LA-UR-10-06217

Approved for public release;
distribution is unlimited.

Title:	MCNP5-1.60 Feature Enhancements & Manual Clarifications
Author(s):	Brian C. Kiedrowski Thomas E. Booth Forrest B. Brown Jeffrey S. Bull Jeffrey A. Favorite R. Arthur Forster Roger L. Martz
Intended for:	MCNP5 Documentation RSICC Release

Form 836 (7/06)

LA-UR-10-06217

MCNP5-1.60 Feature Enhancements & Manual Clarifications

B.C. Kiedrowski, T.E. Booth, F.B. Brown, J.S. Bull,
J.A. Favorite, R.A. Forster, R.L. Martz

X-Computational Physics Division
Los Alamos National Laboratory

MCNP5-1.60 has new features of interest to the nuclear design, radiation transport, and criticality safety communities: adjoint-weighted point kinetics parameters, isotopic mesh tallies, enhancements for large problems, and additional external scripts to assist users with variance reduction. New internal parameterizations to aid with MCNP development are given; this is important for those who modify MCNP5 for specific applications. Additionally, the following items in the MCNP5-1.50 manual are clarified: perturbations and sensitivity coefficient calculations, tally fluctuation chart history score plots, how weight cutoffs and forced collisions work together in the presence of pulse height (F8) tallies, and some discussion on parallelism in MCNP5.

Contents

Adjoint-Weighted Point Kinetics Parameters

MCNP5-1.60 can now produce the point-kinetics parameters for criticality (KCODE) problems using the newly introduced criticality calculation options, or KOPTS, card.

The point-kinetics parameters are the neutron generation time Λ, the effective delayed neutron fraction β_{eff}, and Rossi-α. Assuming space-time separability, the neutron population $n(t)$ as a function of time t in a system perturbed from a critical configuration (no external source) can be found by the differential equation

$$\frac{dn}{dt} = \frac{\rho - \beta_{\text{eff}}}{\Lambda} n(t) + \sum_i \bar{\lambda}_i C_i(t). \tag{1}$$

Here ρ denotes the reactivity (typically defined by $\rho = (k-1)/k$), and the second term deals with delayed neutron precursors (fission products that emit neutrons on timescales of seconds). The precursors are broken into typically six or eight groups each with index i, having an average decay constant $\bar{\lambda}_i$ and concentration $C_i(t)$. Each precursor group has a differential equation coupled to the neutron population,

$$\frac{dC_i}{dt} = \frac{\beta_i}{\Lambda} n(t) - \bar{\lambda}_i C_i(t). \tag{2}$$

Here, β_i is the fraction of the effective delayed neutron fraction that produces neutrons of group i. By solving these coupled differential equations, it is therefore possible to model the time-dependence of a system for small perturbations in reactivity.

To solve these equations, the parameters need to be either measured experimentally or computed should no such measurement be available. Each of these parameters is defined as a ratio of integrals of some quantity weighted by an importance or adjoint function (these arise from solving the forward and adjoint transport equations with some simplifying assumptions on space-time separability). These parameters are

$$\Lambda = \frac{\langle \psi^\dagger, 1/v\ \psi \rangle}{\langle \psi^\dagger, F\psi \rangle}, \tag{3}$$

$$\beta_{\text{eff}} = \frac{\langle \psi^\dagger, B\psi \rangle}{\langle \psi^\dagger, F\psi \rangle}, \tag{4}$$

$$\alpha = -\frac{\beta_{\text{eff}}}{\Lambda}. \tag{5}$$

The brackets denote an integration over all space, energy, and direction, ψ is the forward flux and $\psi^\dagger$ is the adjoint function, v denotes the neutron speed, F is an operator for total fission (prompt plus delayed), and B is the operator for the delayed component of fission.

To solve for the precursors, a β_i and $\bar{\lambda}_i$ are needed for each precursor i. The former is an adjoint-weighted quantity similar to β_{eff},

$$\beta_i = \frac{\langle \psi^\dagger, B_i\psi \rangle}{\langle \psi^\dagger, F\psi \rangle}, \tag{6}$$

where B_i is the source of delayed neutrons emitted only from precursor group i. The average decay constant is simply the average of the decay constants λ_{ij}, which are different for the different isotopes fissioned j and may vary with energy of the neutron causing fission, of neutrons emitted from that precursor group. For ENDF/B-VII.0 data, the decay constants are only functions of the isotope fissioned; therefore, the average decay constant for precursor i is found by

$$\bar{\lambda}_i = \sum_j h_{ij}\lambda_{ij}, \tag{7}$$

where h_{ij} is the fraction of fissions from isotope j that produced precursor i in the system. Note that the sum over j of h_{ij} is unity so that Eq. (7) can be thought of as a weighted average.

For a criticality problem, the adjoint function at a location in phase space is proportional to a measured detector response, after infinitely many fission generations, caused by a neutron introduced into the system at that location in phase space.

MCNP5 computes the point-kinetics parameters in a forward calculation with only the existing random walks by breaking the active cycles of a KCODE calculation into sequential blocks of fission generations. In the first cycle of the block, tally contributions (the term "contribution" is used to distinguish a tally that is eventually weighted by importance, making the adjoint-weighted score) are accumulated, and in the final cycle of the block, the detector response function resulting from all its progeny is measured by tally estimators. The product of these two form a score for an adjoint or importance-weighted tally. MCNP5 uses the points of introduction as the locations of various tally scores; these neutrons are called progenitors and tagged with indices that are passed onto subsequent progeny. The detector response function chosen is the total neutron production, which is estimated by a track-length neutron production estimator, caused by all progeny of a specific progenitor in the last generation of the block.

