Serpent Basics

Hacker Within

Wednesday, September 17th 2014

Overview

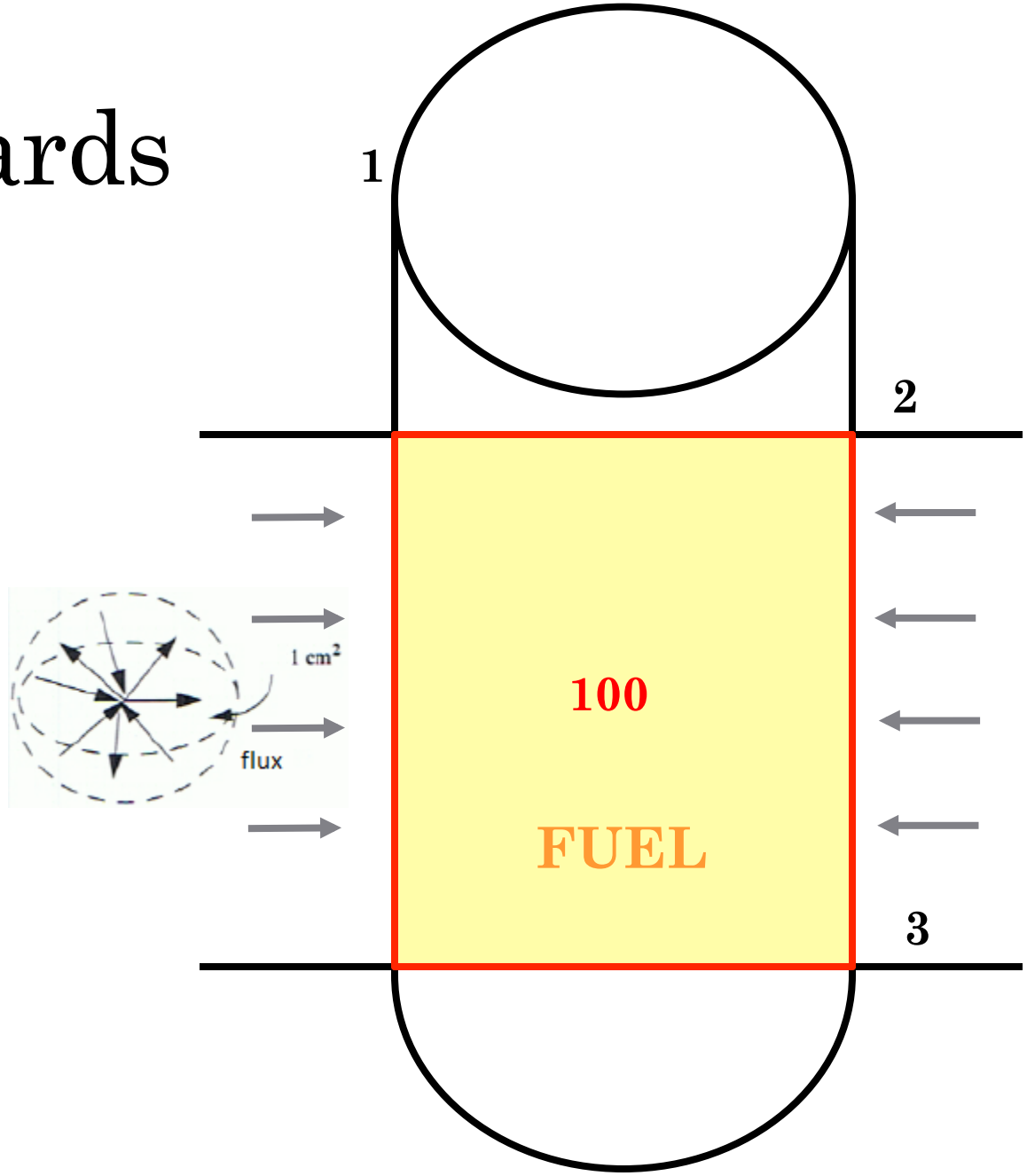
- Monte Carlo code that simulates neutron transport
  - Specifically for reactor physics
- Developed in 2004 at the VTT technical research center in Finland
- Serpent 1.0.0 released in 2008
  - Latest version up to 1.1.19
- Serpent 2 under development and should be available by the end of 2014
  - Beta version available to users of Serpent 1
- User community composed of 300+ users in 112 university and research organizations in 30 countries

