

LA-UR-04-2506

Approved for public release;  
distribution is unlimited.

*Title:* MONTE CARLO PARAMETER STUDIES  
AND UNCERTAINTY ANALYSIS WITH MCNP5

*Author(s):* FORREST B. BROWN, JEREMY E. SWEEZY,  
& ROBERT B. HAYES

*Submitted to:* PHYSOR-2004, American Nuclear Society  
Reactor Physics Topical Meeting,  
Chicago, IL, April 25-29, 2004



Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Form 836 (8/00)

# Monte Carlo Parameter Studies & Uncertainty Analyses With MCNP5

**Forrest B. Brown, Jeremy E. Sweezy**

X-5, Los Alamos National Laboratory

`fbrown@lanl.gov, jsweezy@lanl.gov`

**Robert Hayes**

Radiological Technology, WIPP

`robert.hayes@wipp.ws`

## Monte Carlo Parameter Studies & Uncertainty Analyses with MCNP5

**Forrest B. Brown, Jeremy E. Sweezy (LANL), & Robert Hayes (WIPP)**

A software tool called *mcnp\_pstudy* has been developed to automate the setup, execution, and collection of results from a series of MCNP5 Monte Carlo calculations. This tool provides a convenient means of performing parameter studies, total uncertainty analyses, parallel job execution on clusters, stochastic geometry modeling, and other types of calculations where a series of MCNP5 jobs must be performed with varying problem input specifications.

- **Introduction**
- **mcnp\_pstudy**
- **Examples**
- **Usage**
  - **Parameter definition**
  - **Parameter expansion**
  - **Constraints**
  - **Case setup & execution**
  - **Collecting & combining results**
- **Statistics**
- **Examples**

## How are calculated results affected by:

- **Nominal dimensions**
  - With minimum & maximum values ?
  - With as-built tolerances ?
  - With uncertainties ?
- **Material densities**
  - With uncertainties ?
- **Data issues**
  - Different cross-section sets ?
- **Stochastic materials**
  - Distribution of materials ?

**Monte Carlo perturbation theory can handle the case of independent variations in material density, but does not apply to other cases.**

## **Brute force approach:**

**Run many independent Monte Carlo calculations, varying the input parameters.**

- To simplify & streamline the setup, running, & analysis of Monte Carlo parameter studies & total uncertainty analyses, a new tool has been developed: **mcnp\_pstudy**
- **Control directives are inserted into a standard MCNP input file**
  - Define **lists** of parameters to be substituted into the input file
  - Define parameters to be **sampled from distributions** & then substituted
  - Define **arbitrary relations between parameters**
  - Specify **constraints** on parameters, even in terms of other parameters
  - Specify **repetitions** of calculations
  - Combine parameters as **outer-product for parameter studies**
  - Combine parameters as **inner-product for total uncertainty analysis**
- **Sets up separate calculations**
- **Submits or runs all jobs**
- **Collects results**

- **Completely automates the setup/running/collection for parameter studies & total uncertainty analyses**
  - Painless for users
  - 1 input file & run command can spawn 100s or 1000s of jobs
  - Fast & easy way to become the #1 user on a system  
(Added bonus: make lots of new friends in computer ops & program management.)
- **Ideal for Linux clusters & parallel ASC computers:**
  - Can run many independent concurrent jobs, serial or parallel
  - Faster turnaround: Easier to get many single-cpu jobs through the queues, rather than wait for scheduling a big parallel job
  - Clusters always have some idle nodes

- **mcnp\_pstudy is written in *perl***
  - 640 lines of perl (plus 210 lines of comments)
  - Would have taken many thousands of lines of Fortran or C
- **Portable to any computer system**
  - Tested on Unix, Linux, Mac OS X, Windows
  - For Windows PCs, need to execute under the Cygwin shell
- **Can be modified easily if needed**
  - To add extra features
  - To accommodate local computer configuration
    - Node naming conventions for parallel cluster
    - Batch queueing system for cluster
    - Names & configuration of disk file systems (ie, local or shared)
    - Location of MCNP5 and MCNP5.mpi

## MCNP input for simple Godiva calculation

## MCNP input using *mcnp\_pstudy*, Run 50 different cases - Each with a distinct (odd) random seed

```
gdv
1  -18.74  -1  imp:n=1
2   0      1  imp:n=0

1      so 8.741

kcode 10000  1.0  15  115
ksrc   0 0 0
m1     92235 -94.73  92238 -5.27
prdmp  0 0 1 1 0
```

```
gdv-A
C @@@  RNSEED = ( 2*int(rand(1000000))+1 )
C @@@  xxx    = REPEAT 50
1  -18.74  -1  imp:n=1
2   0      1  imp:n=0

1      so 8.741

kcode 10000  1.0  15  115
ksrc   0 0 0
m1     92235 -94.73  92238 -5.27
prdmp  0 0 1 1 0
rand   seed=RNSEED
```

