include the rejected benchmarks.

7.19 Subroutine AdjustNuclearData

This subroutine either reads or calculates the adjusted covariance data, depending on the command line options. If a path to the adjusted covariance data is specified on the command line, Whisper calculates the adjusted covariance data from the base (unadjusted) covariance data and the application data. If a path is not specified and an application file is specified, Whisper reads previously calculated adjusted covariance data using the subdirectory 'Adjusted' in the directory specified by environment variable WHISPER_COVDATA_PATH. Whisper is distributed with adjusted covariance data that was precomputed based upon the benchmark suite and the recommended rejection set [LA-UR-14-26436, Sec. 3.3 and 3.5]. This subroutine is in the DataAdjustmentMod file.

7.19.1 Inputs

7.19.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
BenchmarkData	BenchmarkData	in
BenchmarkCorrel	BenchmarkCorrel	in
ApplicationData	ApplicationData	in
CovarianceData	CovarianceData	in
AdjustedCovarianceData	AdjustedCovarianceData	out

7.19.2 Results

This subroutine provides the adjusted covariance data (variable AdjustedCovarianceData) to the main program.

7.19.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module DataAdjustmentMod.

1. If a path to the adjusted covariance data was *not* specified on the command line (line 181):
    a. If an application file was specified on the command line (line 182):
        1. Line 190 – List all of the isotopes in the benchmark data.
        2. Line 191 – List all of the isotopes in the application data.
        3. Line 192 – Merge the benchmark and application isotopes lists into a single list.
        4. Line 194 – Read the adjusted covariance data (i.e., covariance data in the 'Adjusted' subdirectory) for the isotopes listed in the benchmark and application data by calling subroutine ReadCovarianceData.
2. If a path to the adjusted covariance data is specified on the command line:
    a. Lines 206 to 223 – Combine the benchmark and application data into a single array (variable CombinedData).
    b. Line 226 – Create the expanded covariance matrix and sensitivity matrix (variable ExpCov) by calling ExpandCovarianceMatrixSensitivityMatrix.
    c. Lines 244 to 248 – Calculate the covariance matrix of the relative difference vector (variable Cdd) according to eq. 36 [LA-UR-14-26558, eq. 40]. The steps of this calculation include:
        1. Line 244 – Calculate second RHS term of eq. 36.
        2. Line 247 – Calculate first RHS term of eq. 36.
    d. Lines 251 to 286 – Calculate the adjusted or residual covariance matrix according to eq. 44 [LA-UR-14-26558, eq. 41]. The steps of this calculation include:
        1. Line 251 – Invert the covariance matrix of the relative difference vector, i.e., variable Cdd becomes C_{dd}^{-1}.
        2. Line 259 – Calculate $C_{xx}S_{B,kx}^{T}$.

            Note: In variable ValidSkx, each column is a sensitivity vector. In LA-UR-14-26558, each row of the sensitivity matrix ($\mathbf{S}_{B,kx}$) is a sensitivity vector [LA-UR-14-26558, pg. 28]. Therefore, Whisper does not transpose the sensitivity matrix when LA-UR-14-26558 does, and vice versa.
        3. Line 266 – Calculate $C_{xx}S_{B,kx}^{T}C_{dd}^{-1}$.
        4. Line 274 – Calculate $C_{xx}S_{B,kx}^{T}C_{dd}^{-1}S_{B,kx}$.

5. Line 282 – Calculate $C_{xx}S_{B,kx}^{T}C_{dd}^{-1}S_{B,kx}C_{xx}$.

e. Line 289 – Compress the adjusted covariance data by calling subroutine CompressExpandedCovarianceMatrix.

f. Line 290 – Write the adjusted covariance data (for use in subsequent runs of Whisper) by calling subroutine WriteCovarianceData.

7.20 Subroutine ExpandCovarianceMatrixSensitivityMatrix

This subroutine generates the expanded covariance matrix and the sensitivity matrix (for LA-UR-14-26558 eq. 40 to 42). This subroutine is in the CovarianceMatrixMod file. This subroutine is called by the following subroutines:

- RejectBenchmarks (via GetDiscrepancyCovarianceMatrix), and
- AdjustNuclearData.

Subroutines ExpandCovarianceMatrixSensitivityVectors and ExpandCovarianceMatrixSensitivityMatrix are similar in that both produce the expanded covariance matrix and organize the sensitivity data. However, there are differences in the format of the input and output data. For the sensitivity data, subroutine ExpandCovarianceMatrixSensitivityMatrix requires an array of benchmark data (i.e., an array with each element of type KeffSenDataType) for input and produces a matrix (one 2-d array) of sensitivity data. The subroutine output has the derived type ExpandedCovarianceMatrixSensitivityMatrixType.

7.20.1 Inputs

7.20.1.1 Arguments

Whisper Variable*	Subroutine Variable	Intent
CovarianceData	This	in
BenchmarkData	kSen	in
CombinedData+		
ExpCov	EC	out
UnknownDataUncertainty	Unc	in (optional)

*A vertically split cell indicates different variables used in separate call statements.
+CombinedData is a combination (concatenation) of the benchmark and application data.

7.20.2 Results

The results for this subroutine are:

- the expanded covariance matrix
  (variable EC % C in subroutine ExpandCovarianceMatrixSensitivityMatrix)

- the sensitivity matrix
  (variable EC % S in subroutine ExpandCovarianceMatrixSensitivityMatrix).

Note: In variable EC % S, each column is a sensitivity vector. In LA-UR-14-26558, each row of the sensitivity matrix ($\mathbf{S}_{B,kx}$) is a sensitivity vector [LA-UR-14-26558, pg. 28]. Therefore, Whisper does not transpose the sensitivity matrix when LA-UR-14-26558 does, and vice versa.

7.20.3 Summary of Source Code

The logic is similar to that in subroutine ExpandCovarianceMatrixSensitivityVector (see Section 7.14.3 above).

7.21 Subroutine CalculateNuclearDataUncertainties

This subroutine calculates the 1σ uncertainty for the adjusted covariance data, which is used to calculate the baseline USL. (The 1σ uncertainty for the original covariance data is also calculated, but only for user information.) This subroutine is in the UncertaintyMod file.

7.21.1 Inputs

7.21.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
ApplicationData	ApplicationData	inout
CovarianceData	CovarianceData	in
AdjustedCovarianceData	AdjustedData	in

7.21.2 Results

The result of this subroutine is the data uncertainty component (DataUnc) for each application (in variable ApplicationData).

7.21.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module UncertaintyMod.

1. If an application file was specified on the command line (line 108):
    a. For each application (line 118):
        1. Line 119 – Create the covariance matrix and the sensitivity vector for the base (unadjusted) covariance data by calling subroutine ExpandCovarianceMatrixSensitivityVectors.

            Note: Variables ExpCov % S1 and ExpCov % S2 are identical sensitivity vectors because the same application data is given to subroutine ExpandCovarianceMatrixSensitivityVectors in the argument list.

        2. Line 120 – Create the covariance matrix and the sensitivity vector for the adjusted covariance data by calling subroutine ExpandCovarianceMatrixSensitivityVectors.

Note: Variables ExpAdj % S1 and ExpAdj % S2 are identical sensitivity vectors because the same application data is given to subroutine ExpandCovarianceMatrixSensitivityVectors in the argument list.

3. Line 122 – Calculate the 1σ uncertainty for the original covariance data according to eq. 11.

4. Lines 123 to 125 – Calculate the 1σ uncertainty for the adjusted covariance data (see explanation above).

7.22 Subroutine CalculateUpperSubcriticalLimits

This subroutine calculates the Upper Subcritical Limit (by first calculating the contributing factors to the USL). This subroutine is listed in the UpperSubcriticalLimitMod file.

