OpenMC Workshop

Paul Romano, Sterling Harper, Will Boyd, Benoit Forget

May 1, 2016

Massachusetts Institute of Technology

Outline

Massachusetts Institute of Technology

Goals of the OpenMC Project

- Develop a Monte Carlo framework for reactor analysis that scales on leadership-class supercomputers (100,000+ cores)
- Provide a simple tool to analyze performance and limitations on proposed architectures
- Objectives:
  - Realistic physics
  - Modern programming style and data structures
  - Extensible for research purposes
  - Open source and freely available

Massachusetts Institute of Technology

Characteristics of OpenMC

Current features

- Fixed source and k-eigenvalue calculations
- **Geometry**: CSG with second-order surfaces, universes, rotations/translations, rectangular/hexagonal lattices
- **Cross sections**: ACE format or multi-group
- **Physics**: neutron transport, $S(\alpha, \beta)$ thermal scattering, probability tables, target thermal motion
- Flexible general tally system
- **Parallelism**: Distributed/shared-memory via MPI/OpenMP
- Rich, extensible Python API
- **Input**: XML or Python API-driven
- **Output**: HDF5
- Multi-group cross section generation

Short-term

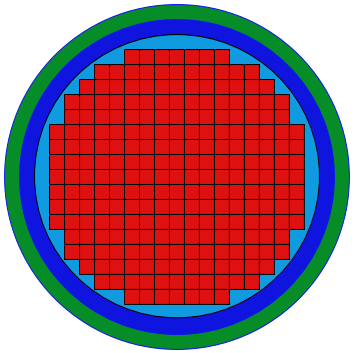
- HDF5 format cross sections
- Photon transport
- Depletion
- Transient-capability
- Temperature dependence
- Domain decomposition

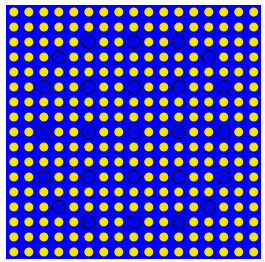
Medium-term

- In-house nuclear data processing
- Thermal-hydraulics coupling
- Non-CSG deformable geometry
- Coupling with MOOSE