Serpent input file

- User defines certain characteristics to correctly model the system:
  - Geometry
  - Materials
    - Isotopes
    - Cross sections
    - Density
  - Boundary Conditions
  - Calculations for the output
    - Flux, Reaction rates, leakage, etc.
- Input file is in plain text
  - The number of characters per line is unlamented (MCNP ≤ 80)
- The order of the cards does not matter since it is defined by a keyword
  - MCNP order matters
- %This is a comment in Serpent (indicated by %)

Input File Main Cards

- The volumes (3D) that make up the system are defined by their surfaces
- A cell is bounded by defined volumes
- A material is defined for each cell
- A boundary conditions is given
- Calculations are assigned to cells, surfaces, or materials

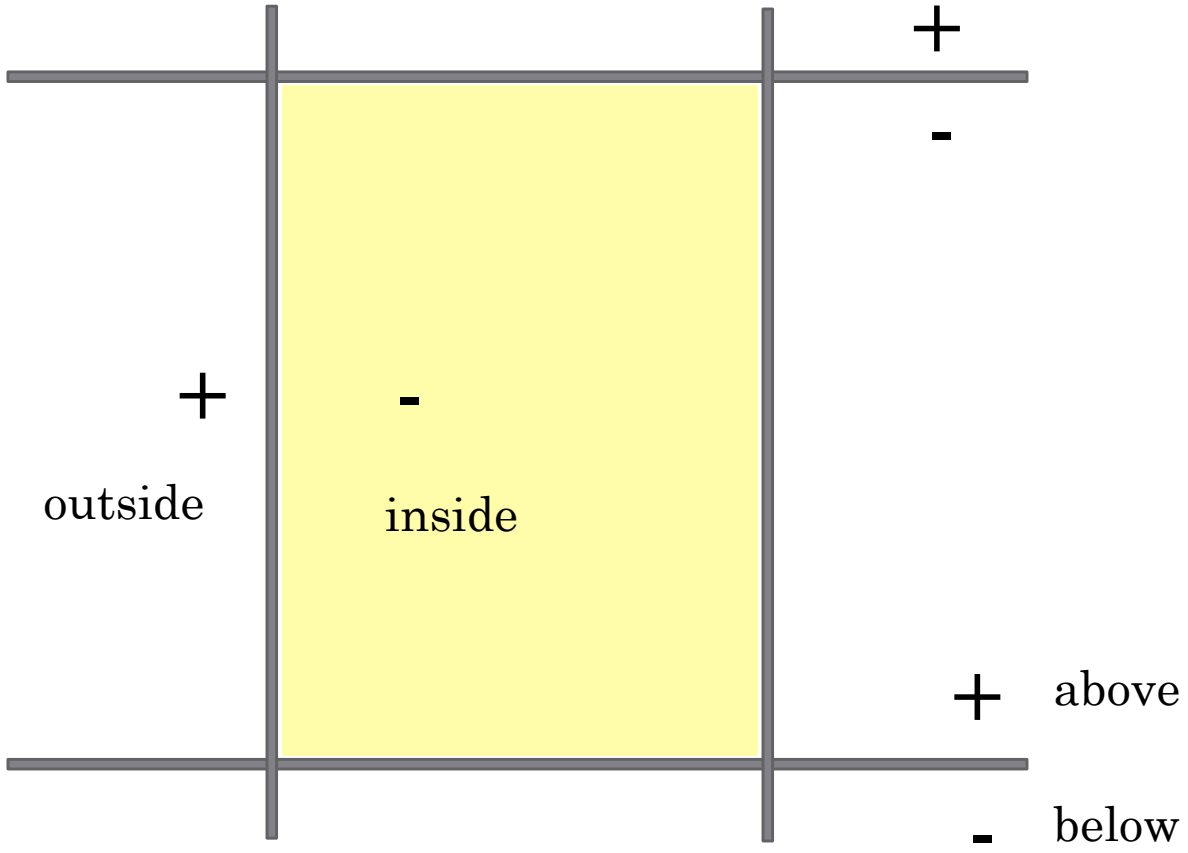


List of Cards

Card	Description	Chapter / Section	Page
cell	cell definition	3.3	24
dep	irradiation history	8.3	110
det	detector definition	7.1	95
disp	implicit HTGR particle fuel model	3.8.1	39
ene	detector energy binning	7.1.2	99
include	read a new input file	2.2.3	16
lat	lattice definition	3.6.2	30
mat	material definition	4.1.2	48
mesh	reaction rate mesh plotter	10.1	131
nest	nest definition	3.5	27
particle	particle definition	3.8	39
pbed	explicit HTGR particle / pebble bed fuel model	3.8.2	40
pin	pin definition	3.4	27
plot	geometry plotter	3.9	42
set	misc. parameter definition	5.1	53
src	external source definition	9.2	126
surf	surface definition	3.2	20
therm	thermal scattering data definition	4.2	49
trans	universe transformation	3.6.1	29