7.22.1 Inputs

7.22.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
BenchmarkData	BenchmarkData	in
ApplicationData	ApplicationData	inout
CovarianceData	CovarianceData	in

7.22.2 Results

The following ApplicationData components in the Whisper main program are set:

- CalcMargin – initially by a call to CalculateCalculationalMargin, could be increased by subroutine CalculateUpperSubcriticalLimits
- Bias – by a call to CalculateCalculationalMargin
- BiasUnc – by a call to CalculateCalculationalMargin
- USL
- KeffOverUSL

7.22.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module UpperSubcriticalLimitMod.

1. If an application file was specified on the command line (line 45):

   a. Line 53 – Calculate the calculational margin using unweighted benchmark data by calling subroutine CalculateCalculationalMargin.

   b. If the calculation margin is less than the minimum non-coverage penalty (the default value of 0.05 can be changed by user options), set the calculation margin to the minimum non-coverage penalty (line 54).

   c. For each application file (line 57):

1. Line 58 – Calculate the calculational margin using weighted benchmark data by calling subroutine CalculateCalculationalMargin.

2. If the residual weight fraction, $(1 - w_{sum}/w_{req})$ in LA-UR-14-26558, is greater than zero (line 62):

    a. Line 63 to 67 – Recalculate the calculational margin using eq. 29 [LA-UR-14-26558 eq. 36].

3. Lines 72 to 75 – Calculate USL according to eq. 31 [LA-UR-14-26558 eq. 2, 37, and 43].

4. Lines 77 to 79 – Calculate the amount the application k_{eff} (upper limit of the confidence interval) exceeds the application USL according to LA-UR-14-26558, eq. 3.

7.23 Subroutine CalculateCalculationalMargin

7.23.1 Inputs

7.23.1.1 Arguments

Whisper Variable*	Subroutine Variable	Intent
BenchmarkData	BenchmarkData	in
UnweightedData ApplicationData(i)	ApplicationData	inout
CovarianceData	CovarianceData	in
SimilarityInfo	SimilarityInfo	out
Unweighted	WeightFlag	in

*A vertically split cell indicates different variables used in separate call statements.

7.23.2 Results

The following components of the application data in the Whisper main program, and the unweighted data in subroutine CalculateUpperSubcriticalLimits, are set:

- CalcMargin
- Bias
- BiasUnc

7.23.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module UpperSubcriticalLimitMod.

1. If non-uniform weighting factors on the normal distribution cumulative distribution functions (CDFs) in the extreme value CDF has been selected (line 149):

    a. Line 150 – Calculate the non-uniform weighting factors by calling subroutine CalculateSimilarityWeights.

2. If uniform weighting factors on the normal distribution CDFs in the extreme value CDF has been selected (line 149):

   a. Line 155 – Set all weighting factors to one.

3. Line 158 – Make new arrays (vectors) of the benchmark k_{eff} biases and bias uncertainties (variables muVec and sigmaVec) by calling subroutine GetBenchmarkBiasandUncertaintyVectors.

4. Lines 160 to 167 – Find the calculational margin where the extreme value CDF equals, or exceeds, the confidence level [LA-UR-14-23202, Sec. 3, last paragraph], i.e., eq. 16. The extreme value CDF is calculated with subroutine ExtremeNormalCDF. The default confidence level of 99% may be changed in the user options.

5. Line 169 – Calculate the application bias. Subroutine ExtremeNormalMean evaluates the integral in eq. 23 using the trapezoidal rule (as stated in LA-UR-14-26558).

6. Line 170 – Calculate the bias uncertainty according to LA-UR-14-26558 eq. 26.

7. Line 173 – Calculate the calculation margin according to eq. 26 and 27 [LA-UR-14-26558, eq. 27 and 7].

7.24 Subroutine CalculateSimilarityWeights

7.24.1 Inputs

7.24.1.1 Arguments

Whisper Variable	Subroutine Variable	Intent
BenchmarkData	BenchmarkData	in
ApplicationData(i)	ApplicationData	in
CovarianceData	CovarianceData	in
Wgt	Wgt	out
SimilarityInfo	SimilarityInfo	out

7.24.2 Results

This subroutine calculates the non-uniform weighting factors on the normal distribution CDFs in the extreme value CDF. This subroutine also calculates the similarity information (variable SimilarityInfo). The residual weight fraction (variable SimilarityInfo % ResidualWeightFrac) is used in subroutine CalculateUpperSubcriticalLimits to calculate the calculational margin.

7.24.3 Summary of Source Code

The following is a summary of the calculations performed in this subroutine. The line numbers are from the file for module UpperSubcriticalLimitMod.

1. Line 214 – Create the covariance matrix and the sensitivity vector for the application data by calling ExpandCovarianceMatrixSensitivityVectors.

2. Line 215 – Calculate the variance of the application data according to eq. 5 [LA-UR-14-26558, eq. 31].

3. For each benchmark (line 218):

    a. Line 220 – Create the covariance matrix and the sensitivity vector for the benchmark data by calling ExpandCovarianceMatrixSensitivityVectors.

    b. Line 221 – Calculate the variance of the benchmark data according to eq. 5 [LA-UR-14-26558, eq. 31].

    c. Line 222 – Calculate the covariance according to eq. 6 [LA-UR-14-26558, eq. 30].

    d. Line 224 – Calculate the correlation coefficient according to eq. 7 [LA-UR-14-26558, eq. 32] and LA-UR-14-26558 (Sec. III.A.2) statement that "Whisper sets negative c_k to zero."

4. Lines 233 to 251 – Choose acceptance criteria ($c_{k,acc}$ in LA-UR-14-26558) such that eq. 15 [LA-UR-14-23202 eq. 8] {a variation of LA-UR-14-26558 eq. 33} is satisfied or the acceptance criteria reaches zero.

    a. Line 236 – Calculate required weight according to eq. 13 [LA-UR-14-26558, eq. 35].

    b. Lines 242 to 247 – Calculate weighting factors according to eq. 14 [LA-UR-14-26558, eq. 34].

    c. Line 249 – Calculate LHS of eq. 15 [LA-UR-14-23202 eq. 8]{a variation of LA-UR-14-26558 eq. 33}.

5. Line 265 – Calculate the residual weight fraction, $\left(1 - w_{sum}/w_{req}\right)$, which will be used later in LA-UR-14-26558 eq. 36.

7.25 Subroutine WriteUSLOutputSummaryTable

If an application file was specified on the command line, the USL Summary Table is written to the output file.

8. References

B. T. Rearden, M. L. Williams, M. A. Jessee, D. E. Mueller, and D.A. Wiarda, *Sensitivity and Uncertainty Analysis Capabilities and Data in SCALE,* Nuclear Technology, Vol. 174, pp. 236-288, May 2011

LA-CP-13-00634, Rev. 0, *MCNP6 User's Manual*, Version 1.0, May 2013

LA-UR-14-23202, *Methodology for Sensitivity and Uncertainty-Based Criticality Safety Validation,* Brian C. Kiedrowski, 2104-05-07

LA-UR-14-23352, *Validation of MCNP6.1 for Criticality Safety of Pu-Metal, -Solution, and –Oxide Systems*, Brian C. Kiedrowski, et. al., 2104-05-13

LA-UR-14-26436, *User Manual for Whisper (v1.0.0), Software for Sensitivity- and Uncertainty-Based Nuclear Criticality Safety Validation,* Brian Christopher Kiedrowski, 2104-08-13

LA-UR-14-26558, *Whisper: Sensitivity/Uncertainty-Based Computational Methods and Software for Determining Baseline Upper Subcritical Limits*, Brian Christopher Kiedrowski, et. al., 2104-08-19

ORNL/TM-2005/39, Version 6.1, Scale: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design, June 2011

Whisper Program Suite Validation and Verification Report, R. F. Sartor and F. B. Brown, Los Alamos National Laboratory, LA-UR publication pending

Appendix A

Equation to Subroutine Crosswalk

The following table is a crosswalk between the equations in LA-UR-14-26558 and the Whisper subroutines. Note: Not every equation in LA-UR-14-26558 is listed.

