**Serpent User’s Manual: Condensed**

Serpent is a continuous-energy Monte Carlo reactor physics code developed at the VTT Technical Research Centre of Finland that is distributed in the United States by Oak Ridge National Laboratory (ORNL) through the Radiation Safety Information Computational Center (RSICC). Serpent can be used to model the neutronic behavior of a wide range of geometries and materials, and is capable of calculating an extensive range of parameters, such as those relating to point reactor kinetics, reaction rates, cross sections, factors in the six-factor formula, and diffusion coefficients. When used on a more advanced level, Serpent can be used to generate 2- and 3-D power profiles, fuel composition over time (burnup calculation), and iterations in some user-specified parameter (such as the amount of boron in PWR coolant) needed to maintain criticality, among many other quantities. This manual will cover the basic use of Serpent, while a short section will be devoted to discussing more advanced applications.

Serpent relies on the repeated random sampling of neutrons within a user-defined geometry through a random number generating process. The transport simulation runs in cycles, and the neutron source distribution in each cycle is determined by the fission reaction distribution of the previous cycle. Serpent by default uses a 2-group neutron structure, with the boundary between fast and thermal neutrons set to 0.625 eV. To perform the transport calculation, Serpent reads cross section data from large ACE data libraries according to the materials in the geometry specified by the user. This condensed manual is written with the assumption that Serpent has been correctly compiled, and focuses on simple input decks that when run will provide neutronic information on the user-defined geometry. To access the code: <https://rsicc.ornl.gov>. For the full manual: <http://montecarlo.vtt.fi/download/Serpent_manual.pdf>.

**1. Input File**

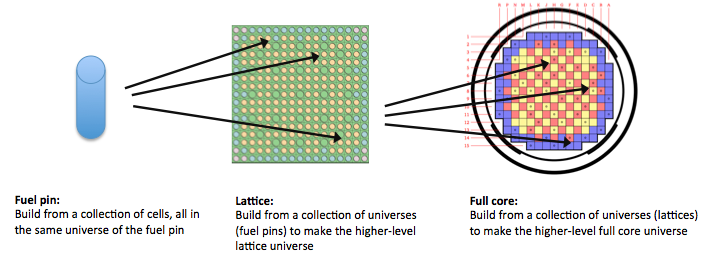
All interaction between the user and Serpent is through the command line interface. Once the code has been compiled, it is run simply by typing:

sss <inputfile>

where <inputfile> is the name of the input file. You may have to use ./sss to run files on Mac systems. Input files are written in a text editor, such as Sublime Text 2. The input file consists of “cards,” or separate data blocks. The file will be processed one card at a time, and there is no restriction on what order the cards may be placed in. Comments are made in the input file by the /\*comment here\*/ C-style comment. The input file will consist of a user-defined geometry card, material definitions cards, and other options.

**2. User-Defined Geometry**

Serpent uses a universe-based geometry composed of *homogeneous* cells that are ordered into *heterogeneous* universes, with higher-order universes composed of combinations of lower-order universes. For example, to define a core, first the user would define the (homogeneous) composition of each part of a fuel pin (fuel, gap, and clad), and then would assign these cells to a universe encompassing a fuel pin, the first-order universe. Then, the locations of the fuel pins would be specified within the heterogeneous fuel pin lattice, the second-order universe. This would then be repeated for each fuel assembly to be included in the core, where each assembly is it’s own universe. At the highest level, the user would then specify the layout of the assemblies within the core, making the core the highest universe in the geometry. The basic building block is a cell, a region of space defined by a collection of surfaces that is filled by a *homogeneous* material. These cells are then combined to form a *heterogeneous* universe.



*Figure 1: Example of the construction of universes from cells for a reactor core geometry [1, 2].*

**2.1 Surfaces**

The basic building block, or cell, is defined in terms of a collection of surfaces. The syntax of the surface card is:

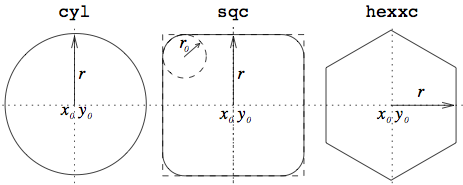
surf <id> <type> <parameter 1> <parameter 2> ...

where <id> is the surface identifier (arbitrary number)

<type> is the surface type

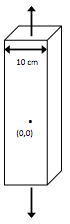
<parameter 1> <parameter 2> ... are the surface parameters

The surface identifier is an arbitrarily chosen number used to reference that particular surface later in the cell definitions. Table 1 contains a list of several surface types, as well as their surface parameters. See the Serpent manual for the complete list of surface types. The parameters , refer to the coordinate distance of the center from the origin. The radius of a surface is . The cyl, sqc, and hexxc surfaces are infinite in the direction. If the circular, square, or hexagonal cylinders have rounded edges, then is the radius of curvature of the rounded edge – if they do not have rounded edges, than there does not need to be an entry for .