Surfaces

- Surfaces are divided by two zones with opposite orientations
- Cells define in what surfaces a material is bounded by (the orientations)
  - Cells in serpent are only defined by the intersection of surfaces



- Planes
  - Above (+)
  - Below (-)
  - Left (-)
  - Right (+)
- 3D volumes: cylinders, spheres, etc.
  - Inside (-)
  - Outside (+)

Surface syntax

```
surf <id> <type> <param 1> <param 2> ...
```

`<id>` is user defined and can be either a word or number

`<type>` is a predefined shape

Example:

```
surf 1 cyl 0.0 0.0 10.0 200 -200
```

OR

```
surf 1 cyl 0.0 0.0 10.0

surf 2 pz 200

surf 3 pz -200
```

Serpent Sufaces

Type	Description	Parameters
`inf`	all space	-
`px`	plane perpendicular to x-axis	x_0
`py`	plane perpendicular to y-axis	y_0
`pz`	plane perpendicular to z-axis	z_0
`sph`	sphere	x_0, y_0, z_0, r
`cylx`	circular cylinder parallel to x-axis	y_0, z_0, r, x_1, x_2
`cyly`	circular cylinder parallel to y-axis	x_0, z_0, r, y_1, y_2
`cylz or cyl`	circular cylinder parallel to z-axis	x_0, y_0, r, z_1, z_2
`sqc`	square cylinder parallel to z-axis	x_0, y_0, r, r_0
`cube`	cube	x_0, y_0, z_0, r
`cuboid`	cuboid	$x_1, x_2, y_1, y_2, z_1, z_2$
`hexxc`	x-type hexagonal cylinder parallel to z-axis	x_0, y_0, r, r_0
`hexyc`	y-type hexagonal cylinder parallel to z-axis	x_0, y_0, r, r_0
`hexxprism`	x-type hexagonal prism parallel to z-axis	x_0, y_0, r, z_1, z_2
`hexyprism`	y-type hexagonal prism parallel to z-axis	x_0, y_0, r, z_1, z_2
`cross`	cruciform cylinder parallel to z-axis	x_0, y_0, r, d, r_0
`pad`	(see description below)	$x_0, y_0, r_1, r_2, \theta_1, \theta_2$
`conx`	cone oriented in the x-axis	x_0, y_0, z_0, r, h
`cony`	cone oriented in the y-axis	x_0, y_0, z_0, r, h
`conz or cone`	cone oriented in the z-axis	x_0, y_0, z_0, r, h
`dode`	dodecagonal cylinder parallel to z-axis	x_0, y_0, r_1, r_2
`octa`	octagonal cylinder parallel to z-axis	x_0, y_0, r_1, r_2
`plane`	general plane	A, B, C, D
`quadratic`	general quadratic surface	$A, B, C, D, E, F, G, H, J, K$

Cell Syntax

```
cell <name> <u0> <mat> <surf 1> <surf 2> ...
```

`<name>` user defined name (or number)

`<u0>` the cell universe

`<mat> the material`

`<surf 1>` the surfaces that bind the cell

Example:

```
cell 1 0 fuel -1 -200 200
```

- ‘Outside’ is defined as the space (material) that is not part of geometry
- ‘void’ is material that defines an empty cell

Universes

- A universe allows the geometry of the model to be divided into separate levels
  - Constructed independently and nested one inside the other
- Each universe is made up of one or multiple cells
- Universe ‘0’ is the ‘real’ onec
  - Each other universe (with a number greater than 0) are placed inside cells that have universe 0
- Example:
  - Fuel pin is a universe (2), moderator channel universe (3)
  - These are placed inside a fuel assembly, another universe (1)
  - The assembly is placed in the real universe (0)

Serpent has built in pin syntax

```
pin <id>
<mat 1> <r1>
<mat 2> <r2>
...
<mat n>
```

`<id>` is the pin identifier (universe number)

`<mat 1> <mat 2> ..` are the materials

`<r1> <r2> ..` are the outer radii of the material regions

- Note that `<mat n>` fills the universe space so it does not have a defined radius

Repeated Structures - Lattices

```
lat <u0> <type> <x0> <y0> <nx> <ny> <p>
```

`<u0>` is the universe number

`<type>` is the lattice type (1, 2, or 3)