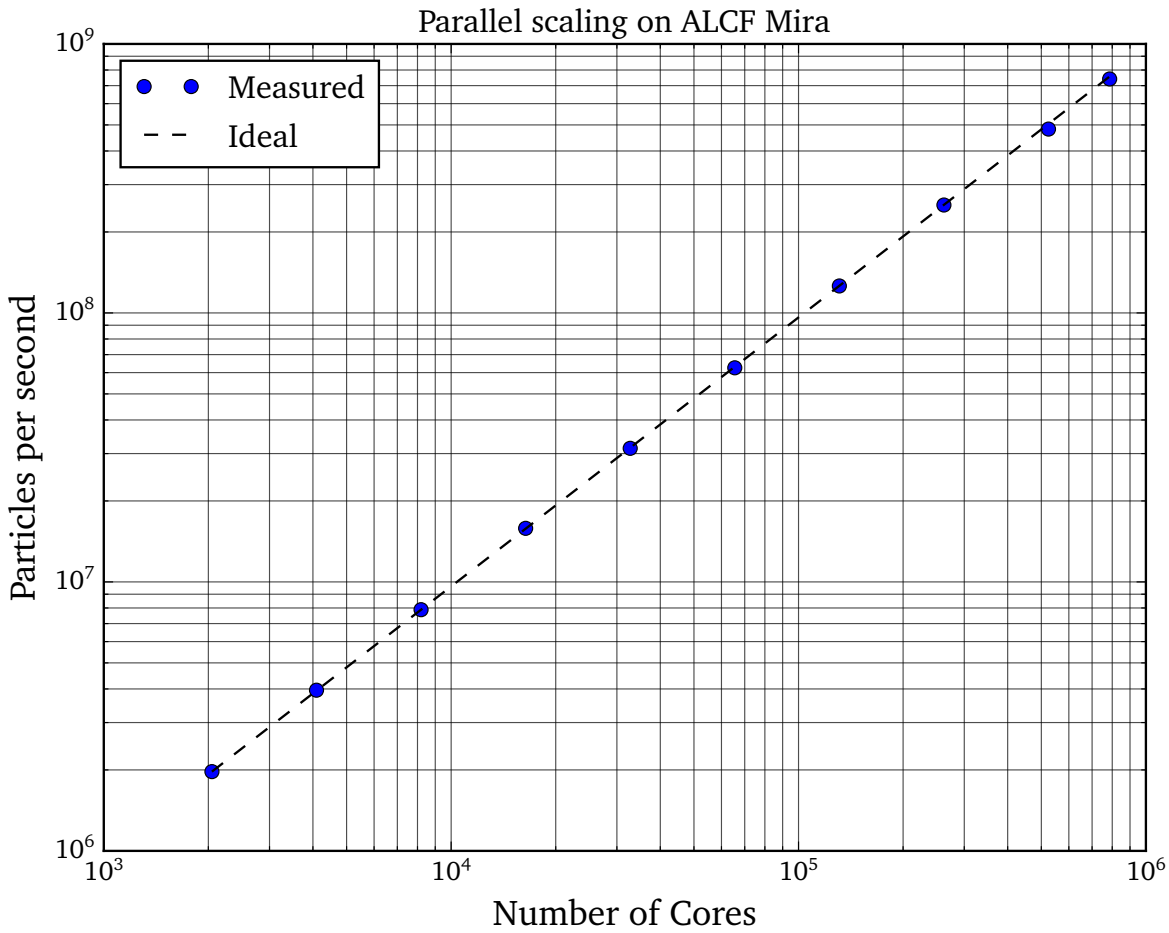
Massachusetts Institute of Technology

Parallel Performance


Core


Assembly

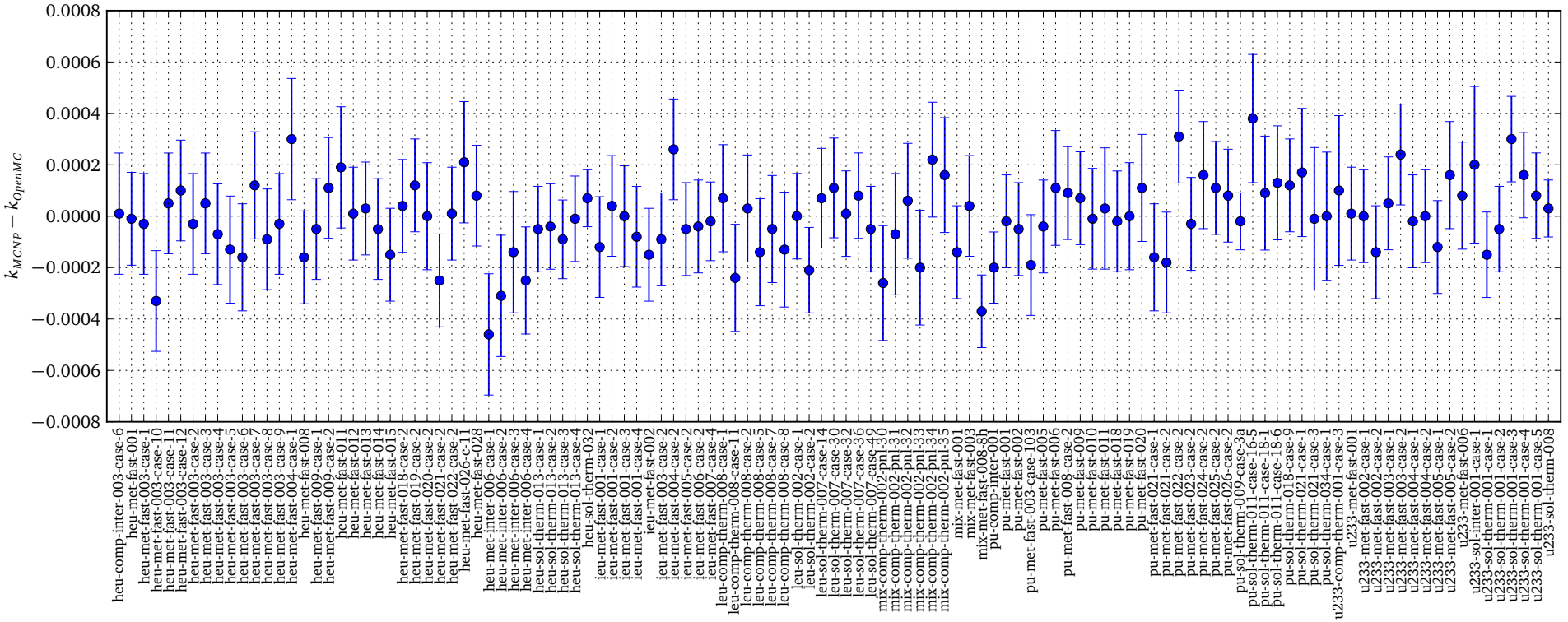
- Mira – #5 on TOP 500
- 49,152 nodes, 786,432 cores
- 4 hw threads/core = 3,145,728 threads



Institute of Technology

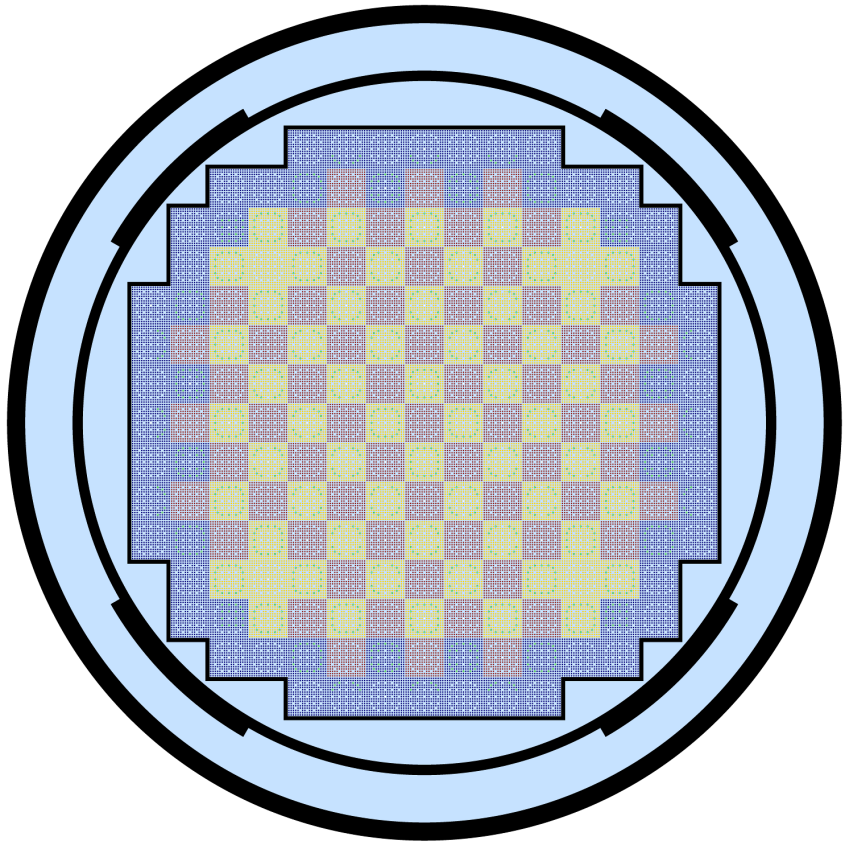
Validation and Verification

- MCNP Criticality Benchmark Suite
  - 119 configurations — different spectra, materials, enrichment
  - Models built for all but two