LA-UR-14-26558 Equation	Whisper Subroutine (Report Section)
2	Subroutine CalculateUpperSubcriticalLimits (7.22)
3	Subroutine CalculateUpperSubcriticalLimits (7.22)
7	Subroutine CalculateCalculationalMargin (7.23)
19	Function ExtremeNormalPDF Function ExtremeNormalCDF
20	Function ExtremeNormalPDF
21	Subroutine ReadAndSetupBenchmarks (7.5)
22	Subroutine ReadAndSetupBenchmarks (7.5)
23	Function NormalCDF
24[13]	Subroutine CalculateCalculationalMargin (7.23)
25	Subroutine CalculateCalculationalMargin (7.23)
26	Subroutine CalculateCalculationalMargin (7.23)
27	Subroutine CalculateCalculationalMargin (7.23)
30	Subroutine EstimateUnknownBenchmarkUncertainties (7.13) - and – Subroutine CalculateSimilarityWeights (7.24)
31	Subroutine EstimateUnknownBenchmarkUncertainties (7.13) - and – Subroutine CalculateSimilarityWeights (7.24)
32	Subroutine EstimateUnknownBenchmarkUncertainties (7.13) - and – Subroutine CalculateSimilarityWeights (7.24)
33	Subroutine CalculateSimilarityWeights (7.24)
34	Subroutine CalculateSimilarityWeights (7.24)
35	Subroutine CalculateSimilarityWeights (7.24)
36	Subroutine CalculateUpperSubcriticalLimits (7.22) - and – Subroutine CalculateSimilarityWeights (7.24)
37	Subroutine CalculateUpperSubcriticalLimits (7.22)
40	Subroutine AdjustNuclearData (7.19)
41	Subroutine AdjustNuclearData (7.19)

[13] Equation (24), $F(m) = q$, is implemented as $F(m) \geq q$ in Whisper.

LA-UR-14-26558 Equation	Whisper Subroutine (Report Section)
42	Subroutine CalculateNuclearDataUncertainties (7.21)
43	Subroutine CalculateUpperSubcriticalLimits (7.22) where the square root operation is in: Subroutine CalculateNuclearDataUncertainties (7.21)
44	Subroutine EstimateUnknownBenchmarkUncertainties (7.13)
45	Subroutine RejectBenchmarks (7.18)

Appendix B

Questions and Answers

1. Module ParameterMod – This module defines two variables, NRxn and NumRelevantMTs. Because reaction and MT number correlate to each other, these two variables define the same program parameter. Consider removing the redundancy to avoid possible conflicts due to a change in one but not the other.

   Answer: This is a matter of programming style or choice, not correctness. Such choices may affect code understanding or organization, but do not affect results. (communication from Forrest Brown, XCP-3, 1/16/2015)

2. Function ReadKeffSenData – If an isotope has more than one data set in the benchmark (or application) data file, then the subsequent sensitivity factors are added to the previous sensitivity factors when constructing the sensitivity matrix in Whisper. Why would it be appropriate to sum sensitivity factors?

   Answer: An isotope can appear in more than one material, or more than once in the same material, or at more than one location (tally cell). The sensitivities are computed per isotope and reaction, not per material or location. (communication from Forrest Brown, XCP-3, 1/16/2015)

3. Subroutine ReadAndSetupCovarianceData – This subroutine ends with two calls to subroutine MakeThermalScatterConsistent with the benchmark and application data (InputProcessingMod lines 663 to 668). Since subroutine MakeThermal ScatterConsistent uses and modifies only benchmark and application data, IsoSen(i) % Sk(j,k), why are these calls in subroutine ReadAndSetupCovarianceData rather than the ReadAndSetupBenchmarks and ReadAndSetupApplications subroutines?

   Answer: This is a matter of programming style or choice, not correctness. Such choices may affect code understanding or organization, but do not affect results. (communication from Forrest Brown, XCP-3, 1/16/2015)

4. Subroutine ReadCovarianceFile – CovarianceMatrixMod line 476 is:

```
    read(ScratchUnit,*) ZA1, MT1, ZA2, MT2
```

   Using cov.1001.data as an example, the corresponding input line is:

```
                1001      1   1001      1      1
```

   Is Whisper executing properly while reading four variables when the covariance data file has five data values?

   Answer: Fortran input/output works with records. The above input line is one record. The Fortran read statement inputs the entire record although only the first four values are transferred to variables. ("Advancing input/output is record oriented and leaves a file positioned between records even if an input operation does not completely consume a record." [Fortran 2003 Handbook; Sec. 9.1.4])

5. Subroutine ReadCovarianceData – CovarianceMatrixMod line 477

```
    read(ScratchUnit,*) ( Map(j), NRow(j), j=1, n )
```

What is the data in variables Map(j) and NRow(j)?

Answer: The general format for the covariance files permits large datasets to be stored in "blocks", excluding large sections of the covariance matrix containing only zeros from the file storage. The initial coding effort for Whisper made some attempt to handle the general file storage format, with the complicated block-oriented storage scheme. However, because the only available covariance data was the relatively compact 44-group ORNL data, which does not require or use the block-oriented storage scheme, the coding for reading the covariance data files was simplified. The items mentioned in questions 5 and 6 are leftover remnants from the initial coding. While not necessary, they do nothing to interfere with the correct processing of the 44-group covariance files.

This restriction to the 44-group covariance data does not affect results to date. In the future, some additions to the coding will be required if it is desired to read larger, block-oriented covariance data files (if & when such files are produced).

(communication from Forrest Brown, XCP-3, 1/16/2015)

6. Subroutine ReadCovarianceFile – CovarianceMatrixMod lines 479 and 480 are:

```
    Map(:) = Map(:) - NRow(:) + 1
    NumCov = sum( Map )
```

Map and NumCov are local variables for subroutine ReadCovarianceFile. Are variables Map and NumCov used after line 480? Are lines 479 and 480 superfluous code?

Answer: See answer to question 5 above.

7. Subroutine ReadCovarianceFile – Different variables, nErg and HeaderData % nErg, are used to allocate and populate matrix M (in This % Data); the code specifics are given in the bullets below. If the values of these two variables are ever different, subroutine ReadCovarianceFile will not execute properly. Should Whisper verify that nErg and HeaderData % nErg have the same value to ensure proper operation of subroutine ReadCovarianceFile?

   - CovarianceMatrixMod line 489 performs the allocation M(1:nErg,1:nErg). Variable nErg is a parameter set to 44 in module ParametersMod.

   - Lines 494 to 498 use variable n to populate M. Variable n is set to HeaderData % nErg in line 461. HeaderData is an argument variable in Subroutine ReadCovarianceFile. Variable HeaderData % nErg is read from the covariance file by subroutine ReadCovarianceFileHeader (line 396).

   Answer: This is a matter of programming style or choice, not correctness. Such choices may affect code understanding or organization, but do not affect results. (communication from Forrest Brown, XCP-3, 1/16/2015)

8. Subroutine EstimateUnknownBenchmarkUncertainties – The experimental uncertainty for each benchmark (variable BenchmarkData(i) % KeffBenchUnc) is initially an input value read from a file (subroutine ReadAndSetupBenchmarks). This input value is used to calculate the bias uncertainty (BenchmarkData(j) % SigmaBias) in subroutine ReadAndSetupBenchmarks. For benchmarks with unknown (or unrealistically low) experimental uncertainty, subroutine EstimateUnknownBenchmarkUncertainties calculates an estimate of the experimental uncertainty. However, subroutine EstimateUnknownBenchmarkUncertainties (or any subroutine) does not recalculate BenchmarkData(j) % SigmaBias. Subsequently, BenchmarkData(j) % SigmaBias is used to calculate the calculational margin, application bias, and application bias uncertainty.

