Unclassified **NEA/NSC/DOC(2007)1/REV1**

Organisation de Coopération et de Développement Economiques
Organisation for Economic Co-operation and Development **16-May-2007**

English text only

NUCLEAR ENERGY AGENCY
NUCLEAR SCIENCE COMMITTEE

NEA/NSC/DOC(2007)1/REV1
Unclassified

Benchmarking the Accuracy of Solution of 3-Dimensional Transport Codes and Methods over a Range in Parameter Space

Yousry Y. Azmy
The Pennsylvania State Univeresity, University Park, USA

English text only

JT03227274

Benchmarking the Accuracy of Solution of 3-Dimensional Transport Codes and Methods over a Range in Parameter Space

Approved by the

OECD Nuclear Energy Agency Nuclear Science Committee (NEA-NSC)

Yousry Y. Azmy

The Pennsylvania State University, 229 Reber Building

University Park, PA 16801

yya3@psu.edu

Introduction

Particle transport calculations in three-dimensional configurations are becoming prevalent in nuclear applications. This places a premium on determining the accuracy of numerical solutions to practical configurations as this might have strong implications on the level of confidence in various design and system analysis activities based on the transport equation. Computational benchmark exercises are essential to shed light on this question and raise awareness of additional issues that influence the performance on numerical methods and computer codes when deployed in modeling complex practical applications.

This is the third in a sequence of computational benchmarks organized by the NEA's *Expert Group on 3-D Radiation Transport Benchmarks* aimed at comparing the results of advanced three-dimensional transport methods and codes to high quality reference solutions. *Mathematical* benchmarks are designed to limit the validation exercise to the numerics of the solution method and algorithm, not the quality of the data. Namely, in the previous exercises, as in this proposed one, the cross section data and the full specification of the problem geometry are provided to the participants. This way the observed error in the results reported by the participants is a consequence only of the codes they employ and the underlying approximations made in the implemented numerical method and computational model.

The proposed benchmark is designed to elucidate three important issues necessary to judge the quality of numerical solutions obtained with particle transport software. The geometric configuration and nuclear data (i.e. cross sections) are intentionally simple and unsophisticated to avoid diverting the participants' attention and efforts towards modeling details. Instead, they are required to illustrate their software's performance in view of the following criteria:

1. The vast majority of benchmark exercises, mathematical, computational or physical, specify a single configuration to be solved by the participants. This limits the results of the comparison of the obtained solutions to reference values, and validity of reached conclusions, both general and code/method specific, to that particular configuration. Deviations from the benchmark configuration, a highly likely eventuality for most users, require extrapolation of the learned lessons in an *ad hoc*, subjective, fashion that is difficult to quantify. We propose addressing this issue by requiring solutions to an entire suite of benchmark configurations that cover a wide range in the parameter space spanned by the problem geometry and nuclear data. As detailed below, these cover two orders of magnitude in each relevant parameter permitting users to interpolate, or extrapolate, somewhat objective estimates of a given code's performance in a specific application.

2. Derivation of numerical methods for the transport equation involves discretization of the energy variable into energy groups, the angular variable into discrete ordinates or moments, and the spatial variables over computational cells. Typically, but not necessarily, the solution error decreases with the increased number of discrete variables, and this process is collectively referred to as *model refinement*. The meaningfulness of a numerical solution is confined to the range in model-refinements referred to as the *asymptotic regime* characterized by a monotonic approach to the limit solution with further refinement of the computational model. This follows from the fact that outside the asymptotic regime, the solution can fortuitously possess small errors that unexpectedly increase with model refinement. Since, in general, the user does not know the exact, or reference, solution, such behaviour can lead to erroneous extrapolation of the solution away from the exact value potentially leading to invalid conclusions and deficient design. A straightforward, albeit non-rigorous, numerical procedure to verify that the solutions are obtained within the asymptotic regime involves solving the problem on a sequence of refined models to observe a monotonic approach of the solution to a limit. While this is typically impractical in the majority of applications, it is feasible in a benchmark exercise with the simple geometry proposed here. Since the present case involves fictitious one-group nuclear data, model refinement along the energy variable is irrelevant. In contrast, the participants are free to select the method and level of discretization of the angular and spatial variables, and are required to verify that the reported solutions are within the asymptotic regime near the provided reference solution.

3. Particle transport calculations in three-dimensional configurations are notorious for their difficulty and high demand on computational resources. This has compelled nuclear computational scientists involved in the development and implementation of numerical methods and solution algorithms for the transport equation to pursue novel approaches and schemes to resolve specific difficulties. Once these are successfully tested and verified on simple test codes, they are implemented into major production level codes. Indeed, most production level computer codes employed in the solution of particle transport problems are endowed with a multitude of options and adjustable parameters designed to address a broad variety of difficult problem configurations. In the hands of an expert user, these options and parameters comprise powerful tools that are invaluable in solving the most difficult problems encountered by nuclear scientists and engineers involved in innovative research outside the normal range of operation for traditional systems. Nevertheless, it is rare for a nuclear scientist with the physics expertise necessary to advance the design problem at hand also to possess sufficient expertise in a specific transport code to be able to navigate through the multitude of options and parameter settings. In recognition of this difficulty many code developers set defaults, or provide recommended settings that the non-expert user can start with. Failure to obtain a satisfactory solution then should prompt the user to experiment with the default/recommended setting, but this is often an unguided adventure. Hence, in the proposed benchmark exercise, the participants will be required to state any non-standard settings that are necessary to yield the reported solutions. Hopefully this will help future users understand the available options in a given transport code, and aid their selection of adjustable options and parameters when the defaults do not work satisfactorily.

Description of the Suite of Benchmark Problems

The geometric configuration of the proposed suite of benchmark problems is very simple: two nested parallelepipeds as depicted in Fig. 1, with the origin of the Cartesian coordinate system fixed at the shown corner of the outer one. The outer parallelepiped is referred to with the index 1, while the inner is referenced with the index 2. Parallelepiped 1 is surrounded by vacuum on all six faces, and has a unit square base and height L, while parallelepiped 2 is scaled down by a factor γ, i.e. it has dimensions $\gamma\times\ \gamma\times\ \gamma L$. The total macroscopic cross section and the scattering ratio are denoted σ_i and c_i, $i=1,2$, respectively.