A definition of a progenitor must be selected such that it encompasses the branching that may occur within a history. At the first tally score (source emission in the case of point-kinetics parameters) in the history, the neutron is tagged with a progenitor index. Whenever there is a branching in the history path, which may arise from an n,2n reaction or an implicit capture generating a fission neutron for the next cycle where the original neutron continues in the current history, a new progenitor index is required. Each progenitor index is tied to a set of tally contributions that will later be multiplied by an importance score – a possible way to think of this is the set of tally contributions $T \in \{t_1, t_2, ..., t_K\} = T(p)$ are functions of progenitor indices denoted by p. The tally contributions referenced by a progenitor index contain not only those for the portion of the current index, but for all other indices on branches leading back to the parent state. An example of a history and its associated progenitors (denoted by a set Π) is given in Fig. 1.

Whenever a fission neutron for the next cycle is generated, the current progenitor index p is passed onto any progeny for all cycles within the block. Note that it is only the first generation in the block where new progenitor indices are assigned; however, the information continues to be passed onto all progeny. In the last cycle in the block, the neutron production is measured with a track-length estimator all neutron progeny

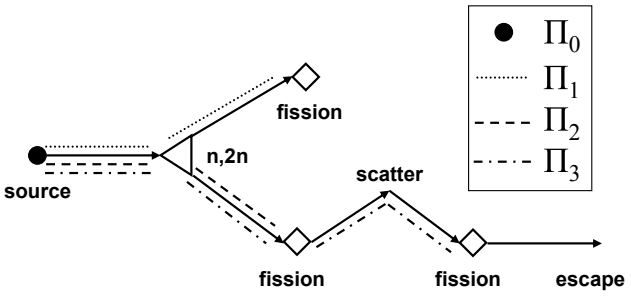

Figure 1: Example of a neutron history. The four progenitor sets Π for this history are specified.

with the same progenitor index p (typically covering many histories in this cycle). Within history n, the neutron production is given by f_n,

$$f_n = \sum_{\text{tracks} \in n} \nu\Sigma_f w d, \tag{8}$$

where $\nu\Sigma_f$ is the mean neutrons produced per fission event times the macroscopic fission cross section, w is the particle weight, and d is the length of the current track. The detector response for progenitor p, $R(p)$, is the sum over all f_n for all histories n that have a progeny neutron with progenitor index p. An adjoint-weighted tally A is then the sum over p of all products of $T(p)$ and $R(p)$ (denoted by $P(p)$), or

$$A = \frac{1}{M}\sum_p T(p)R(p) = \frac{1}{M}\sum_p P(p), \tag{9}$$

where M is the total number of histories where progenitor indices are assigned (which, again, only occur in the first cycle in the block). An example of a history with neutron progenitors propagating through various cycles is given in Fig. 2 with an example computation of the adjoint-weighted tally scores in Table I.

Table I: Example of adjoint-weighted scoring quantities, $T(p)$, $R(p)$, and $P(p)$ for the example in Fig. 2.

p	$T(p)$	$R(p)$	$P(p)$
1	t_1	$f_2 + f_3$	$t_1\,(f_2 + f_3)$
2	$t_1 + t_2 + t_3$	f_1	$(t_1 + t_2 + t_3)\,f_1$

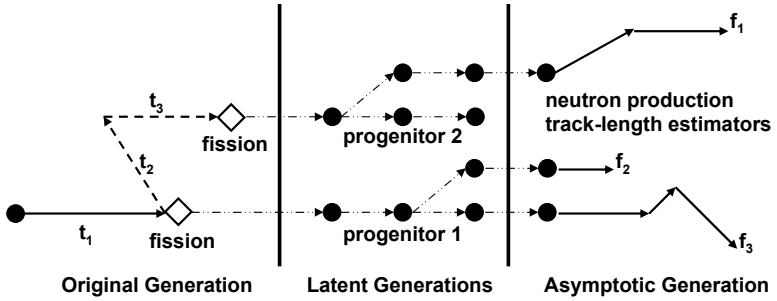

Figure 2: Example of a random walk of a single history in the original generation and subsequent generations within a block.

A subtle detail is reconciling the impact of a non-analog simulation with the forward interpretation of the adjoint function. In an analog simulation, what occurs in the next event only depends upon the current state in phase space (a purely Markov process) – the effect of taking any state of the neutron in its random walk and following it to calculate a response is the same as if the neutron had been introduced into that point. Unfortunately, in non-analog simulations, the behavior of the random walk depends upon previous states through the particle weight w and is, strictly speaking, non-Markovian with respect to the traditional phase space. The weighting is done in such a way to preserve the expected scores, but it is no longer permissible to equate taking a snapshot of a neutron in a forward random walk with introducing a hypothetical neutron at that point because of this extra information carried in the particle weight. It can be shown that the expected number of eventual progeny produced (detector response) between the analog and non-analog cases are different by a factor of particle weight w and that this equivalence can be restored by a factor of C/w where C is some normalization constant and, for convenience, will be taken to be unity. This factor of $1/w$ is applied to each component of $T(p)$ making the tally contributions appear to be not multiplied by particle weight.

To solve for the point-kinetics parameters, it is necessary to solve for the integrals (each forming a tally) in Eqs. (3), (4), and (6). Observe that they all share the same denominator but have different numerators, meaning three different integrals, plus an additional one for each precursor group, need to be evaluated. The integral of the non-importance weighted fission source can be tallied in MCNP by simply summing the weight of every fission source particle (the mechanics of neutron production between cycles handles the details) and multiplying by the estimate of k_{eff} (this arises from source normalization). To perform the importance weighted version, each source emission must be multiplied by its detector response $R(p)$. Because of the factor of $1/w$, the tally contribution is therefore $T(p) = 1$, and the tally is

$$\left\langle \psi^{\dagger}, F\psi \right\rangle = \frac{k}{M} \sum_{p} R(p). \tag{10}$$