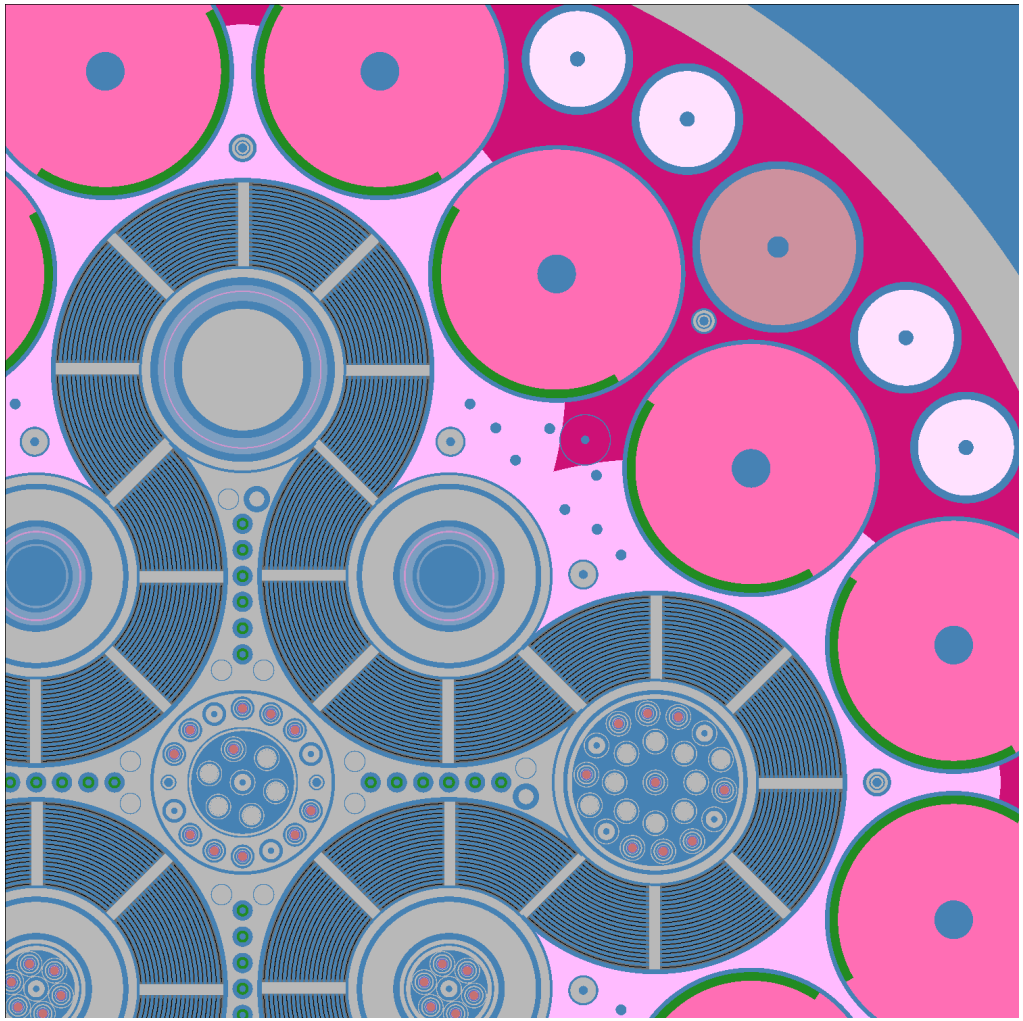
Example: MIT BEAVRS Benchmark

- Radial enrichment zoning
- Guide tubes/instrument tubes
- Burnable absorbers/control rods
- Grid spacers, core support
- Core baffel, core barrel
- Thermal shield pads
- Reactor pressure vessel



Massachusetts Institute of Technology

Example: Advanced Test Reactor



Massachusetts Institute of Technology

Software Architecture

- ∼40,000 lines of **Fortran 2008** code
- Distributed-memory parallelism via **MPI**
- Shared-memory parallelism via **OpenMP**
- **CMake** build system for portability
- ∼20,000 lines of **Python** API
- Version control through **git**
- Code hosting, bug tracking through **GitHub**
  - `https://github.com/mit-crpg/openmc`
- Automatic documentation generation with **Sphinx**
  - `http://openmc.readthedocs.io`

Massachusetts Institute of Technology

Resources

- User's Group: `openmc-users@googlegroups.com`
- Developer's Group: `openmc-dev@googlegroups.com`
- Documentation: `http://openmc.readthedocs.io`

Massachusetts Institute of Technology

Outline

Massachusetts Institute of Technology

XML Input

- When you run **openmc** command, it looks for a set of XML files:
  - Paramters for the simulation — **settings.xml**
  - Material definitions — **materials.xml**
  - Geometric model, mapping of materials — **geometry.xml**
  - Determine which quantities to score — **tallies.xml**
  - Create geometry plots — **plots.xml**
- Writing XML files is now considered "old-school"

Massachusetts Institute of Technology

Python API Input

- OpenMC includes a rich Python API that enables programmatic pre- and post-processing
- To construct input files (and analyze output) you, as the user, *must write a Python script* utilizing classes defined by the OpenMC API
- There are many benefits to using Python API over direct use of XML, e.g.,
  - All the usual benefits of Python – variables vs. hard-coded numbers, standard library, third-party packages
  - Run-time type checking
  - Perturbation or sensitivity studies
  - Extra features in our API – data analysis, multi-group cross section generation, nuclear data parsers, etc.
- **Caveat:** You need to know Python!

Massachusetts Institute of Technology

Python 101

Python is often thought of as a "scripting" language

- Dynamically typed (type of variable interpreted at runtime)
- Implicitly typed (type of variable never declared)
- Case-sensitive (var and VAR are different)
- Object oriented (everything is an object)
- Containers are zero-indexed

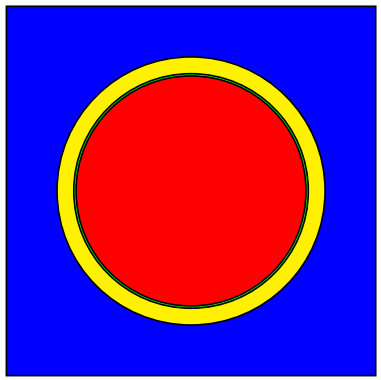
Massachusetts Institute of Technology

Problem to Solve

- Simple pin cell comprised:
  - UO_2 ($10.3\,g/cm^3$)
  - Clad ($6.55\,g/cm^3$)
  - Water ($0.701\,g/cm^3$)
- Dimensions:
  - Fuel OR $= 0.392\,18$ cm
  - Clad IR $= 0.400\,05$ cm
  - Clad OR $= 0.457\,20$ cm
  - Pitch $= 1.259\,84$ cm

Material	Isotope	Composition Fraction (a or w)
Fuel	U-235	0.02115 (w)
Fuel	U-238	0.86032 (w)
Fuel	O-16	0.11852 (w)
Clad	Zr	elemental
Water	H-1	2.0 (a)
Water	O-16	1.0 (a)

w = weight fraction, a = atom fraction



Massachusetts Institute of Technology

Creating Materials

- Material defined as collection of nuclides (`openmc.Nuclide`) and/or elements (`openmc.Element`) with assigned atom/weight fractions
- Each nuclide/element has a `name` and `xs` which link it to an ACE file
  - `name` is a nuclide's or element's name (e.g., H-1 or Zr)
  - `xs` is the ACE file cross section extension (for temperature)
- Total density of material can be specified in units of `g/cc` or `atom/b-cm` (renormalization based on total density)
- Thermal scattering libraries should be assigned to moderator materials
- In many cases, specifying default cross section is useful

Massachusetts Institute of Technology

Geometry and Material Assignments

- **Cell** defined by assigning a material/fill to a region
- **Region** defined as intersection, union, and/or complement surface half-spaces
- **Surface** is a locus of zeros of a function
  - Planes, e.g., $x - x_0 = 0$
  - Cylinders parallel to axis, e.g., $y^2 + z^2 - R^2 = 0$
  - Sphere, e.g., $x^2 + y^2 + z^2 - R^2 = 0$
  - Cones parallel to axis, e.g., $x^2 + z^2 - R^2 y^2 = 0$
  - General quadratic
- Surface **half-space** is the region whose points satisfy a positive/negative inequality of the surface equation
- Universes, rectangular/hexagonal lattices, rotation/translation

Massachusetts Institute of Technology

Simulation Settings

- At a minimum, one should specify how many particles to run and initial starting source
- The `openmc.Settings` object contains these and other settings:
  - `Settings.source` – list of sources
  - `Settings.batches` – number of batches
  - `Settings.inactive` – number of inactive batches
  - `Settings.particles` – number of particles per batch