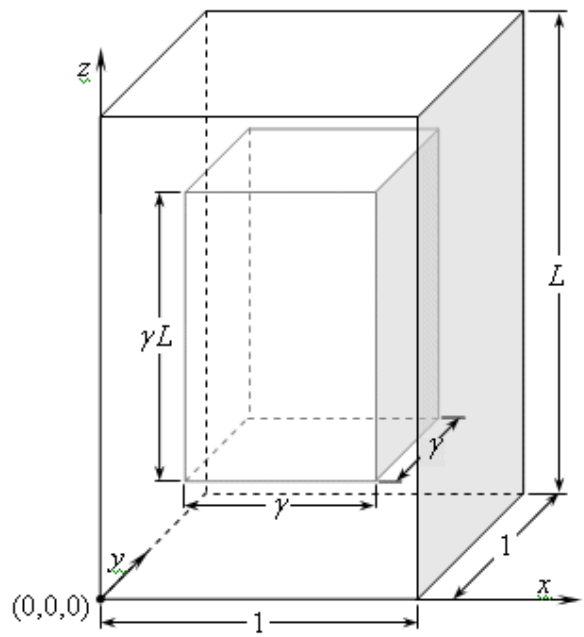

Fig. 1. Geometric Configuration of the Benchmark Problem

A fixed, distributed unit source is located in parallelepiped 1, cornered at the origin and occupying a region of dimensions $(1-\gamma)/2 \times (1-\gamma)/2 \times L(1-\gamma)/2$ as illustrated in Fig. 2.

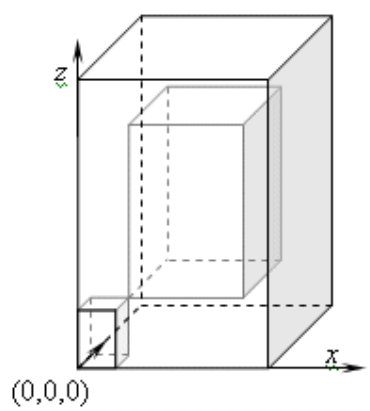

Fig. 2. Location of the Fixed Unit Source

The *suite* of problems is now constructed by independently varying each of the parameters L, γ, σ_i and c_i, $i = 1, 2$. The range for each of these parameters is presented in Table 1. Since each of the designated parameters is permitted three values, the entire suite of benchmark problems is comprised of a total of 729 cases to be executed multiple times in order to verify asymptotic convergence of the reported results. While this is admittedly a very large number of cases to be executed by the participants, the simplicity of the problem configurations should mitigate this burden. Furthermore, we propose that the

organizers of the benchmark implement a shell script template that permit automatic execution of the member configurations in the suite, and automatic collection of the benchmark results for immediate analysis by the participants.

Table 1. Range of Parameters Comprising the Suite of Benchmark Problems

Parameter	Range		
L	0.1	1.0	5.0
γ	0.1	0.5	0.9
σ_1	0.1	1.0	5.0
c_1	0.5	0.8	1.0
σ_2	0.1	1.0	5.0
c_2	0.5	0.8	1.0

Benchmark Quantities

For each run in the collection of benchmark configurations the participants will be required to report:

1. Three model refinement levels of the angular and spatial variables.
2. Any non-default settings of optional methods or adjustable parameters implemented in the employed code.

For each benchmark configuration in the suite, the participants will be required to provide the following information for each benchmark quantity:

1. Three values corresponding to three levels of model refinement.
2. The relative error compared to the reference value provided by the organizers for each level of model refinement
3. An indication of whether the solution is in the asymptotic regime.

The benchmark quantities which the participants will be required to obtain values for are:

1. Scalar flux averaged over:
   1.a. The region in parallelepiped 1 not in 2.
   1.b. Parallelepiped 2.
2. Net leakage out of the following faces:
   2.a. $x = 0,\ y \in [0,1],\ z \in [0, L]$
   2.b. $x = 1,\ y \in [0,1],\ z \in [0, L]$
   2.c. $z = 0,\ x \in [0,1],\ y \in [0,1]$

2.d. $z = L,\ x \in [0,1],\ y \in [0,1]$

2.e. $x = (1-\gamma)/2,\ y \in \left[(1-\gamma)/2, (1+\gamma)/2\right],\ z \in \left[L(1-\gamma)/2, L(1+\gamma)/2\right]$

2.f. $x = (1+\gamma)/2,\ y \in \left[(1-\gamma)/2, (1+\gamma)/2\right],\ z \in \left[L(1-\gamma)/2, L(1+\gamma)/2\right]$

2.g. $z = L(1-\gamma)/2,\ x \in \left[(1-\gamma)/2, (1+\gamma)/2\right],\ y \in \left[(1-\gamma)/2, (1+\gamma)/2\right]$

2.h. $z = L(1+\gamma)/2,\ x \in \left[(1-\gamma)/2, (1+\gamma)/2\right],\ y \in \left[(1-\gamma)/2, (1+\gamma)/2\right]$

3. The scalar flux averaged over the following subregions:

3.a. $x, y \in \left[(1-\gamma)/4, (1-\gamma)/2\right],\ z \in \left[L(1-\gamma)/4, L(1-\gamma)/2\right]$

3.b. $x, y \in \left[(1-\gamma)/2, 1/2\right],\ z \in \left[L(1-\gamma)/2, L/2\right]$

3.c. $x, y \in \left[1/2, (1+\gamma)/2\right],\ z \in \left[L/2, L(1+\gamma)/2\right]$

3.d. $x, y \in \left[(1+\gamma)/2, (3+\gamma)/4\right],\ z \in \left[L(1+\gamma)/2, L(3+\gamma)/4\right]$

3.e. $x \in \left[(3+\gamma)/4, 1\right],\ y \in \left[0, (1-\gamma)/4\right],\ z \in \left[0, L(1-\gamma)/4\right]$

3.f. $x \in \left[(3+\gamma)/4, 1\right],\ y \in \left[(3+\gamma)/4, 1\right],\ z \in \left[0, L(1-\gamma)/4\right]$

3.g. $x, y \in \left[0, (1-\gamma)/4\right],\ z \in \left[L(3+\gamma)/4, L\right]$

3.h. $x \in \left[(3+\gamma)/4, 1\right],\ y \in \left[0, (1-\gamma)/4\right],\ z \in \left[L(3+\gamma)/4, L\right]$

3.i. $x \in \left[(3+\gamma)/4, 1\right],\ y \in \left[(3+\gamma)/4, 1\right],\ z \in \left[L(3+\gamma)/4, L\right]$

3.j. $x \in \left[1/2, (1+\gamma)/2\right],\ y \in \left[(1-\gamma)/2, 1/2\right],\ z \in \left[L(1-\gamma)/2, L/2\right]$