The numerator in the expression for β_{eff} is calculated in a similar way except that only delayed neutrons are considered. Suppose m is the emitted precursor index where $m = 0$ implies a prompt neutron. The equation for the adjoint-weighted delayed source is then

$$\left\langle \psi^{\dagger}, B\psi \right\rangle = \frac{k}{M} \sum_{p} R(p)(1 - \delta_{m0}), \tag{11}$$

where δ_{ij} is the Kronecker delta function defined to be one if $i = j$ and zero otherwise. Therefore in this expression, the function is non-zero when $m \neq 0$, or when the neutron is delayed. Likewise, the component-wise delayed fission source for precursor i is found by

$$\left\langle \psi^{\dagger}, B_i\psi \right\rangle = \frac{k}{M} \sum_{p} R(p)\delta_{mi}. \tag{12}$$

All that remains is the numerator of Eq. (3) or the adjoint-weighted neutron density. MCNP5 obtains $T(p)$ in this case from a track-length flux estimator divided by the neutron speed v. The neutron density for progenitor p is this estimator summed over every track with progenitor p, or

$$\left\langle \psi^{\dagger}, 1/v\ \psi \right\rangle = \frac{1}{M} \sum_{p} R(p) \sum_{\text{tracks} \in p} \frac{d}{v}. \tag{13}$$

The average decay constant for precursor i is computed by

$$\bar{\lambda}_i = \frac{1}{N_{fi}} \sum_{\text{fissions} \in i} \lambda_i, \tag{14}$$

where N_{fi} is the number of fissions producing a precursor of type i and λ_i is the decay constant for each individual fission event, which depends upon the isotope causing fission and possibly the energy of the neutron causing fission.

MCNP5 does not compute the point-kinetics parameters by default; they must be asked for by the user via options on the new KOPTS card. The KOPTS card is specified in the data card section and takes the

following format:

```
kopts   keyword1=entry1   keyword2=entry2   ...
```

Table II gives the valid keywords and information as of MCNP5-1.60. To have MCNP5 compute the kinetics parameters the `kinetics` keyword needs to have its associated entry as `yes`. By default, information on the precursors (β_i's and $\bar{\lambda}_i$'s) is not computed. To obtain this output, set the `precursor` keyword entry to `yes`. The default size of the block is ten cycles, but this may be changed with the `blocksize` keyword.

Table II: List of keywords for the KOPTS card.

Keyword	**Description**	**Valid Options**	**Default**
`blocksize`	Number of cycles in blocks for adjoint weighting. Specifying this without setting `kinetics` to `yes` is allowed, but MCNP5 will try to do adjoint weighting without tallying anything.	Integer $\geq$ 2	10
`kinetics`	Whether to calculate point-kinetics parameters.	`yes/no`	`no`
`precursor`	Whether to calculate detailed precursor information. If this is `yes`, then kinetics must be set to `yes` also.	`yes/no`	`no`

The default has been shown to lead to results with sufficient accuracy for most problems of interest; however, fewer or more KCODE cycles per block may be desired or required. Using fewer introduces greater bias from truncation but results in a more statistically efficient calculation. Larger blocks are more accurate, but the accuracy gained for larger block sizes is usually small (law of diminishing returns) relative to the increased run time required to preserve the statistical precision. In either case, users are encouraged to check whether the default (or their selected block size) is sufficient for their applications by running a larger block to observe the difference.

Example:

Suppose a user wishes to compute both the standard kinetics parameters and the information on precursors and that the user has determined from empirical studies that 15 generations per block are needed for the application. The user would add the following line in the data card block of the input file:

```
kopts   blocksize=15   kinetics=yes   precursor=yes
```

Prior to MCNP5-1.60, users would need to use the prompt removal time t_{rem} as opposed to the more correct neutron generation time Λ for estimates of neutron lifetime. The primary difference between the two is that the prompt removal time is not adjoint weighted; namely,

$$t_{\text{rem}} = \frac{\langle 1, 1/v\ \psi \rangle}{\langle 1, F\psi \rangle} \tag{15}$$

would be computed by MCNP5. For many problems, this would adequately approximate the neutron generation time. Typically, the error would be on the order of several percent. For example, accounting for the importance weighting from leakage effects of Godiva results in about a 10% difference between the two. However, for some problems, especially those involving thick reflectors, t_{rem} may differ from the correct value, Λ, by as much as two or three orders of magnitude.

Isotopic Mesh Tallies

A new feature in MCNP5-1.60 is the isotopic reaction mesh tally, which will allow the user to calculate isotopic reaction rates throughout the problem geometry. This feature can be used, for example, to estimate

the total number of fissions from a specific isotope in a problem, or the production rates of isotopes in a material.

To use this feature, a user will typically define a new dummy material card containing only the isotope(s) of interest. This dummy material is not used in the problem geometry, but instead is used exclusively on the FM card. The dummy material is defined using the normal material card (m), with the isotope ZAID followed by the atomic/gram density. For these materials, however, the densities are not used but must exist as placeholders. Instead, the atom fractions of the isotopes needed for the calculation are taken from the materials used in the geometry during transport.

To specify a reaction rate mesh tally, a + symbol is placed in front of the FM card associated with the mesh tally. The rest of the FM card is set up the regular way, with a multiplicative constant followed by the dummy material number and the ENDF reaction numbers of interest.

When calculating the mesh tally, MCNP will multiply the particle flux times the cross sections for the isotopes defined in the material card. This value is then multiplied by the atom fraction of the isotope of the material in which the particle is traveling to calculate the isotopic reaction rate tally. The units of the results will be (# reactions/cm^3) or (# reactions/cm^3/shake) depending upon the units of the source (If the source is time-dependent, tally results are interpreted as time-integrated. If the source is steady state, tally results are per unit time. A steady-state tally result may be multiplied by a time interval to change the units from a reaction rate density to a reaction density.). Note that placing a minus sign in front of the multiplicative constant will multiply the results by the atom density of the cell. This is usually NOT the desired result.

For more information on tally multipliers and the FM card, see pages 2-105 and 3-99 of the MCNP manual.

