Title: User Manual for Whisper (v1.0.0), Software for Sensitivity- and Uncertainty-Based Nuclear Criticality Safety Validation

Author(s): Kiedrowski, Brian Christopher

Intended for: Report

Issued: 2014-08-13

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

Brian C. Kiedrowski

1 Introduction & Overview

Whisper is computational software designed to assist the nuclear criticality safety (NCS) analyst with validation studies with the Monte Carlo radiation transport package MCNP. Standard approaches to validation rely on the selection of benchmarks based upon expert judgment. Whisper uses sensitivity/uncertainty (S/U) methods to select relevant benchmarks to a particular application or area of applicability (AOA), or set of applications being analyzed. Using these benchmarks, Whisper computes a calculational margin from an extreme value distribution. In NCS, a margin of subcriticality (MOS) that accounts for unknowns about the analysis. Typically, this MOS is some prescribed number by institutional requirements and/or derived from expert judgment, encompassing many aspects of criticality safety. Whisper will attempt to quantify the margin from two sources of potential unknowns, errors in the software and uncertainties in nuclear data. The Whisper-derived calculational margin and MOS may be used to set a baseline upper subcritical limit (USL) for a particular AOA, and additional margin may be applied by the NCS analyst as appropriate to ensure subcriticality for a specific application in the AOA.

Whisper provides a benchmark library containing 1,086 MCNP input files spanning a large set of fissionable isotopes, forms (metal, oxide, solution), geometries, spectral characteristics, etc. Along with the benchmark library are scripts that may be used to add new benchmarks to the set; this documentation provides instructions for doing so. If the user desires, Whisper may analyze benchmarks using a generalized linear least squares (GLLS) fitting based on nuclear data covariances and identify those of lower quality. These may, at the discretion of the NCS analyst and their institution, be excluded from the validation to prevent contamination of potentially low quality data. Whisper provides a set of recommended benchmarks to be optionally excluded, bringing the total benchmarks to 972.

Whisper also provides two sets of 44-group covariance data. The first set is the same data that is distributed with SCALE 6.1 in a format that Whisper can parse. The second set is an adjusted nuclear data library based upon a GLLS fitting of the 972 benchmarks following rejection. Whisper uses the latter to quantify the effect of nuclear data uncertainties within the MOS. Whisper also has the option to perform a nuclear covariance data adjustment to produce a custom adjusted covariance library for a different set of benchmarks.

This document gives an overview of the use of the Whisper software. Details of the theory and methodology may be obtained from the companion theory document, which is a manuscript submitted to Nuclear Science & Engineering, a journal of the American Nuclear Society. This Manual begins with a section (Sec. 2) on installing the Whisper software and demonstrating its basic use. The following three sections discuss different capabilities of Whisper: calculating baseline USLs for applications (Sec. 3), analyzing and rejecting benchmarks (Sec. 4), and performing nuclear covariance data adjustment (Sec. 5). The final section discusses advanced options (Sec. 6).

2 Installation & Getting Started

This section gives instructions for installing the Whisper software and running scripts distributed as part of Whisper to compute baseline USLs. It is likely the instructions in this section will be sufficient for most users.

2.1 Overview of Whisper Directory Structure

Whisper has a hierarchically-structured set of directories that are organized by function. Each of these is discussed at a high level:

- `Benchmarks` – This directory contains information about the benchmark suite and serves as source material to get started with Whisper. Most users will likely be content with the benchmark suite as is, but it may be optionally added to (see Sec. 6. Within this directory: the MCNP input files are provided in the `Inputs` subdirectory, the respective MCNP sensitivity profiles on the SCALE 6.1 energy grid are in the `Sensitivities` directory, a file containing the available benchmark experimental correlation data is in the `Correlations`, a file containing a listing of benchmarks that the user should consider excluding based upon a GLLS nuclear data adjustment is in the `Exclude` directory, and the table of contents file that is set up during the installation is in the `TOC` directory.

- `bin` – Contains executables and shell scripts for ease of use in running Whisper. The installation instructions will put this directory as part of the system path so that the executables/scripts in here may be executed as Linux commands. The items are: `whisper`, the executable for the Whisper source code; `whimcnp`, a csh script that runs MCNP input files for use with Whisper; and `ww`, the "Whisper Wrapper", which takes the results of the `whimcnp` script and runs Whisper in the mode to calculate baseline USLs, where the MCNP runs are the applications being analyzed.

- `CovarianceData` – This is the directory containing the covariance data. There is a subdirectory called `SCALE6.1` that contains the default covariance data, which is shipped with SCALE 6.1. Within this are subdirectories called: `Native`, which is file distributed with SCALE6.1; `Data`, which is that data processed into a format for Whisper; and `Adjusted`, which is the covariance library following a GLLS nuclear covariance data adjustment using the benchmarks (the outliers specified in the `Benchmarks/Exclude` subdirectory were excluded) provided with Whisper.

- `Documentation` – This contains the Whisper Manual (this document) and the document detailing the theory and methods.

- `Install` – This directory contains compressed data and scripts for the installation. Users should not modify this directory.

- `Source` – Contains the Whisper Fortran 2003/2008 source code. A Makefile is also provided for building the code with the Intel Fortran compiler. Compiling the code here overwrites the executable in the `bin` directory.

- `Testing` – Contains installation tests for Whisper. New installations should run these tests to ensure that everything is set up correctly.
- `Utilities` – Contains scripts and data for running and maintaining Whisper that are discussed throughout this documentation.