3.k. $x, y \in \left[1/2, (1+\gamma)/2\right],\ z \in \left[L(1-\gamma)/2, L/2\right]$

3.l. $x, y \in \left[(1-\gamma)/2, 1/2\right],\ z \in \left[L/2, L(1+\gamma)/2\right]$

3.m. $x \in \left[1/2, (1+\gamma)/2\right],\ y \in \left[(1-\gamma)/2, 1/2\right],\ z \in \left[L/2, L(1+\gamma)/2\right]$

Annex I

Benchmark Package Description and README

1. 3D-Transport-Accuracy-Benchmark.doc
   This is the present file in computer readable format and describes the benchmark specification, the script for generating the benchmark cases and the approved mandate of the OECD/NEA/NSC ***Expert Group on 3D Radiation Transport Benchmarks***
2. bechrun-1.2 folder of the distribution; includes a shell script that sets up and executes the benchmark cases with multiple options for the user to exercise, in addition to the MCNP reference solution as best we can obtain it so far. The script uses the reference solution to compute errors in the benchmark quantities. A full description of the script is found in the following Readme text included also in the benchrun-1.2 folder of the package (available on CD-ROM – requests should be made to programs@nea.fr). As it was decided to keep the benchmark "blind", the reference results (folder rdpool) will be made available on CD-ROM at a later data to participants having submitted results.

README

```
benchrun-1.2

========
AUTHOR
========
      Kursat B. Bekar <kbb141@psu.edu>
      Department of Mechanical and Nuclear Engineering
      The Pennsylvania State University


======================
COMMAND LINE OPTIONS
======================
     (for details see EXAMPLES: how to run script?)

usage: benchrun [options]

  -a|--all           Run all cases: Script generates all inputs and
                     cross-sections, then executes and post-processes all
                      cases.

  -x|--xsec          Only generate cross-sections.
  -i|--input         Only generate transport code inputs.
  -p|--postprocess   Process transport code outputs.
  -e|--execute       Execute transport code with previously generated inputs.
  -t|--table         Tabulate results in LaTeX table format (requires LaTeX and
                     DVIutils)
  -h|--help          What you're reading

  --single L gamma sigma1 c1 sigma2 c2
                     Perform calculation for a single case with L,gamma,sigma1,
                      c1,sigma2 and c2  values.


  --group [ L=LL gamma=GG s1=SS1 c1=C1 s2=SS2 c2=CC2 ]
                     Perform calculations for a given group of cases. A group
                     can be defined by fixing any subset of the 6 parameters
                     following this option, implying all three values of the
                     parameters not specified are included in the group. Example:

                     --group L=0.1 s1=0.1 c1=0.5 s2=0.1 c2=0.5

                     means that the script fixes the values for L,sigma1,c1,
                     sigma2, and c2 as specified, then uses the three possible
```

```
                    values of gamma: 0.1, 0.5, 0.9 to define a group of cases.

                     If there is no parameter following this option, the actions
                    selected by the -i,-p,-e,... options are performed for all
                    cases. If all 6 parameters are provided with the --group
                    option, it runs only the selected single case (similar to
                    --single option).

  In addition, the user can define the following parameters on the command line:

  --code=[NAME]       Name of the tranport code executable
  --codedir=[DIR]     Directory where the transport code executable resides
  --codetmp=[NAME]    File name for the transport code input template
  --codetmpdir=[DIR] Directory where the input template file resides
  --postp=[DIR/NAME] Full pathname for the post-processor of the transport
                      code output: collect and tabulate data from output file
  --prep=[DIR/NAME]  Full pathname of the pre-processor of the transport
                      code input: generates input to the transport code from
                      input template for specified case(s)
  --rdpool=[DIR]      Directory where the results of the reference MCNP5
                      calculation reside


==============
INTRODUCTION
==============

This Bourne Again Shell script is designed to facilitate participation in
the NEA Benchmark exercise "Benchmarking the Accuracy of Solution of
3-Dimensional Transport Codes and Methods over a Range in Parameter Space".
This exercise comprises a large collection of cases that must be solved and
various benchmark quantities determined from the solution form comparison
with the corresponding reference values as a measure of the target transport
method/code. To reduce the chances of error if a participant in the exercise
were to execute the multitude of cases and extract the required results
manually this script automates:
1. Setup of the input files (per case) for the target transport code.
2. Execution of the target transport code for all cases in the suite.
3. Collection of the benchmark quantities (per case) from the output file(s).
4. Tabulation of the benchmark quantities for documentation or archiving.

The script is composed with the intent to provide the participant in the
Benchmark exercise with a template suitable for their preferred target
transport code. Hence, the user must edit the script and modifiy it to make
it use the target transport code, with all peripheral requirements, e.g.
cross sections library. This might require the user to implement a code-
specific postprocessor to collect the necessary data from the output file,
to be followed by this script's tabulation capability.

In addition to enabling the execution of all cases included in the Benchmark
suite, the shell enables the execution of individual cases and ranges of
cases. In addition it serves several other functions that the user might
find worthwhile during their participation in the Benchmark exercise.

The script has a modular structure that should make its modification simple.
In order to maintain modularity, two aspects are important to elaborate:
the naming convention for the various cases in the suite of benchmarks,
and the structure/format of some files manipulated by the script.

=================
NAME GENERATION
=================

In the suite of Benchmark problems 6 parameters are varied and each is
allowed to take one of three values thus comprising all 3^6 cases comprising
the suite. The following convention is used in naming files employed by the
script, where each file name is composed of a "prefix" and an "extension":

- The prefix denotes the specific case in the suite that this file belongs to.
It is an integer whose 6-digits are 1,2, or 3 corresponding to the value of
each of the 6 parameters:
```

```
  PREFIX=abcdef

where
a indicates index of L      {1,2,3} => {0.1,1.0,5.0.}
b indicates index of gamma  {1,2,3} => {0.1,0.5,0.9}
c indicates index of sigma1 {1,2,3} => {0.1,1.0,5.0}
d indicates index of c1     {1,2,3} => {0.5,0.8,1.0}
e indicates index of sigma2 {1,2,3} => {0.1,1.0,5.0}
f indicates index of c2     {1,2,3} => {0.5,0.8,1.0}

- The extension denotes the contents of the file; the three possible
extensions are:
i    => input for the target transport code
o    => output from the target transport code
data => data processed from the output file(s)

Example:

   123121  =>  L=0.1, gamma=0.5, sigma1=5.0, c1=0.5, sigma2=1.0, c2=0.5

   input file  : 123121.i
   output file : 123121.o
   data file   : 123121.data

Retaining this naming convention allows the user to re-execute the transport
code with a previously generated input file, e.g. tighten iterative
convergence, or post-process previosly executed outputs of the transport
code to extract the benchmark quantities and tabulate them. In addition,
this enables the automatic execution of a single case or a group of cases.

======================
INPUT FILE STRUCTURE
======================
The input file must have the required structure by the target transport code.
Three examples of input files are included in this distribution corresponding
to three of the MCNP code cases executed in the process of generating the
reference solutions for the entire Benchmark suite. Normally the user would
provide this script with a template for the input file abiding by the format
required by the target transport code, then the script will modify the
template and generate input files for all cases in the suite.

=======================
OUTPUT FILE STRUCTURE
=======================
The structure of the output file(s) depends on the target transport code and
the selected options. Three examples of MCNP output files, corresponding to
the three sample input files mentioned above, are included in this
distribution. Since the format and structure of these output files might
vary drastically across transport codes, it is not possible to compose a
single, generic procedure to extract all the Benchmark quantities. Instead,
the user must implement a transport code-specific "extractor" to collect the
desired Benchmark quantities from the output file(s), write them to the
corresponding .data file in the fixed format described below for subsequent
post-processing.

=====================
DATA FILE STRUCTURE
=====================
In order to enable the table generation function of the script, the data
file generated by the transport code-specific "extractor" from the output
file(s) must possess the following structure:


--------------------------------------------------
username        : USERNAME
date            : DATE of post-processing
pwd             : WORKING DIRECTORY
input file name : INPNAME, input filename
--------------------------------------------------
Parameters:
```

```
-------------------------------------------------
L         : L  value of this case
Gamma     : GAMMA value of this case
Sigma1    : SIGMA1 value of this case
c1        : C1   value of this case
Sigma2    : SIGMA2 value of this case
c2        : C2 value of this case
-----------------------------------
          R E S U L T S
-----------------------------------
 Item       Value
-----------------------------------
1.a     x.xxxxxEsxx             [ shown format is E12.5 ]
1.b     x.xxxxxEsxx
-----------------------------------
2.a     x.xxxxxEsxx
...
3.m     x.xxxxxEsxx
```