Example:

In this problem, there are two cells, one composed of natural uranium and the other depleted uranium. To calculate the fission rates for each isotope in both cells a mesh tally is used:

```
c    Cells
900    100    -19.1   -1     $ Natural Uranium
901    200    -19.1    1     $ Depleted Uranium

c Surfaces
...

c   Problem materials
c     Natural Uranium
m100  92238     0.992745
      92235     0.007200
c     Depleted Uranium
m200  92238     0.998049
      92235     0.001951
c   Dummy materials for FM mesh tallies
m235 92235   1.0
m238 92238   1.0
c
fmesh04:n  ...
fmesh14:n  ...
fmesh24:n  ...
...
+fm04   1   235   -6   $ fission rate from U235
+fm14   1   238   -6   $ fission rate from U238
+fm24   1   100   -6   $ total fission rate from both U235 and U238
```

Notes:
1) The default units of the results are the number of fissions per cm^3 (or per cm^3 per shake) in each mesh

cell.
2) For tally 24, material 200 could be used instead of material 100 since both materials contain the same isotopes.

Example:

This problem contains a single cell composed of concrete. We want to calculate the production rate of ^{24}Na and ^{55}Fe in the material.

```
100 10 -2.35  -5

c Ordinary Concrete (rho = 2.35 g/cc)
m10      1001     -0.00600
         8016     -0.50000
        14028     -0.28940
        14029     -0.01518
        14030     -0.01042
        13027     -0.04800
        20000     -0.08300
        26054     -0.00068
        26056     -0.01106
        26057     -0.00026
        11023     -0.01700
        19000     -0.01900
m20   11023     1
m21   26054     1
fmesh4:n  ...
fmesh14:n  ...
...
C    102 = (n,gamma) reaction
+fm4     1 20   102   $ Na-24 production
+fm14    1 21   102   $ Fe-55 production
```

Notes:
1) The ^{23}Na and ^{54}Fe isotopes are used on the dummy materials since a (n,γ) reaction on these isotopes produce ^{24}Na and ^{55}Fe respectively.
2) The production rate is calculated by multiplying the (n,γ) reaction cross section times the atomic fraction of the isotope in material 10.

Greatly Increased Limits for Geometry, Tally, Material, and Source Specifications

MCNP5-1.60 includes modifications that extend the limits on the permitted cell numbers, surface numbers, material numbers, etc., from 99,999 (or $10^5 - 1$) to 99,999,999 (or $10^8 - 1$). The maximum number of tally card identifiers was also raised from 999 to 9999.

The maximum permitted card number for MCNP5-1.60 input (e.g., m12345 is material 12345) has been raised from 99,999 to parameter MAX_CARD_NUMBER = 99,999,999 or $10^8 - 1$).

Many print statements were changed to allow for larger cell and surface numbers. Typically *I5* or *I6* Fortran formats were changed to *I9*. This results in more blanks in many lines.

Table III summarizes the previous and current limits.

The line length for the *big_strings* print routines was changed from 90 to 120. As a result, some lines sent to the screen window may now be up to 120 characters long.

The format of *mctal* files was modified to accommodate the larger permitted values for cell and surface numbers: The cell and surface numbers for the problem are first checked to see if any are greater than 99,999. If not, then the traditional formatting is used when writing the *mctal* file. If there are numbers greater than

Table III: Previous and current limits for MCNP5 problem specifications.

	MCNP5-1.51	**MCNP5-1.60**
Allowed cell numbers	1 - 99,999	1 - 99,999,999
Allowed material numbers	0 - 99,999	0 - 99,999,999
Allowed surface numbers	1 - 99,999	1 - 99,999,999
Allowed universe numbers	1 - 99,999	1 - 99,999,999
Allowed surface numbers for transformations	1 - 999	1 - 999
Allowed cell numbers for transformations	1 - 999	1 - 999
Allowed tally numbers	1 - 999	1 - 9,999
Allowed perturbation numbers	1 - 999	1 - 999
Allowed source distribution numbers	1 - 999	1 - 999
Allowed "card numbers"	1 - 99,999	1 - 99,999,999
Maximum levels for geometry	10	20
Maximum number of detector tallies	100	100
Maximum number of *dxtran* spheres per particle type	10	10
Maximum length for cell geometry list (*mlgc*)	1,000	9,999
Maximum length of file names	256	256
Maximum entries for *dbcn*	30	100
Maximum entries for *idum* and *rdum*	50	2,000
Maximum number of *uran* universes	2	unlimited

99,999, then *I9* format is used instead for all integers written to the *mctal* file. It should be noted that the mctal file is defined to be free-format, without rigid output format requirements. However, some simple user-written utility programs that read the *mctal* file may expect fixed format. If such user-written programs cannot be modified to handle the larger integers, then users should be careful to use only numbers less than 99,999 for cell and surface numbers in their input files.

It should be noted that the output formats for PTRAC and event logs were not changed. The formats for those output options are complex, and many code users have invested large amounts of time in creating utilities to read the PTRAC and event log output. As a result, users intending to use PTRAC or event logs should avoid the use of cell, surface, or material numbers greater than 99,999.

Most integer quantities in MCNP5 are still defined as the default integer size, 32 bits on most systems and 64 bits on some systems. While changing many limits from 99,999 to 99,999,999 was relatively straightforward, further extension to one billion or more would require using 64-bit integers for everything.

It is important to note that while individual material numbers may now be as large as 99,999,999, that does not imply that the total number of materials in a problem can be as large as 99,999,999. The total size possible for a problem depends on many different parameters in a complex manner. For the 32-bit MCNP5 executable, a single array cannot exceed 2.147 GB in size (268 M real-8 words or 536 M integers). Since many arrays are multidimensional, with sizes such as (number of materials) $\times$ (number of isotopes), problems may not fit in the allowable virtual address space for a computer if there are very large total numbers of materials, cells, surfaces, or tallies. It has been observed, for example, that for reactor problems with about 200 nuclides per fuel material, only around 15-20 K fuel materials can be accommodated before exhausting the memory address space and aborting. Using a 64-bit MCNP5 executable on systems that support it may permit larger problems.