Massachusetts Institute of Technology

Source Definitions

- The `openmc.Source` object allows one to specify separate spatial, angular, and energy distributions
  - Spatial distributions: independent (x, y, z), box, point
  - Angular distributions: independent (μ, ϕ), isotropic, monodirectional
  - Energy distributions: discrete, tabular, uniform, Maxwell, Watt

Defining Tallies

Tallies are defined as combinations of *filters* and *scores*

$$X = \underbrace{\int d\mathbf{r} \int d\mathbf{\Omega} \int dE}_{\text{filters}} \underbrace{f(\mathbf{r}, \mathbf{\Omega}, E)}_{\text{scores}} \psi(\mathbf{R}, \mathbf{\Omega}, E)$$

- Filters: cell, material, energy, outgoing energy, mesh, ...
- Scores: flux, total, fission, nu-fission, scatter, absorption, scattering moments, individual reactions, ...
- Can also specify nuclides

Massachusetts Institute of Technology

Plotting Geometry

- OpenMC can be run in plotting mode to create either slice or voxel plots
- Slice plots are `.ppm` files – usually best to convert to `.png`

Running OpenMC

With a complete set of inputs, we can run OpenMC by either

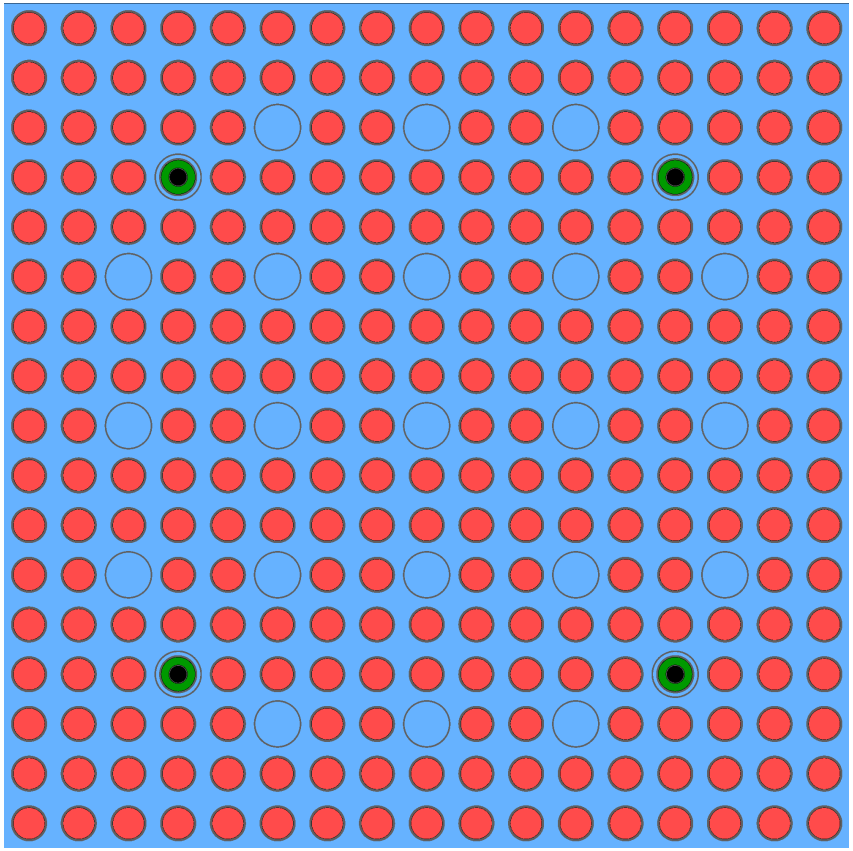
1. Dropping into a terminal and running `openmc`
2. Calling `openmc.run()`

Outline

Massachusetts Institute of Technology

Lattices

Pin cells are cool, but some physics can only be captured in a lattice calculation. This section will show how to construct and tally lattices in OpenMC.

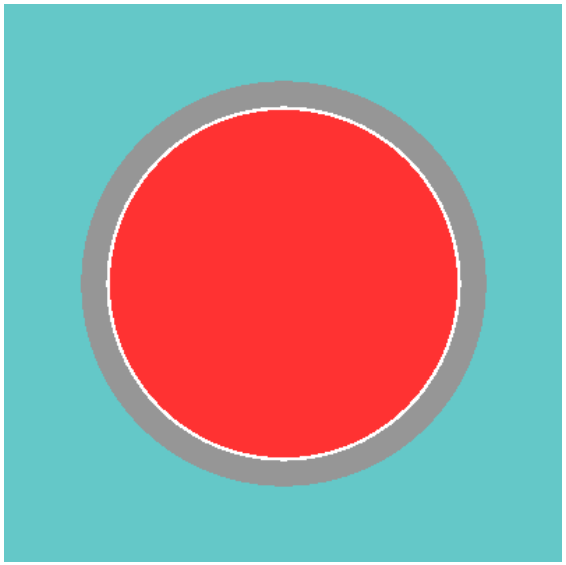


Massachusetts Institute of Technology

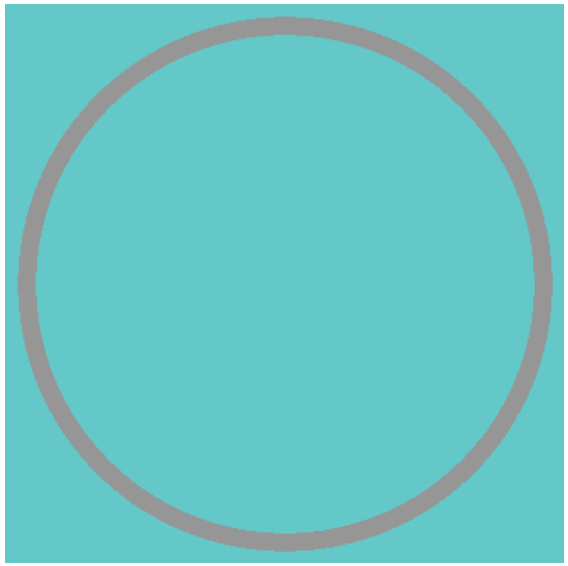
Piece-by-piece

Build each component of your lattice in its own universe, and test it as you go along

