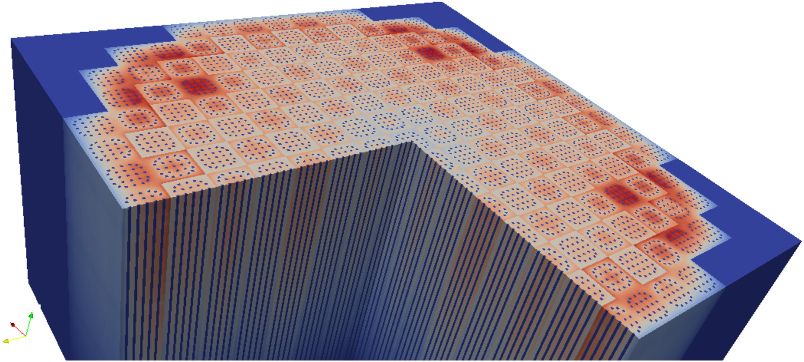
OpenMC Module: Tally Specification and Data Extraction

Computational Reactor Physics Group

Department of Nuclear Science and Engineering
Massachusetts Institute of Technology

How Do We Get Results?



Tallying Physics

- Method to extract quantities out of Monte Carlo run (e.g., flux, reaction rates, current, etc.)
- Tallies are made using different estimators (e.g., analog, collision, track-length)
- When an event occurs, the following formula is used for tallies:

$$\text{tally} = \sum_{i \in \text{events}} \frac{R_i w_i \phi_i}{W}$$

- w_i: neutron statistical weight, where W is total starting weight
- R_i: response function (flux=1, reaction rates=Σ)
- ϕ: estimator (analog=1, collision=Σ_t^{-1}, track=d)

Tally Specifications in OpenMC

- Tallies are specified on `tallies.xml`
  - See xml files in examples/demos/tallying
- Each tally has *scores* and *filters*
  - *Scores* specify <u>what</u> to tally
  - *Filters* specify <u>where</u> to tally
- Can choose tally estimator type: analog, tracklength, collision

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tallies>

  <tally id="1">
    <label>My Tally Label</label>
    <filter type="cell" bins="10 20 30"/>
    <scores>flux</scores>
    <estimator>tracklength</estimator>
  </tally>

</tallies>
```

Tally Scores

- Multiple responsed can be scored at the same time
- Can tally scattering moments, number of interaction events, arbitrary MT's
- See the users's guide for the full list of responses: http://mit-crpg.github.com/openmc/

```
<tally id="1">
  <label>My Awesome Tally 1</label>
  <filter type="mesh" bins="2"/>
  <filter type="energy">
   <bins>0.0 0.625e-6 20.0<bins/>
  </filter>
  <scores>total flux fission absorption nu-scatter scatter-P3</scores>
</tally>
```

Tally Filters

- Can specify cells, materials, universes, etc. to tally
- Easily specify meshes in `tallies.xml` for spatial tallies
- Can specify tallying for interactions only with certain nuclides (`<nuclides>` tag)
- See the users's guide for the full list of filters: http://mit-crpg.github.com/openmc/

```
<mesh id="2">
  <type>rectangular</type>
  <origin>0.0 0.0 0.0</origin>
  <upper_right>85.8774 85.8774 81.662</upper_right>
  <lower_left>-85.8774 -85.8774 -81.662</lower_left>
  <dimension>105 105 20</dimension>
</mesh>

<tally id="2">
  <label>My Scientific Tally #2</label>
  <filter type="mesh" bins="2"/>
  <filter type="material" bins="2"/>
  <nuclides>U-235 total</nuclides>
  <scores>fission absorption</scores>
</tally>
```

Tallies Are Set, OpenMC is Run – Now What?

- The manual way: Look at tallies.out

```
===================>     TALLY 1: MY TEST TALLY #1     <===================

Mesh Index (1, 1, 1)
  Incoming Energy [0.0, 6.25000E-07)
    Total Material
      Flux                        0.0            +/- 0.0
 ...
 ...
Mesh Index (24, 15, 3)
  Incoming Energy [0.0, 6.25000E-07)
    Total Material
      Flux                        7.42110E-08    +/- 3.14470E-08
  Incoming Energy [6.25000E-07, 20.0000)
    Total Material
      Flux                        2.07275E-07    +/- 6.29534E-08
Mesh Index (24, 15, 4)
  Incoming Energy [0.0, 6.25000E-07)
    Total Material
      Flux                        1.20947E-07    +/- 6.08719E-08
  Incoming Energy [6.25000E-07, 20.0000)
    Total Material
      Flux                        1.39299E-07    +/- 4.56898E-08
Mesh Index (24, 15, 5)
  Incoming Energy [0.0, 6.25000E-07)
    Total Material
 ...
 ...
```

Tallies Are Set, OpenMC is Run – Now What?

- The Better Way™: Parse statepoint files
- Run OpenMC to generate statepoint files
  - A final statepoint is always generated by default
  - Additional statepoints can be generated at any time throughout the run with the <state_point> tag in the `settings.xml`
- Use the `statepoint.py` utility to parse them
  - Once in python, it's easy to output to anything you want

```xml
<?xml version="1.0" encoding="UTF-8"?>
<settings>

  ...

  <state_point>
    <batches>7 11 15</batches>
    <interval>10</interval>
    <source_separate>true</source_separate>
  </state_point>

</settings>
```

Using Statepoint.py

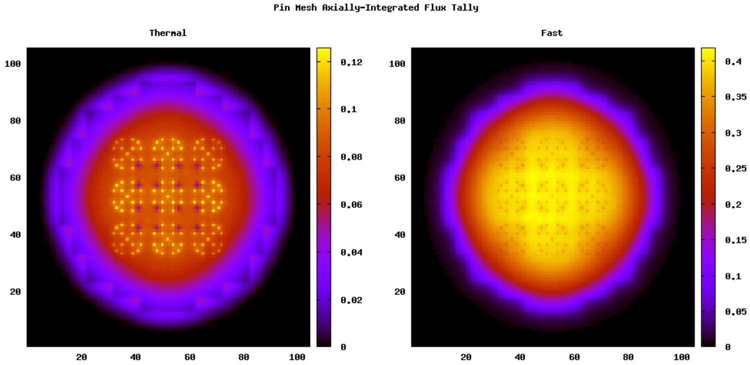
- Located in the src/utils directory of the OpenMC source
- Provides a simple user front-end for extracting tally data from statepoint files

```
sp = statepoint.StatePoint('statepoint.300.binary')
sp.read_results()

for tally in sp.tallies:
    print tally.id
    print tally.scores
    print tally.filters

print sp.meshes
```

Using Statepoint.py

```
tallyid = 0
scoreid = 0
egroupid = 0
values = {}
for x in range(1,nx+1):
    for y in range(1,ny+1):
        for z in range(1,nz+1):
            val,err = sp.get_value(tallyid,
                                   [('mesh',(x,y,z)),('energyin',egroupid)],
                                   scoreid)
            values[(x,y,z)] = val
```

See parse.py in examples/demos/tallying



Questions?

- Users's guide: http://mit-crpg.github.com/openmc/