2.2 Installation

The top-level of Whisper contains a Makefile. The first step on a fresh install is to type `make install` on the command line (GNU-Make must be installed on the system). This Makefile will run a setup script in the Install directory that checks the directories, sets the table of contents file path for the initial suite of benchmarks, and provides a set of commands that the user should put in their `.cshrc` file in their home directory (the user should then type `source $HOME/.cshrc` to ensure that the current shell has everything set up). These commands set specific environment variables that Whisper uses as defaults during its execution; these may be specified explicitly or overridden on the command line to execute Whisper.

Once this is complete, enter the `Testing` directory and type `make install`. This runs the installation tests, which typically takes about 10 minutes. The tests are:

1. The first test checks the ability of Whisper to compute baseline USLs. The test uses the benchmark suite provided with Whisper and excludes a majority of them. The benchmark information is obtained from the `WHISPER_BENCHMARK_TOC` environment variable, which tests this part of the installation. This test then computes the baseline USLs for a set of applications provided in the test. The output file `Test1.out` is compared with a reference file.
2. The second test checks the benchmark rejection ability of Whisper. A list of benchmarks is provided by the test; a user options file is also provided that changes the cutoff χ^2 value for acceptance from the default of 1.2 to 1.6. The test uses the iterative diagonal method to reject a series of benchmarks until the χ^2 of the suite falls below 1.6. The test compares the output file `Test2.out` and the list of rejected benchmarks in `Reject2.out`.
3. The third test checks the GLLS nuclear covariance data adjustment capability of Whisper. A list of benchmarks is provided by the test along with a set of benchmark correlations. The GLLS adjustment produces a new set of adjusted covariance data. The output file `Test3.out` adjusted covariance file for ^{235}U are compared with the reference files.

If everything has been copied and installed correctly, the message indicating the all installation tests having passed will be displayed. If failures occur, there are difference files `Diffx.y.dat` (here `x` is the test index and `y` is a file number within that test) that provide differences from reference files.

Note that if the benchmark suite distributed with Whisper is modified, these tests may not pass. In this case, advanced users may wish to replace the template files once they have checked that the results are otherwise correct so that this may be used as a check on any other potential changes to the system. A description of how to do this is given in Sec. 6.

Uninstalling Whisper may be done by typing `make uninstall` in the top-level Whisper directory. The user will be warned that this operation cannot be undone and advised to make a backup. Before proceeding, the string `confirm` must be typed. If the user does so, then the Whisper directory will remove the benchmark information, covariance data, source code, and installation tests, which may be modified to meet end-user needs. This makes the directory ready for distribution. Typing `make install` will restore the aforementioned files to their default state. Finally, typing `make reinstall` does an uninstall immediately followed by an installation.

2.3 Tutorial for Running Whisper

Next, this document will explain how to run Whisper using the scripts in the `bin` directory.

Leave the Whisper directory tree completely and create a new directory to serve as a personal workspace. The user must now provide a few MCNP input decks for criticality problems. These will serve as input files for applications being analyzed. The input decks must contain the `kcode` card and may not contain `prdmp`, `kopts`, or `ksen` cards.

Once these input files are available, the `whimcnp` script will be used to run these files with MCNP. Type:

```
whimcnp inp1 [inp2] ...,
```

where `inp1` and `inp2` are the names of the MCNP input files. A directory called `Calcs` is created, and modified copies of the MCNP input files are placed there. Jobs are then submitted to the cluster that run MCNP with the modified input files. Note that the file `MCNPInputList.toc` has been created. This file contains the listing of inputs that will be used by `ww`, the Whisper Wrapper. Also, a directory called `KeffSenLib` has been created. For now, leave these alone.

Before leaving `whimcnp`, note that the default wall-clock time is one hour. If the user wishes to change this, the user may type `walltime hh:mm:ss` as the first two arguments of `whimcnp`. Note that the entire `hh:mm:ss` must be specified, so to request 30 minutes, the command would be `walltime 00:30:00`. For completeness, the syntax for `whimcnp` is

```
whimcnp [walltime hh:mm:ss] inp1 [inp2] ...
```

Once these jobs have finished running, in the same directory, type `ww` to run Whisper. Note that the `ww` script also has an optional command line argument for a file containing a list of benchmarks (one per line) to be excluded from the validation. Such a feature is useful to leave out irrelevant benchmarks for the validation, e.g., exclude uranium systems for a plutonium validation, to save time. For completeness, the syntax is

```
ww [ExcludeFile.dat],
```

This Whisper Wrapper will read the file `MCNPInputList.toc` and parse the sensitivity profiles. The sensitivity profiles are placed into the `KeffSenLib` and a new file called `KeffSenList.toc` is created, which is a table of contents file run by Whisper. The contents of this file are discussed in Sec. 3.

The Whisper Wrapper than runs Whisper itself using the applications provided by the user and the benchmark files distributed with Whisper. Results are printed out to the screen and to the output file `Whisper.out`.

After Whisper finishes running, baseline USLs should be displayed. This now completes the tutorial.

2.4 Whisper Command Arguments

It is also possible to run Whisper directly. The first argument is `-h`, which displays the user help information, giving a list of command line arguments. The user help can be seen by the command

```
whisper -h
```

Also useful is to have Whisper display the version information (this is also written to the screen and to the output file upon execution). This is done with the `-v` argument, as

```
whisper -v
```

To illustrate a simple example, consider the case of running Whisper directly on the application sensitivity table of contents file `KeffSenList.toc` produced by the `ww` script. The syntax is

```
whisper -a KeffSenList.toc
```

Here the `-a` denotes that the next argument is the name of a file containing application information about the calculated k and its statistical uncertainty, which is discussed in Sec. 3.