It is also important to note that the MCNP5 input processing and setup routines were developed with the assumption of a few dozen or hundred cells, surfaces, and materials. Some of the checking algorithms in the input processing routines, for example, require computer time proportional to the **square** of the number of cells, surfaces, or materials. While these algorithms may be tolerable when extended to 1000s, the input checking times may be excessive for problems with millions of cells, surfaces, or materials. The MCNP5 input processing routines are being revised as computational bottlenecks are discovered, but that process will take much time and has not been completed for the current MCNP5-1.60.

Variance Reduction Scripts

Included in this release are two external Perl scripts that are routinely used in the MCNP classes taught at LANL. Attendees always seem to find these scripts useful, so both of them are now available with this release. Both of these scripts are not necessarily finished products, but are considered developmental and works in progress. Nevertheless, they can be quite useful for users dealing with variance reduction issues.

Event Log Analyzer (ELA)

The Event Log Analyzer (ELA) has been available since release 1.50. ELA can be considered a series of Perl scripts that function together. The initial implementation was a GUI-only version that required that the user have installed Perl version 5.8.5 or later and the Tk package. Recently, we have added a command line option for those who are having problems with the Tk install or who prefer command line programs. The command line option permits execution of only the surface analysis and distance analysis features. For more details on the command line feature, see the README in the Utilities directory or supply the `-help` switch to the master control script, *ela.pl*, from the command line.

This program is demonstrated and taught in the advanced variance reduction class at LANL. It is recommended only for serious variance reduction users.

Cell Importance Input

The Perl script *vrtsplit1.pl* is a command line script that creates new importances (values for the `imp` card) from old ones. It adjusts the cell importances that are currently in use and in an existing output file to obtain a constant track distribution from source to tally. The script requires one or two arguments on the command line. The first argument to appear on the command line must be the name of an MCNP output file (*outp*) with Print Tables 60 and 126 present in the file.

The second argument is optional. This argument is the name of a file that contains cell numbers, separated by white space, for which the ratios are to be calculated. Cell numbers may be all on one line, one per line, or a mixture. If the second file is not present, the script will calculate cell importances and ratios for all cells present in Print Table 60 that have non-zero tracks and non-zero importances.

If the second argument is present, the cell importances are calculated for only those cells named in the file for all that have non-zero tracks and non-zero importances. Ratios are calculated for cell pairs starting from the first pair of cell numbers. The first two cell numbers form the first pair. The second and third cell numbers form the second pair, etc. This file of cell numbers gives the user the ability to select the important line of sight cells from the source to the tally in a highly three-dimensional geometry. As always, the success of the importance splitting game is dependent upon the cell layout.

This script generates two results files. New cell importances appear in the default or specified order in the *importance.inp* file. Cell importance ratios appear with the "m-notation" (see Vol. II, Chapter 3, Sec. I.D.1 on horizontal input in the MCNP5-1.50 manual) in the default or specified order in the *splitratio.inp* file. Results from either of these files may be copied into the MCNP input deck. These results are also displayed in the command window.

A help message is displayed if the script is run without any arguments.

This program is demonstrated and taught in both our introductory MCNP class and the advanced variance reduction class at LANL.

Parameterization of Special Constants for Geometry, Materials, Tallies, and Sources

The changes described in this section do not affect user input specifications for MCNP5. Users who have modified previous versions of MCNP5 or have patches to the code should examine their patches carefully, however, altering them according to the discussion below and parameter values given in Tables IV and V.

Many previously hardwired constants, such as 1000003, were used in geometry, source, and tally routines as flags for operators (e.g., ":", "(", "[", etc.). These have been parameterized, with parameters defined in *mcnp_params.F90*. The previous usage of these special constants is given in Table IV and a summary of the replacement parameters and their values is given in Table V. It should be noted in Table IV that a number of symbols such as "(", ")", "[", "]" were represented internally in MCNP inconsistently, with one special

integer value used in tally routines and a different special integer in geometry or source routines. This was a source of great confusion in maintaining MCNP; the consistent parameterization of the special integers to represent the symbols greatly reduces confusion and the maintenance burden.

The parameterization of input constants representing symbols was accomplished in several passes, addressing just geometry parameters, then tally parameters, then source parameters. Testing was performed in each pass and upon completion of the parameterization. No differences were introduced into the Regression Test Suite results or the Criticality Validation Suite, other than extra blanks in many of the printed output lines.

In many places throughout the MCNP source coding, the number 100,000 was used either as a flag or an upper limit for testing. That is, since the number used for a material was required to be 99,999 or less, undefined materials could be given the number 100,000 during input processing and problem set up operations. Similar treatment was used for cell and surface numbers. These occurrences of 100,000 were replaced by the parameters MAX_CELL, MAX_CELLP, MAX_SURF, MAX_SURFP as appropriate.

Table IV: Parameterized special constants for tallies, sources, and geometry.

original	input	used by			new parameter	
integer	symbol	tal	sdef	geom	NAME	VALUE
1000000		x	x	x	I_FLAGS	1050000000
1000001	(	x	x	x	I_LEFT_PAREN	I_FLAGS + 1
1000002	)	x	x	x	I_RIGHT_PAREN	I_FLAGS + 2
1000003	:	x	x	x	I_COLON	I_FLAGS + 3
1000004	#	x	x		I_POUND	I_FLAGS + 4
1000005	<	x			I_LESS_THAN	I_FLAGS + 5
1000005	[		x		I_LEFT_SQUARE	I_FLAGS + 6
1000006	[	x			I_LEFT_SQUARE	I_FLAGS + 6
1000006	]			x	I_RIGHT_SQUARE	I_FLAGS + 7
1000007	]	x			I_RIGHT_SQUARE	I_FLAGS + 7
1000007	(		x		I_LEFT_PAREN	I_FLAGS + 1
1000007				x	I_PATH_BEGIN	I_FLAGS + 11
1000008	,	x			I_COMMA	I_FLAGS + 8
1000008	)		x		I_RIGHT_PAREN	I_FLAGS + 2
1000008				x	I_PATH_END	I_FLAGS + 12
1000009	u	x			I_U	I_FLAGS + 9
1000009	<		x		I_LESS_THAN	I_FLAGS + 5
1000010	=	x			I_EQUALS	I_FLAGS + 10
10000000		x			I_TAL_FLAG1	I_FLAGS + 13
20000000		x			I_TAL_FLAG2	I_TAL_FLAG1 × 2

Table V: Parameter names and values.