- Within an MCNP input file, all directives to mcnp\_pstudy must begin with

**c**    **@@@**

- To continue a line, use "\" as the last character

**c**    **@@@**    **XXX = 1    2    3    4    5    6    \**  
**c**    **@@@**                    **7    8    9    10**

- Parameter definitions have the form

**c**    **@@@**    **P =**    *value or list*  
**c**    **@@@**    **P = (**    *arithmetic-expression*    **)**

- Constraints have the form

**c**    **@@@**    **CONSTRAINT = (**    *expression*    **)**

- Control directives have the form

**c**    **@@@**    **OPTIONS =**    *list-of-options*

- Parameters

- Like C or Fortran variables
- Start with a letter, contain only letters, integers, underscore
- Case sensitive
- Parameters are assigned values, either number(s) or string(s)
- Examples: **R1, r1, U\_density, U\_den**

- Single value

**C @@@ P1 = value**

- List of values

**C @@@ P2 = value1 value2 ... valueN**

- List of N random samples from a Normal probability density

**C @@@ P3 = normal N ave dev**

- List of N random samples from a Uniform probability density

**C @@@ P4 = uniform N min max**

- Arithmetic expression

```
C   @@@   P5   = ( arithmetic-statement )
```

- Can use numbers & previously defined parameters
- Can use arithmetic operators **+**, **-**, **\***, **/**, **%** (mod), **\*\*** (exponentiation)
- Can use parentheses **( )**
- Can use functions: **sin()**, **cos()**, **log()**, **log10()**, **exp()**, **int()**, **abs()**, **sqrt()**
- Can generate random number in (0,N): **rand(N)**
- Must evaluate to a single number
- Examples:

```
C   @@@   FACT = normal 1 1.0 .05
```

```
C   @@@   UDEN = ( 18.74 * FACT )
```

```
C   @@@   URAD = ( 8.741 * (18.74/UDEN)**.333333 )
```

- Repetition (list of integers, 1..N)

```
C   @@@   P6   = repeat N
```

# Parameter Definition

- Examples

```
C  rod height in inches, for search
C  @@@  HROD = 5  10  15  20  25  30  35  40  45  50

C  nominal dimension, with uncertainty
C  @@@  X1 = normal  25  1.234  .002

C  dimension, with min & max
C  @@@  X2 = uniform 25  1.232  1.236

C  try different cross-sections
C  @@@  U235 = 92235.42c  92235.49c  92235.52c  \
C  @@@           92235.60c  92235.66c

C  different random number seeds (odd)
C  @@@  SEED = ( 2*int(rand(1000000)) + 1 )
```

## Random Sampling of Parameters

- For parameters sampled from a **Uniform** probability density, each sample is obtained as

$$P = x_{min} + (x_{max} - x_{min}) * \text{rand}()$$

- For parameters sampled from a **Normal** probability density, each sample is obtained using the Box-Muller scheme

$$P = ave + dev * \text{sqrt}(-2 * \log(\text{rand}())) * \sin(2 * \pi * \text{rand}())$$

- Other probability densities could easily be added

## Arithmetic Expressions & Constraints

- Evaluated within **perl**, using the **eval** function
- Must conform to **perl** rules for arithmetic

# Parameter Expansion

- After all parameters are defined, mcnp\_pstudy expands them into sets to be used for each separate MCNP calculation
  - Outer product expansion: All possible combinations. Parameters specified first vary fastest.
  - Inner product expansion: Corresponding parameters in sequence. If not enough entries, last is repeated.