   Answer: Subroutine EstimateUnknownBenchmarkUncertainties should recalculate BenchmarkData(j) % SigmaBias after updating BenchmarkData(i) % KeffBenchUnc. In the interim, the benchmark uncertainties calculated by Whisper can be used in subsequent Whisper runs by inserting the Whisper results in the benchmark library (TOC) file.

   This issue is identified in a problem report (NCS-SQM-WHISPER-PROBID07) in order to initiate corrective actions.

9. Subroutine RejectBenchmarks – When calculating the chi-squared diagonal values for deciding whether a benchmark should be rejected (DataAdjustmentMod lines 337 and 425), the **fractional difference** between KeffCalc and KeffBench is used as the discrepancy value. When calculating the chi-squared diagonal values for selecting a benchmark for rejection (DataAdjustmentMod lines 396 to 400), the **magnitude of the difference** between KeffCalc and KeffBench is used as the discrepancy value. Furthermore, LA-UR-14-26558, Sec. III.D describes the discrepancy as the difference between the calculated k_{eff} from its reference value without mentioning any scaling.

   Answer: For consistency, the treatment for rejecting benchmarks (using the magnitude of the difference) should be changed to use fractional differences. The practical effect on results should be entirely negligible, however, since the absolute differences only need to be divided by values that are very close to 1.0 (that is, the benchmark k-effectives, which differ from 1.0 by only fractions of a percent). (communication from Forrest Brown, XCP-3, 1/16/2015)

   This issue is identified in a problem report (NCS-SQM-WHISPER-PROBID06) in order to initiate corrective actions.

10. Subroutine CalculateUpperSubcriticalLimits – UpperCriticalLimitMod line 54 performs UnweightedData % CalcMargin = max(UnweightedData % CalcMargin, MinimumNonCoveragePenalty). Why is it necessary/appropriate to have a minimum limit (MinimumNonCoveragePenalty) for the unweighted calculation margin?

    Answer: This is just conservatism. It would be unthinkable to have a calculation margin of zero, or one that is artificially small due to lack of a sufficient number of cases. (communication from Forrest Brown, XCP-3, 1/16/2015)

11. The value of the calculational margin due to the code and method (variable CodeAndMethodMargin) is a named constant with the value 0.005 in the module ParametersMod. In the next revision, consider changing variable CodeAndMethodMargin from being a parameter to a user option variable to give users control over this value.

Answer: This parameter is based on the carefully considered expert judgment of the most experienced MCNP code developer. It is not a parameter that should be overruled by a user, no matter how experienced. If a user disagrees with the value of this parameter, it is a simple matter to add or subtract from the final baseline USL determined by Whisper. Such changes, however, would require complete documentation and justification.

Appendix C

Treatment of Sensitivity Factors for S(α, β) Cross Sections

For materials with S(α, β) cross sections (hydrogen bound in light water, zirconium hydride, polyethylene, and heavy water; deuterium in heavy water; beryllium metal; and graphite), the covariance data for the inelastic scattering cross sections will be approximated with the covariance data for the elastic scattering cross sections. To use the elastic scattering covariance data, the inelastic scattering cross section sensitivity coefficients calculated by MCNP6 are added to the elastic scattering cross section sensitivity coefficients. The original inelastic scattering cross section sensitivity coefficients are then set to zero to avoid double counting the sensitivity to low-energy inelastic scattering.

This appendix justifies why this is the appropriate covariance data and sensitivity coefficient treatment. The following discussion begins with a review of thermal scattering kernels, MCNP6 sensitivity factors, and covariance data. Then the sources of the covariance data are reviewed to discover the applicability.

Thermal Scattering Kernels

Molecular effects or crystalline structure effects can affect the neutron scattering cross sections. These effects are included in the S(α, β) thermal neutron scattering data. Every S(α, β) material will have inelastic scattering and may have either coherent or incoherent elastic scattering (but not both). [LA-UR-12-00800]

To treat a particular isotope (or isotopes) as a molecular compound in the thermal regime in MCNP6, the MT card with one of the available S(α,β) data sets is used. As MCNP6 transports particles, the free-gas treatment is used down to the energy where S(α,β) data are available. At that point, the S(α,β) treatment automatically overrides the free-gas treatment; there is no mixing of the two treatments for the same isotope in the same material at a given energy. Typically the free-gas model is used for each isotope of a material down to a few electron volts and then the S(α,β) treatment takes over for the isotope(s) comprising the substance specified on the MT card. In general, S(α,β) effects are most significant below 2 eV. [LA-CP-13-00634; Sec. 3.3.2.2]

The following table lists the MCNP6 S(α,β) identifiers used in the Whisper program suite and the corresponding SCALE identifier.

MCNP6 S(α,β) Identifier	Description[(a)]	SCALE ZA Identifier	Description[(b)]
lwtr	Hydrogen in Light Water	1001	Hydrogen in water with a S(α,β) thermal kernel
h-zr	Hydrogen in Zirconium Hydride	1701	Hydrogen in zirconium hydride with a S(α,β) thermal kernel
poly	Hydrogen in Polyethylene	1901	Hydrogen in polyethylene with a S(α,β) thermal kernel
hwtr	Deuterium in Heavy Water	1002	Deuterium in heavy water with S(α,β) thermal kernel
be	Beryllium Metal	4309	Beryllium metal with a S(α,β) thermal kernel (Bebound)

MCNP6 S(α,β) Identifier	Description[(a)]	SCALE ZA Identifier	Description[(b)]
grph	Graphite	6312	Graphite carbon

(a) LA-UR-13-21822; Table 7: Thermal S(α,β) cross-section libraries, and Table 8: continuous-energy neutron data library

(b) ORNL/TM-2005/39; Table M8.2.3, Elements and special nuclide symbols, Table M8.2.4, Compounds, and Section M4.A.2, The 44-Group ENDF/B-V Library

MCNP6 produces sensitivity factors for the S(α, β) cross section separate from the parent material cross sections. For all cross sections, including the S(α, β) cross sections, MCNP6 produces separate sensitivity factors for elastic and inelastic scattering.

MCNP6 Sensitivity Factors

MCNP6 calculates sensitivity factors for the cross sections used. There are separate sensitivity factors for elastic and inelastic scattering.

Covariance Data in the Whisper Program Suite

The covariance data in SCALE6 was transferred to the Whisper program suite. In the Whisper program suite, the covariance data is used with sensitivity factors from MCNP6. The covariance data available for the S(α, β) identifiers is:

Nuclide	Covariance Data Reactions (MT numbers)
lwtr	total (1), elastic scatter (2), capture (102)
poly	total (1), elastic scatter (2), capture (102)
grph	total (1), elastic scatter (2), inelastic scatter (4), capture (102), n,p (103), n,d (104), n,alpha (107)
be	total (1), elastic scatter (2), n,2n (16), capture (102), n,p (103), n,d (104), n,t (105), n,alpha (107)
hwtr	total (1), elastic scatter (2), n,2n (16), capture (102)
h-zr	total (1), elastic scatter (2), capture (102)

For neutron scattering, most of these nuclides have only one set of covariance data, identified as elastic scattering. These nuclides do not have separate covariance data for inelastic scattering (the primary reason for the S(α, β) thermal neutron scattering data). Graphite is the only exception with covariance data for both elastic and inelastic scattering.