In addition, a file containing a list of benchmarks to exclude may be specified on the command line as well (this again, is the optional argument of the `ww` script). This file simply lists the name of each benchmark to exclude with one per line. If the file is called `Exclude.dat`, then the syntax would be

```
whisper -a KeffSenList.toc -x Exclude.dat
```

The other option of most use is modifying the name of the output file, which has a default name of `Whisper.out`. This may be changed via the `-o` option,

```
whisper -o MyOutputFile.dat
```

Note that Whisper overwrites any output file whether it is specified on the command line or not.

The entire list of command line arguments and a brief description of their use is given in Table I.

Table I: Whisper Command Line Arguments

`-a`	Followed by an application table of contents file containing a file path header and a list of names of application sensitivity profiles. The presence of this argument causes Whisper to calculate baseline USLs for the applications listed in the file. This is described in detail in Sec. 3.
`-b`	Followed by an benchmark table of contents file containing a file path header and a list of names of benchmark sensitivity profiles. The presence of this argument overrides the benchmark table of contents file given by the environment variable `WHISPER_BENCHMARK_TOC`. Either this argument be specified or the environment variable `WHISPER_BENCHMARK_TOC` set to a valid file for Whisper to run. This is described in detail in Sec. 3.
`-c`	Followed by a path to covariance data. The presence of this argument overrides the path specified by the environment variable `WHISPER_COVDATA_PATH`. This is described in detail in Secs. 3 and 5.
`-d`	Followed by a path to a directory (must already exist) where adjusted covariance data will be placed after a GLLS nuclear covariance data adjustment. The presence of this argument activates the GLLS nuclear covariance data adjustment. This is described in detail in Sec. 5.
`-h`	Followed by no argument. Displays help screen listing command line arguments.
`-k`	Followed by a file listing experimental correlations of benchmark used in GLLS benchmark rejection or nuclear covariance data adjustment, and is strongly recommended when using Whisper for either of those purposes. The file and how it is used is described in detail in Secs. 4 and 5.
`-o`	Followed by the name of the output file. If this is not specified, the default file name `Whisper.out` is used. Note that Whisper overwrites output files regardless of whether this argument is given.
`-r`	Followed by a the name of a new file that will list benchmarks that have been rejected by the GLLS rejection. The presence of this argument activates the GLLS benchmark rejection. This is described in detail in Sec. 4.
`-t`	Followed by an integer listing the number of OpenMP threads for parallel computing. The default is 16.
`-u`	Followed by a file listing user options. This is an advanced feature described in Sec. 6.
`-v`	Followed by no argument. Displays Whisper version information, copyright, and authorship.
`-x`	Followed by a file listing benchmarks to be excluded from the validation. This is described in detail in Sec. 3.

3 Estimation of Baseline Upper Subcritical Limits

The primary use of Whisper is to estimate baseline USLs. Reading this section and the two previous will probably satisfy most users of the software. This section discusses the two steps for calculating baseline USLs: running MCNP input files specifically set up for Whisper and running Whisper itself with the results of those MCNP calculations. The Whisper baseline USL calculation and output are then discussed. Finally, this section gives information on the sensitivity and covariance data libraries.

3.1 Running MCNP for Whisper

The first step is to run MCNP for Whisper. This could, in principle, be done manually, but it is best to use either the `whimcnp` script in the `bin` directory or the `RunMCNPInputs.csh` script in the `Utilities` directory. The `whimcnp` script (see Sec. 2) actually calls `RunMCNPInputs.csh` with specific command line arguments, and is slightly easier to use but less flexible. The `RunMCNPInputs.csh` has the following command arguments:

```
./RunMCNPInputs.csh [walltime hh:mm:ss] MCNPInput.toc inp1 [inp2] ...
```

Here `walltime hh:mm:ss` is an optional argument to change the wall-clock submission time from the default of one hour, which is identical to the syntax for `whimcnp`. The argument `MCNPInput.toc` is the specified name of an intermediate file listing the MCNP inputs and any benchmark k values and uncertainties that are provided in the file. When using `whimcnp`, the name of this is hard coded to `MCNPInputList.toc`, and `whimcnp` will not run if this is present to prevent accidentally overwriting it; conversely, `RunMCNPInputs.csh` overwrites any such file specified.

The benchmark k and uncertainty are provided by a comment at the bottom of the MCNP input file that is

```
c k(bmk) = [keff] +- [unc]
```

where `keff` and `unc` get replaced with the benchmark values of k and the uncertainty. If this string is absent, which is desirable in the case of applications, then `RunMCNPInputs.csh` writes zeroes to the intermediate file `MCNPInput.toc`.

Upon writing this file `RunMCNPInputs.csh`, new inputs are written with the `kcode` line replaced, a `prdmp` card added to suppress dumping to `runtpe` files, and the sensitivity profiles inserted with the `kopts` and `ksen` cards. The energy grid is obtained from a file called `SCALE44ErgGrid.dat`, which must be available either locally (takes precedence) or in the directory `Whisper/Utilities/Data`. Once this has been done, the script submits jobs to the cluster for each modified MCNP input. The user must then wait until all jobs have finished.