Example:

```
c @@@ A = 1 2
c @@@ B = 3 4
c @@@ C = 5
```

**Outer:**

```
Case 1: A=1, B=3, C=5
Case 2: A=2, B=3, C=5
Case 3: A=1, B=4, C=5
Case 4: A=2, B=4, C=5
```

**Inner:**

```
Case 1: A=1, B=3, C=5
Case 2: A=2, B=4, C=5
```

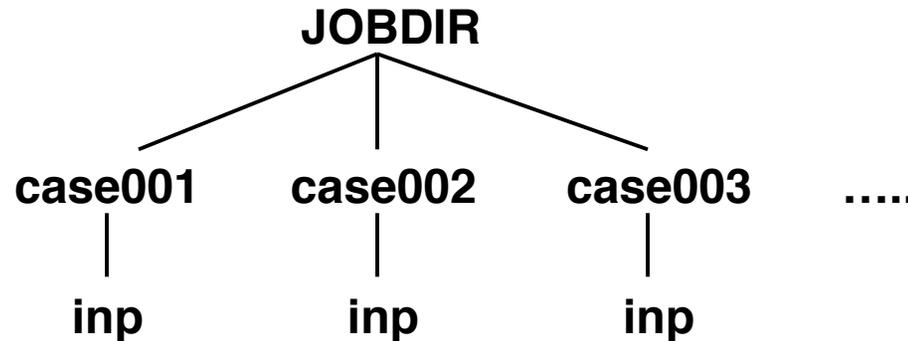
# Constraint Conditions

- After all parameters are defined & expanded, constraint conditions are evaluated
- Constraints involve comparison operators ( >, <, >=, <=, ==, != ) or logical operators ( && (and), || (or), ! (not) ), and may involve arithmetic or functions
- Constraints must evaluate to True or False
- If a any constraint is not met, the parameters for that case are discarded & re-evaluated until all of the constraints are satisfied

## Example

```
C pick a random direction
C @@@ ANGLE = ( 6.2831853 * rand(1) )
C @@@ UUU   = ( cos(ANGLE) )
C @@@ VVV   = ( sin(ANGLE) )
C
C same, using CONSTRAINT to implement rejection scheme
c @@@ RN1 = ( 2.*rand(1) - 1. )
C @@@ RN2 = ( 2.*rand(1) - 1. )
C @@@ CONSTRAINT = ( RN1**2 + RN2**2 < 1.0 )
C @@@ UUU = ( RN1 / sqrt(RN1**2 + RN2**2) )
C @@@ VVV = ( RN2 / sqrt(RN1**2 + RN2**2) )
```

- Directory structure for MCNP5 jobs



- Unix filesystem conventions followed

`JOBDIR/case001/inp`, `JOBDIR/case002/inp`, etc.

- Values of parameters are substituted into the original MCNP5 input file to create the input files for each case
  - Parameters substituted only when exact matches are found
  - Example: **UDEN** matches **UDEN**, and not **UDEN1**, **UDENS**, **uden**

- **Specifying options for running jobs**

- Can be specified on the **mcnp\_pstudy** command-line

```
mcnp_pstudy -inner -setup -i inp01
```

- Within the INP file

```
c @@@ OPTIONS = -inner
```

- **Common options**

<b>-i str</b>	The INP filename is <i>str</i> , default = inp
<b>-jobdir str</b>	Use <i>str</i> as the name of the job directory
<b>-case str</b>	Use <i>str</i> as the name for case directories
<b>-mcnp_opts str</b>	Append <i>str</i> to the MCNP5 run command, may be a string such as 'o=outx tasks 4'
<b>-bsub_opts str</b>	<i>str</i> is appended to the LSF bsub command
<b>-inner</b>	Inner product approach to case parameter substitution
<b>-outer</b>	Outer product approach to case parameter substitution
<b>-setup</b>	Create the cases & INP files for each
<b>-run</b>	Run the MCNP5 jobs on this computer
<b>-submit</b>	Submit the MCNP5 jobs using LSF bsub command
<b>-collect</b>	Collect results from the MCNP5 jobs

## Running or Submitting Jobs

- Jobs can be run on the current system, or can be submitted to a batch queueing system (e.g., LSF)
- Tally results & K-effective can be collected when jobs finish

Examples:

```
bash: mcnp_pstudy -inner -i inp01 -setup
```

```
bash: mcnp_pstudy -inner -i inp01 -run
```

```
bash: mcnp_pstudy -inner -i inp01 -collect
```

```
bash: mcnp_pstudy -inner -i inp01 -setup -run -collect
```

```
bash: mcnp_pstudy -inner -i inp01 -setup -submit
```

```
... wait till all jobs complete...
```

```
bash: mcnp_pstudy -inner -i inp01 -collect
```

# Combining Results

- Tally results & K-effective from separate cases can be combined using batch statistics:

$$\bar{X} = \frac{1}{M} \cdot \sum_{k=1}^M X_k \quad \sigma_{\bar{X}} = \sqrt{\frac{1}{M-1} \cdot \left[ \frac{1}{M} \sum_{k=1}^M X_k^2 - \bar{X}^2 \right]}$$

where **M** is the number of cases & **X<sub>k</sub>** is some tally or Keff for case **k**