Covariance Data Sources

The source of the covariance data is:

Nuclide	Data source	Comments
Bebound	BNL-LANL-ORNL (BLO) approximate data	
C-graphite	ENDF/B-VI	Duplicate of carbon

Nuclide	Data source	Comments
C	ENDF/B-VI	
^{1}H	above 5 keV :BLO LANL evaluation below 5 keV: JENDL 3.3	
H-ZrH	above 5 keV :BLO LANL evaluation below 5 keV: JENDL 3.3	Duplicate of ^{1}H
H-poly	above 5 keV :BLO LANL evaluation below 5 keV: JENDL 3.3	Duplicate of ^{1}H
Hfreegas	above 5 keV :BLO LANL evaluation below 5 keV: JENDL 3.3	Duplicate of ^{1}H
^{2}H	BLO approximate data	
Dfreegas	BLO approximate data	Duplicate of ^{2}H

[ORNL/TM-2005/39; Table M19.4.1]

Beryllium and Deuterium (BLO Approximate Data)

The SCALE-6 covariance data includes approximate uncertainties from the collaborative effort between Brookhaven National Laboratory (BNL), Los Alamos National Laboratory (LANL), and ORNL to produce "low fidelity" (lo-fi) covariances spanning the full energy range [Williams and Rearden]. Los Alamos had responsibility for covariance data for the light isotopes (up through ^{19}F) over the entire energy range [Little]. The covariance evaluation for the light elements includes high-fidelity R-matrix analyses for a few reactions {^{1}H(n,n), ^{1}H(n,γ), ^{10}B(n,n), and ^{10}B(n,α)}, but the majority are low-fidelity estimates [Evaluation of Covariances for Actinides and Light Elements at LANL].

Low fidelity uncertainties in the thermal range were produced by a simple integral approximation which uses uncertainties in the integral measurements of thermal cross sections to approximate differential data uncertainties [NEA/NSC/DOC(2007)23]. The method does not provide uncertainties for inelastic scattering or other types of data with no integral measurements [Williams, et. al.].

In the case of molecules or crystals, the integral measurements will inherently include the effects of the molecule or crystal on the neutron scattering cross sections. Therefore, the covariance data from the integral approximation method includes the low-energy inelastic scattering. The covariance data from integral approximations is for all scattering, elastic and inelastic. For beryllium metal (be) and deuterium (^{2}H), it is appropriate for the Whisper program to add the inelastic scattering cross section sensitivity coefficients calculated by MCNP6 to the scattering cross section sensitivity coefficients.

Hydrogen (JENDL 3.3 Data)

Although the table above (from ORNL/TM-2005/39) states that the covariance data for free gas hydrogen is a duplicate of the covariance data for ^{1}H, i.e., hydrogen in water, this statement is backwards. For a point of reference, the current ENDF/B-VII.1 nuclear data does not have covariance data specifically for hydrogen in water[14], although there is covariance data for free gas hydrogen. Therefore, free gas hydrogen is considered to be the source of the bound hydrogen covariance data.

[14] http://www.nndc.bnl.gov/exfor/servlet/E4sGetTabSect?EvalID=22417&req=6525

For scattering (mt = 2), the JENDL 3.3 covariances of the hydrogen elastic scattering (mt = 2) cross sections is based on ENDF/B-V data except that the standard deviation below 1 keV was changed to 0.1% [JAERI-Data/Code 2002-26]. ENDF/B-V was the first time ENDF/B included covariance data (file 33) for carbon, and only the total (mt = 1), elastic scattering (mt = 2), and capture (mt = 102) cross sections were included [BNL-NCS-17541 (1979)].

At this point there is a choice for the covariance data for inelastic scattering with bound hydrogen: use the default covariance matrix (diagonal elements of 0.10 or the user input value) or the hydrogen S(α, β) elastic scattering covariance matrix. The elastic scattering covariance matrix was selected as a better estimate of the low-energy inelastic scattering behavior. Therefore, for bound hydrogen (in light water, ZrH, or polyethylene), it is appropriate for the Whisper program to add the inelastic scattering cross section sensitivity coefficients calculated by MCNP6 to the elastic scattering cross section sensitivity coefficients.

Carbon Graphite (ENDF/B-VI)

The covariance data for carbon graphite is the covariance data for carbon [ORNL/TM-2005/39, Table M19.4.1]. Carbon is one of the fifty materials in the SCALE6 covariance data with high-fidelity covariance data. For these fifty materials, the evaluation of nuclear data utilized a regression algorithm to determine the nuclear physics model parameters that fit the experimental measurements. Information from the regression analysis can be propagated to uncertainties and correlations in the evaluated differential data. [ORNL/TM-2005/39; Sec. M19.2.1]

Unlike the other isotopes listed in the table above, carbon graphite has both an S(α, β) scattering kernel and covariance data for the inelastic scattering (mt = 4) cross sections. At this point, the question is: does the inelastic scattering cross section covariance data include low-energy scattering behavior due to molecule or crystal effects? The ENDF/B-VI documentation for carbon has no indication that the inelastic cross section covariance data includes molecular or crystalline effects. For example:

1. In BNL-NCS-17541 (1991), the two energy ranges discussed for the uncertainty files are < 2 MeV and above.

2. In BNL-NCS-17541 Supplement I (1996), the lowest energy discussed for the uncertainty (of total elastic scattering cross section) is 1 keV.

3. In Fu & Perey (1978), the thermal cross sections section discusses the total, capture, and scattering cross sections but scattering is not divided further into elastic and inelastic types.

Furthermore, the covariance data for the inelastic scattering cross sections (with the total, elastic scattering, and inelastic scatter cross sections) has non-zero values only for energies above 10 keV. Therefore, the available covariance data for carbon graphite inelastic scattering cross sections is not applicable to low-energy inelastic scattering or the S(α, β) cross sections.

At this point there is a choice for the covariance data for inelastic scattering with carbon graphite, use the default covariance matrix (diagonal elements of 0.10 or the user input value) or the carbon S(α, β) elastic scattering covariance matrix. The elastic scattering covariance matrix was selected as a better estimate of the low-energy inelastic scattering behavior. Therefore, for carbon graphite (grph), it is appropriate for the Whisper program to add the inelastic scattering cross section sensitivity coefficients calculated by MCNP6 to the elastic scattering cross section sensitivity coefficients.

References

BNL-NCS-17541, *ENDF-102 ENDF/B-VI Summary Documentation*, Brookhaven National Laboratory, National Nuclear Data Center, Oct. 1991

BNL-NCS-17541 Supp. I, *ENDF-102 ENDF/B-VI Summary Documentation Supplement I, ENDF/HE-VI Summary Documentation*, Brookhaven National Laboratory, National Nuclear Data Center, Dec. 1996

BNL-NCS-50496, 2nd Ed. (ENDF/B-V) Revised, *ENDF-102 Data Formats and Procedures for the Evaluated Nuclear Data File, ENDF/B-V*, 1983

Fu, C. Y. and Perey, F. G., *Neutron Scattering Cross Sections of Carbon Below 2 MeV Recommended from R-Matrix Fits to Data*, Atomic Data and Nuclear Data Tables 22, 1978, pages 249-267

JAERI-Data/Code 2002-26, *Descriptive Data of JENDL-3.3 Part I (Z=1-50)*, Jan. 2003 [http://jolissrch-inter.tokai-sc.jaea.go.jp/pdfdata/JAERI-Data-Code-2002-026-Part1.pdf]

Kawano, T., et. al., *Evaluation of Covariances for Actinides and Light Elements at LANL*, Nuclear Data Sheets 109, 2008, pages 2817-2821

LA-CP-13-00634, Rev. 0; *MCNP6 User's Manual*, Los Alamos National Laboratory, 2013

LA-UR-12-00800, *Release of Continuous Representation for S(α, β) ACE Data*, Los Alamos National Laboratory, 2012

LA-UR-13-21822, *Listing of Available ACE Data*, Los Alamos National Laboratory, 2013

Little, R.C., et. al., *Low-fidelity Covariance Project*, Nuclear Data Sheets 109, 2008, pages 2828–2833

NEA/NSC/DOC(2007)23, *Benchmark for Uncertainty Analysis in Modelling (UAM) for Design, Operation, and Safety Analysis of LWRs*, Volume 1, Organization for Economic Co-operation and Development (OECD), 21-Jan-2008

ORNL/TM-2005/39, Version 6.1, *Scale: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design*, June 2011

Williams, M.L. and Rearden, B.T., *SCALE-6 Sensitivity/Uncertainty Methods and Covariance Data*, Nuclear Data Sheets 109, 2008, pages 2796–2800

Williams, Mark L., et al. *Approximate Techniques for Representing Nuclear Data Uncertainties*, Eighth International Topical Meeting on Nuclear Applications and Utilization of Accelerators (AccApp'07), 2007

Appendix D

Derived Variable Types in Whisper

The following tables list the derived variable types used in the program Whisper. Declarations of private or public are appended to the type name (in **bold**) in the table title. The subtitle identifies the module that defines the derived variable type. The table provides the component name, data type, and initial value (if set). The components that are not for data storage, i.e., references to other derived variables and pointers to procedures, are in light grey highlighting. Extend operations, and the variables incorporated by the extend operation, are identified with a triple line (|||) border on the left edge of the table.