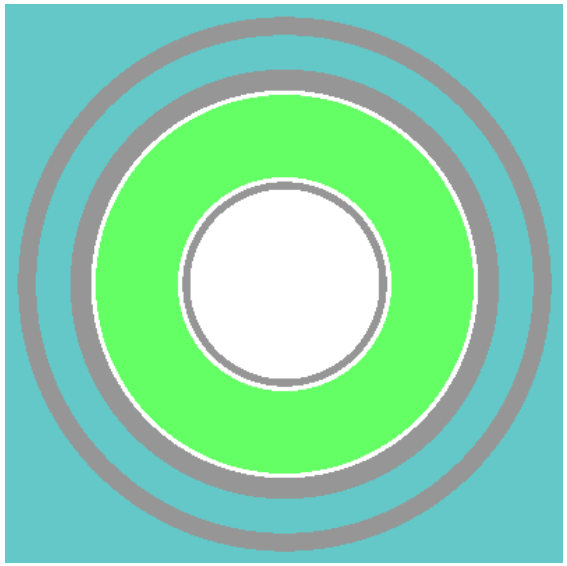
Fuel pin:



Guide tube:



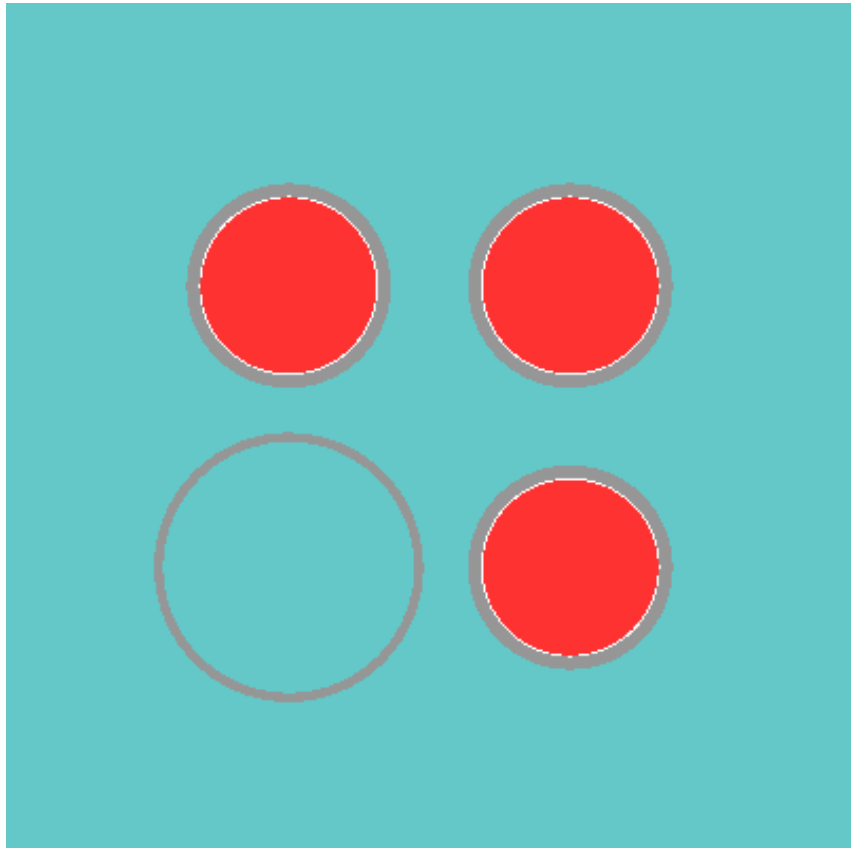
Burnable absorber:



Massachusetts Institute of Technology

A simple lattice

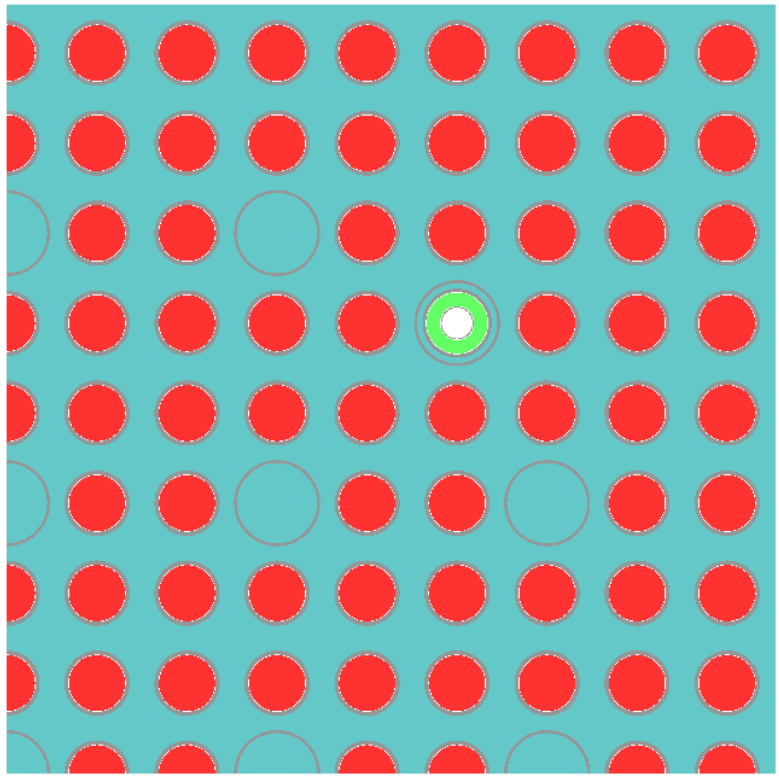
```
lattice = openmc.RectLattice()
lattice.dimension = [2, 2]
lattice.lower_left = [0.0, 0.0]
lattice.pitch = [pitch]*2
lattice.universes = [
    [fuel_pin,   fuel_pin],
    [guide_tube, fuel_pin]
]
lattice.outer = all_water
```



Massachusetts Institute of Technology

The quarter assembly

```
lattice = openmc.RectLattice()
lattice.dimension = [9, 9]
lattice.pitch = [pitch]*2
lattice.outer = all_water
lattice.lower_left = [-pitch/2.0]*2
lattice.universes = [
    [fuel_pin for i in range(9)] for j in range(9)]
lattice.universes[2][0] = guide_tube
lattice.universes[2][3] = guide_tube
lattice.universes[5][0] = guide_tube
lattice.universes[5][3] = guide_tube
lattice.universes[5][6] = guide_tube
lattice.universes[8][0] = guide_tube
lattice.universes[8][3] = guide_tube
lattice.universes[8][6] = guide_tube
lattice.universes[3][5] = burn
```