Strictly speaking, the header lines are for the user's benefit to maintain execution records and ensure back-tracking. Hence the user may change the format of the header lines and/or add/delete information to the suggested list. All that matters for the postprocessing module provided with this script is that each case's results be included in a file with the naming convention described above, and that each benchmark quantity be preceded with its identifier, e.g. "1.a", "1.b", etc. The postprocessing module searches for these strings and reads the data appearing following the string as the value of that benchmark quantity for the case identified by the name of the input file. The user is cautioned, therefore, that the parameter values listed in this file have no bearing on the postprocessing activiy. Only the file name and the real numbers listed behind the identifier of each benchmark quantity are consequential for the correct functioning of the postprocessing module.

Recall that the .data file is written by the user-implemented extractor, hence it is the user's responsiblity to ensure that their implementation abides by the specified standard. In such case, the script should successfully read all data in the file and prepare it in LaTeX table format. If the computer has DVIUTILS and PSUTILS packages installed, then it can also generate the PDF output of this table.

```
=====================================
CONSTANT AND USER DEFINED PARAMETERS
=====================================
```

The script has two parameter groups:

1. Constant parameters: These are defined and used to determine benchmark cases, to check consistency of cases given by user, etc.

```
     L_cdata="0.1 1.0  5.0"
 Gamma_cdata="0.1 0.5  0.9"
Sigma1_cdata="0.1 1.0  5.0"
    C1_cdata="0.5 0.8  1.0"
Sigma2_cdata="0.1 1.0  5.0"
    C2_cdata="0.5 0.8  1.0"
```

2. User defined parameters: These are defined by the user either in the script, via edits, or interactively on the command line executing script; for details see the OPTIONS section below.

```
CODEDIR     : directory where the code executable resides
CODE        : name of executable
CODETMPDIR  : directory where the code input template resides
CODETMP     : name of code input template
POSTP       : post-processing code with its full path
PREP        : pre-processing code with its full path
RDPOOL      : directory where the reference MCNP results reside
```

```
==========================
SOME IMPORTANT VARIABLES
==========================
```

1. Control variables:

```
These parameters are set by the script to 0 or 1 in some functions
according to the options selected by the user on the command line
and the availablity of the requested processes.

   do_input : input generation          (Enable=1,disable=0)
   do_exec  : code execution            (Enable=1,disable=0)
   do_postp : post-processing           (Enable=1,disable=0)
   do_xsec  : cross-section generation  (Enable=1,disable=0)
   do_table : table generation          (Enable=1,disable=0)
   do_single: perform single case       (1 on, 0 off)
   do_group : perform a group of cases  (1 on, 0 off)
   do_latex : process LaTeX table       (1 on, 0 off)
   do_dvips : convert table DVI -> PS   (1 on, 0 off)
   do_ps2pdf: convert table PS -> PDF   (1 on, 0 off)

2. List of case parameters:
In addition to enabling the execution of all cases included in the Benchmark
suite, the shell enables the execution of individual cases and ranges of
cases. In order to avoid running unrequested case(s), the script saves a list
of the case parameters requested to the following variables:

         L       list of lengths along z
        Gamma   list of aspect ratios
         Sigma1  list of total cross sections for the 1st material
         C1      list of scattering ratios for the 1st material
         Sigma2  list of total cross sections for the 2nd material
         C2      list of scattering ratios for the 2nd material

Each list holds the three possible values of the benchmark parameters as
 stated in the CONSTANT and USER DEFINED PARAMETERS  section above.

===================
MODULES of SCRIPT
===================

The script has two groups of functions:

1. Utility functions:
   ------------------
Some utility functions were written and are used in other functions.
  a) Function decho: This function writes messages (Warnings/Error/Info)
     to the STDIO and to a message file.
  b) Function generate_input_filename: This function generates a unique filename
     for the given case to provide modularity; see NAME GENERATION section.
  c) Function generate_XS_ID1: Generate a unique ID for the first material using
     sigma1 and c1 values; the ID is used in generating the input file as
     elaborated below.
  d) Function generate_XS_ID2: Generate a unique ID for the second material using
     sigma2 and c2 values.
  e) Function initialize_message: Initializes the message file before running
     script.
  f) Function usage: Prints the cmmand line options to the screen if either any
     option is wrong or help is requested.
  g) Function check_options: Checks consistency of the command line options and
     sets the control variables and the list of cases requested.
  h) Function check_udp: Checks consistency of the user defined parameters, and
     availability of the codes given.

2. Modules:
   --------
Modules are also functions designed to perform the following actions (detailed
below):

  a) generate_inputs: produces input file for a case from the template
  b) execute_cases: executes the target transport code for selected cases
  c) postp_cases: extract benchmark quantities from transport code output
  d) generate_XS: produce cross sections library
  e) make_tables: tabulate benchmark quantities from .data file(s)
  f) merging_data_files: merge multiple .data files

Modules a,b, and c have the following general structure:
```

```
        loop l in L
          loop gamma in Gamma
            loop sigma1 in Sigma1
           loop c1 in C1
               loop sigma2 in Sigma2
              loop c2 in C2
              ...
              --> generate some data for the case
                       --> generate input file name
              ################################
              #BEGIN:MODIFICATION
                      #
                       action(input_generation, execution, or post_processing)
                      #
                      #END:MODIFICATION
                      ################################
                       ...
              end loop c2
            end loop sigma2
           end loop c1
          end loop sigma1
        end loop gamma
      end loop l
```