`<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>` same as above but the y direction

`<p>` is the lattice pitch

Materials Defintion

```
mat <name> <dens> [<options>]
<iso 1> <frac 1>
<iso 2> <frac 2>
```

`<name>` is the material name

`<dens>` is the density, mass (-) or atomic (+)

`<options>` depend on the case

`<iso 1> <iso 2>` .. are the names of the constituent nuclides

`<frac 1> <frac 2>` .. are the corresponding fractions, mass (-) or atomic (+)

Material isotopes

- The isotopes are defined by ZAID (like MCNP)
  - Z is the element
  - A is the isotope mass number (three digits)
  - ID is the cross sectional library ID
- Example: 92238.09c (^{238}U)
  - Z = 92
  - A = 238
  - ID = 09.c (library data generated at 900 K)
- If you don't want the material as found in nature and not list all the isotopes use 000 for A
  - 40000.06c is natural zirconium
- The cross section ID is part of library whose path is defined in the input
  - Example: `set acelib "/usr/local/serpent/xsdata/endfb7/sss_endfb7u.xsdata"`

Materials options

- Volume: `mat <name> <dens> vol <V>`
- Mass: `mat <name> <dens> mass <M>`
- Thermal scattering cross sections replace the low-energy free-gas scattering (moderators such as hydrogen in water or carbon in graphite)
  - Defined by card: `therm <thname> <lib>`
  - Also added in material definition: `mat <name> <dens> moder <thname> <ZA>`

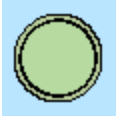
    `<thname>` name of data library

    `<lib>` library identifier as defined by directory in file

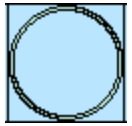
    `<ZA>` moderator nuclide ZA
- Doppler Broadening initiated by adding "tmp" entry to material card:
  - `Mat <name> <dens> tmp <T>`
  - The temperature T must be greater than cross section temperature

Fuel Assembly Example

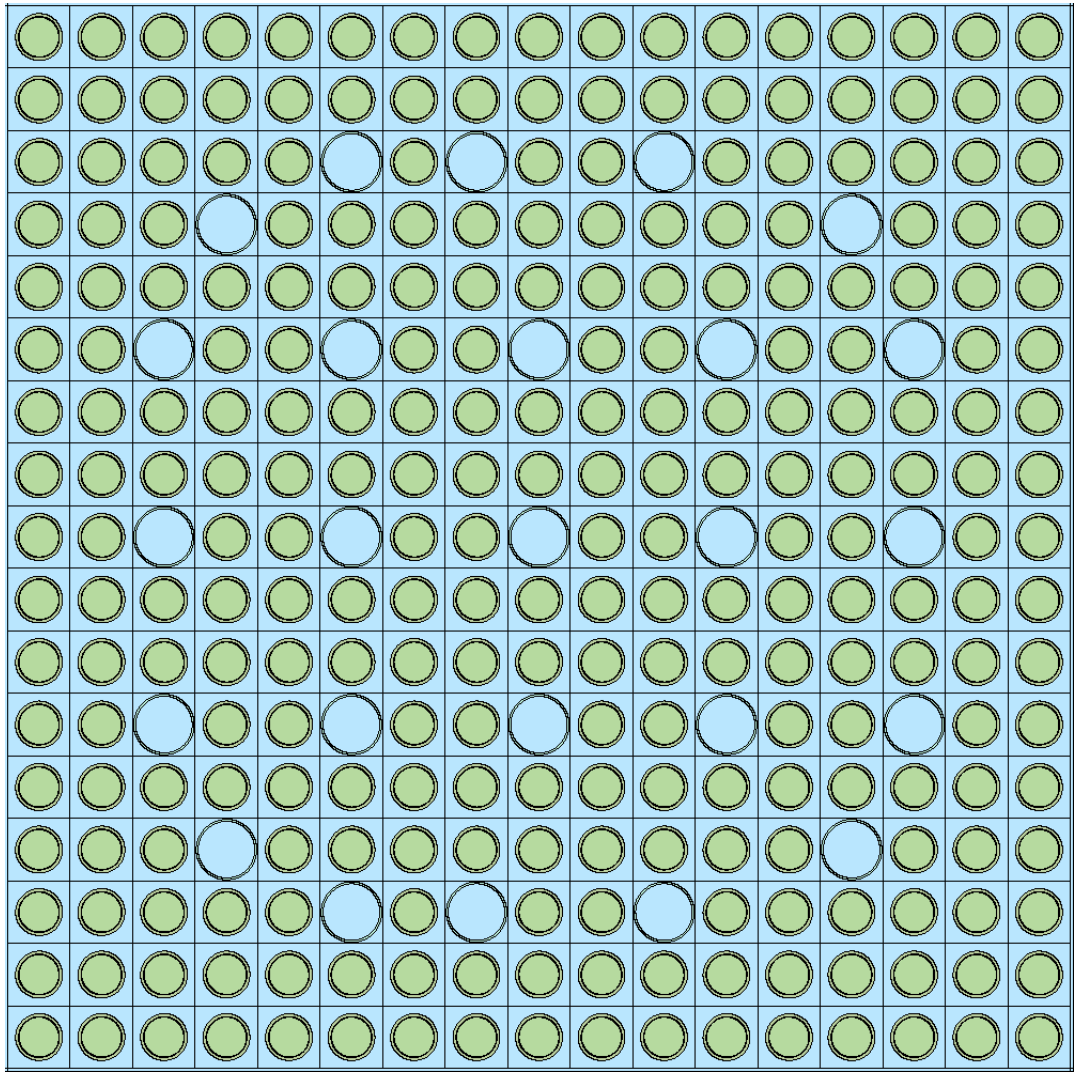
```
pin 2
fuel  0.4025
void  0.4150
clad  0.4750
water
```