- Variance due to randomness in histories decreases as 1/M, but variance due to randomness in input parameters is constant

$$\sigma_{\bar{X}}^2 \approx \sigma_{\bar{X}, \text{ Monte Carlo}}^2 + \sigma_{\bar{X}, \text{ Initial Conditions}}^2$$

**Varies as 1/M**

**~ Constant**

# Examples

Vary the fuel density randomly & adjust radius for constant mass, for 50 cases

Vary fuel density & mass independently, for 50 cases

```
gdv-D
c vary fuel density - normal, 5%sd,
c adjust the radius to keep constant mass
c
c @@@ FACT= normal 50 1.0 .05
c @@@ UDEN= ( 18.74*FACT )
c @@@ URAD= ( 8.741*(18.74/UDEN)**.333333 )
c
1 1 -UDEN -1 imp:n=1
2 0 1 imp:n=0

1 so URAD

kcode 10000 1.0 15 115
ksrc 0. 0. 0.
m1 92235 -94.73 92238 -5.27
prdmp 0 0 1 1 0
```

```
gdv-E
c vary fuel radius - normal, 5%sd
c vary fuel density- normal, 5%sd
c
c @@@ OPTIONS = -inner
c
c @@@ DFACT = normal 50 1.0 .05
c @@@ UDEN = ( DFACT * 18.74 )
c
c @@@ UFACT = normal 50 1.0 .05
c @@@ URAD = ( UFACT * 8.741 )
c
1 1 -UDEN -1 imp:n=1
2 0 1 imp:n=0

1 so URAD

kcode 10000 1.0 15 115
ksrc 0. 0. 0.
m1 92235 -94.73 92238 -5.27
prdmp 0 0 1 1 0
```

# Examples

**Table 1. Results from varying parameters in the Godiva problem**

Problem	Description	K-effective	$\sigma_{K\text{-eff}}$
base	<b>Base case</b> , discard 15 initial cycles, retain 100 cycles with 10K histories/cycle, <b>1M total histories</b>	0.9970	<b>0.0005</b>
A	Repeat the base problem 50 times, <b>50M total histories</b>	0.9972	<b>0.0001</b>
B	<b>Vary the fuel density only</b> : sample from a normal distribution with 5% std.dev, <b>50M total histories</b>	0.9961	<b>0.0061</b>
C	<b>Vary the fuel radius only</b> : sample from a normal distribution with 5% std.dev, <b>50M total histories</b>	1.0057	<b>0.0051</b>
D	<b>Vary the enrichment only</b> , sample from a normal distribution with 5% std.dev, <b>50M total histories</b>	0.9890	<b>0.0027</b>
E	<b>Sample the fuel density from a normal distribution with 5% std.dev, and adjust the fuel radius to keep constant fuel mass, 50M total histories</b>	0.9966	<b>0.0042</b>
F	<b>Sample the fuel density from a normal distribution with 5% std.dev, and independently sample the radius from a normal distribution with 5% std.dev, 50M total histories</b>	1.0073	<b>0.0076</b>

- **Parameter studies**
  - Run a series of cases with different control rod positions
  - Run a series of cases with different soluble boron concentrations
  - Run a series of cases sampling certain dimensions from a Uniform or Normal probability density
  - Run a series of cases substituting different versions of a cross-section
- **Total uncertainty analysis**
  - Run a series of cases varying all input parameters according to their uncertainties
- **Parallel processing using a "parallel jobs" approach**
  - Running N separate jobs with 1 cpu each will be more efficient than running 1 job with N cpus
  - Eliminates queue waiting times while cpus are reserved
  - Take advantage of cheap Linux clusters
- **Simulation of stochastic geometry**
  - Run a series of cases with portions of geometry sampled randomly, with a different realization in each case

- **mcnp\_pstudy works**
  - In use regularly at LANL for a variety of real applications
  - Developed on Mac & PC, runs anywhere
  - Easy to customize, if you have special needs
- **To get it:**
  - MCNP5 website: **[www-xdiv.lanl.gov/x5/MCNP](http://www-xdiv.lanl.gov/x5/MCNP)**

FB Brown, JE Sweezy, RB Hayes, "Monte Carlo Parameter Studies and Uncertainty Analyses with MCNP5", PHYSOR-2004, Chicago, IL (April, 2004)