3.2 Running Whisper for Calculating Baseline USLs

Upon completion of the jobs, the user may run the Whisper wrapper `ww` or run `MakeKeffSenLib.csh` and `whisper` separately. The former is easier to use, whereas the latter offers greater flexibility. This Whisper Wrapper `ww` calls `MakeKeffSenLib.csh` with fixed arguments. The arguments for `MakeKeffSenLib.csh` are

```
./MakeKeffSenLib.csh MCNPInput.toc KeffLib.toc [run] [Exclude.dat]
```

The arguments `MCNPInput.toc` are the name of the MCNP input list file that was given to `RunMCNPInputs.csh` and `KeffLib.toc` is the name of the sensitivity library that will be used by Whisper. When running `ww`, these file names are fixed to be `MCNPInputList.toc` and `KeffSenList.toc` respectively. The argument `run` is optional, which has the script run `whisper` with the application file of `KeffLib.toc` and the benchmarks being obtained from the `WHISPER_BENCHMARK_TOC` environment variable. The fourth optional argument, `Exclude.dat` is a file that lists benchmarks (one per line) to exclude from the validation when running Whisper. This is also an optional argument on the `ww` command.

To illustrate, if `whimcnp` has been run on a set of inputs previously, the command

```
ww Exclude.dat
```

is equivalent to

```
./MakeKeffSenLib.csh MCNPInputList.toc KeffSenList.toc run Exclude.dat
```

Once the file `KeffSenList.toc` (or any equivalent sensitivity profile table of contents file with a different name) is given, it can run Whisper directly as opposed to though `ww`. Again, the advantage of using `ww` is that it is easier to use for this purpose, but does not have as much flexibility. If the sensitivity profile table of contents file is `KeffSenList.toc`, then the Whisper command to run the calculation without excluding any benchmarks is

```
whisper -a KeffSenList.toc
```

The benchmark sensitivity profile table of contents file is obtained from the environment variable `WHISPER_BENCHMARK_TOC`. Suppose a different benchmark table of contents file called `Benchmark.toc` is to be used and is in the current directory. The Whisper command is then

```
whisper -a KeffSenList.toc -b Benchmark.toc
```

Additionally, if a set of benchmarks should be excluded and are listed in a file called `Exclude.dat`, then the Whisper command is

```
whisper -a KeffSenList.toc -b Benchmark.toc -x Exclude.dat
```

Recall that results are written to an output file with a default name of `Whisper.out`. The name can be changed with the `-o` option when using `whisper`; there is no way to do this with `ww`. Note that Whisper always overwrites the output file regardless of whether it is specified on the command line or not or whether Whisper is run directly or through its wrapper `ww`. The contents of this output file along with the calculational flow is discussed next.

3.3 Baseline USL Calculation and Output

Upon running Whisper, the software reads the benchmark and application sensitivities and the covariance data (base data and adjusted data that has been precomputed based upon the benchmark suite) to compute similarity coefficients and weighting factors for the benchmarks. Any benchmarks that do not have a known uncertainty associated with them is assigned a value that is from a variance that is a similarity coefficient weighted average of the benchmarks with quantified uncertainties. Note that this has already been done for the benchmarks distributed with Whisper.

The benchmark and application data that was read is listed to the output file with any exclusions noted. Which covariance data files that were read are listed in the output too. Any estimated benchmark uncertainties are also listed.

Once all data has been read in and any unknown uncertainties estimated, the nuclear data uncertainty based upon the adjusted nuclear data library is estimated. For each application the nuclear data uncertainties from the adjusted covariance data and the covariance data prior to adjustment is listed in the Whisper output. This is used to help determine the additional margin of subcriticality to help account for other unknowns about the simulation process. An additional term of 0.005 is added to this margin of subcriticality to account for undetected errors in the transport software.

Once this margin of subcriticality has been developed for all the benchmarks, Whisper computes the calculational margin for each benchmark. The sensitivity profiles and the covariance data are used to compute weighting factors for each benchmark with respect to each application, so that for each application all benchmarks are weighted by their degree of relevance to the application. Based on these weighting factors, a calculational margin is estimated from an extreme value distribution, which represents the value to some degree of confidence that bounds the worst case bias and bias uncertainty of the weighted population of benchmarks.

For each application the Whisper output file lists the calculational margin, the 1-σ nuclear data uncertainty, the derived baseline USL, and the amount k exceeds the USL, which if negative, can be assured to be subcritical absent any additional margin to be added by the analyst. The output file also lists, for each application, the number of benchmarks used for validating that application, the maximum degree of similarity, the bias and bias uncertainty, and any non-coverage penalty that is applied in cases where the benchmark suite is inadequately populated for the specific application being analyzed. The benchmarks that were used and their respective similarity coefficients and weight factors are then listed.

At the bottom of the output file, a summary table is printed that gives the calculational margin, the 1-σ nuclear data uncertainty, the derived baseline USL, and the amount k exceeds the USL. This exact same information is printed out to the screen during the calculation.

3.4 Sensitivity Profile Table of Contents File

Whisper requires that both the benchmark and application sensitivity files have table of contents file. The benchmark table of contents file is either specified after the `-b` option or obtained from the `WHISPER_BENCHMARK_TOC` environment variable. The command line option takes precedence over the environment variable. The Whisper Wrapper script `ww` assumes the environment variable is set.

