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
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
Abstract

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.
Outline

• Introduction
• mcnp_pstudy
• Examples
• Usage
  – Parameter definition
  – Parameter expansion
  – Constraints
  – Case setup & execution
  – Collecting & combining results
• Statistics
• Examples
Frequent Questions

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

<table>
<thead>
<tr>
<th>gdv</th>
<th></th>
<th></th>
<th>imp:n=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-18.74</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>imp:n=0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>so</th>
<th>8.741</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>kcode</th>
<th>10000</th>
<th>1.0</th>
<th>15</th>
<th>115</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ksrc</th>
<th>0 0 0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ml</th>
<th>92235</th>
<th>-94.73</th>
<th>92238</th>
<th>-5.27</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>prdmp</th>
<th>0 0 1 1 0</th>
</tr>
</thead>
</table>

### MCNP input using `mcnp_pstudy`, Run 50 different cases - Each with a distinct (odd) random seed

```plaintext
<table>
<thead>
<tr>
<th>gdv-A</th>
<th></th>
<th></th>
<th>imp:n=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C @@ RNSEED = ( 2*int(rand(1000000))+1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C @@ xxx = REPEAT 50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-18.74</td>
<td>-1</td>
<td>imp:n=1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>imp:n=0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>so</th>
<th>8.741</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>kcode</th>
<th>10000</th>
<th>1.0</th>
<th>15</th>
<th>115</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ksrc</th>
<th>0 0 0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ml</th>
<th>92235</th>
<th>-94.73</th>
<th>92238</th>
<th>-5.27</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>prdmp</th>
<th>0 0 1 1 0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>rand</th>
<th>seed=RNSEED</th>
</tr>
</thead>
</table>
```
• Within an MCNP input file, all directives to mcnp_pstudy must begin with
  
  \begin{verbatim}
  C @@@
  \end{verbatim}

• To continue a line, use "\" as the last character
  
  \begin{verbatim}
  C @@@ XXX = 1 2 3 4 5 6 \\
  C @@@ 7 8 9 10
  \end{verbatim}

• Parameter definitions have the form
  
  \begin{verbatim}
  C @@@ P = value or list
  C @@@ P = ( arithmetic-expression )
  \end{verbatim}

• Constraints have the form
  
  \begin{verbatim}
  C @@@ CONSTRAINT = ( expression )
  \end{verbatim}

• Control directives have the form
  
  \begin{verbatim}
  C @@@ OPTIONS = list-of-options
  \end{verbatim}
Parameter Definition

- 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
Parameter Definition

- 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
  \[ P = xmin + (xmax-xmin) \times \text{rand()} \]

- For parameters sampled from a **Normal** probability density, each sample is obtained using the Box-Muller scheme
  \[ P = ave + dev \times \sqrt{-2 \times \log(\text{rand()}) \times \sin(2\pi \times \text{rand()})} \]

- 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
  B @@@  =  3  4
  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 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**

```
JOBDIR
   case001    case002    case003    ..... 
   inp        inp        inp
```

  - 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**
Job Options

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

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

where \( M \) is the number of cases & \( X_k \) 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

<table>
<thead>
<tr>
<th>Vary the fuel density randomly &amp; adjust radius for constant mass, for 50 cases</th>
<th>Vary fuel density &amp; mass independently, for 50 cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdv-D</td>
<td>gdv-E</td>
</tr>
<tr>
<td>c vary fuel density - normal, 5%sd,</td>
<td>c vary fuel radius - normal, 5%sd</td>
</tr>
<tr>
<td>c adjust the radius to keep constant mass</td>
<td>c vary fuel density - normal, 5%sd</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>c @@@ FACT = normal 50 1.0 0.05</td>
<td>c @@@ OPTIONS = -inner</td>
</tr>
<tr>
<td>c @@@ UDEN = ( 18.74*FACT )</td>
<td>c @@@ DFECT = normal 50 1.0 0.05</td>
</tr>
<tr>
<td>c @@@ URAD = ( 8.741*(18.74/UDEN)**.333333 )</td>
<td>c @@@ UDEN = ( DFECT * 18.74 )</td>
</tr>
<tr>
<td>1 1 -UDEN -1 imp:n=1</td>
<td>1 1 -UDEN -1 imp:n=1</td>
</tr>
<tr>
<td>2 0 1 imp:n=0</td>
<td>2 0 1 imp:n=0</td>
</tr>
<tr>
<td>1 so URAD</td>
<td>1 so URAD</td>
</tr>
<tr>
<td>kcode 10000 1.0 15 115</td>
<td>kcode 10000 1.0 15 115</td>
</tr>
<tr>
<td>ksrc 0.0 0.0</td>
<td>ksrc 0.0 0.0</td>
</tr>
<tr>
<td>m1 92235 -94.73 92238 -5.27</td>
<td>m1 92235 -94.73 92238 -5.27</td>
</tr>
<tr>
<td>prdmp 0 0 1 1 0</td>
<td>prdmp 0 0 1 1 0</td>
</tr>
</tbody>
</table>

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

Vary fuel density & mass independently, for 50 cases
### Table 1. Results from varying parameters in the Godiva problem

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

- **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
Conclusions

- **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)