ApplicationDataType
(module SensitivityMod)

Component	Data type	Init. Value
extends(KeffSenDataType)		
NIso	integer	
MaxIso	integer	
ZA(:)	character(len=20)	allocatable
Init	procedure	
AddIso	procedure	
WriteIso	procedure	
IsoExists	procedure	
FindIso	procedure	
FileName	character(len=40)	
KeffCalc	real(8)	
KeffCalcUnc	real(8)	
IsoSen(:)	type(IsoSenType)	allocatable
IsoSen(:) % Sk(:,:)	real(8)	allocatable
UnionizeKeffSen	procedure	
UnionizeKeffSenVector	procedure	
Union[6]	generic => UnionizeKeffSen, UnionizeKeffSenVector	
DataUnc	real(8)	
CalcMargin	real(8)	
Bias	real(8)	
BiasUnc	real(8)	
USL	real(8)	
KeffOverUSL	real(8)	

BenchmarkCorrelationMatrixType (public)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
Name(:)	character(len=40)	allocatable
Mat(:,:)	real(8)	allocatable

BenchmarkDataType
(module SensitivityMod)

Component	Data type	Init. Value
extends(KeffSenDataType)		
NIso	integer	
MaxIso	integer	
ZA(:)	character(len=20)	allocatable
Init	procedure	
AddIso	procedure	
WriteIso	procedure	
IsoExists	procedure	
FindIso	procedure	
FileName	character(len=40)	
KeffCalc	real(8)	
KeffCalcUnc	real(8)	
IsoSen(:)	type(IsoSenType)	allocatable
IsoSen(:) % Sk(:,:)	real(8)	allocatable
UnionizeKeffSen	procedure	
UnionizeKeffSenVector	procedure	
Union[6]	generic => UnionizeKeffSen, UnionizeKeffSenVector	
KeffBench	real(8)	
KeffBenchUnc	real(8)	
KeffBias	real(8)	
SigmaBias	real(8)	

CovarianceMatrixType (public)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
Data(:)	type(MatrixType)	allocatable
M(:,:)	real(8)	allocatable
Iso(:,:)	type(IsotopeLevelCovarianceMatrixType)	allocatable
Iso(:,:) % Rxn(:,:)	type(ReactionLevelCovarianceMatrixType)	allocatable
Iso(:,:) % Rxn(:,:) % Data	type(MatrixType), pointer	=> null()
Iso(:,:) % Rxn(:,:) % doTranspose	logical	= .false.
Iso(:,:) % mt1(:)	integer	allocatable
Iso(:,:) % mt2(:)	integer	allocatable
ZA(:)	character(len=6)	allocatable
Read	procedure, public => ReadCovarianceData	
ReadFile	procedure, private => ReadCovarianceFile	
Write	procedure, public => WriteCovarianceData	
FindZAIndex	procedure, public => CovarianceMatrixFindZAIndex	
FindMTIndex	procedure, public => CovarianceMatrixFindMTIndex	
FindNextFreeMatrix	procedure, private => CovarianceMatrixFindNextFreeMatrix	
ConstructSUMap	procedure, public => CovarianceMatrixConstructSUMapVector	

ExpandCovarianceMatrixSensitivity Vectors	procedure, public	
ExpandCovarianceMatrixSensitivity Matrix	procedure, public	
Expand[15]	generic, public => ExpandCovarianceMatrixSensitivityVectors, & ExpandCovarianceMatrixSensitivityMatrix	

CovHeaderDataType (private)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
FileZA	character(len=6)	
NErg	integer	
NumFileZAMT	integer	
NumUsedZAMT	integer	
NumUniqueMT	integer	
UsedZAMT(:)	type(ZAMTPairType)	allocatable
UsedZAMT(:) % ZA1	character(len=6)	
UsedZAMT(:) % ZA2	character(len=6)	
UsedZAMT(:) % MT1	integer	
UsedZAMT(:) % MT2	integer	
UniqueMT(:)	integer	allocatable

[15] Component Expand selects subroutine ExpandCovarianceMatrixSensitivityVectors or ExpandCovarianceMatrixSensitivityMatrix depending on the arguments supplied.

- If the argument types are KeffSenDataType, KeffSenDataType, ExpandedCovarianceMatrixSensitivityVectorType, and (optionally) real(8), then Expand calls subroutine ExpandCovarianceMatrixSensitivityVectors.
- If the argument types are an array KeffSenDataType, ExpandedCovarianceMatrixSensitivityMatrixType, and (optionally) real(8), then Expand calls subroutine ExpandCovarianceMatrixSensitivityMatrix.

ExpandedCovarianceMatrixSensitivityMatrixType (public)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
extends(ExpandedCovarianceMatrix Type)		
Cov	type(CovarianceMatrixType), pointer	=> null()
SUMap	type(SUMapVectorType)	
SUMap % nBlocks	integer	= 0
SUMap % TotalSize	integer	= 0
SUMap % SU(:)	Type(SUMapType)	
SUMap % SU(:) % ZA	character(len=6)	
SUMap % SU(:) % Rxn	integer	
SUMap % SU(:) % MinEBin	integer	
SUMap % Cov	type(CovarianceMatrixType), pointer	
SUMap % GetSensitivityVector	procedure => SUMapGetSensitivityVector	
SUMap % GetExpandedCovarianceMatrix	procedure => SUMapGetExpandedCovarianceMatrix	
C(:,:)	real(8)	allocatable
FixUp	procedure => FixUpExpandedCovarianceMatrix	
Compress	procedure => CompressExpandedCovarianceMatrix	
S(:,:)	real(8)	allocatable

ExpandedCovarianceMatrixSensitivityVectorType (public)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
extends(ExpandedCovarianceMatrix Type)		
Cov	type(CovarianceMatrixType), pointer	=> null()
SUMap	type(SUMapVectorType)	
SUMap % nBlocks	integer	= 0
SUMap % TotalSize	integer	= 0
SUMap % SU(:)	Type(SUMapType)	
SUMap % SU(:) % ZA	character(len=6)	
SUMap % SU(:) % Rxn	integer	
SUMap % SU(:) % MinEBin	integer	
SUMap % Cov	type(CovarianceMatrixType), pointer	
SUMap % GetSensitivityVector	procedure => SUMapGetSensitivityVector	
SUMap % GetExpandedCovarianceMatrix	procedure => SUMapGetExpandedCovarianceMatrix	
C(:,:)	real(8)	allocatable
FixUp	procedure => FixUpExpandedCovarianceMatrix	
Compress	procedure => CompressExpandedCovarianceMatrix	
Sen1	type(KeffSenDataType), pointer	=> null()
Sen2	type(KeffSenDataType), pointer	=> null()
S1(:)	real(8)	allocatable
S2(:)	real(8)	allocatable