Data is generated in these loops and the code/script corresponding to a particular module (a, b, or c) is executed. In order to accommodate different transport codes a fixed interface structure is necessary between each module (a, b, or c) and the user-selected codes. If the codes provided by the user to (a) generate input file, (b) execute transport code, or (c) postprocess output file are compatible with this fixed interface structure, then the script can be used directly in conjunction with these codes. Otherwise, the user must modify the section of the modules between the labels "BEGIN:MODIFICATION" and "END:MODIFICATION" according to the interface structure of their codes.

a) Input generation:
For any given case, input generator for the transport codes needs the following data from the script:

(1) input file name
(2) material ID/name of two materials
(3) input template filename & pathname
(4) boundary (x,y,z extent) of the problem
(5) boundary (x,y,z extent) of the fixed source region
(6) boundary (x,y,z extent) of the sub-regions

In this module, after generating the input file name, material IDs are generated for the two materials involved in the selected configuration in terms of their sigma1,c1 and sigma2,c2 values.

By using the values of variables l and gamma, all the necessary boundary points are calculated in the x,y,z direction and are used to define items 4,5,6.

By preparing an input template suitable for the target transport code, and using the above parameters with this template, the input processor generates input files for the given cases. Additional parameters required by the target transport code, e.g. convergence criterion, must be inserted by the user in the input template. Only geometric configuration and material assignment input values are manipulated by modeule generate_inputs (for details see EXAMPLES:input template).

The interface block between the input generation module and the preprocessor code is as follows:

```
(1) inpname   : input file name
(2) ID1       : ID/name of the first material
(3) ID2       : ID/name of the second material
(4) CODETMPDIR: directory where input template file resides
(5) CODETMP   : input template filename
(6) x vector  : 7 elements
```

```
(7) y vector  : 7 elements
(8) z vector  : 7 elements

The user-defined parameter PREP holds the full pathname of the
pre-processing code (input generator). By executing this code with the
necessary command line arguments input file(s) are  generated for the
given case(s) determined by the loops indices.

b) Code execution:
   For any given case, this module executes the target transport code with the
   input corresponding to that case in a file previously generated. First the
   script checks the existence and accessibility of the input file then it
   it executes the transport code.

   The user-defined parameters CODEDIR and CODEXEC hold the name and path of
   the transport code in the file system.  The user must modify the command
   line according to the target transport code needs, e.g. command-line
   arguments, and to save the output from the transport calculation in a
   file with a legtimate name to enable the post-processing function.

c) Post-processing:
   This module post-processes the output of the transport code as saved in
   a previously generated output file.

   The user has to provide the name of the output file of the transport code,
   as well as other parameters, for proper operation of this module.

   If the user employs the following file structure for the .data files
   generated by their own post-processor code, the table generation and data
   merging modules that are part of this script, see below, correctly read
   the data and tabulate the results:

            ---------------------------------------------------
            username        : USERNAME                 (optional)
            date            : DATE of post-processing  (optional)
            pwd             : WORKING DIRECTORY        (optional)
            input file name : INPNAME, input filename  (optional)
            ---------------------------------------------------
            Parameters:
            ---------------------------------------------------
            L         : L  value of this case
            Gamma     : GAMMA value of this case
            Sigma1    : SIGMA1 value of this case
            c1        : C1   value of this case
            Sigma2    : SIGMA2 value of this case
            c2        : C2 value of this case
            -----------------------------------
                      R E S U L T S
            -----------------------------------
             Item       Value
            -----------------------------------
            1.a     x.xxxxxEsxx  format(E12.5)
            1.b     x.xxxxxEsxx
            -----------------------------------
            2.a     x.xxxxxEsxx
            ...
            3.m     x.xxxxxEsxx


   The interface block between the input generation module and the preprocessor
   code is as follows:

   (1) inpname   : input file name
   (2) ID1       : ID/name of the first material
   (3) ID2       : ID/name of the second material
   (4) x vector  : 7 elements
   (5) y vector  : 7 elements
   (6) z vector  : 7 elements

   The user-defined parameter POSTP holds the full pathname of the
   post-processing code. By executing this code with the necessary command
```

line arguments the transport code output files are processed for the given case(s) determined by the loops indices.

d) Cross-section generation:
Usually the cross-section library preparation is completed seperately from the transport code execution. All cross-sections are prepared and saved in a file, then this file is used by the transport code. If the user replaces the cross-section preparation code/algorithm with an alternative algorithm, they can still generate the cross-sections by means of this script.

e) Table generation:
One of the primary difficulties with presenting results of this suite of benchmarks is the large number of cases it comprises. In order to present all/some of the results of the suite in a readable form this module was designed. In this module, all data in the .data file is read if and only if it has a proper format described above, then it is processed to produce a LaTeX table. If the post-processor code output format is incompatible with the format given in section "DATA FILE STRUCTURE", the data file cannot be read and table cannot be generated.

In addition, this module prepares same table in comma seperated format for use by MS-Excel or other applications.