The sensitivity profile table of contents file of the benchmarks and the applications have the same format. The first line is a file path where Whisper gets the sensitivity profiles. The subsequent lines consist of the name of a benchmark input file, the benchmark k (this is normally zero for an application and is ignored if it is not), the benchmark uncertainty in k (again, normally zero for an application and is ignored if it is not), the calculated k, and the calculated uncertainty in k.

Whisper reads the first line to determine the file path. Following this, each line is read along with its benchmark information. Upon reading each line, if the benchmark is not excluded, Whisper reads in the sensitivity profile for that file (which has the same name as the input with the letter `k` appended to it). This continues until all benchmarks are read. The exact same process is then repeated for the applications.

The nuclear data sensitivity profiles that are generated are for the following data: elastic scattering, inelastic scattering, fission, (n,2n), (n,γ), (n,p), (n,d), (n,t), (n,^{3}He), (n,α), fission ν, and fission χ. All other reactions are going to be minor to criticality safety and are ignored.

3.5 Nuclear Covariance Data

Whisper requires nuclear covariance data to compute the baseline USLs in the estimation of the similarity coefficients for the weighting factors and as part of the additional margin of subcriticality. The covariance data distributed with Whisper was obtained from SCALE6.1 in 44-group format, which was the most comprehensively available processed set available at the time.

The covariance data is nominally located in `Whisper/CovarianceData`. Within the `CovarianceData` subdirectory are three other subdirectories: `Native`, `Data`, and `Adjusted`. The `Native` subdirectory contains the original, unprocessed file distributed with SCALE6.1. Whisper has utility scripts to process this. The `Data` subdirectory contains the processed covariance data from the file in the `Native` subdirectory representing the best estimated differential uncertainties and correlations of the nuclear data. These files are used in the computation of the similarity coefficients. The `Adjusted` subdirectory contains the adjusted covariance data that was generated with a GLLS nuclear covariance data adjustment using Whisper with the benchmark suite provided with the recommended set rejected.

During runtime, Whisper normally obtains the covariance data from the environment variable `WHISPER_COVDATA_PATH`. This can be overriden with the `-c` option followed by the path to new directory. To be valid, this directory must have subdirectories named `Data` and `Adjusted` that contains the respective covariance data libraries.

4 Benchmark Rejection

Whisper may also be used to identify benchmark descriptions that may be of lower than typical quality. The assumption made by the methodology is that the primary source of bias arises from the nuclear data uncertainties; therefore, if this is strictly true, the discrepancy for a set of benchmarks should be possible to eliminate for a consistent adjustment of nuclear data within the constraints of the nuclear covariance data.

The rejection is done by a GLLS nuclear data adjustment. The χ^2 statistic is calculated following the adjustment. If the computed χ^2 is greater than a given threshold, the default being 1.2, then the benchmark set is accepted. If it is not, this indicates an inconsistency with the stated assumption, and the iterative diagonal method is used to identify the benchmark that is likely contributing most to this inconsistency. That benchmark is rejected and the χ^2 statistic is computed on the new set. Benchmarks are rejected one by one until the threshold χ^2 is met.

In using the GLLS method, it is advisable to consider the experimental correlations of the benchmarks. This section describes what is available with Whisper and how to use them. Then the section gives the mechanics of performing a benchmark rejection and how it may be integrated into a future validation.

4.1 Benchmark Experimental Correlation Data

Whisper provides experimental correlation data where it is available. These correlations were lifted from the ICSBEP Handbook in 2014, and more are likely to become available in the future. The impact of correlations is that it further constrains the adjustment leading to a more realistic rejection, assuming the provided correlation coefficients are representative.

The benchmark correlations distributed with Whisper are in the directory `Benchmark/Correlations` in the file `BenchCorrel.dat`. The file lists one correlation per line. Each line has the names of two benchmark input files followed by a correlation coefficient. Whisper may read this file by employing the `-k` command line argument followed by the name of the experimental correlation file.

4.2 Performing a Benchmark Rejection

Whisper's GLLS benchmark rejection is activated with the `-r` command line argument followed by a name of a file to write the names of the rejected benchmarks. Benchmark information is obtained by having the sensitivity table of contents file specified on the command line followed by the `-b` argument or from the `WHISPER_BENCHMARK_TOC` environment variable. Unadjusted covariance data is used and the path must either be specified following the `-c` argument or obtianed from the `WHISPER_COVDATA_PATH` environment variable.

Applications, if specified, do not influence the benchmark rejection. If applications are specified, Whisper will calculate the baseline USLs as described in Sec. 3 using the benchmark specified suite excluding those identified in the GLLS rejection. Benchmarks may also be excluded prior to the rejection by specifying a file listing those benchmarks followed by the `-x` argument. The excluded benchmarks will be written to the benchmark rejection file prior to performing the GLLS rejection.

The most basic command for having Whisper do a benchmark rejection (assuming benchmarks and covariance data have been specified by the `WHISPER_BENCHMARK_TOC` and `WHISPER_COVDATA_PATH` environment variables respectively) is

```
whisper -r RejectedBenchmarks.dat
```

This command will read the benchmarks and covariance data specified by the environment variables and then perform the GLLS rejection until the threshold χ^2 is met. The rejected benchmarks will be listed in the new file `RejectedBenchmarks.dat` and the output file. Like with the output file, the file listing the rejected benchmarks is always overwritten.