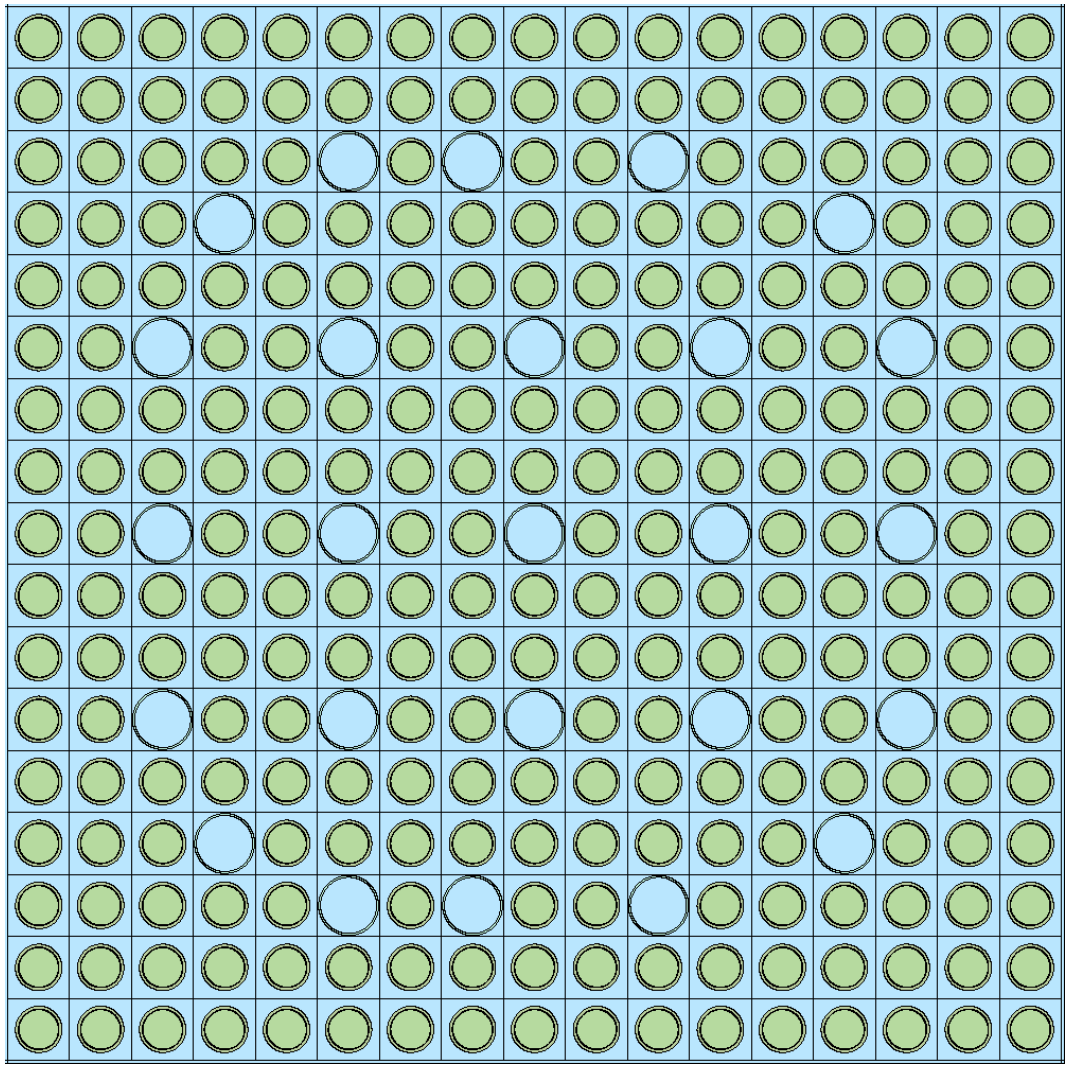
```
% --- Guide tube
pin 3
water   0.5730
tube    0.6130
water
```



```
% ---  pin lattice
lat 10 1 0.0 0.0 17 17 1.265
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 3 2 3 2 2 3 2 2 2 2 2 2
2 2 2 3 2 2 2 2 2 2 2 2 2 3 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 3 2 2 3 2 2 3 2 2 3 2 2 3 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 3 2 2 3 2 2 3 2 2 3 2 2 3 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 3 2 2 3 2 2 3 2 2 3 2 2 3 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 3 2 2 2 2 2 2 2 2 2 3 2 2 2
2 2 2 2 2 3 2 3 2 2 3 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

% --- assembly:
surf 50  cuboid   -10.752  10.752  -10.752 10.752  -200 200
surf 60  cuboid   -10.805  10.805  -10.805  10.805 -200 200

% --- Cell definitions:
cell  6   0  fill 10     -50
cell  7   0  water      -60 50
cell 99  0  outside    60
```



```
% --- Fuel materials:

mat fuel    -10.41
  8016.09c   6.6667E-01
 92238.09c   3.1667E-01
 92235.09c   1.6667E-02

% --- Cladding
mat clad    -6.56  tmp 800
 40090.06c   5.145E-01
 40091.06c   1.122E-01
 40092.06c   1.715E-01
 40094.06c   1.738E-01
 40096.06c   2.800E-02

% --- Moderator:
mat water  -0.7   moder lwtr 1001
 1001.06c   6.6667E-01
 8016.06c   3.3333E-01

mat tube   4.3206E-02
26000.06c  1.4838E-04
24000.06c  7.5891E-05
40000.06c  4.2982E-02

% --- Thermal scattering data for light water:
therm lwtr lwe7.12t

% --- Reflective boundary condition:

set bc 2

% --- Neutron population and criticality cycles:

set pop 2000 500 50 1.0

% --- Geometry and mesh plots:

plot 3 1000 1000
mesh 3 1000 1000
```