This module merges results from three different data files corresponding to the same case with three different aspect ratios in a table. The format of the table is as follows:

```
##table structure
     caption=>    ..................................  L=..
                  sigma1=.., c1=.., and sigma2=.., c2=..
  ---------------------------------------------------------------------------
   item         gamma=0.1              gamma=0.5               gamma=0.9
           value     (%error)     value     (%error)      value    (%error)
  ---------------------------------------------------------------------------
   1.a   x.xxxxxEsxx  xxx.xx    x.xxxxxEsxx  xxx.xx    x.xxxxxEsxx  xxx.xx
   1.b   x.xxxxxEsxx  xxx.xx    x.xxxxxEsxx  xxx.xx    x.xxxxxEsxx  xxx.xx
  ---------------------------------------------------------------------------
   2.a   x.xxxxxEsxx  xxx.xx    x.xxxxxEsxx  xxx.xx    x.xxxxxEsxx  xxx.xx
   ...
   3.m   x.xxxxxEsxx  xxx.xx    x.xxxxxEsxx  xxx.xx    x.xxxxxEsxx  xxx.xx
  ---------------------------------------------------------------------------
##END - table structure
  (formats: E12.5 and F6.2)
```

The postprocessing module of this script calculates and prints the "% error" in each benchmark quantity computed by the target transport code relative to the reference MCNP value provided in the reference data directory (RDPOOL).

f) Merging data files:
This module merges results from three different data files corresponding to the same case with three different aspect ratios in a table. Table is prepared both in comma seperated format for use by MS-Excel or other applications and in text format. The format of the table is as follows:

```
##table structure
    L=.., sigma1=.., c1=.., and sigma2=.., c2=..

 item   gamma=0.1      gamma=0.5     gamma=0.9
         value          value         value

 1.a,   x.xxxxxEmxx,  x.xxxxxEmxx,  x.xxxxxEmxx    [ format = 1pe11.5 ]
 1.b,   x.xxxxxEmxx,  x.xxxxxEmxx,  x.xxxxxEmxx

 2.a,   x.xxxxxEmxx,  x.xxxxxEmxx,  x.xxxxxEmxx
 ...
 3.m,   x.xxxxxEmxx,  x.xxxxxEmxx,  x.xxxxxEmxx
```

```
##END - table structure

     This module is called by the post-processing module to merge into one file
     all data that has been post-processed in the working directory.



===========
 EXAMPLES
===========

  How to run script?
  ------------------
  Script can be run without any command line arguments if the user defined
  parameters were set/modified in the script.

  ./benchrun

  # for cross-section generation:

  ./benchrun -x

  # for input generation:

  ./benchrun -i

  # for execution of previously generated inputs:

  ./benchrun -e

  # for post-processing of transport code output for previously executed case:

  ./benchrun -p

  # for table generation of existing .data files

  ./benchrun -t

  # The user can specify two or more options simultaneously to perform multiple
    actions in a reasonable order, e.g. input generation precedes execution,
    the post-processing.

  ./benchrun -i -e -p

  or

  ./benchrun -p -t


  # The script enables the user to perform some of the above actions for a
    single case.

    The option --single for the script achieves this purpose.

    For example the command line:
    ./benchrun --single 0.1 0.5 5.0 0.5 1.0 0.5 -i

    generates the input file for the case:
    L=0.1, gamma=0.5, s1=5.0, c1=0.5, s2=1.0, c2=0.5

    Similarly, the command line:
    ./benchrun --single 0.1 0.5 5.0 0.5 1.0 0.5 -i -p -e

    generates input, executes the target transport code and post-processes the
    outputs file for the indicated case

    The -t (table) option is incompatible with the --single option.

  # The script enables the user to perform some of the above actions for a set
    of cases.
```

```
    The option --group for the script achieves this purpose.

    For example the command line:
    ./benchrun --group L=0.1 gamma=0.5 s1=5.0 c1=0.5 s2=0.1 -i

    generates three input files for the set of cases:
    L=0.1, gamma=0.5, sigma1=5.0, c1=0.5 , sigma2=0.1, c2=0.5,0.8,1.0

    Similarly, the command line:
    ./benchrun --group L=0.1 Gamma=0.5 s1=5.0 c1=0.5 s2=0.1 -p -e

    executes then postprocesses the set of cases indicated above.

  # User-defined paramters can be set on the command line. For example:

    ./benchrun --codetmp=nea.tmp --codetmpdir=/tmp/erase --group L=0.1 -i

    is equivalent to editing the script and setting the parameters as
    CODETMP=nea.tmp and CODETMPDIR=/tmp/erase, then executing the script
    with the command line:
    ./benchrun  --group L=0.1 -i


  Example of an input file template:
  ----------------------------------
  We include the input file template for MCNP5. It uses the mesh tally
  feature to compute the scalar fluxes averaged over the defined subregions
  in the benchmark problem. Also, the multigroup option is
  enabled for the purposes of this benchmark exercise. Our goal is to
  calrify the structure of the template and illustrate how it is modified
  by the generate_inputs module to produce the input files for specific cases.

  In the input template, many input parameters required by the target transport
  code are fixed. The keyword @REPLACE marks lines containing input parameters
  that must be modified to produce an input file containing input parameters
  corresponding to the specific case to be executed. Following the @REPLACE
  another keyword indicates what input parameter to replace the string
  @REPLACE:KEYWORD with. The input processor follows certain rules to apply
  these replacements using the data provided by the pre-processing module of
  the benchrun script.

##############################################################################
   NEA Benchmark template
   c cell declarations
   1 1 1.00  -1  #2 imp:n=1      $ parallelpiped 1
   2 2 1.00  -2     imp:n=1      $ parallelpiped 2
   3   0.00   1     imp:n=0      $ outside

   c surface declarations
   1  rpp @REPLACE:BODY1
   2  rpp @REPLACE:BODY2
   c

   c
   c problem parameters
   c
   mode n
   nps @REPLACE:NOH
   c
   c material declarations
   c
   m1 @REPLACE:MT1  -1
   m2 @REPLACE:MT2  -1
   c
   c source declaration
   c
   sdef cell=1 pos=0 0 0 x=d1 y=d2 z=d3
   si1 @REPLACE:SX
   sp1 0. 1.0
   si2 @REPLACE:SY
```

```
   sp2 0. 1.0
   si3 @REPLACE:SZ
   sp3 0. 1.0
   mgopt f 1
   c
   c
   c tallies
   c
   f4:n   1 2                $ tallies for 1.a and 1.b
   f11:n  1.1 1.2 1.3 1.4 1.5 1.6    $ tallies for 2.a, 2.b, 2.c and 2.d
   c11 0. 1.0
   f21:n  2.1 2.2 2.3 2.4 2.5 2.6    $ tallies for 2.e, 2.f, 2.g and 2.h
   c21 0. 1.0
   c
   c scalar fluxes averaged over the defined subregions
   c
   fmesh14:n   @REPLACE:3.a  $tally for 3.a
   fmesh24:n   @REPLACE:3.b  $tally for 3.b
   fmesh34:n   @REPLACE:3.c  $tally for 3.c
   fmesh44:n   @REPLACE:3.d  $tally for 3.d
   fmesh54:n   @REPLACE:3.e  $tally for 3.e
   fmesh64:n   @REPLACE:3.f  $tally for 3.f
   fmesh74:n   @REPLACE:3.g  $tally for 3.g
   fmesh84:n   @REPLACE:3.h  $tally for 3.h
   fmesh94:n   @REPLACE:3.i  $tally for 3.i
   fmesh104:n  @REPLACE:3.j  $tally for 3.j
   fmesh114:n  @REPLACE:3.k  $tally for 3.k
   fmesh124:n  @REPLACE:3.l  $tally for 3.l
   fmesh134:n  @REPLACE:3.m  $tally for 3.m
##############################################################################
```