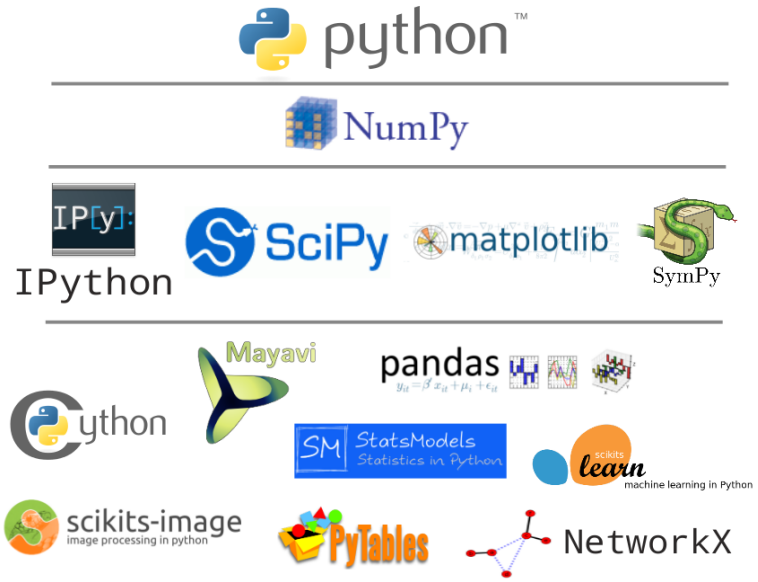
Massachusetts Institute of Technology

Outline

Massachusetts Institute of Technology

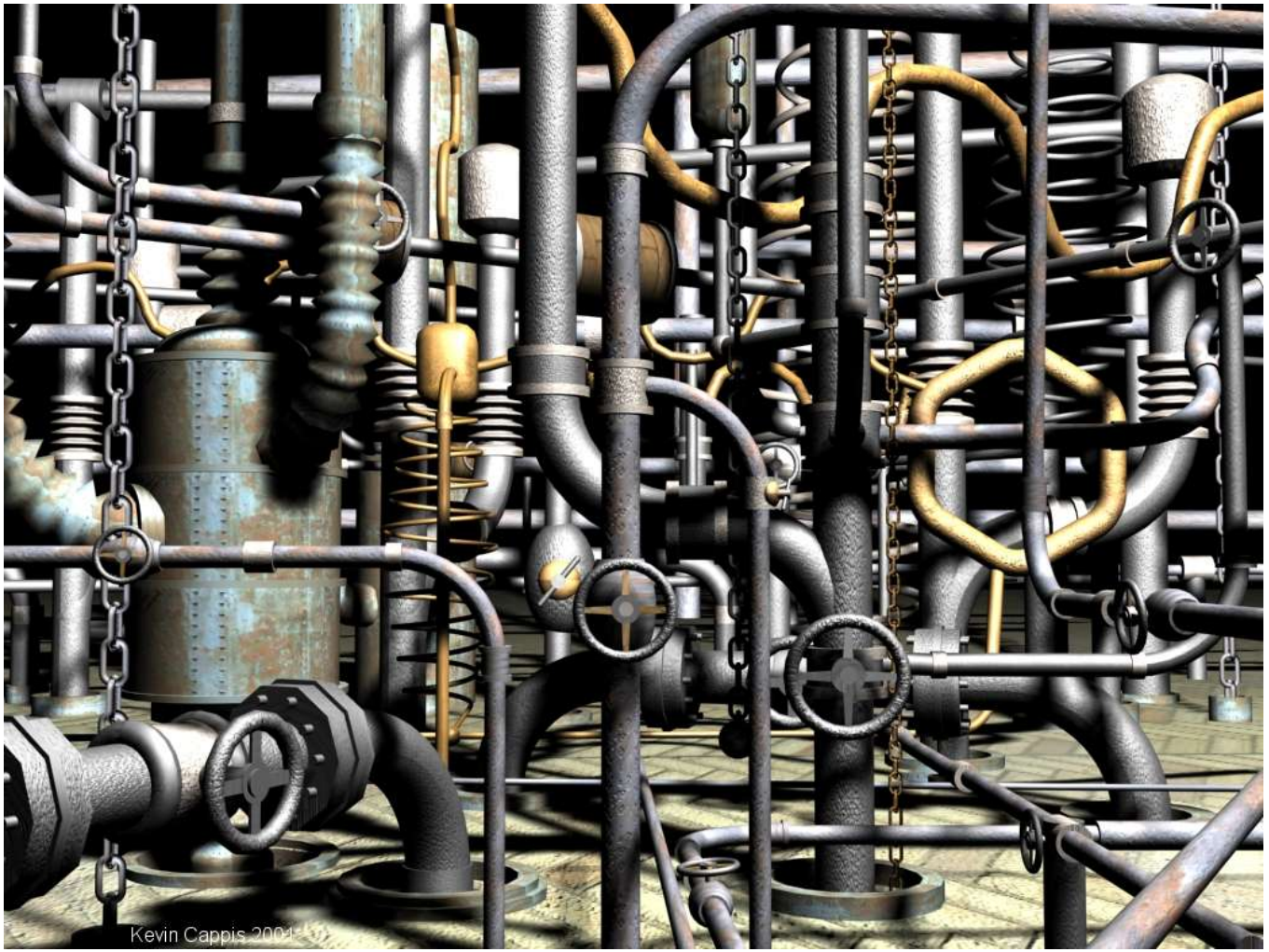
Leverage the Python Ecosystem

- Python API is built to serve your data processing needs
- Enables **tight coupling** with **Python's ecosystem** of scientific computing packages
- We will interactively explore OpenMC's features to **accelerate data workflows**



Massachusetts Institute of Technology

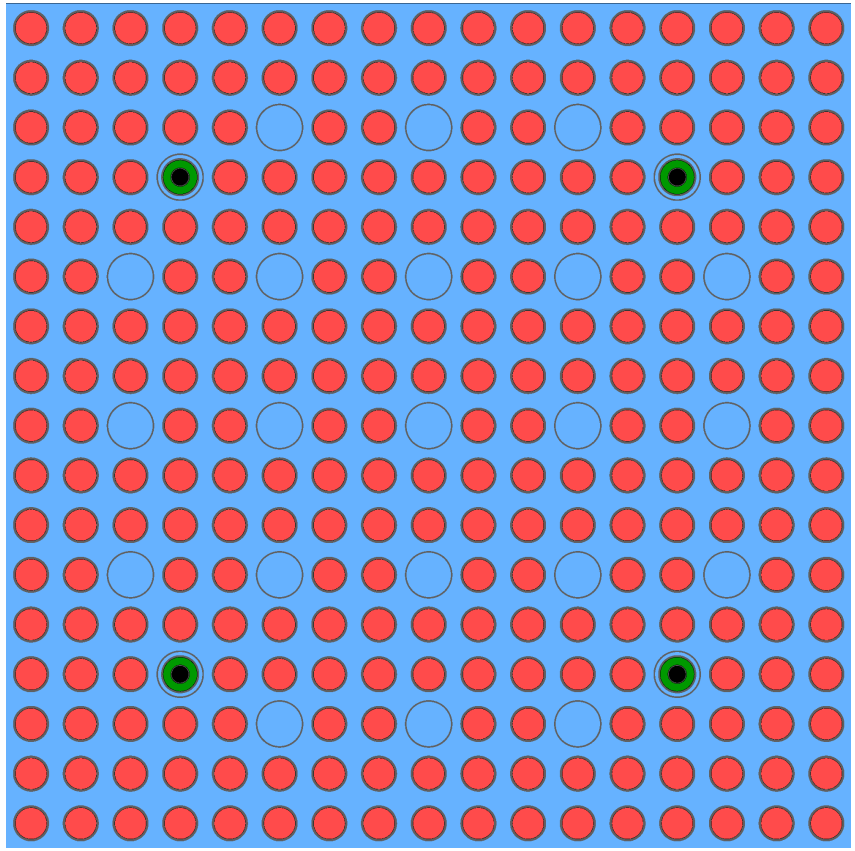
Everything You Need to Build a Data Pipeline!