Other options

- Boundary conditions: `set bc <c>`
  - 1-void, 2-reflective, 3-periodic
- Geometry plot
  - `Plot <or> <nx> <ny> [<p> <min1> <max1> <min2> <max2>]`
    - `<or>` plot orientation where 1 = yz, 2 = xz, 3 = xy
    - `<nx>` is the width of the plot in pixels
    - `<ny>` is the height of the plot in pixels
    - `<p>` is the position on the axis perpendicular to the plot plane
    - `<min1> <min2>` minimum value of the first and second coordinate
    - `<max1> <max2>` maximum value of the first and second coordinate
- Neutron Population and criticality cycles
  - `set pop <npop> <cycles> <skip> [<keff0> <int>]`
    - `<npop>` is the number of source neutrons per cycle
    - `<ycles>` is the number of active cycles run
    - `<skip>` number of inactive cycles run
    - `<keff0>` initial keff guess
    - `<int>` collction interval

Detectors

- Serpent utilizes a collision estimate of neutron flux for calculating reaction rates integrated over space and energy

$$R = \frac{1}{V} \int_V \int_{E_{i+1}}^{E_i} f(\mathbf{r}, E)\phi(\mathbf{r}, E) d^3\mathbf{r} dE \, .$$

- The response function *f(**r**,E)* and spatial and energy domains of the integration are set by detector parameters
- The volume the integral is divided by is set to **unity** by default
- Sytanx: `det <name> <param 1> <param 2>`
  - `<name>` is the name and `<param 1> <param 2>` are the paramenters