It is strongly recommended to apply known benchmark correlations. As discussed previously, a file containing a list is provided with Whisper and this file or another of the user's preference may be specified following the `-k` argument. Suppose the user provides their own correlation file called `Correlations.dat`, then the execution command for using these correlations is

```
whisper -r RejectedBenchmarks.dat -k Correlations.dat
```

This argument will cause Whisper to consider the experimental correlations defined in the file `Correlations.dat` during the GLLS nuclear data adjustment as part of the rejection.

The threshold χ^2 value is 1.2 by default but may be changed via a user options file. To do this, first create a text file with a single line of text,

```
ThresholdChiSquare value
```

where `value` is some number greater than one. Whisper will read this user options file if it is specified after the `-u` argument. For example, if the file is called `Options.dat`, then the command is

```
whisper -r RejectedBenchmarks.dat -k Correlations.dat -u Options.dat
```

Covariance that is not available is, by default, given an uncertainty along the diagonal, i.e., no correlation assumed, of 0.1 or 10%. This represents a crude approximation, but is more realistic than just assuming there is no uncertainty. This value is used during GLLS nuclear data adjustments for benchmark rejections or nuclear covariance data adjustments. The value of 0.1 may be changed by the user option `UnknownDataUncertainty`.

The user options file is discussed in greater detail in Sec. 6, which covers advanced topics.

The resulting rejected benchmark list file, in the example called `RejectedBenchmarks.dat`, is in the exact same format as a file is for excluding benchmarks in a future calculation, which is specified by the `-x` option. Since the process of performing a benchmark rejection may be very time consuming, it is recommended that users do this once and store the list of benchmarks to be excluded in future calculations.

5 Nuclear Covariance Data Adjustment

An adjusted nuclear covariance data library is used in the determination of the margin of subcriticality because of nuclear data uncertainty/variability. The adjusted covariance data is used because it represents a more realistic assessment of the statistical uncertainties. The prior covariance data considers only the uncertainties because of differential measurements and not the critical experiments. In practice, nuclear data libraries are developed with the critical experiments in mind, so the true uncertainty is lower than would indicated by the differential uncertainties alone. This is evidenced by the fact that a 1-σ uncertainty appears to bound the bias rather than the expected rules for normally distributed uncertainties. Note that the similarity coefficients use the prior uncertainties rather than the adjusted ones because the data adjustment tends to introduce unphysical correlations between nuclear data and therefore dissimilar benchmarks may become correlated in unusual ways even if the overall uncertainty computed by the adjusted covariance data is more representative of the actual uncertainty.

Whisper considers a population of benchmarks and uses the GLLS to adjust the nuclear data. Based on this adjustment of the nuclear data, their covariances are also adjusted. Whisper generates a new, adjusted nuclear covariance data library that may be used with future calculations. This section describes how this can be done for a given set of benchmarks followed by a discussion of the covariance data format.

5.1 Adjusting Nuclear Covariance Data with Whisper

To perform a nuclear covariance data adjustment, benchmark sensitivities and covariance data must be available. As discussed in previous sections, the benchmark sensitivity table of contents must either be specified following the `-b` command line argument or from the `WHISPER_BENCHMARK_TOC` environment variable, and the prior covariance data path must either be specified following the `-c` command line argument or from the `WHISPER_COVDATA_PATH` environment variable. As with the GLLS nuclear data adjustment for benchmark rejection, using experimental benchmark correlations is also strongly recommended; the file is benchmark correlation file specified following the `-k` argument and is further discussed in Sec. 4. Finally, the user must create a new directory for Whisper to store the adjusted covariance library. This directory is given to Whisper following the `-d` command line argument; the presence of this activates the nuclear covariance data adjustment.

If the benchmark and prior covariance data are specified with the appropriate environment variables described previously, the benchmark correlation file is called `Correlations.dat`, and the new covariance library directory is called `NewAdjustedData` then the command is

```
whisper -k Correlations.dat -d NewAdjustedData
```

Whisper may also perform the nuclear covariance data adjustment in conjunction with either or both of the options for computing baseline USLs and rejecting benchmarks. The benchmark rejection is performed prior to the nuclear covariance data adjustment and the rejected benchmarks are excluded during the adjustment. The baseline USLs are calculated after the nuclear covariance data adjustment and use the new adjusted covariance data library produced by the current run instead of the `Adjusted` directory from the path given after the `-c` argument or from the `WHISPER_COVDATA_PATH` environment variable.

Note that covariance that is not available is, by default, given an uncertainty along the diagonal, i.e., no correlation assumed, of 0.1 or 10%. This represents a crude approximation, but is more realistic than just assuming there is no uncertainty. This value is used during GLLS nuclear data adjustments for benchmark rejections or nuclear covariance data adjustments. The value of 0.1 may be changed by the user option `UnknownDataUncertainty`. The user options file is discussed in greater detail in Sec. 6, which covers advanced topics.

Since the nuclear covariance data adjustment may take a long time, often over an hour, it is usually beneficial to store these directories for later use. There are two possible approaches for setting this up.

One option for doing this is to create a new nuclear covariance library directory in the users personal space (call it `MyCovData`), copy over the directory `WHISPER_COVDATA_PATH/Data` to get the prior covariance data so that there is a directory `MyCovData/Data`, and copy over the new adjusted library to `MyCovData/Adjusted`. The user then needs to specify `-c MyCovData` on the Whisper command line or set the environment variable `WHISPER_COVDATA_PATH` to point to that new directory. Unfortunately, Whisper only prints out the covariance data files that were used in the benchmark suite for the nuclear covariance data adjustment and the additional isotopes should be copied over. Whisper provides a utility script in `Whisper/CovarianceData/Utilities` called `CopyUnadjustedData.csh` for this purpose. The script requires two command line arguments: the first is the path of the directory containing the prior nuclear covariance data, and the second is the path to the directory containing the new, adjusted data. For example,

```
./CopyUnadjustedData.csh WHISPER_COVDATA_PATH/Data NewAdjustedData
```

The second option is to replace the covariance data in the default directory for the adjusted covariance data `Whisper/CovarianceData/SCALE6.1/Adjusted`. This is a preferred option if an institution wishes to permanently use a set of covariance data. If this option is chosen, there is a script in the `Whisper/Utilities/CovarianceData` directory called `UpdateCovarianceData.csh`. The argument is the new adjusted nuclear covariance data directory,

```
./UpdateCovarianceData.csh NewAdjustedData
```

The script will then have a prompt where the user must type the string `confirm` to proceed. This will copy the new directory overwriting the old one. The covariance files that do not exist will then be copied over from the prior directory as explained by the `UpdateCovarianceData.csh` script.

Users should be aware that is a permanent change to the Whisper distribution users are strongly encouraged to make a backup of the `Whisper/CovarianceData/SCALE6.1/Adjusted` directory. Note that the installation tests will also no longer match the reference values if this is done, so the user should check and update those. The original adjusted covariance data can be restored by reinstalling Whisper, which may clear out other changes. This is done by going to the top-level Whisper directory, typing `make reinstall` and typing `confirm` when prompted. Users should then run the installation tests to verify the install occurred correctly.

5.2 Covariance Data Processing and Format

The nuclear covariance data has a format that closely resembles the format used by SCALE6.1, the ASCII form of the COVERX format. It has been further processed to remove information and heading information irrelevant to Whisper. The processing was done by the `ProcessCovData.csh` script in the `Whisper/Utilities/CovarianceData` directory. This script was run on the original SCALE-6.1 covariance data file residing in the `CovarianceData/SCALE6.1/Native` directory. The `ProcessCovData.csh` is provided so future covariance data releases in the COVERX format may be processed for Whisper. The argument to the `ProcessCovData.csh` is the name of the COVERX file.

The processed covariance files have the name `cov.ZA.dat` where `ZA` is replaced by the appropriate ZA of the isotope, e.g., hydrogen is 1001. Thermal scattering laws have the `ZA` represented by the string used by MCNP, e.g., `lwtr` for hydrogen bonded in water.

The first line of the file is a header that lists the isotope ZA or thermal scattering string, the number of energy groups (Whisper currently supports only the 44 group structure), and the number of ZA/reaction MT pairs that covariance data is given for, i.e., the number of blocks of data in the file. The next block of data is the energy grid in eV; recall that MCNP results are in MeV and Whisper converts appropriately. The following block lists the ZA/reaction MT pairs that are in the file.

For each ZA/reaction MT pair, the following three blocks are given: the header containing the ZA/reaction MT pair; a listing of indices for reading the data (the SCALE6.1 data is always a full matrix), and the covariance data itself listed in column-major order. The covariance data should be in relative covariance format, i.e., it represents the relative uncertainty or correlation of a particular piece of nuclear data. The blocks repeat for each ZA/reaction MT pair.

6 Advanced Topics

This section covers topics that will likely be relevant to only the most advanced users of Whisper. The first topic is the user options file that was briefly mentioned in the section on Benchmark Rejection; a full list of options is given here. The second topic deals with adding benchmarks to Whisper's provided suite, which may be important as new input files and benchmarks become available. The third topic discusses how to update installation tests when changes are made that impact reference solutions, which, in effect, creates a new baseline for the user and their institution. The final topic is a brief discussion of the source code.

6.1 User Options

User options are set in a text file created by the user. Each line has a single option in the format

```
option-name value
```

This file is provided to Whisper upon execution following the `-u` argument. For example, if the file is called `Options.dat`, then the Whisper command is

```
whisper -u Options.dat
```

A list of user options, their default, and a brief explanation is given in Table II. Note that all user options in the file are case sensitive.

Table II: Whisper User Options

`ThresholdChiSquare`	`1.2`	Threshold of acceptable GLLS nuclear data adjustment χ^2 for benchmark rejection.
`CalcMarginConfidenceLevel`	`0.99`	Value of the extreme value cumulative distribution function (CDF) that is used to determine the calculational margin.
`dxCalcMargin`	`1.e-5`	When determining the calculational margin from the extreme value CDF, the candidate calculational margin is increased by this amount until it is found.
`dxAcceptSimilarity`	`1.e-5`	Value that the factor to determine acceptance similarity parameter decreases by until the minimum benchmark weight is met and weight factors are determined.
`MinimumWeightSum`	`25`	The minimum total benchmark weight allowed when determining the calculational margin.
`WeightSumPenalty`	`100`	The penalty applied to the amount of benchmark weight needed for determining the calculational margin based upon the maximum similarity parameter of the benchmark set.
`MinimumNonCoveragePenalty`	`0.05`	The lowest allowed non-coverage penalty applied based upon the degree of inadequacy of entire the benchmark set to meet the minimum benchmark weight.
`DataUncMultiplier`	`2.6`	Confidence level, in number of standard deviations, multiplied by the nuclear data uncertainty to get the nuclear data driven margin of subcriticality.
`UnknownDataUncertainty`	`0.1`	Value of the uncertainty along the diagonal used for covariance data that is needed but unavailable.
`AdjustedCovarianceCutoff`	`1.e-6`	Lowest value of diagonal variance that is for writing a block to the adjusted covariance data file.
`IntegrationTolerance`	`1.e-9`	Integration tolerance for residual when computing the bias from mean of the extreme value density function.
`IntegrationLimitTolerance`	`1.e-6`	Tolerance for limits of integration when computing the bias from mean of the extreme value density function.