INTEGER PARAMETERS	**VALUES**
MAX_CARD_NUMBER	99999999
MAX_CARD_NUMBERP	MAX_CARD_NUMBER + 1
MAX_CARD_NUMBER_DIGITS	99999999
MAX_CELL	99999999
MAX_CELLP	MAX_CELL + 1
MAX_SURF	99999999
MAX_SURFP	MAX_CELL + 1
I_FLAGS	1050000000
I_LEFT_PAREN	I_FLAGS + 1
I_RIGHT_PAREN	I_FLAGS + 2
I_COLON	I_FLAGS + 3
I_POUND	I_FLAGS + 4
I_LESS_THAN	I_FLAGS + 5
I_LEFT_SQUARE	I_FLAGS + 6
I_RIGHT_SQUARE	I_FLAGS + 7
I_COMMA	I_FLAGS + 8
I_U	I_FLAGS + 9
I_EQUALS	I_FLAGS + 10
I_PATH_BEGIN	I_FLAGS + 11
I_PATH_END	I_FLAGS + 12
I_TAL_FLAG1	I_FLAGS + 13
I_TAL_FLAG2	I_TAL_FLAG1 × 2
ipack_nsf_ncl	max(MAX_CELLP, MAX_SURFP)
CHARACTER PARAMETERS	**VALUES**
C_FLAGS	’():#<[],u=’
C_LEFT_PAREN	’(’
C_RIGHT_PAREN	’)’
C_COLON	’:’
C_POUND	’#’
C_LESS_THAN	’<’
C_LEFT_SQUARE	’[’
C_RIGHT_SQUARE	’]’
C_COMMA	’,’
C_U	’u’
C_EQUALS	’=’
C_BLANK	’ ’
C_PERIOD	’.’
C_DOLLAR	’$’
C_AMPERSAND	’&’
C_PLUS	’+’
C_MINUS	’-’

Perturbations and Using MCNP5 to Calculate Sensitivity Coefficients

The sensitivity of the response c to cross section σ_x is defined as

$$S_{c,\sigma_x} = \frac{\sigma_{x,0}}{c_0}\frac{dc}{d\sigma_x}, \tag{16}$$

where $\sigma_{x,0}$ is the reference value of σ_x and $c_0 = c(\sigma_{x,0})$ is the reference value of the response. Defining the relative cross-section perturbation $p = \Delta\sigma_x/\sigma_{x,0}$, the first-order Taylor series term Δc_1 is

$$\Delta c_1 = \frac{dc}{d\sigma_x}\Delta\sigma_x = \frac{dc}{d\sigma_x}\sigma_{x,0}p \tag{17}$$

and the sensitivity is

$$S_{c,\sigma_x} = \frac{\Delta c_1}{c_0 p}. \tag{18}$$

This equation provides a prescription for calculating sensitivities by post-processing MCNP output: Divide the first-order Taylor series term by the unperturbed (reference) response c_0 and by the relative cross-section perturbation p. Since the first-order term is linear with p (with no offset), the computed sensitivity will be completely independent of the user-input value of p. Thus, p is arbitrary.

The (calculated) statistical relative uncertainty in the sensitivity is given by the usual propagation of errors formula to be

$$\frac{s_S}{S} = \sqrt{\left(\frac{s_{\Delta c_1}}{\Delta c_1}\right)^2 + \left(\frac{s_{c_0}}{c_0}\right)^2} \tag{19}$$

(where s_x is the sample standard deviation of quantity x computed by MCNP), assuming that Δc_1 and c_0 are uncorrelated, which is not true if they are computed using the same set of histories. This is, nevertheless, a common approximation.

Detailed parameter studies using a second-order Taylor series expansion can easily be accomplished with only two perturbations. The second-order Taylor series term Δc_2 is

$$\Delta c_2 = \frac{1}{2}\frac{d^2c}{d\sigma_x^2}\left(\Delta\sigma_x\right)^2 = \frac{1}{2}\frac{d^2c}{d\sigma_x^2}\sigma_{x,0}^2p^2. \tag{20}$$

Define coefficients

$$c_1 = \frac{dc}{d\sigma_x}\sigma_{x,0} \quad \text{and} \quad c_2 = \frac{1}{2}\frac{d^2c}{d\sigma_x^2}\sigma_{x,0}^2, \tag{21}$$

which are constants related to the reference case and have nothing to do with a perturbation. However, for any arbitrary perturbation p, the constants can be recovered by post-processing MCNP output, using

$$c_1 = \frac{\Delta c_1}{p} \quad \text{and} \quad c_2 = \frac{\Delta c_2}{p^2}. \tag{22}$$

Then the perturbed response associated with any perturbation p' can be calculated using

$$\Delta c(p') = c_1p' + c_2p'^2, \tag{23}$$

and its standard deviation is

$$s_{\Delta c} = \left|\frac{p'}{p}\right|\sqrt{s_{\Delta c_1}^2 + \left(\frac{p'}{p}\right)^2 s_{\Delta c_2}^2}, \tag{24}$$

assuming Δc_1 and Δc_2 are not correlated; however, they are in actuality.

As an example, suppose it is desired to plot Δc as a function of p in 10% increments between -50% and $+50\%$. This can be done using 10 perturbations, each giving the total perturbation (first- plus second-order

terms), along with another 10, each giving either the first- or second-order term separately so that the effect of ignoring third- and higher-order terms can be estimated for each value of p. Or, this can be done using only two perturbations for an arbitrary value of p, one for the first-order term and another for the second-order term; then the coefficients c_1 and c_2 are recovered and used as shown above. The results for Δc will be exactly the same as if the 20 perturbations had been done individually, but the standard deviations will vary. The limitations on p. 2-202 of the MCNP5-1.50 manual apply to these post-processed results just as they do to the MCNP perturbation results.