Detector Parameters

Param.	Description	Comments
dr	Reaction multiplier	Determines the response function
dv	Detector volume	Used for normalization
dc	Detector cell	Defines the cell where the reactions are scored
du	Detector universe	Defines the universe where the reactions are scored
dm	Detector material	Defines the material where the reactions are scored
dl	Detector lattice	Defines the lattice where the reactions are scored
de	Detector energy grid	Defines the energy bins for the response function
dx	Detector mesh	Defines the x-mesh where the reactions are scored
dy	Detector mesh	Defines the y-mesh where the reactions are scored
dz	Detector mesh	Defines the z-mesh where the reactions are scored
dt	Detector type	Special detector types
ds	Surface current detector	Defines surface for current detector

Response Function

- f =1 , flux
- f = other number for reaction cross section
  - The results will be a reaction rate
- Syntax: `det <name> dr <mt> <mat>`
  - `<name>` detector name
  - `<mt>` response function number
  - `<mat>` material name (or 'void')
- Example: Average one-group absorption and fission rates in the fuel:

```
det 4 dm fuel dr -2 fuel

det 5 dm fuel dr -6 fuel
```

	MT	Reaction mode
Material total reactions	0	None
	-1	Total
	-2	Total capture
	-3	Total elastic
	-5	Total (n,2n)
	-6	Total fission
	-7	Total fission neutron production
	-8	Total fission energy deposition
	-9	Majorant
ENDF Reaction modes	1	Total
	2	Elastic scattering
	16	(n,2n)
	17	(n,3n)
	18	Total fission
	19	First-chance fission
	20	Second-chance fission
	51	Inelastic scattering to 1st excited state
	52	Inelastic scattering to 2nd excited state
	...	
	90	Inelastic scattering to 40th excited state
	91	Continuum inelastic scattering
	102	(n,γ)
	103	(n,p)
	104	(n,d)
	105	(n,t)
	106	(n,^{3}He)
	107	(n,α)

Finding Problems

- Serpent is a code in development, problems may surface ☹
- Patience, young grasshopper, things can be solved.
- My own experience with Macro XS calculation:

```
det 991 dm core1
det 992 dm core2
det 993 dm core3
det 994 dm core4
det 995 dm core5

det 109 dm core1 dr -6  U2381 dt 3 991

mat U2381 7.19049210e-003 92238.12c 1.0
```

```
mat core1 0.038306 vol
389683.8 burn 1 % axial
level 1/5, radial zone
1/1
90232.12c 1.05594290e-032
90233.12c 1.05139577e-032
90234.12c 1.04689439e-032
91231.12c 1.06052329e-032
91232.12c 1.05594049e-032
91233.12c 1.05140181e-032
92232.12c 1.05594700e-032
92233.12c 1.05140457e-032
92234.12c 1.04690624e-032
92235.12c 1.45941431e-005
92236.12c 1.03801542e-032
92237.12c 1.03362267e-032
92238.12c 7.19049210e-003
```

Different Results????

```
DET109_VOL   = 1.00000E+00;
DET109_RBINS = 1;
DET109_CBINS = 1;
DET109_MBINS = 1;
DET109_UBINS = 1;
DET109_XBINS = 1;
DET109_YBINS = 1;
DET109_ZBINS = 1;
DET109_LBINS = 1;
DET109_EBINS = 1;
DET109_VALS  = 1;
DET109 = [
    1    1    1    1    1    1    1
1    1  1.02803E-01  0.00008
```

```
MA        _ADENS = [
7.7       03 7.57325E-03 % 922380
MA        _CAPTXS = [
          E-01 2.10227E-01 % 922380
          XS * ATOMIC DEN = 1.5964E-03
```

Patience young grasshopper, problem will be solved…

The Serpent Forum is there for YOU

- Go here: http://ttuki.vtt.fi/serpent/index.php?sid=d73171cd775041e4a5ef131b40276c3c and submit any issue you have.
  - You will likely get a response! (be patient)