*Table 1: Several surface types in Serpent [1].*

*Figure 2: Several cylinder types.*

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Parameters** |
| inf | all space | N/A |
| px | plane perpendicular to the x-axis |  |
| py | plane perpendicular to the y-axis |  |
| pz | plane perpendicular to the z-axis |  |
| sph | sphere |  |
| cyl | circular cylinder parallel to the z-axis |  |
| sqc | square cylinder parallel to the z-axis |  |
| hexxc | hexagonal cylinder parallel to the z-axis |  |



Below are several examples of surface definitions:

/\* square cylinder at origin with side length of 10 cm \*/

surf 100 sqc 0 0 5

/\* sphere shifted from origin to (1,0,5)cm of radius 3 cm \*/

surf 3 sph 1 0 5 3

/\* plane perpendicular to the z-axis at z=100 cm \*/

surf 123 pz 100

**2.2 Cells**

The user defines an arrangement of surfaces in order to obtain the desired geometry. These surfaces are defined such that any single space between these surfaces consists of a homogeneous material. The regions of the geometry that will contain homogeneous materials are then defined as cells by specifying whether the user-defined materials are inside or outside a collection of the surfaces. Each surface is associated with a positive and negative side. A point is inside a surface if it is on the negative side of the surface, and outside if on the positive side of the surface. The positive side of a plane is the side facing the direction of the positive axis. The syntax of the cell card is:

cell <name> <universe> <mat> <surf 1> <surf 2> ...

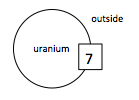
where <name> is the cell name (arbitrary number)

<universe> is the universe number of the cell

<mat> is the cell material

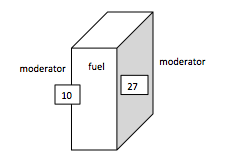
<surf 1> <surf 2> ... are the boundary surfaces (with indication)

<mat> is a user-defined material that is defined to exist either inside or outside of surfaces <surf1>, <surf2> by listing positive or negative versions of the surface numbers. The cell syntax is repeated for each material and each region of space, and cells are defined in groups relating to their common universe number. A universe must be defined to cover all space, and there must always be a universe with number 0. The <mat> card is set to void if there is no material in that region or to outside if that region of space is outside the region of interest of the user. Cells are best illustrated by examples:



/\* solid uranium sphere of radius 17.5 cm \*/

surf 7 sph 0 0 0 17.5

cell 1 0 uranium -7 /\* uranium inside the sphere \*/

cell 2 0 outside 7 /\* outside sphere not of interest \*/

/\* slab of fuel (from x=0 to x=36 cm) in an infinite moderator \*/

surf 10 px 0

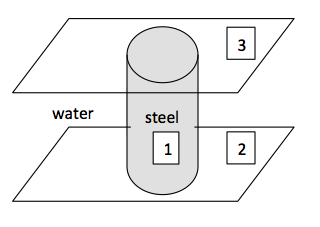
surf 27 px 36

cell 1 0 moderator -10 /\* moderator to the left of the fuel \*/

cell 2 0 moderator 27 /\* moderator to the right of the fuel \*/

cell 3 0 fuel 10 -27 /\* fuel between the two planes \*/

/\* steel cylinder, radius 10 cm and height 300 cm surrounded by water on all sides \*/

surf 1 cyl 0 0 10

surf 2 pz -150

surf 3 pz 150

cell 100 0 steel -1 2 -3 /\* steel inside cyl b/w 2 planes \*/

cell 101 0 water 1 2 -3 /\* water outside cyl b/w 2 planes \*/

cell 102 0 water -2 /\* water below bottom plane \*/

cell 103 0 water 3 /\* water above top plane \*/

**2.2.1 Fuel Pins**

Serpent was originally developed to provide easier definition of fuel pins and lattices. Serpent has a simplified syntax for axially infinite fuel pins based on nesting annular layers instead of defining many cyl surfaces:

pin <id>

<mat 1> <radius 1>

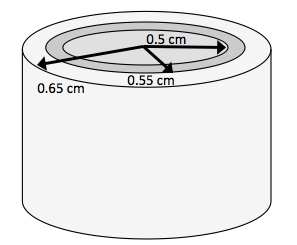
<mat 2> <radius 2> ...