Classical first-order sensitivity analysis requires only the first-order term, `method=2` keyword on the `pert` card; in this case, the relative magnitude of the second-order term is irrelevant. For Example 4 in Volume II, Chap. 3, Sec. IV.J.10, of the MCNP5-1.50 manual, `method=2` should be added to both `pert` cards:

```
60  13   -2.34   105 -106 -74 73   $ mat 14 at 2.34 g/cc
...
m13    1001  -0.2    8016  -0.2   13027  -0.2   26000 -0.2   29000  -0.2
m15    1001  -0.2    8016  -0.2   13027  -0.2   26000 -0.2   29000  -0.4
pert1:p   cell=60   mat=15   rho=-2.808   rxn=51 9i 61,91   erg=1,20
          method=2
pert2:p   cell=60   rho=-4.68   rxn=2   method=2
```

This example illustrates sensitivity analysis. The first `pert` card generates the first-order Taylor series terms Δc_1 for changes in tallies caused by a $p = 100\%$ increase in the Cu cross section (ENDF/B reaction types 51 - 61 and 91) above 1 MeV. To effect a $p\%$ change for a specific isotope, set up a perturbed material mimicking the original material, except multiply the composition fraction of the perturbed isotope by $1 + p$ (0.2 to 0.4). The density of the perturbed material is the density of the original material (2.34 g/cm^3) multiplied by the ratio of the sum of the weight fractions of the perturbed material (1.2) to the sum of the weight fractions of the unperturbed material (1.0), or `rho` = (2.34 g/cm^3 $\times$1.2/1.0) = 2.808 g/cm^3. This change must be made to `rho` to maintain the other nuclides in their original amounts. Otherwise, after MCNP normalizes the `m15` card and multiplies the constituent weight fractions by the unperturbed material density, the density of all of the constituents would be perturbed, which is not the intent. When MCNP normalizes the `m15` card and multiplies the constituent weight fractions by the correctly modified material density, the density of the unperturbed isotopes will be unchanged, but the density of the perturbed isotope will be changed by a factor $1 + p$, as intended.

The sensitivity of response c is calculated in post-processing using $S = \Delta c_1/(c_0 p)$ where p is arbitrary.

The second `pert` card (`pert2:p`) gives the first-order Taylor series terms Δc_1 for changes in tallies caused by a 100% increase in the elastic (`rxn=2`) cross section of material 13. `rho` =2.34 g/cm^3 $\times$ 2 = 4.68 g/cm^3.

Tally Fluctuation Chart History Score Plotting

2-D plots of a tally $F(x) = xf(x)$ are made by dividing the tally bin value by the width of the tally bin $x_{i+1} - x_i$. Visually Accurate Area (VAA) $F(x)$ plots are plots whose visual area under the curve is an accurate representation of the tally in each of the tally bins; i.e., the visual area represents $F(x)(x_{i+1} - x_i)$ for all abscissa values. A VAA plot will be produced for a linlin0 (linear abscissa scale, linear ordinate scale starting at 0) $F(x)$ plot; i.e., the area of $F(x)$ from x_i to x_{i+1} correctly represents the bin tally value visually for all x when both the abscissa and ordinate scales are linear and the smallest ordinate value is zero.

In a similar manner to the linlin0 VAA plot above, a VAA plot of $F(x)$ on a loglin0 (log abscissa scale, linear ordinate scale starting at 0) plot is produced if the tally bin value is divided by the difference in the logarithms of the abscissa values. If $y = \ln(x)$, then $G(y)$ is defined to be

$$G(y) = (\text{tally bin value})/(\ln(x_{i+1}) - \ln(x_i)) = (\text{tally bin value})/(y_{i+1} - y_i). \tag{25}$$

A loglin0 plot of $G(y)$ is a VAA plot of $F(x)$ because y is linear on a log abscissa and $G(y)(y_{i+1} - y_i)$ is the area of the tally bin.

The relation between $G(y)$ and $F(x)$ is

$$G(y)|dy| = F(x)|dx|, \tag{26}$$

where

$$dy = d\ln(x) = dx/x. \tag{27}$$

Therefore

$$G(y) = xF(x). \tag{28}$$

This y norming of the tally bin value to make a loglin $G(y)$ plot is equivalent to making a loglin $xF(x)$ plot. Lethargy plotting of an energy-dependent tally F(energy) (see Appendix B, Section III.F of the MCNP5-1.50 Manual) is the equivalent of plotting $G(\ln(\text{energy})) = \text{energy} \times F(\text{energy})$ on a loglin0 scale to produce a VAA plot for the tally F(energy).

These norming statements can be generalized to any function $h(x)$. The VAA interpretation of a 2-D plot therefore depends on the abscissa axis scale. A linlin0 plot of $h(x)$ is a VAA $h(x)$ plot and a loglin0 $h(x)$ plot is a VAA plot for $h(x)/x$. If $h(x)/x$ is a useful quantity, then the VAA plot of $h(x)/x$ is also useful.

2-D plots from a *runtpe* (not a *mctal*) file of the empirical history score probability density function $f(x)$ moments can be made using the Tally Fluctuation Chart (TFC) tally plot commands (see Appendix B, Section III.C.5.c). The tally $F(x) = xf(x)$. From the discussion above, a loglin0 $F(x)$ plot can be interpreted as a VAA plot for $f(x)$; i.e., the area under the curve on a loglin0 scale represents where the $f(x)$ sampling has occurred. A linlin0 $F(x)$ plot can be interpreted as a VAA plot for the tally $F(x)$.

Based on these observations, the following statements can be made about TFC commands to create $f(x)$ moment plots for the TFC bin of a tally (without the NONORM option):

$f(x)$ TFC bin plots are VAA plots when:

1) TFC p [$f(x)$] is on a linlin0 scale; and
2) TFC 1 [$xf(x)$] is on a loglin0 scale.