cross sections from burnup and detector do not match

by **aleja311** » Wed Aug 07, 2013 11:27 am

Hello, I am having some trouble figuring out why I am getting such different results regarding cross sections created from detectors and those that are in the depletion output.
I am doing parametric studies on a fast reactor transurnamic core and have to calculate the conversion ratio defined as the ratio of the capture cross section of U238 over the fission cross section of TRU.
At first I calculated this by taking the the micro capture XS of U238 times the atomic density from the depletion output at the beginning of life and dividing it by the sum of each TRU isotope fission micro cross section times its corresponding atomic density. This method gave me reasonable results.
Now I tried doing the exact same thing using detectors in the input without having to deplete the reactor. However I am getting some strange results.
Just like it was discussed in a much older post, I created materials for each of the TRU isotopes in the fuel. Since I wanted macro cross sections I included the atomic density of the isotope that is defined in the fuel region. For example with plutonium:
mat Pu239 atomic density 92239.12c 1.0
Then I created a detector to calculate the integral flux such as:
det 1 dm core
And with that I could create a detector that would get the fission reaction rate and divide it by the flux from detector 1 to obtain the macro cross section of Pu239:
det 2 dm core dr -6 Pu239 dt 3 1
I was pretty sure I did this correctly. However the cross sections (once the micro is multiplied by the atom density in the depletion output) do not seem to agree ever so slightly. For example with Pu239, the detector gives me a fission cross section of 5.34502E+00 while the burnup depletion file macro cross section is 0.0025.
The conversion ratio using detectors is way off at about 2% as opposed to 39% when depletion information was used.
What is going on? I do not understand how this could be so off.
Please help! I can also send my input if necessary.

Thank you!
Alejandra

aleja311

Posts: 10
Joined: Tue Jun 04, 2013 10:29 am

Bugs do exist

- In my case there was a bug in Serpent vs 1.1.19
- The source code of scoredetectors.c had to be fixed
  - “Change the if-branch between lines 60 and 69 …”

Re: cross sections from burnup and detector do not match

by **Jaakko Leppänen** » Tue Aug 13, 2013 5:19 pm

To fix the problem in Serpent 1, add new variable of type double "g" in scoredetectors.c, and change the if-branch between lines 60 and 69 from:

CODE: SELECT ALL

```
      if ((f = CurrentDet(det, nn, x, y, z, u, v, w)) != 0.0)
    {
      /* Get bin index */

      bin = GetDetBin(det, 0, 0, 0, ebin, 0, 0, 0, 0, 0);

      /* Score current */

      AddBuf(-sta, bin, f, wgt);
    }
```

to:

CODE: SELECT ALL

```
      if ((g = CurrentDet(det, nn, x, y, z, u, v, w)) != 0.0)
    {
      /* Get bin index */

      bin = GetDetBin(det, 0, 0, 0, ebin, 0, 0, 0, 0, 0);

      /* Score current */

      AddBuf(-sta, bin, g, wgt);
    }
```

i.e. change f to g.

- Jaakko

Jaakko Leppänen
Site Admin

Posts: 983
Joined: Thu Mar 18, 2010 10:43 pm
Location: Espoo, Finland

Problem Solved!

```
DET101_VOL   = 1.00000E+00;
DET101_RBINS = 8;
DET101_CBINS = 1;
DET101_MBINS = 1;
DET101_UBINS = 1;
DET101_XBINS = 1;
DET101_YBINS = 1;
DET101_ZBINS = 1;
DET101_LBINS = 1;
DET101_EBINS = 1;
DET101_VALS  = 8;
DET101 = [
    5    1    1    1    1    1    5    1
1    1  1.58898E-03 0.00221
```

```
MAT_core1_ADENS = [
7.70407E-03 7.57325E-03 % 922380
MAT_core1_CAPTXS = [
2.06239E-01 2.08586E-01 % 922380
MICRO XS * ATOMIC DEN = 1.5889E-03
```

MCNP and Serpent differences

- Mush faster!
  - Woodcock delta-tracking method
    - Serpent utilizes a combination of surface-to-surface ray-tracing method (used by MCNP) and the Woodcock-delta tracking method.
    - Can deal with problems where the mean free path is much longer than the dimensions (cell and surfaces)
      - TRISO particles
    - Drawback: reaction rates have to be calculated using a collision estimator that is not as efficient as the track-length estimate of neutron flux
  - Unionized energy grid
    - Continous-energy cross sctions in the library files are reconstructed on a unionized energy grid, used for all reaction modes
    - Drawback: grid becomes increasingly large with burnup (not enough memory). Serpent 2 has ways to optimize large burnup problems, which the unionized energy grid approach is used selectively

MCNP and Serpent differences

- Doppler-broadening preprocess routine
  - More accurate description of interaction physics in temperature-sensitive applications
- Burnup calculation
  - Built-in calculation routines without any external coupling
  - Uses TTA and CRAM methods (described last week)
- Can run in parallel
- Much more user friendly
- Output in matlab .m files for easy plotting
- Pretty gravy buuuut …
  - No union operator (serpent 2)
  - No photon transport
  - No variance reduction
  - No fixed source problems
  - No track length detectors and other available in MCNP
- Room for improvement


Serpent is the future