ExpandedCovarianceMatrixType (public)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
Cov	type(CovarianceMatrixType), pointer	=> null()
SUMap	type(SUMapVectorType)	
SUMap % nBlocks	integer	= 0
SUMap % TotalSize	integer	= 0
SUMap % SU(:)	Type(SUMapType)	
SUMap % SU(:) % ZA	character(len=6)	
SUMap % SU(:) % Rxn	integer	
SUMap % SU(:) % MinEBin	integer	
SUMap % Cov	type(CovarianceMatrixType), pointer	
SUMap % GetSensitivityVector	procedure => SUMapGetSensitivityVector	
SUMap % GetExpandedCovarianceMatrix	procedure => SUMapGetExpandedCovarianceMatrix	
C(:,:)	real(8)	allocatable
FixUp	procedure => FixUpExpandedCovarianceMatrix	
Compress	procedure => CompressExpandedCovarianceMatrix	

IsoListType (public)
(module IsotopeListMod)

Component	Data type	Init. Value
NIso	integer	
MaxIso	integer	
ZA(:)	character(len=20)	allocatable
Init	procedure	
AddIso	procedure	
WriteIso	procedure	
IsoExists	procedure	
FindIso	procedure	

IsoSenType
(module SensitivityMod)

Component	Data type	Init. Value
Sk(:,:)	real(8)	allocatable

IsotopeLevelCovarianceMatrixType (private)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
Rxn(:,:)	type(ReactionLevelCovarianceMatrixType)	allocatable
Rxn(:,:) % Data	type(MatrixType), pointer	=> null()
Rxn(:,:) % doTranspose	logical	= .false.
mt1(:)	integer	allocatable
mt2(:)	integer	allocatable

KeffSenDataType
(module SensitivityMod)

Component	Data type	Init. Value
extends(IsoListType)		
NIso	integer	
MaxIso	integer	
ZA(:)	character(len=20)	allocatable
Init	procedure	
AddIso	procedure	
WriteIso	procedure	
IsoExists	procedure	
FindIso	procedure	
FileName	character(len=40)	
KeffCalc	real(8)	
KeffCalcUnc	real(8)	
IsoSen(:)	type(IsoSenType)	allocatable
IsoSen(:) % Sk(:,:)	real(8)	allocatable
UnionizeKeffSen	procedure	

UnionizeKeffSenVector	procedure	
Union[16]	generic => UnionizeKeffSen, UnionizeKeffSenVector	

MatrixType (private)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
M(:,:)	real(8)	allocatable

ReactionLevelCovarianceMatrixType (private)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
Data	type(MatrixType), pointer	=> null()
doTranspose	logical	= .false.

SimilarityInfoType
(module SensitivityMod)

Component	Data type	Init. Value
NSimilar	integer	
CalcMargin	real(8)	
WeightSum	real(8)	
MaxSimilarity	real(8)	
CutoffRatio	real(8)	
ResidualWeightFrac	real(8)	
CoveragePenalty	real(8)	
Name(:)	character(len=40)	allocatable
Ck(:)	real(8)	allocatable
Wgt(:)	real(8)	allocatable

SUMapType
(module CovarianceMatrixMod)

Component	Data type	Init. Value
ZA	character(len=6)	
Rxn	integer	
MinEBin	integer	

[16] Component Union selects subroutine UnionizeKeffSen or UnionizeKeffSenVector depending on the arguments supplied.

- If there are two arguments that are type KeffSenDataType, then Union calls subroutine UnionizeKeffSen.
- If there is one or two arguments that are arrays of type KeffSenDataType, then Union calls subroutine UnionizeKeffSenVector.

SUMapVectorType (public)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
nBlocks	integer	= 0
TotalSize	integer	= 0
SU(:)	type(SUMapType)	allocatable
SU(:) % ZA	character(len=6)	
SU(:) % Rxn	integer	
SU(:) % MinEBin	integer	
Cov	type(CovarianceMatrixType), pointer	
GetSensitivityVector	procedure => SUMapGetSensitivityVector	
GetExpandedCovarianceMatrix	procedure => SUMapGetExpandedCovarianceMatrix	

ZAMTType (public)
(module CovarianceMatrixMod)

Component	Data type	Init. Value
ZA	character(len=6)	
MT	integer	

ZAMTPairType
(module CovarianceMatrixMod)

Component	Data type	Init. Value
ZA1	character(len=6)	
ZA2	character(len=6)	
MT1	integer	
MT2	integer	

Appendix E

Whisper Files/Modules and Subroutines/Functions

Appendix F

Whisper Tree Structure

The structure tree below is organized as follows:

a) The structure tree does not include any logic as to if or when a subroutine or function is called.

b) The first level subroutines, i.e., subroutines called by the main program, are in bold text.

c) Type-bound procedures are shown with a pointer to the subroutine or function name.

d) Underlining on the subroutine/function name indicates that other calls to the subroutine/function occur below but the tree for the subroutine/function will not be repeated.

e) Multiple calls are indicated after the subroutine/function name by the number of call statements in parantheses.

Program Whisper

1. **WriteHeader**
   - 1.1. GetVersion
2. **ParseCommandLine**
   - 2.1. DisplayHelp
   - 2.2. DisplayVersion
3. **CheckFiles**
   - 3.1. GetVersion
4. **ReadAndSetupUserOptions**
5. **ReadAndSetupBenchmarks**
   - 5.1. AppendForwardSlash
   - 5.2. FindBenchmarkIndex
   - 5.3. ReadKeffSenData
     - 5.3.1. ConvertZAIDtoZA
     - 5.3.2. ks % FindIso => FindIso
     - 5.3.3. ks % AddIso => AddIso
       - 5.3.3.1. This % Init => Init
6. **ReadAndSetupBenchmarkCorrelations**
7. **ReadAndSetupApplications**
   - 7.1. AppendForwardSlash
   - 7.2. ReadKeffSenData – see 5.3 above
8. **ReadAndSetupCovarianceData**
   - 8.1. UnionIsoListVector (2)
     - 8.1.1. UnionList % Init => Init
     - 8.1.2. UnionList % AddIso => AddIso

8.2. UnionIsoLists
8.2.1. UnionList % Init => Init
8.2.2. UnionList % AddIso => AddIso (2)
8.3. AppendForwardSlash
8.4. CovarianceData % Read => ReadCovarianceData
8.4.1. ReadCovarianceFileHeader
8.4.2. This % ReadFile => ReadCovarianceFile
8.4.2.1. This % FindNextFreeMatrix => CovarianceMatrixFindNextFreeMatrix
8.4.2.2. This % FindZAIndex (2) => CovarianceMatrixFindZAIndex
8.4.2.3. This % FindMTIndex (2) => CovarianceMatrixFindMTIndex
8.5. MakeThermalScatterConsistent (2)

9. EstimateUnknownBenchmarkUncertainties

9.1. CovarianceData % Expand[17] => ExpandCovarianceMatrixSensitivityVectors (2)
9.1.1. UnionList % Union[18] => UnionizeKeffSen
9.1.1.1. UnionIsoLists – see 8.2 above
9.1.1.2. kSen1 % FindIso => FindIso
9.1.1.3. kSen2 % FindIso => FindIso
9.1.2. This % ConstructSUMap => CovarianceMatrixConstructSUMapVector
9.1.2.1. This % FindZAIndex => CovarianceMatrixFindZAIndex
9.1.2.2. This % FindMTIndex => CovarianceMatrixFindMTIndex
9.1.3. EC % SUMap % GetSensitivityVector => SUMapGetSensitivityVector (2)
9.1.3.1. Sen % FindIso => FindIso
9.1.4. EC % SUMap % GetExpandedCovarianceMatrix => SUMapGetExpandedCovarianceMatrix (2)
9.1.4.1. This % Cov % FindZAIndex => CovarianceMatrixFindZAIndex (2)
9.1.4.2. This % Cov % FindMTIndex => CovarianceMatrixFindMTIndex (2)

[17] Component Expand selects subroutine ExpandCovarianceMatrixSensitivityVectors or ExpandCovarianceMatrixSensitivityMatrix depending on the arguments supplied.