where <id> is the pin identifier (arbitrary number)

<mat 1> <mat 2> are the materials

<radius 1> <radius 2> are the outer radii of the material regions

The material regions and their outer radii are listed in ascending order, and Serpent constructs cells using concentric cylindrical regions. The outermost material is given without a radius, and is assumed to fill the remaining space.



/\* fuel, helium gap, and cladding, surrounded by water \*/

pin 35

fuel 0.50

helium 0.55

cladding 0.65

water

**2.3 Universes**

Constructing higher-order universes from a collection of lower-order universes is accomplished by placing the lower-level universes in a geometric arrangement by specifying the of the universes relative to the origin. Or, to fill a cell with another universe, replace the <mat> entry in the cell card with fill <universe>, where <universe> is the number of the universe that you are filling in the cell. As an example, if instead you wanted to fill the axially finite cylinder in the third example on this page with a universe already defined (for example, universe 13), then instead of steel, you could type fill 13.

**2.3.1 Lattices**

Lattices are special universes that are filled with a regular structure of other universes (fuel pins). The syntax of the lattice card is:

lat <universe> <type> <x0> <y0> <nx> <ny> <pitch>

where <universe> is the universe number of the lattice

<type> is the lattice type (=1, 2, or 3, where 1 = square cylinder, 2 = x-type hexagonal cylinder, and 3 = y-type hexagonal cylinder)

<x0> is the x-coordinate of the lattice origin

<y0> is the y-coordinate of the lattice origin

<nx> is the number of lattice elements in the x-direction

<ny> is the number of lattice elements in the y-direction

<pitch> is the pin pitch, or distance between neighbor fuel pin centers

The lattice card is then followed by a list of the pin identifiers that define the fuel pins in the lattice, which have been defined previously. As an example:

lat 10 1 0 0 8 8 1.25 /\*Square lattice with 8x8 elements with pitch 1.25 cm\*/

3 3 3 3 3 3 3 3 /\* Pins 3, 4, and 5 have been defined with the pin syntax \*/

3 4 3 3 3 3 4 3 /\* i.e., 3, 4, and 5 are the pin identifiers for three \*/

3 3 5 3 3 5 3 3 /\* different pin types \*/

3 4 3 5 5 3 4 3

3 4 3 5 5 3 4 3

3 3 5 3 3 5 3 3

3 4 3 3 3 3 4 3

3 3 3 3 3 3 3 3

**3. User-Defined Materials**

The user defines all materials used. Each material must be defined homogeneously – inhomogeneities in the geometry are addressed through the use of different surfaces bounding separate, homogeneous regions. Nuclide temperatures are fixed when the cross-section data is generated. The syntax of the material card is:

mat <name> <density>

<isotope 1> <fraction 1>

<isotope 2> <fraction 2>

...

where <name> is the material name (arbitrary user’s choice)

<density> is the density (mass or atomic)

<isotope 1> <isotope 2> are the names of the constituent nuclides

<fraction 1> <fraction 2> are the corresponding fractions (mass or atomic)

The <density> is the mass or atomic *density* of the whole material, while <fraction 1> <fraction 2> are the mass or atomic *densities* or *fractions* of the isotopes that make up the homogeneous material. Negative values for <density> and <fraction> indicate mass densities (g/cm3) and mass fractions, while positive values indicate atomic densities (in units of 1024 atoms/cm3) and atomic fractions. The <isotope> entry is of the format (1000\*Z+A).09c for fuel material and (1000\*Z+A).06c for non-fuel material. The .09c and .06c extensions correspond to entries in the cross section libraries for cross sections evaluated at 900 K and 600 K, respectively. In full power reactor applications, because the fuel is at an average temperature around 900 K and the other materials such as cladding and coolant are around average temperatures of 600 K, use the .09c extension for fuel materials and the .06c extension for non-fuel materials. Isotopes of natural composition are defined as (1000\*Z).09c or (1000\*Z).06c. For reference, oxygen in UO2 composes about 0.1185 weight percent of the material. Several examples:

/\* UO2 fuel, 4.64% of the uranium is U-235, remainder of uranium is U-238, overall density of 10.97 g/cm3 \*/

mat UO2fuel -10.97

92235.09c -0.0409

92238.09c -0.8406

8016.06c -0.1185

/\* Cladding made of natural zirconium (98 atomic%), natural iron (0.5 atomic%), and Ni-58 (1.5 atomic%), overall density of (3.851E-02)\*1024 atoms/cm3 \*/

mat cladding 3.851E-02

40000.06c 0.98 /\* natural zirconium \*/

26000.06c 0.005 /\* natural iron \*/

28058.06c 0.015 /\* Ni-58 \*/

**4. Other (Necessary) Options**

The statistical accuracy of a Monte Carlo code is directly related to how many neutrons are sampled each cycle, and for how many cycles. The user must set the number of neutrons sampled for each cycle, and the number of cycles. 500 active cycles of 5000 source neutrons is generally sufficient. Set the number of inactive cycles run to 20 to allow the initial fission source distribution to converge before collecting results.

set pop <npop> <cycles> <skip>

where <npop> is the number of source neutrons per cycle

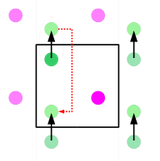
<cycles> is the number of active cycles

<skip> is the number of inactive cycles run

The user must specify in the input file where the cross section library files are located:

set acelib “<file path>”

There are several cross section data libraries available with the installation package. You may use whichever you like, but be aware that using different cross section libraries may give slightly different results and special syntax is needed for each used when specifying the thermal scattering cross section libraries.



There are 3 options for boundary conditions to be applied at surfaces:

set bc <option>

where <option> is the boundary condition of 1) the neutron vanishes, 2) the neutron is reflected back into the geometry, and 3) the neutron moves to the opposite side of the geometry. Condition 3 is a periodic boundary condition used to simulate an infinite geometry as a unit cell. For example, neutrons leaving the right side of the cell will reappear on the left side of the cell.

*Figure 3: Periodic boundary condition [3]*

A random number seed is used to initiate each transport calculation. To keep the same random number seed between calculations (such as seed# 420515206), which may be beneficial when testing your input, set:

set seed <seed number>

It needs to be specified for which universe to calculate the delayed neutron group constants. For simple examples with only one universe (which must be defined to 0, since a 0 universe must always exist), the group constant universe is set to 0. To specify this:

set gcu <universe number>

When Serpent pulls cross section data from the ACE libraries, it discretizes the cross sections based on the neutron energies. This leads to a very fine energy grid, which can be thinned down by recombining cross section values at energies very close to each other, which saves memory usage. Include the following line in your input to combine points with a relative difference of less than 5E-5 for neutron energies ranging from 1E-9 MeV to 15.0 MeV.

set egrid 5E-5 1E-9 15.0

**5. Full Example Input**

The example below is a full input for an axially infinite slab of water from cm, 3.75 weight% enriched UO2 fuel from cm, and a symmetric reflector of water from cm. Regions outside of cm is considered outside the geometry. Bolded text should be copied and pasted into your input – it consists of options that are needed to read the correct bound-atom cross sections for light water. These libraries are needed to replace the low-energy free-gas elastic scattering cross sections (that are used for most materials) for important bound nuclides such as hydrogen in water. To use this specific bolded text, you must be using the ENDF/B-VI.8 cross section library, or else the format of the bolded words changes slightly. The moder lwtr 1001 text should follow the first line of the water material card, as this specifies that the cross section for H-1 is being changed. You will be using only one universe, so be sure to set the universe number of each cell of that universe to zero.