$xf(x) = F(x)$ TFC bin tally plots are VAA plots when:

1) TFC 1 [$xf(x) = F(x)$] is on a linlin0 scale; and
2) 2 [$x^2 f(x) = xF(x)$] is on a loglin0 scale.

$x^n f(x)$ TFC bin tally moment plots are VAA plots when:

1) TFC n [$x^n f(x)$] is on a linlin0 scale; and
2) TFC n+1 [$x^{n+1} f(x)$] is on a loglin0 scale.

The VAA contributions to the nth $f(x)$ moment can be viewed with an $x^n f(x)$ linlin0 plot or an $x^{n+1} f(x)$ loglin0 plot. The empirical $f(x)$ slope result can be checked by viewing a TFC n plot. If the high-score $f(x)$ tail for a long-tailed distribution (not a finite distribution) is proportional to $1/x^n$, then the TFC n plot will be a statistical constant at the high x scores. A large score $f(x)$ slope of at least n exists if the high-score TFC n [$x^n f(x)$] values are decreasing. VAA $f(x)$ moment plots can be a useful tool in studying the detailed impact of variance reduction techniques on $f(x)$ (not the history sampling time as function of x) and the efficiency of a calculation.

Weight Cutoffs and Forced Collisions with Pulse-Height Tallies

In MCNP, the weight cutoff variance reduction method is on by default, unless a pulse-height tally (F8) is included in the problem. With the introduction of pulse-height tally variance reduction in MCNP5-1.50, in most cases this default behavior (weight cutoffs not employed) is kept; the weight cut game had to be explicitly turned on if it was to be used with pulse-height tallies. The exception to this behavior is if forced collisions are used in the problem. Since this variance reduction technique can produce several very low-weight particles, it was decided to keep the weight cutoff game turned on by default when using pulse-height

tallies and forced collisions. Note that any of these default settings can be overridden by explicitly setting the weight cutoffs on the CUT card.

MCNP5 Use on Multicore Processors

Since its beginning 10 years ago, MCNP5 has fully supported parallel computing using **both** threads (OpenMP) and message-passing (MPI) parallelism. Threading is used on a single compute node (e.g., PC or Mac) with either a single multicore processor or more than one multicore processor (as long as they share memory). Message-passing (MPI) is used for parallelism on clusters or distributed architectures, where processors do not share a common memory. MCNP5 uses a hierarchical parallelism – MPI between nodes and threading within a node.

Only in the last 5 years or so have PCs and Macs appeared with multicore processors. Older versions of MCNP5 for PCs or Macs were of course only single-threaded since there were not any multicore PCs or Macs. Today, almost every laptop or office computer has multicore processors, and threaded executables are provided for MCNP5-1.60 on all platforms.

To complicate the discussion of threading, many of the Intel processors also provide "hyperthreading" on the CPU-cores (see "Multi-Core Programming" from the Intel Press, 2006). The current Intel Core i7 with two CPU-cores also permits two hyperthreads per CPU-core. Since hyperthreads share a CPU-core (the computational unit), hyperthreads do not provide any significant speedup for MCNP5. Running with two threads results in two cpu-cores sharing the work for two MCNP5 computational tasks, while running with four threads results in two cpu-cores sharing four MCNP5 computational tasks. The result is not a speedup, but a slight slowdown because of the overhead of context switching between the four threads. Hyperthreading only seems to help MCNP5 on the Intel Atom processor used in netbooks, where using two hyperthreads for the single CPU compensates for the lack of out-of-order execution features in the CPU architecture.

The default number of threads used by MCNP5-1.60 is one. To use more than one thread when running MCNP5-1.60, users must supply the options "`tasks 2`" or "`tasks 4`" on the MCNP5 execution line to get two or four threads, respectively. For example, to have MCNP5 use four threads, the user would type

```
mcnp5 i=inp-file tasks 4
```

on the execute line.

The number of threads used in a calculation should be chosen to be no larger than the number of CPU-cores, and not the number of hyperthreads. Even on a multicore processor with two CPU-cores and two hyperthreads per core (for a total of four hyperthreads), MCNP5 should be run with "`tasks 2`". If a user is running other applications concurrently with MCNP5, such as web browsers, videos, compilations, etc., it may be preferable to run MCNP5 with just one thread, so that other applications have better response time.

Fig. 3 shows an example of the speedup in MCNP5 execution on a Mac Pro computer with two quad-core Xeon processors. For this example, speedups are excellent in going from one to two to eight MCNP5 threads. In the MCNP5 coding, there are some threading inefficiencies because of lock/unlock or critical sections in the source and tally routines. Problems that have relatively simple geometry and collision physics may spend a greater percentage of time in the source or tally sections, and hence not be as efficient as the problems shown in Fig. 3.

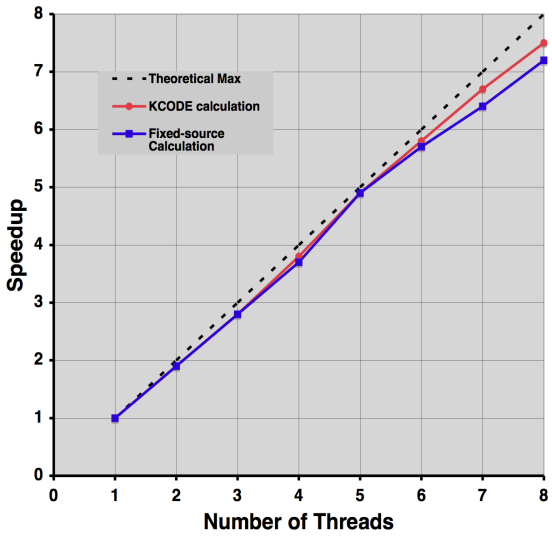

Figure 3: MCNP5-1.60 speedup, on a Mac Pro, versus number of OpenMP threads.