6.2 Adding Benchmarks

Adding benchmarks is done similarly to how applications are run.

First, benchmark MCNP input files should be prepared. At the bottom of the input file, the user must place the following line:

```
c k(bmk) = [keff] +- [unc]
```

where `keff` and `unc` get replaced with the benchmark values of k and the uncertainty. The script `RunMCNPInputs.csh` will take this line and ensure it gets placed in the sensitivity table of contents files. The `RunMCNPInputs.csh` script is run first, which submits the MCNP jobs to the cluster. Once these complete, the `MakeKeffSenLib.csh` script is run to extract the sensitivity profiles and produce the sensitivity table of contents; it is not necessary to use the script to run Whisper. Instructions for running these scripts are given in Sec. 3.

Next, there is a script called `AppendBenchmarks.csh` in the directory `Utilities/Benchmarks`. This script requires two arguments: the current sensitivity table of contents file and the one produced by the `MakeKeffSenLib.csh` script. Suppose these files are named `Current.toc` and `New.toc`; the command to execute the script is then

```
./AppendBenchmarks.csh Current.toc New.toc
```

Since this is a permanent change that can only be undone by a reinstall of Whisper, the user is required to type `confirm` before the script will proceed. The script appends the new benchmark information to the old sensitivity table of contents file and copies over the sensitivity profiles to the directory specified in the old sensitivity table of contents file from the one specified in the sensitivity table of contents file produced by the `MakeKeffSenLib.csh` script.

If this new file is pointed to by the `WHISPER_BENCHMARK_TOC` environment variable, the installation tests should now be rerun, as it may change reference solutions. If it does, the user should verify this change is desired and update the installation test reference solutions as described next.

6.3 Updating Installation Test Reference Solutions

Upon changing the Whisper directory structure by adding benchmarks, changing covariance data, etc., the reference installation test solutions may change. These changes should intentional and should be judged by the user to be what they expect. If this is so, the reference files may be updated. Following the update, the installation tests allow users to verify that future changes do not have unintentional effects.

To perform an update of the installation test reference solutions, execute the script `UpdateInstallSolutions.csh` in the `Utilities/Testing` directory. This will require the user to type the string `confirm` before proceeding to ensure this is not done by mistake. As the text of the script indicates, this change is permanent and can only be undone by a complete reinstall of Whisper. Upon execution, the script runs the installation tests, overwrites the reference results files with the new result files, and runs the tests again to verify that they pass.

After this is complete, typing `make install` in the `Testing` directory will, from this point on, run the tests with the new reference solutions.

6.4 Source Code

The Fortran source code for Whisper is provided in the `Source` directory. A `Makefile` is also provided so that the user can rebuild the code by typing `make` in the `Source` directory. This requires that the Intel Fortran compiler be present; if it is not, the `Makefile` must be modified to use the available compiler. The recompilation produces a new executable and overwrites the current one in the `bin` directory. Generally speaking, after recompiling, the user should run the installation tests again to verify that no unintentional changes have occurred.

Advanced users may find a need to modify the source code. For this reason, a description of each source routine is provided.

- `CovarianceMatrixMod.F90` – Module containing data structures and procedures for reading, writing, and processing covariance data.
- `DataAdjustmentMod.F90` – Module containing routines to perform GLLS nuclear data adjustment. This module has both the benchmark rejection and nuclear covariance adjustment routines.
- `FilesMod.F90` – A low-level module containing variables pertaining to files the Whisper reads and writes and whether those files have been specified on the command line.
- `InputProcessingMod.F90` – Module containing routines to parse the command line, read files, and populate data structures needed by other parts of Whisper.
- `IsotopeListMod.F90` – A low-level module defining the isotope list class and its methods. The isotope list class is a base class that is extended to derived classes in the `SensitivityMod.F90` module.
- `MatrixInverseMod.F90` – Module containing a LINPACK routines for matrix inversion and a wrapper to call them.
- `OptionsMod.F90` – A low-level module containing the list of user options and defaults. The user options may be overwritten with a user options file with the routines to do so in the `InputProcessingMod.F90` module.
- `ParametersMod.F90` – A low-level module containing compile-time constants used by Whisper.
- `SensitivityMod.F90` – Module containing class extensions for the isotope list class defined in `IsotopeListMod.F90`. The first extension adds the ability to store and manipulate sensitivity profile data. This derived class is further extended to be benchmark and application classes containing appropriate information and methods. This routine contains methods for these classes.

- `StatsMod.F90` – Module containing functions for computing properties of statistical distributions.
- `UncertaintyMod.F90` – Module containing routines to estimate nuclear data uncertainties and correlations for given sensitivity vectors and a covariance matrix. Contains the routine that estimates unknown benchmark uncertainties.
- `UpperSubcriticalLimitMod.F90` – Module containing routines and data structures to compute baseline upper subcritical limits and write them out.
- `WhisperMain.F90` – Main program for Whisper. This contains the high-level instances of data structures that are passed into routines given in the other modules. The main program is simply a list of function calls.