To illustrate the functionality of the input-generator "imckbb" included in this distribution, following is the input file, named 111111.i, that it prepares from the above template for the case:

L=0.1 Gamma=0.1 S1=0.1 C1=0.5 S2=0.1 C2=0.5

```
##############################################################################
NEA Benchmark template
c cell declarations
1 1 1.00  -1  #2 imp:n=1      $ parallelpiped 1
2 2 1.00  -2     imp:n=1      $ parallelpiped 2
3   0.00   1     imp:n=0      $ outside

c surface declarations
1  rpp 0.0 1.0 0.0 1.0 0.0 0.1
2  rpp 0.45 0.55 0.45 0.55 0.045 0.055
c

c
c problem parameters
c
mode n
nps 1.e7
c
c material declarations
c
m1 1003.10m  -1
m2 1003.10m  -1
c
c source declaration
c
sdef cell=1 pos=0 0 0 x=d1 y=d2 z=d3
si1 0.0 0.45
sp1 0. 1.0
si2 0.0 0.45
sp2 0. 1.0
si3 0.0 0.045
sp3 0. 1.0
```

```
mgopt f 1
c
c
c tallies
c
f4:n   1 2                $ tallies for 1.a and 1.b
f11:n  1.1 1.2 1.3 1.4 1.5 1.6    $ tallies for 2.a, 2.b, 2.c and 2.d
c11 0. 1.0
f21:n  2.1 2.2 2.3 2.4 2.5 2.6    $ tallies for 2.e, 2.f, 2.g and 2.h
c21 0. 1.0
c
c scalar fluxes averaged over the defined subregions
c
fmesh14:n  ORIGIN=0.225 0.225 0.0225 IMESH=0.45 JMESH=0.45 KMESH=0.045 $tally for 3.a
fmesh24:n  ORIGIN=0.45 0.45 0.045 IMESH=0.5 JMESH=0.5 KMESH=0.05  $tally for 3.b
fmesh34:n  ORIGIN=0.5 0.5 0.05 IMESH=0.55 JMESH=0.55 KMESH=0.055  $tally for 3.c
fmesh44:n  ORIGIN=0.55 0.55 0.055 IMESH=0.775 JMESH=0.775 KMESH=0.0775 $tally for 3.d
fmesh54:n  ORIGIN=0.775 0.0 0.0 IMESH=1.0 JMESH=0.225 KMESH=0.0225  $tally for 3.e
fmesh64:n  ORIGIN=0.775 0.775 0.0 IMESH=1.0 JMESH=1.0 KMESH=0.0225  $tally for 3.f
fmesh74:n  ORIGIN=0.0 0.0 0.0775 IMESH=0.225 JMESH=0.225 KMESH=0.1  $tally for 3.g
fmesh84:n  ORIGIN=0.775 0.0 0.0775 IMESH=1.0 JMESH=0.225 KMESH=0.1  $tally for 3.h
fmesh94:n  ORIGIN=0.775 0.775 0.0775 IMESH=1.0 JMESH=1.0 KMESH=0.1  $tally for 3.i
fmesh104:n  ORIGIN=0.5 0.45 0.045 IMESH=0.55 JMESH=0.5 KMESH=0.05  $tally for 3.j
fmesh114:n  ORIGIN=0.5 0.5 0.045 IMESH=0.55 JMESH=0.55 KMESH=0.05  $tally for 3.k
fmesh124:n  ORIGIN=0.45 0.45 0.05 IMESH=0.5 JMESH=0.5 KMESH=0.055  $tally for 3.l
fmesh134:n  ORIGIN=0.5 0.45 0.05 IMESH=0.55 JMESH=0.5 KMESH=0.055  $tally for 3.m
##############################################################################
```

Upon executing MCNP on the input file above, it produces the two files:
output file   : 111111.o
mesh tally file: 111111.m