- If the first argument types are KeffSenDataType, KeffSenDataType, and ExpandedCovarianceMatrixSensitivityVectorType, then Expand calls subroutine ExpandCovarianceMatrixSensitivityVectors.
- If the first argument types are an array KeffSenDataType, and ExpandedCovarianceMatrixSensitivityMatrixType, then Expand calls subroutine ExpandCovarianceMatrixSensitivityMatrix.

[18] Component Union selects subroutine UnionizeKeffSen or UnionizeKeffSenVector depending on the arguments supplied.

- If there are two arguments that are type KeffSenDataType, then Union calls subroutine UnionizeKeffSen.
- If there is one or two arguments that are arrays of type KeffSenDataType, then Union calls subroutine UnionizeKeffSenVector.

10. RejectBenchmarks

- 10.1. GetDiscrepancyVector (2)
- 10.2. GetDiscrepancyCovarianceMatrix (2)
  - 10.2.1. CovarianceData % Expand[17] => ExpandCovarianceMatrixSensitivityMatrix
    - 10.2.1.1. UnionList % Union[18] => UnionizeKeffSenVector
      - 10.2.1.1.1. UnionIsoListVector (3) – see 8.1 above
      - 10.2.1.1.2. UnionIsoLists – see 8.2 above
      - 10.2.1.1.3. kSen1(n) % FindIso => FindIso
      - 10.2.1.1.4. kSen2(n) % FindIso => FindIso
    - 10.2.1.2. This % ConstructSUMap => CovarianceMatrixConstructSUMapVector – see 9.1.2 above
    - 10.2.1.3. EC % SUMap % GetSensitivityVector => SUMapGetSensitivityVector – see 9.1.3 above
    - 10.2.1.4. EC % SUMap % GetExpandedCovarianceMatrix => SUMapGetExpandedCovarianceMatrix (2) – see 9.1.4 above
  - 10.2.2. GetCovKeffMeasurement
- 10.3. InvertMatrix (2)
  - 10.3.1. dgefa
    - 10.3.1.1. idamax
    - 10.3.1.2. dscal
    - 10.3.1.3. daxpy
  - 10.3.2. dgedi
    - 10.3.2.1. dscal
    - 10.3.2.2. daxpy (2)
    - 10.3.2.3. dswap
- 10.4. RemoveBenchmarkCorrel

11. AdjustNuclearData

- 11.1. UnionIsoListVector (2) – see 8.1 above
- 11.2. UnionIsoLists – see 8.2 above
- 11.3. AdjustedCovarianceData % Read => ReadCovarianceData – see 8.4 above
- 11.4. CovarianceData % Expand => ExpandCovarianceMatrixSensitivityMatrix – see 10.2.1 above
- 11.5. GetCovKeffMeasurement
- 11.6. InvertMatrix – see 10.3 above
- 11.7. ExpCov % Compress => CompressExpandedCovarianceMatrix
  - 11.7.1. Cov % FindZAIndex => CovarianceMatrixFindZAIndex
- 11.8. AdjustedCovarianceData % Write => WriteCovarianceData

12. CalculateNuclearDataUncertainties

- 12.1. CovarianceData % Expand[17] => ExpandCovarianceMatrixSensitivityVectors – see 9.1 above
- 12.2. AdjustedData % Expand[17] => ExpandCovarianceMatrixSensitivityVectors – see 9.1 above

13. CalculateUpperSubcriticalLimits

13.1. CalculateCalculationalMargin (2)

13.1.1. CalculateSimilarityWeights

13.1.1.1. CovarianceData % Expand[17] =>
ExpandCovarianceMatrixSensitivityVectors (2) – see 9.1 above

13.1.2. GetBenchmarkBiasAndUncertaintyVectors

13.1.3. ExtremeNormalCDF

13.1.3.1. NormalCDF

13.1.4. ExtremeNormalMean

13.1.4.1. ExtremeNormalMoment

13.1.4.1.1. ExtremeNormalCDF (2)

13.1.4.1.1.1. NormalCDF

13.1.4.1.2. ExtremeNormalPDF (2)

13.1.4.1.2.1. NormalCDF

13.1.4.1.2.2. NormalPDF

14. WriteUSLOutputSummaryTable

End of Whisper Tree

Appendix G

Unused Whisper Source Code Routines

The following Whisper source code routines are not executed in the Whisper program logic. The routines may possibly be used in definitions of type-bound procedures.

	Routine*	Module
1.	subroutine FixUpExpandedCovarianceMatrix	CovarianceMatrixMod
2.	subroutine WriteIso	IsotopeListMod
3.	logical function IsoExists	IsotopeListMod
4.	subroutine IntersectIsoLists	IsotopeListMod
5.	subroutine AssignKeffSenDataType	SensitivityMod
6.	ExtremeNormalVariance	StatsMod

Appendix H

Covariance File Formats

The Whisper program suite has three types of covariance data: native, original, and adjusted.

1. The native data file (ORNL_SCALE6.1.cov) is a COVERX file. The format of COVERX files is documented in Table M18.A.8 of ORNL/TM-2005/39. Each data block has a flag or prefix. For example, the neutron group energy boundaries are prefixed with '3D' and each covariance matrix is prefixed with '9D'.

2. The original data files are extracts from the native file. The extracted covariance data is identical to the native data. The original data files do not use the block flags. The original data files also separate the different isotopes into separate files (rather than a single covariance data file).

3. The adjusted covariance files have covariance values calculated by Whisper but use the same format as the original data files. In some cases, the adjusted covariance file is identical to the original covariance data file because Whisper does not calculate any adjusted covariance values for the isotope.

Energy Structure

In the covariance data files, the energy structure is high to low. This is illustrated by the '3D' block in the native covariance data file 'ORNL_SCALE6.1.cov' [ORNL/TM-2005/39; Table M18.A.8], which is listed below.

```
3d   2.0000E+07  8.1873E+06  6.4340E+06  4.8000E+06  3.0000E+06
  2.4790E+06  2.3540E+06  1.8500E+06  1.4000E+06  9.0000E+05  4.0000E+05
  1.0000E+05  2.5000E+04  1.7000E+04  3.0000E+03  5.5000E+02  1.0000E+02
  3.0000E+01  1.0000E+01  8.1000E+00  6.0000E+00  4.7500E+00  3.0000E+00
  1.7700E+00  1.0000E+00  6.2500E-01  4.0000E-01  3.7500E-01  3.5000E-01
  3.2500E-01  2.7500E-01  2.5000E-01  2.2500E-01  2.0000E-01  1.5000E-01
  1.0000E-01  7.0000E-02  5.0000E-02  4.0000E-02  3.0000E-02  2.5300E-02
  1.0000E-02  7.5000E-03  3.0000E-03  1.0000E-05
```

The energy bins in Whisper are identical but are organized from low to high (see Sec. 5.2).

Matrix Structure

Table M18.A.8 of ORNL/TM-2005/39 identifies that the matrix data in the COVERX file is prefixed with '9D', but refers to the matrix data as a one-dimensional array, COV(K). However, the covariance data matrix follows the row, column standard with the matrix written/read column by column. This is illustrated by the following read statement from SCALE 6.1 subroutine print_matrix where the row index is the inner loop and the column index is the outer loop:

```
read(77, iostat=io) ((cov(i,j), i=1+j-ijj(j),jband(j)+j-ijj(j)),j=1,ngroup)
```