mat water  -0.8362 **moder lwtr 1001** /\* water density at 500K and 8 MPa \*/

1001.06c   -0.1119 /\* mass fraction of H-1 \*/

8016.06c   -0.8881 /\* mass fraction of O-16 \*/

**therm lwtr lwe6.12t**

mat UO2fuel -10.9700 /\* fuel density of 10.97 g/cm^3 \*/

8016.09c    -0.11850 /\* 11.85% of fuel weight is oxygen (approximate)\*/

92235.09c   -0.03306 /\* 3.75 wt% of the uranium is U-235 \*/

92238.09c   -0.84844

surf 100 px 0

surf 200 px 5

surf 300 px 12.5

surf 400 px 17.5

cell 1 0 outside -100 /\* outside \*/

cell 2 0 water    100 -200 /\* water from 0 to 5 cm \*/

cell 3 0 UO2fuel       200 -300 /\* fuel from 5 to 12.5 cm \*/

cell 4 0 water              300 -400 /\* water from 12.5 to 17.5 cm \*/

cell 5 0 outside  400 /\* outside \*/

set acelib “/home/codes/serpent/xsdata/endfb68/sss\_endfb68.xsdata"

set pop 5000 500 20

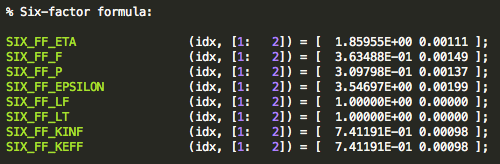
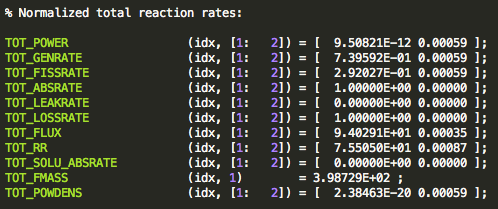
set bc 1 /\* “black” boundary condition \*/

set egrid 5E-5 1E-9 15.0 /\* thins energy grid \*/

set gcu 0 /\* group constants generated in universe 0 \*/

**6. Output File**

Successfully running the input file will produce a Matlab output file titled “<input>\_res.m”, where <input> is the name of your input file. This file contains easy-to-read calculation results, and will be created in the same location as your input file. Below is part of the output file for the full input given above. The full manual provides a description of each parameter listed in the output file. Generally, the first entry for a parameter will be the numeric value, and the second will be the relative statistical error in that value.



**7. More Advanced Uses of Serpent**

Serpent can be used in a more advanced manner to calculate 2- and 3-D power profiles, fuel composition over time, and many other options. The manual should be referenced to understand how to perform these activities. Briefly, when simulations are performed for multi-region problems such as fuel assemblies or even a whole core, normalized power distributions will be calculated for each fuel pin. Likewise, if a 3-D power profile is desired, then the geometry, such as an assembly, must be discretized into axial nodes. The output will then provide normalized power in the radial and axial directions. Figure 4 shows an example of a full-core power distributions for uniform fuel in a 37-assembly core. These distributions have been extracted from the Matlab output file and manipulated in Matlab to provide color plots.

In burnup calculation, a “burnup card” is included, in which you can specify the power level of the universe for a set length of time. In this way, the actual variable power level of the core can be modeled over an entire fuel cycle. At the end of the burnup card, a list of fission products is specified by the user, and after each burnup step, Serpent will determine the concentration of these fission products in the fuel in a separate output file. Burnup calculation is significantly more computationally intensive, and is primarily used to model fuel depletion over time, as well as decay heat once the fuel is removed from the reactor.

*Figure 4: NuScale full core power distribution.*

Serpent can also be used to model hexagonal and CANDU fuel assemblies, as well as pebble-bed configurations. In an iterative method, Serpent can calculate the necessary value in some parameter, such as the amount of boron in the coolant, to maintain .

**8. Units**

|  |  |
| --- | --- |
| **Quantity** | **Unit** |
| Distance | cm |
| Area | cm2 |
| Volume | cm3 |
| Time | s |
| Energy | MeV |
| Microscopic cross section | b |
| Macroscopic cross section | 1/cm |
| Mass | g |
| Mass density | g/cm3 |
| Atomic density | 1024 atoms/cm3 |
| Power | W |
| Neutron flux | 1/cm2s |
| Reaction rate density | 1/cm3s |

Serpent uses a set collection of units– be sure when entering numeric values that they are in the correct units.

**9. References**

A majority of this condensed manual was derived from the official Serpent manual [1].

[1] Leppanen, Jaakko. *PSG2/Serpent – a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code. Methodology – User’s Manual – Validation Report.* November 6, 2009.

[2] N. Horelik, B. Herman, B. Forget, and K. Smith. Benchmark for Evaluation and Validation of Reactor Simulations (BEAVRS), v1.0.1. Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuc. Sci. & Eng. 2013. Sun Valley, Idaho.

[3] Limiteperiodicite.svg. Grimlock. 02-01-2015.