The post-processor code "pmckbb" included in this distribution reads both output files, obtains the necessary data from them to compute the benchmark quantities, then writes these to the data file "111111.data". These results are illustrated as follows:

```
##############################################################################

-------------------------------------------------------------------
username       : kbb
date           : Wed Feb 21 07:10:18 EST 2007
pwd            : /research1/NEA-Benchmark/NEA-2B/2Azmy/package-new-1.1
input file name: 111111.i
-------------------------------------------------------------------
Parameters     :
-------------------------------------------------------------------
L              : 0.1
Gamma          : 0.1
Sigma1         : 0.1
C1             : 0.5
Sigma2         : 0.1
C2             : 0.5
----------------------------------
          R E S U L T S
----------------------------------
 Item       Value
----------------------------------
 1.a     1.33269e+00
 1.b     7.85628e-01
----------------------------------
 2.a     -9.09779e-02
 2.b     9.74575e-03
 2.c     -4.41191e-01
 2.d     3.50766e-01
 2.e     5.88300e-04
 2.f     4.25550e-04
```

```
 2.g     8.98850e-04
 2.h     1.20850e-03
---------------------------------
 3.a     6.79136e+00
 3.b     1.10047e+00
 3.c     5.69566e-01
 3.d     2.27892e-01
 3.e     1.87467e-01
 3.f     8.87461e-02
 3.g     2.68040e+00
 3.h     1.82032e-01
 3.i     8.76687e-02
 3.j     7.38238e-01
 3.k     5.95117e-01
 3.l     1.06293e+00
 3.m     7.28418e-01

##############################################################################


  The post-processing module calls the data file-merging module after
  processing the outputs files for the selected cases. A sample data file
  generated by module merge_data_files is shown as follows based on the data
  files 111111.data 121111.data and 131111.data included in this
  distribution:



##############################################################################
Results for the cases with  L=0.1  sigma1=0.1 c1=0.5 and sigma2=0.1 c2=0.5

Item           gamma=0.1                gamma=0.5                gamma=0.9
          Value   (% Error)         Value    (% Error)        Value    (% Error)

 1.a,   1.33269e+00,    0.01,   1.23257e+00,     0.01,   1.57273e+00,     0.00
 1.b,   7.85628e-01,    0.12,   3.64969e-01,     0.25,   3.40076e-01,     0.10
 2.a,  -9.09779e-02,    0.13,  -1.23824e-01,     0.11,  -1.92668e-01,     0.07
 2.b,   9.74575e-03,    0.08,   7.10965e-03,     0.20,   5.39705e-03,     0.11
 2.c,  -4.41191e-01,    0.01,  -4.44076e-01,     0.01,  -4.45850e-01,     0.01
 2.d,   3.50766e-01,    0.04,   2.88550e-01,     0.03,   1.54879e-01,     0.03
 2.e,   5.88300e-04,    1.14,   9.34835e-03,     0.12,   4.11083e-02,     0.12
 2.f,   4.25550e-04,    1.62,   3.07465e-03,     0.45,   4.75830e-03,     0.14
 2.g,   8.98850e-04,    1.40,   3.70860e-03,     0.18,   4.98510e-03,     0.22
 2.h,   1.20850e-03,    0.02,   1.59653e-02,     0.39,   7.63719e-02,     0.02
 3.a,   6.79136e+00,    0.05,   2.20345e+01,     0.05,   5.51320e+02,     0.05
 3.b,   1.10047e+00,    0.32,   7.99943e-01,     0.01,   1.09173e+00,     0.15
 3.c,   5.69566e-01,    1.07,   1.56064e-01,     0.51,   7.59116e-02,     1.56
 3.d,   2.27892e-01,    0.20,   7.50361e-02,     3.91,   4.01892e-02,     1.09
 3.e,   1.87467e-01,    1.49,   1.14942e-01,     0.81,   9.12466e-02,    17.72
 3.f,   8.87461e-02,    0.28,   5.43536e-02,     1.44,   4.07519e-02,     9.93
 3.g,   2.68040e+00,    0.06,   4.35567e+00,     0.05,   7.83570e+00,     0.43
 3.h,   1.82032e-01,    0.15,   1.13538e-01,     0.64,   6.66546e-02,    14.61
 3.i,   8.76687e-02,    0.26,   5.38505e-02,     1.48,   3.89634e-02,     3.90
 3.j,   7.38238e-01,    0.37,   2.61921e-01,     0.34,   1.44664e-01,     0.08
 3.k,   5.95117e-01,    3.21,   1.55149e-01,     1.48,   7.69956e-02,     0.63
 3.l,   1.06293e+00,    1.24,   7.61350e-01,     0.57,   9.00189e-01,     0.00
 3.m,   7.28418e-01,    0.83,   2.60677e-01,     0.02,   1.43118e-01,     0.06


##############################################################################


=======================
FILES IN THIS PACKAGE
=======================

         - benchrun      Bourne Again Shell script
        - Readme        a brief manual benchrun (this document)
        - Changelog     modifications from version 1.0 to 1.2
        - pmckbb        post-processor module
        - imckbb        pre-processor (input file generation) module
```

```
        - nea.tmp       input file template for MCNP
        - kbbXS,kbbxsd  cross-section files for MCNP

        - message                              sample message file produced by benchrun
        - 111111.i,121111.i,131111.i           sample MCNP inputs
        - 111111.o,121111.o,131111.o           sample MCNP outputs
        - 111111.m,121111.m,131111.m           sample MCNP mesh tally files
        - 111111.data,121111.data,131111.data  sample result files
        - data.txt                             sample data file merged
        - data.xls                             sample data file merged for MS-Excel
        - table_sample.pdf                     sample table file
        - table_sample.xls                     sample table file for MS-Excel

        - rdpool/       directory where reference MCNP results of all cases
                        reside (contains 4x729 files)
========
AUTHOR
========
      Kursat B. Bekar <kbb141@psu.edu>
      Department of Mechanical and Nuclear Engineering
      The Pennsylvania State University

=============
BUG REPORTS
=============
If you find a bug in "benchrun" or any component of this package, please report
it to the author. Comments and bug reports concerning this package should be
directed to kbb141@psu.edu
```

Annex II

Expert Group on 3D Radiation Transport Benchmarks

Approved MANDATE

Scope

The Expert Group deals with scientific issues in the field of deterministic and stochastic methods and computer codes relative to three dimensional radiation transport. Applications encompass transport through large and complex shields including void regions and highly heterogeneous reactor cores. Methods include discrete ordinates, nodal transport, finite elements with spherical harmonics, collision probabilities, Monte Carlo, etc.

Objectives

- to develop benchmarks and comparison exercises for 3D radiation transport computer codes
- to carry out validation of methods and identify their strength, limitations and accuracy
- to suggest needs for method development

Deliverables

- "Benchmarking the Accuracy of Solution of 3-dimensional Transport Codes and Methods over a Range in Parameter Space", Report summarizing comparative analysis, conclusions and lessons learned.

Schedule

- Start: March 2007.
- Meeting: 2008
- End: March 2009

Co-ordinator / Chair: Prof. Yousry Azmy, PSU, USA

Completed Benchmarks

- **3D Radiation Transport Benchmarks for Simple Geometries with Void Regions** (Kobayashi Benchmarks), Index of Special Issue of Progress in Nuclear Energy" , Vol. 39, Number 2, 2001
- Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation - A 2-D/3D MOX Fuel Assembly Benchmark, OECD/NEA Nuclear Science ISBN 92-64-02130-6, NEA/NSC/DOC(2003)16
- Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation - MOX Fuel Assembly 3-D Extension Case, OECD/NEA Nuclear Science ISBN 92-64-01069-6, NEA/NSC/DOC(2005)16

Annex III

Benchmarking the Accuracy of Solution of 3-Dimensional Transport Codes and Methods over a Range in Parameter Space

Participation Form

(to be sent to Yousry Azmy yya3@psu.edu and Enrico Sartori sartori@nea.fr
by 30 June 2007

I wish to participate in this benchmark study with the following computer code(s):

Code name(s): Method(s):

Given & Family Name: Organisation: Address: e-mail:

Comments: