The Monte Carlo Method: Versatility Unbounded In A Dynamic Computing World
Chattanooga, Tennessee, April 17–21, 2005, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2005)

AUTOMATED CONVERSION OF KENO INPUT FILES TO MCNP INPUT FILES AND COMPARISON OF KENO AND MCNP RESULTS

Koji Sugawara[*]
Japan Patent Office
3-4-3 Kasumigaseki, Chiyoda-ku, Tokyo, 100-8915 Japan
sugawara-koji@jpo.go.jp

Rizwan-uddin
Department of Nuclear, Plasma and Radiological Engineering
University of Illinois
103 S. Goodwin Ave., Urbana, IL 61801 USA
rizwan@uiuc.edu

ABSTRACT

We report the development of a program, called *KtoMTran*, that converts KENO input files to MCNP input files. The code is based on parser and lexical analyzers Bison and Flex for translation. It has been tested with the sample cases that are provided with the KENO code as part of the SCALE package by RSICC. The input file conversion program still lacks capability to identify and translate certain special features in KENO input files. However, twenty-six out of the thirty-two input files for KENO sample problems are converted correctly and do not need any manual "fixing" before they can be run with MCNP. Five cases required minor modifications due to special boundary conditions. Since the translator program is not yet capable of handling criticality problems with a hole at the center of the geometry, it is not able to translate one such case in the KENO package. We also report a comparison of results of some selected problems obtained using KENO and MCNP.

Key Words: *KENO, MCNP, geometry, input file, translator*

1 INTRODUCTION

KENO [1] and MCNP [2, 3] are well-known Monte Carlo simulation codes. Input files for both are in text format. However, the input file formats for the two programs are not compatible, and hence a separate input file must be developed for each code to solve the same problem. Hence, having developed an input file for one, it is desirable to have a "translator code" that would translate, for example, a KENO input deck to an MCNP input deck, and vice versa. Even if the converted input file still needs to be examined and "touched up" before it can be used with the other program, the *zeroth order approximation* – automatically carried out by a computer program – will be quite useful. Availability of such a code might even encourage users of one code to compare results obtained with those obtained using the other code. In this paper we present a file conversion program, *KtoMTran* (KENO to MCNP Translation) that converts

[*] This work was carried out at the University of Illinois during a sabbatical as a visiting scholar.

KENO input files to MCNP input files. The program is tested with the sample programs that accompany the KENO code in the SCALE package from RSICC.

Below, after reviewing KENO and MCNP input file formats, we describe the translator program to convert KENO V.a input files to MCNP5 input files. Finally, we test the conversion program by converting some KENO input files to MCNP input files, solving the problems using the two codes, and comparing results. These samples use several geometric features available in KENO. A variety of criticality experiments, including a *2 x 2 x 2* array bare reactor, a *2 x 2 x 2* array with reflector, infinite cylinder, infinite array of slab, water reflected sphere, triangular pitched array, etc., are tested. A comparison of results obtained using the two codes (KENO and MCNP) is also presented.

2 REVIEW OF KENO AND MCNP INPUT FILES

As described above, input for both KENO and MCNP is in the form of text files. We first manually developed MCNP input decks for fourteen of the thirty-two sample KENO problems. It was based on a combination of line-by-line conversion of the KENO input deck and using problem description itself. The exercise provided MCNP input decks for comparison with the MCNP input decks created by the translator program. For completeness, we here briefly summarize the KENO and MCNP input file formats. Input decks for both codes have geometric, material and simulation parameters. Since geometry modeling is the most challenging part of the translator code, it is described first in detail.

2.1 Review of Geometry part of KENO and MCNP input files

Summary of KENO V.a input format

Model geometry in KENO input file is in the form of nested basic building blocks, starting from the innermost, moving outward. Building blocks are one of the following: rectangular parallelepiped, cuboid, sphere (full/hemi), and finite size cylinder (full/semi). Each building block has it's own origin. Each line contains information about a building block. A set of building blocks makes a unit. Each unit can contain other units, and units can touch but cannot intersect. KENO V.a can handle array of units. Geometry part of the KENO input file for a simple example, shown in Fig. 1, is given in List 1. The model

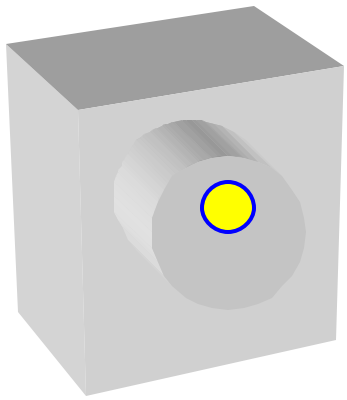

Fig.1. A simple geometry including a cuboid, cylinder and a sphere.

contains a sphere inside a cylinder, which is inside a cuboid. Both, the sphere and the cylinder have offsets from the center of the cuboid.

List 1: KENO geometry description for model in Fig.1

```
read geometry
sphere 1 1 1.0 orig 3.0 -1.0 6.0
zcylinder 0 1 6.0 9.0 1.0 orig -1.0 0.0
cuboid  2 1 10.0 -10.0 10.0 -10.0 10.0 0.0
end geometry
```

In this list, line 2 represents the region of the innermost sphere, line 3 represents the cylinder minus the sphere, and line 4 represents the cuboid minus the cylinder and sphere. Each line has geometry keyword, material ID, calculation bias, position parameters, and optional parameters. In this case, sphere and cylinder have *'orig'* option to offset the origin. Moreover, there is only one unit in the problem, and hence there is no need for a unit declaration. [In Appendix C, a KENO input file for a problem with 5 types of units placed in a *2 x 2 x 2* array is given. Each unit begins with 'box type' and ends with next unit declaration.]

Summary of MCNP input format

There are two major components: surfaces; and cells. Each surface is defined by its type (plane, cylinder, etc) and its position in *global* coordinate system. Surfaces are used to form cells. Each cell in the problem is defined by a logical combination (*and*, *or*) of surfaces and directions (inward or outward face of the surface). For example, a cuboid is defined by six planes. In addition, generic models of commonly used shapes, like cuboid, rectangular parallelepiped, finite size cylinder, etc, are available. Replication function with rotation and parallel translation of a region is also available. Geometry part of the MCNP input file for the model shown in Fig. 1 is given in List 2. The line beginning with 'c' is a comment line (optional). In List 2, the upper part contains cell definitions and lower part contains surface definitions. In the list, upper block (lines 2 through 5) declares cells for each material. Each line begins with a cell ID, followed by a mixture ID, density (omitted if void), list of surfaces, and other parameters. Cell 1 (line 2) corresponds to the region inside the sphere (surface 10), where 10 is the surface ID, defined in the last line of List 2. Cell 2 is the inside of the cylinder (-9 6 7) [and outside of the sphere (-10)]. Finite height cylinder is represented by infinite cylinder and the top and bottom planes. Cell 3 is the inside of the cuboid except for the cylinder (#2) and sphere (#1). The '#' and cell ID is used to exclude other cells. Cell 4 is the outside of the cuboid. Different from KENO, each cell declaration in MCNP contains surfaces or cells to exclude its inner contents explicitly. Also, MCNP input files have a cell definition for the outside (rest of the world) with importance set equal to zero.

The lower block declares each surface. Each line contains surface ID, surface keyword, and parameter(s). In the block, 'px', 'py' and 'pz' are infinite planes perpendicular to each axis. 'c/z' indicates infinite cylinder parallel to the z-axis, and 's' means sphere. Figure 2 shows a detailed correspondence of geometry description for the sample problem in KENO and in MCNP input files.

List 2: MCNP geometry description for model in Fig.1

```
c cell card
1 1 -18.76 -10 imp:n=1
2 0 -9 6 -7 10 imp:n=1
3 2 -0.99791 1 -2 3 -4 5 -8 #2 #1 imp:n=1
4 0 -1:2:-3:4:-5:8 imp:n=0

c surface card
1 px -10
2 px 10
3 py -10
4 py 10
5 pz 0
6 pz 1
7 pz 9
8 pz 10
9 c/z  -1 0 6
10 s 3 -1 6 1
```

2.2 Rest of the input files (material properties and simulation parameters)

In addition to geometric description, both KENO and MCNP input files contain information about materials and simulation parameters. Complete KENO and MCNP input files for problem shown in Fig. 1 are given in Appendix A and B, respectively. The material mixture parts for the two codes are fairly similar. However, simulation parameters for the two codes are quite different.

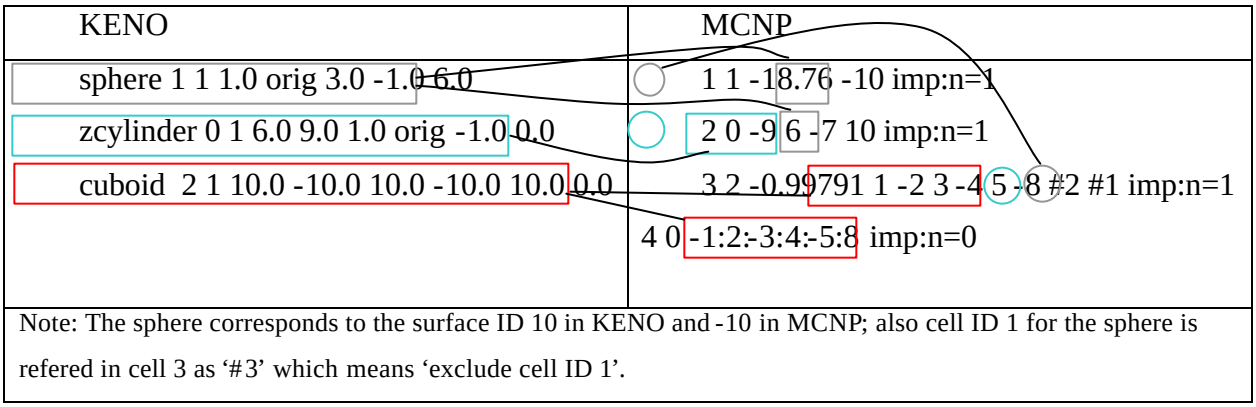

Fig. 2 Correspondence of geometric parameters in KENO and MCNP input files

3 KTOMTRAN

It is obvious that the geometry conversion, that includes translating building blocks to cells and surfaces and translating from local coordinate system in KENO to a global coordinate system for MCNP, is relatively the most difficult part of the conversion program. In fact, if it is not to be done automatically by a computer code, it is probably easier to develop an MCNP input file from scratch using the problem description, than via a line-by-line conversion from a KENO input deck. Since writing a translator code to convert MCNP input files to KENO input files is a much more challenging task, at this stage we have focused on a translator to convert KENO input files to MCNP input files.

There are three parts in the KENO input deck that need conversion: geometry; material; and simulation options.

3.1 Geometry Conversion

Geometry conversion from KENO to MCNP is accomplished in three steps:

1) Break down each basic building block in the unit definition into surfaces and their combination.
2) Calculate the position in global coordinate system for each unit and each surface, starting from outer to inner units, recursively.
3) Write out surface description and cell description for MCNP.

In KENO, all nested units have their own origin and they are placed in the outer unit with offsets. On the other hand, MCNP uses global coordinates for all surfaces. So converting the input file from KENO to MCNP is an exercise in coordinate translations (additions). In KENO

when two units touch each other, two or more surfaces may refer to the same plane. In MCNP, input files with such duplicate surfaces may cause an error. This type of error is likely to occur in a line-by-line conversion of a KENO file. It rarely occurs when writing an MCNP input file from scratch. To avoid this error, the program checks surface duplication when assigning actual location of each surface, and reuses the same surface number when a duplication is found.

3.1.1 Structure and implementation of *KtoMTran*

Since both, KENO and MCNP input files are text-files, we used parser and lexical analyzer Bison [4] and Flex [5] for translation. Bison and Flex generate C (or C++) source codes from a source file that defines rules and behavior. The behavior part is written in C language. These are popular GNU products, compatible with *yacc* [6] and *lex* [6], used to create text file processing programs like computer language compilers (e.g. cc). While a compiler makes binary executable program (or assembler source) from a text source file, *KtoMTran* creates an MCNP input (text) file from a KENO input (text) file. To develop and test, we used Cygwin [7], GNU and UNIX tools running under Microsoft Windows. It is also used to plot the MCNP results for Windows.

The lexical analyzer first analyzes the input file and cuts out "tokens." The parser then parses the stream of tokens with rules and creates internal data. The internal data thus generated consists of a tree of units and building blocks, a list of surfaces and arrays, and a list of materials. Building blocks in KENO are nested in a unit from outside to inside. A unit may contain two or more units by using an array or a "hole" feature. The nested feature leads the internal data to be in the form of a tree. After scanning the input file, the surfaces and their coordinates corresponding to each building block are noted. For example, 6 planes for a cuboid, 1 infinite cylinder and 2 planes for a finite cylinder, etc., are created as surface description. Units and arrays form a tree and refer to other units and arrays. The program determines the root of the tree, i.e. the outermost unit. Because KENO has several ways to designate the outermost unit both explicitly and implicitly, the program determines the root according to a priority rule.

The program then determines the dimensions of all units and arrays. KENO's geometry looks like a stack of units, each with its own origin (see Fig. 3). On the other hand, MCNP uses a global coordinate system. The translation hence, requires the offset of each unit's origin from the global origin. To determine the offset, the dimension of each stacked unit is needed. The program inquires the dimension from the root, recursively. If unit's dimension has not been known because it contains array or replication, the program inquires the dimension of the member parts recursively. The offset calculation from global origin to each unit is now complete.

The program then writes cell and surface definitions for MCNP. To write the surface definition, the program calculates the location of each 'placed' unit and all surfaces. Because MCNP surface definition uses global coordinate system to locate each surface, the program first assigns (0,0,0) as global origin to the outermost unit. Then it calculates the offset from the global origin for each inner unit recursively using the dimension calculated earlier. For each unit, cell and surface definitions are generated. Each cell is represented as the region of the building block, except for the region of the building blocks inside. For example, in Fig. 4, the cell for cuboid is represented as "cuboid – zcylinder". The innermost building block (sphere in the sample

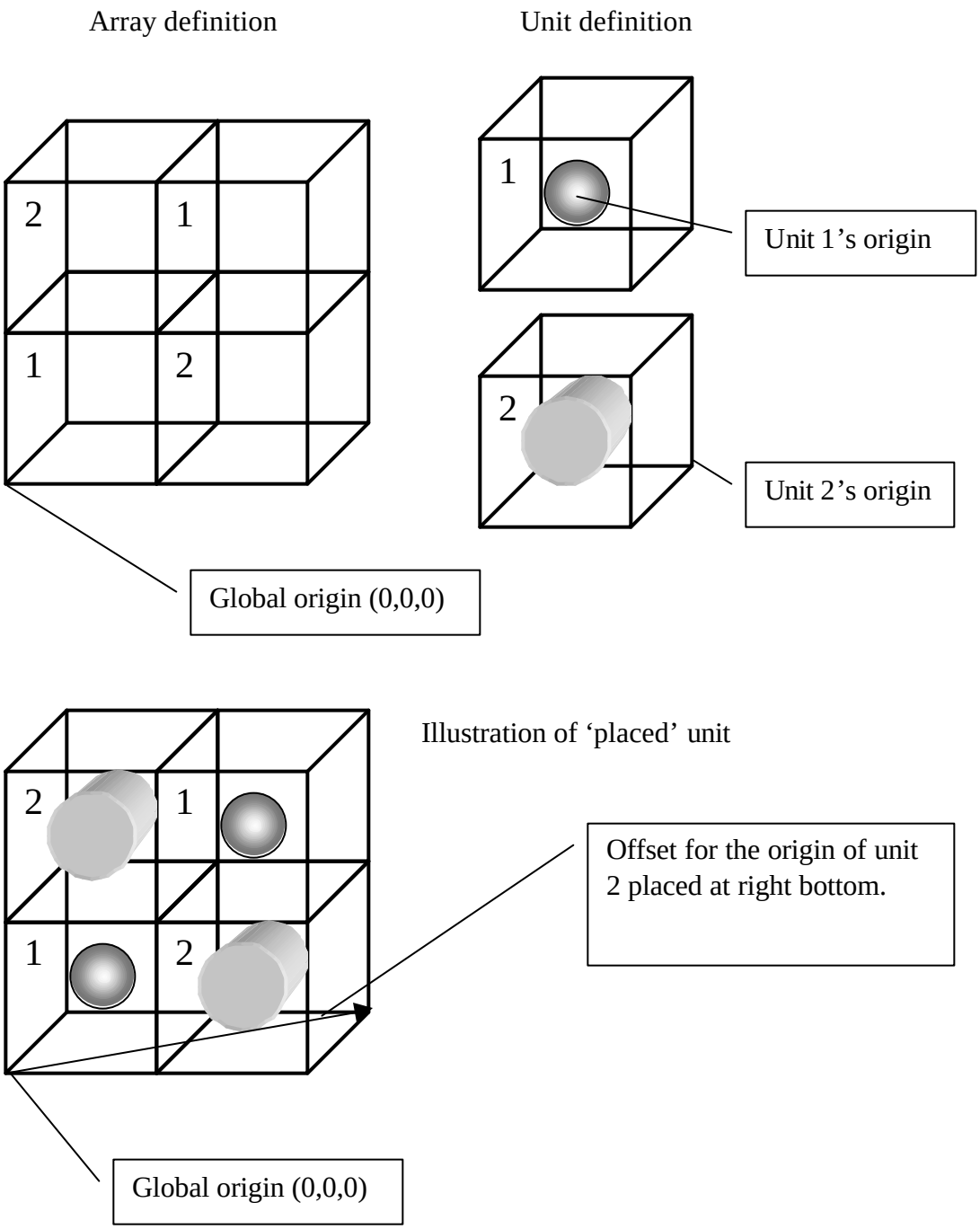

Fig. 3. Schematic diagram of arrays and units; and local and global coordinates.

geometry) is represented as the region itself. The surface definition comprises the type of surface and associated parameters. The program calculates the coordinates associated with the surface with the offset from global origin.

The program then checks duplication of surfaces (type and location). If two or more units use the same surface, only one such surface is created for MCNP. For all surfaces, a unique surface ID is assigned.

Figure 4 shows the internal geometry data diagram for the sample problem shown in Fig.1. There is only one unit (implicit). The null building block in Fig. 4 is the reverse of the outermost building block and used for void cell 4 (*the rest of the world*).

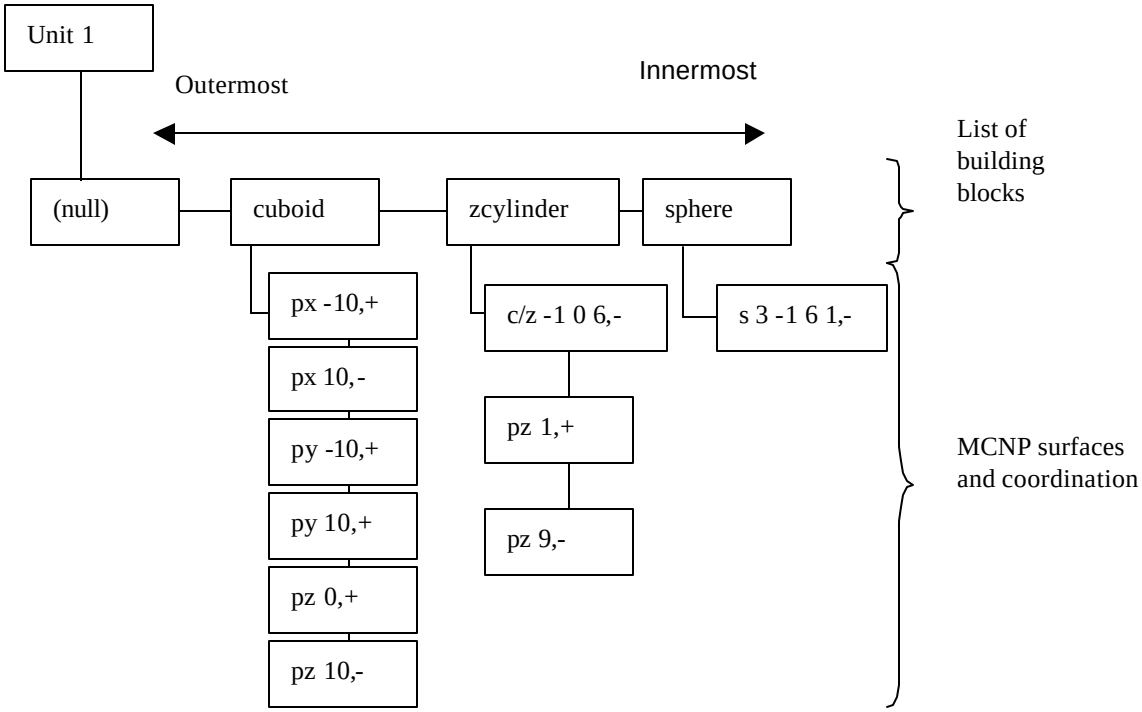

Fig. 4 Internal data diagram for the sample problem shown in Fig.1

3.2 Material and Simulation Parameters

Material description is similar in both KENO and MCNP input files, and is easy to convert. Translated material description (for MCNP) is first written in a separate temporary file.

Simulation options or run time parameters are different in KENO and MCNP. However, choosing these options for MCNP – for which default parameters can always be very easily selected – is not difficult (when compared with geometry development). We ignore the run time simulation options provided in the KENO deck. Instead, the default set is chosen for MCNP simulations [3]. These can be relatively easily changed by the user.

This completes the separate identification of geometry, material and simulation parameters for the MCNP run. Finally, three files of cells, surfaces, and materials are combined into one file, and "default" MCNP options are added for the simulation parameters [3]. An example of a KENO input file and the corresponding translated MCNP input files are given in Appendix C and D.

3.3 Some Limitations of *KtoMTran*

Source point description is required for MCNP criticality calculations. In KENO, it is optional. So, at this point we have added a simple source point search feature: If the innermost unit contains fissile material, the program adds the center of the region (cell) to the list of source points. Consequently, if the input file contains only a hollow unit; i.e., a unit with a hole at its center (donut shape), the translator program fails to accurately generate MCNP input file for such a geometry. Moreover, replication features of both semi-cylinder and hemisphere are not implemented.

[A Comment on *MtoKTran*: Developing a "translator" to convert MCNP input files to KENO input files is significantly more complicated. It would need a pattern matching technique to find basic building blocks of KENO from the set of surfaces provided in MCNP input deck, and it would need restriction and error detection features to limit the cells to fit into the KENO format.]

4 RESULTS

The *KtoMTran* program is tested with thirty-two of the thirty-three sample input files that accompany the KENO V.a package. [One of the input file is a demonstration of the "restart" feature of KENO and contains no geometry data.] All the input files are converted using *KtoMTran*. Only one – a criticality problem with a hole at the center of the geometry – failed to convert. Five of the thirty-two MCNP input files generated by KtoMTran needed slight modifications. Two of these have albedo-reflector and auto-reflector boundary conditions using implicit materials, and the other three use mirror boundary conditions. These modifications can be easily made by hand in the computer generated MCNP input file. Remaining twenty-six cases ran without any difficulty. All thirty-one cases – twenty-six needing no human intervention and five requiring minor modification of the computer generated MCNP input file – lead to results similar to those obtained using KENO. A comparison of keff for all cases that did not require any correction to the computer generated MCNP input file is given in Table 1. These cases are run using a script that translates the KENO input file to the MCNP input file, and runs both KENO and MCNP to solve each of the twenty-six cases. Note that differences in KENO and MCNP results are partly due to different simulation parameters used in the two codes.

5 SUMMARY

A computer code, based on lexical analyzer, has been written to translate KENO input files to MCNP input files. Though there are some limitations, the code has been tested with a large number of KENO input files. These were accurately translated to MCNP input files. Results obtained using KENO and those obtained using MCNP (with input files generated using the translator code developed here) have also been compared.

ACKNOWLEDGMENTS

This project was supported in part by the *Innovations in Nuclear Infrastructure and Education* (INIE) grant from the US Department of Energy.

REFERENCES

1. L. M. Petrie, N. F. Landers, *KENO V.a: AN IMPROVED MONTE CARLO CRITICALITY PROGRAM WITH SUPERGROUPING*, Oak Ridge National Laboratory, Oak Ridge, TN (2000).

2. X-5 Monte Carlo Team, MCNP — A General Monte Carlo N-Particle Transport Code, Version 5 Volume II: User's Guide, Los Alamos National Laboratory, Los Alamos, NM (2003).

3. Charles D. Harman, II, Robert D. Busch, Judith F. Briemeister, R. Arthur Forster, *Criticality Calculation with MCNP: Primer*, Los Alamos National Laboratory, Los Alamos, NM (1984)

4. http://www.gnu.org/software/bison/bison.html

5. http://www.gnu.org/software/flex/

6. John Levine, Tony Mason, Doug Brown, *lex & yacc, 2nd Edition (A Nutshell Handbook)*, O'Reilly Media Inc., Sebastopol, CA. (1992)

7. http://www.cygwin.com/

	Keff-KENO	Keff-MCNP		**Table 1**
#	average k-eff	col/abs/ trk len	notes	Description
1	1.00059	0.9944		case 2c8 bare
2	1.00059	0.9944		2c8 bare with 8 unit types matrix calculation
3	1.00287	1.01485		2c8 15.24 cm paraffin refl
4	1.00738	1.0137	*	2c8 15.24 cm paraffin refl automatic refl
5	1.02374	1.01763	*	2c8 12 inch paraffin albedo reflector
6	0.747901	0.74411		one 2c8 unit (single unit)
7	1.00111	0.9957	*	bare 2c8 using specular reflection
8	0.941021	0.93876	*	infinitely long cylinder from 2c8 unit
9	2.28604	2.25805	*	infinite array of 2c8 units
10	1.00059	0.9944		case 2c8 bare write restart
12	1.00281	1.00152		4 aqueous 4 metal mixed units
13	0.999279	0.99287		two cuboids in a cylindrical annulus
14	0.99734	0.9917		u metal cylinder in an annulus
15	1.00185	1.02083		small water reflected sphere on plexiglas collar
16	0.989058	0.98893	*	uo2f2 infinite slab k-infinity
17	1.00351	0.98625		93% uo2f2 solution sphere adjoint calculation
18	1.01058	1.03717		1f27 demonstration of options problem
19	1.00348	1.00065		4 aqueous 4 metal array of arrays (samp prob 12)
20	0.999139	0.99351		triangular pitched array
21	0.991288	0.97693		samppartially filled sphere
22	1.00215	0.99415		case 2c8 bare with 3 nested, equal volume holes
23	0.999209	0.99178		case 2c8 bare as mixed zhemicylinders
24	1.00044	0.99371		case 2c8 bare as mixed xhemicylinders
25	0.998998	0.99117		case 2c8 bare as mixed yhemicylinders
26	0.999632	0.99178		case 2c8 bare as mixed zhemicylinders with origins
27	1.00273	0.99371		case 2c8 bare as mixed xhemicylinders with origins
28	0.99807	0.99117		case 2c8 bare as mixed yhemicylinders with origins
29	0.995883	0.99402		bare critical sphere 3.4420" radius
30	0.992751	0.99721		bare critical sphere z hemisphere model 3.4420" radius
31	0.992751	0.99378		bare critical sphere x hemisphere model 3.4420" radius
32	0.992751	0.99386		bare critical sphere y hemisphere model 3.4420" radius
			*	using reflecter function

APPENDIX A

Complete KENO input file for model in Fig.1 *(Comments are in the right column.)*

```
#kenova                                                          Program name
                                                                 
sample problem toy problem for describe geometry                 title
read parameters                                                  parameters
lib=4 flx=yes fdn=yes                                            library
npg=1000 gen=203 nsk=3                                           number of calculations
                                                                 
end parameters
read mixt  sct=2                                                 Mixture definition
                                                                 
mix=1 1092234 4.82716e-04 1092235 4.47971e-02 1092236
9.57231e-05
1092238 2.65767e-03
mix=2 12001001 6.67514e-2 12008016 3.33757e-2
end mixt
read geometry                                                    Geometry definition
                                                                 
sphere 1 1 1.0 orig 3.0 –1.0 6.0
zcylinder 0 1 6.0 9.0 1.0 orig -1.0 0.0
cuboid  2 1 10.0 -10.0 10.0 -10.0 10.0 0.0
end geometry
end data
End
```

APPENDIX B

Complete MCNP input file for model in Fig.1 *(Comments are in the right column.)*

```
Sample toy problem                          Title of the problem
c cell card                                 Comment
1 1 -18.76 -10 imp:n=1                      Cell definitions
2 0 -9 6 -7 10 imp:n=1                      Id, material , density (omit if void), surfaces,
                                            calculation importance
3 2 -0.99791 1 -2 3 -4 5 -8 #2 #1           # indicates excluding the other cell
imp:n=1
4 0 -1:2:-3:4:-5:8 imp:n=0                  ':' indicates "OR" logical operation

c surface card                              Comments
1 px -10                                    Surface definitions
2 px 10                                     Planes perpendicular to x/y/z axis
3 py -10
4 py 10
5 pz 0
6 pz 1
7 pz 9
8 pz 10
9 c/z  -1 0 6                               Cylinder along with Z axis
10 s 3 -1 6 1                               Sphere (center position and radius)

c data card                                 Comments
m1 92234 4.82716e-4 92235 4.47971e-2        Material definitions
92236 9.57231e-5
       92238 2.65767e-3
m2 1001 6.67514e-2 8016 3.33757e-2
kcode 1000 1.0 15 115                       k-effective calculation parameter
ksrc 3 -1  6                                Source point(s)
```

APPENDIX C

(Sample KENO input file from KENO manual)

```
#kenova
sample problem 12 4 aqueous 4 metal mixed units
read param
  lib=4 fdn=yes nub=yes smu=yes mkp=yes
mku=yes fmp=yes fmu=yes end param
read mixt  sct=2
mix=1 1092234 4.82716e-04 1092235 4.47971e-02 1092236 9.57231e-05
      1092238 2.65767e-03
mix=2 2001001 5.77964e-02  2007014 2.13092e-03  2008016 3.74130e-02
      2092234 1.06784e-05  2092235 9.84599e-04  2092236 5.29385e-06
      2092238 6.19413e-05
mix=3 11001001 5.68187e-02  11006012 3.55117e-02  11008016 1.42047e-02
end mixt
read geom
box type 1
cylinder  2 1 9.525 8.89 -8.89
cylinder  3 1 10.16 9.525 -9.525
cuboid  0 1 10.875 -10.875 10.875 -10.875 10.24 -10.24
box type  2
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 6.59 -15.16 6.225 -14.255
box type  3
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 15.16 -6.59 6.225 -14.255
box type  4
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 6.59 -15.16 14.255 -6.225
box type  5
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 15.16 -6.59 14.255 -6.225
end geom  read array  nux=2 nuy=2 nuz=2  loop
1 3r2 1 2 1 1 2 1  2 9r1  3 3r1 2 2 1 3r1  4 6r1 2 2 1 5 3r1 2 2 1 2 2 1
end array
end data
end
```

APPENDIX D

(MCNP input file obtained by translating using *KtoMTran* the KENO input file given in Appednix C)

```
sample problem 12 4 aqueous 4 metal mixed units
1 0      (1:-2:3:-4:5:-6) imp:n=0
2 0      -7 2 -8 4 -9 6 (-10:11:12) imp:n=1
3 1 4.803321e-02 10 -11 -12 imp:n=1
4 0      -1 7 -8 4 -9 6 (-13:14:15) imp:n=1
5 3 1.065351e-01 13 -14 -15 (-16:17:18) imp:n=1
6 2 9.840283e-02 16 -17 -18 imp:n=1
7 0      -7 2 -3 8 -9 6 (-10:11:19) imp:n=1
8 1 4.803321e-02 10 -11 -19 imp:n=1
```

```
9 0      -1 7 -3 8 -9 6 (-13:14:20) imp:n=1
10 3 1.065351e-01 13 -14 -20 (-16:17:21) imp:n=1
11 2 9.840283e-02 16 -17 -21 imp:n=1
12 0      -7 2 -8 4 -5 9 (-22:23:12) imp:n=1
13 1 4.803321e-02 22 -23 -12 imp:n=1
14 0      -1 7 -8 4 -5 9 (-24:25:15) imp:n=1
15 3 1.065351e-01 24 -25 -15 (-26:27:18) imp:n=1
16 2 9.840283e-02 26 -27 -18 imp:n=1
17 0      -7 2 -3 8 -5 9 (-22:23:19) imp:n=1
18 1 4.803321e-02 22 -23 -19 imp:n=1
19 0      -1 7 -3 8 -5 9 (-24:25:20) imp:n=1
20 3 1.065351e-01 24 -25 -20 (-26:27:21) imp:n=1
21 2 9.840283e-02 26 -27 -21 imp:n=1

1 px 43.5
2 px 0
3 py 43.5
4 py 0
5 pz 40.96
6 pz 0
7 px 21.75
8 py 21.75
9 pz 20.48
10 pz 8.8725
11 pz 19.6375
12 c/z 15.16 15.16 5.748
13 pz 0.715
14 pz 19.765
15 c/z 32.625 10.875 10.16
16 pz 1.35
17 pz 19.13
18 c/z 32.625 10.875 9.525
19 c/z 15.16 28.34 5.748
20 c/z 32.625 32.625 10.16
21 c/z 32.625 32.625 9.525
22 pz 21.3225
23 pz 32.0875
24 pz 21.195
25 pz 40.245
26 pz 21.83
27 pz 39.61

m1 92234 4.827160e-04 92235 4.479710e-02 92236 9.572310e-05
     92238 2.657670e-03
m2 1001 5.779640e-02 7014 2.130920e-03 8016 3.741300e-02 92234 1.067840e-05
     92235 9.845990e-04 92236 5.293850e-06 92238 6.194130e-05
m3 1001 5.681870e-02 6012 3.551170e-02 8016 1.420470e-02
kcode 1000 1.0 15 115
ksrc  32.6 32.6 30.7 15.2 28.3 26.7 32.6 10.9 30.7
       15.2 15.2 26.7 32.6 32.6 10.2 15.2 28.3 14.3
       32.6 10.9 10.2 15.2 15.2 14.3
```