Massachusetts Institute of Technology

Data Processing Demo Case Study

- A 17 $\times$ 17 PWR fuel assembly
- Pin cells comprised of:
  - UO2 fuel
  - Helium gap
  - Zircaloy clad
  - Light water moderator
  - Pyrex burnable poisons
- A variety of tallies for us to interactively analyze



Massachusetts Institute of Technology

StatePoint and Summary Files

StatePoint Files

- HDF5 output files with **source sites** and **tally data**
- Output **statepoint.*.h5** files for some or all batches
- Easily processed with `openmc.StatePoint`

Summary Files

- HDF5 output files with **geometric** and **materials**
- Output **summary.h5** file at beginning of simulation
- Easily processed with `openmc.Summary`

Massachusetts Institute of Technology

Simulation Metadata

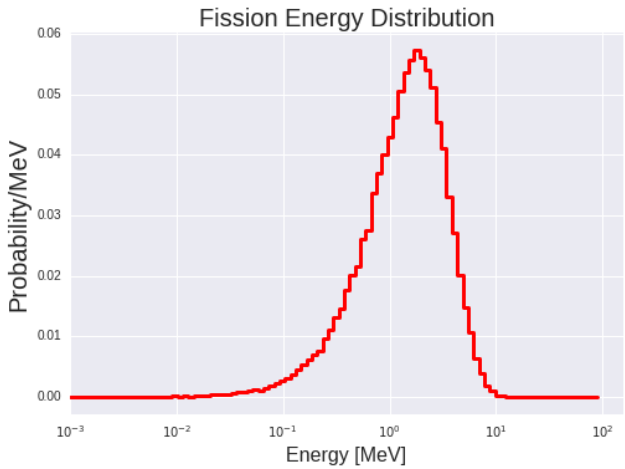
- Metadata in a `StatePoint`
  - Various estimators for k_{eff} (*i.e.*, `StatePoint.k_combined`)
  - Simulation parameters (*i.e.*, # batches, #particles / batch, etc.)
  - Simulation date, time, version, etc.

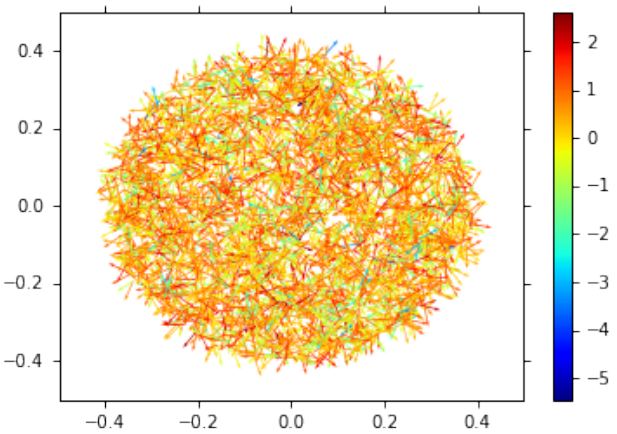
- Metadata in a `Summary`
  - Complete description of model geometry and materials
  - Methods to retrieve `Material`, `Surface`, `Cell`, etc. objects
  - **Link metadata** with `Tally` objects using `StatePoint.link_with_summary(...)`

Massachusetts Institute of Technology

Neutron Source Sites

- A `StatePoint` contains complete **source bank** for current batch
- Access source sites with `StatePoint.source` property
- Each **source site** includes the following data:
  - Spatial location `xyz`
  - Angular trajectory `uvw`
  - Energy [MeV] `E`
- Source sites can be left out of statepoint files to save space





Massachusetts Institute of Technology

Extracting Tallies from a StatePoint

- A `StatePoint` encapsulates metadata and `Tally` objects
- The `StatePoint.tallies` property is a Python dictionary of `Tally` objects indexed by integer ID
- The `StatePoint.get_tally(...)` method is highly recommended over direct access to `tallies` property
- Retrieves the first `Tally` found in the `StatePoint` matching one or more of the following descriptors:
  - Integer ID
  - String name
  - One or more `Filters`
  - One or more nuclides
  - One or more scores

Massachusetts Institute of Technology

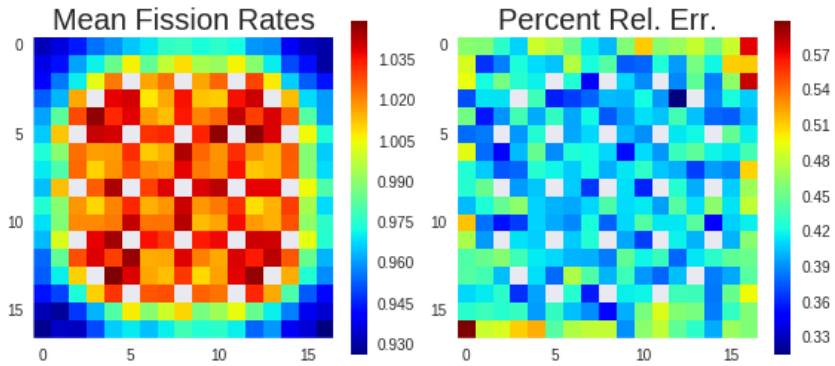
Inspecting Tallies

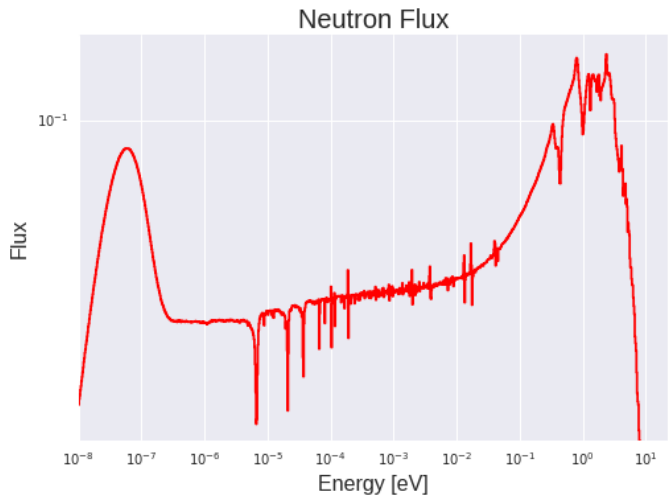
- View description of `Tally` with Python's `print(...)` method
- A few key `Tally` properties:
  - `Tally.id` - integer ID
  - `Tally.name` - string name
  - `Tally.scores` - list of scores
  - `Tally.nuclides` - list of nuclides
  - `Tally.filters` - list of `Filters`
  - `Tally.mean` - **3D array of sample means**
  - `Tally.std_dev` - **3D array of standard deviations**
- The sample means and uncertainties are stored in NumPy arrays indexed by filter bin, nuclide and score

Massachusetts Institute of Technology

Plotting Tally Data

- Python API closes the gap between your tally data and a rich variety of plotting tools:
  - **matplotlib** - Python's *de facto* general purpose plotting package
  - **seaborn** - High-level statistical graphics interface
  - **bokeh** - Interactive visualization in the browser
  - **statsmodels** - Statistical plotting inspired by R
- Share your coolest visualizations with the OpenMC community!





Massachusetts Institute of Technology

Pandas DataFrames for Tally Data

Problem: Tallies are Opaque

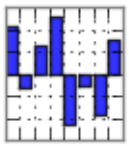
- OpenMC tally data is stored by filter, nuclide and score
- Difficult to rapidly index contiguous tally data arrays
- Need to quickly inspect data to debug simulation data workflows

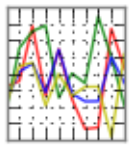
Solution: Pandas DataFrames

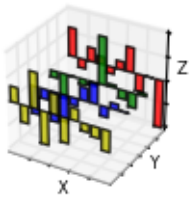
- Pandas `DataFrame` containers inspired by R
- A 2D data structure of columns and rows
- Includes an amazing assortment of features:
  - Mixed type data (*i.e.*, strings, integers)
  - Fancy indexing, slicing and subsetting
  - And a LOT more!

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



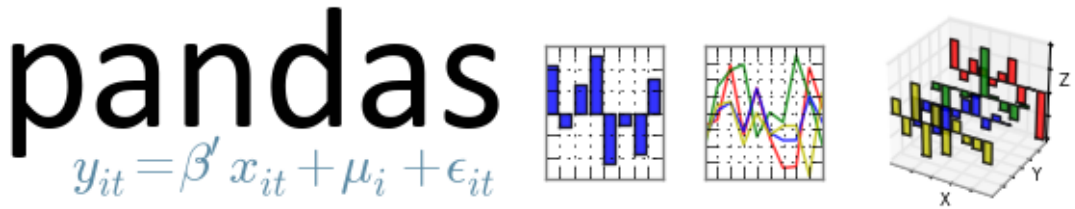




Massachusetts Institute of Technology

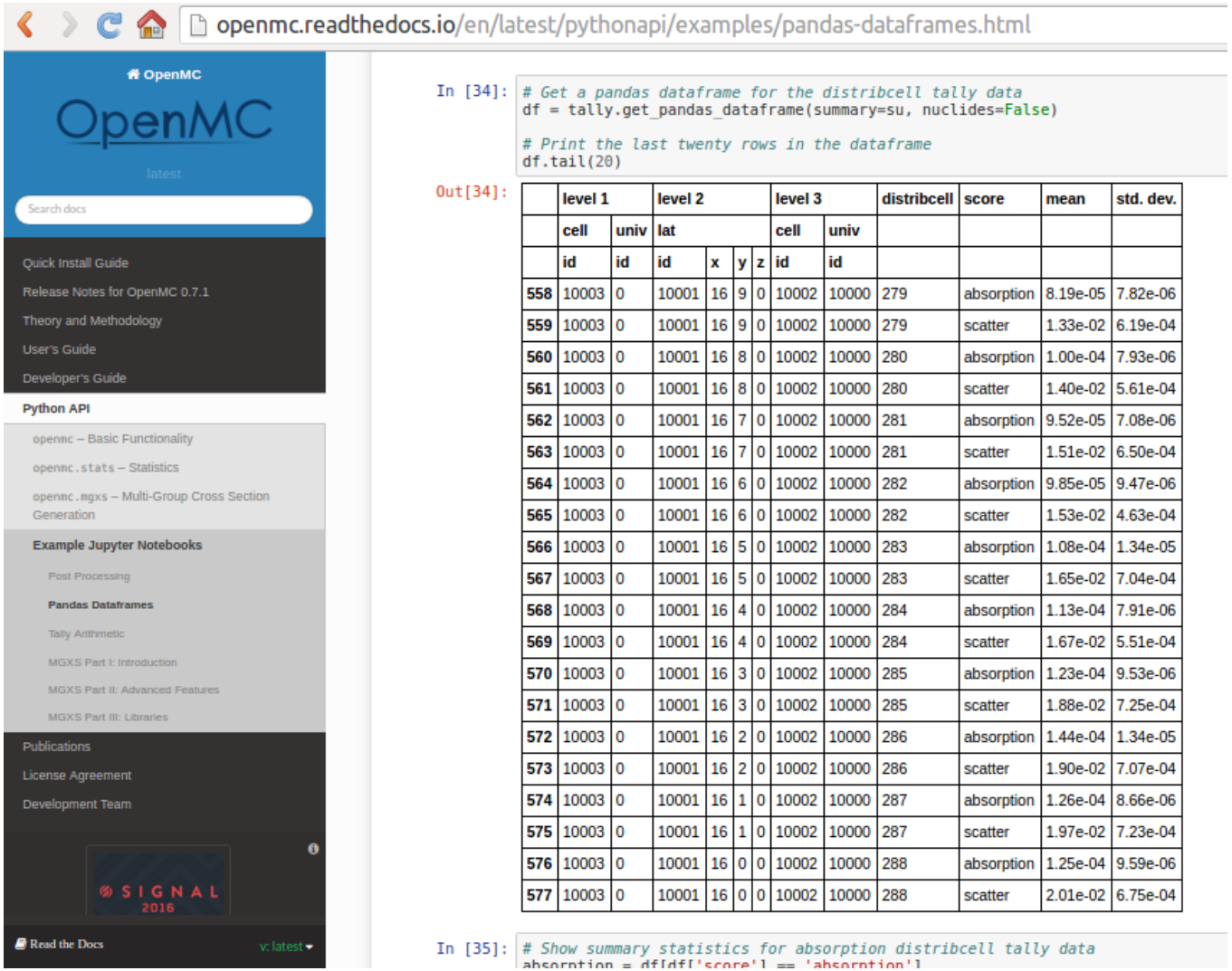
OpenMC + Pandas

- OpenMC `Tally` objects can construct a `DataFrame` with columns for filters, nuclides, and scores
- Each row corresponds to a sample mean and uncertainty for one combination of filter, nuclide and score bins
- Retrieve a Pandas `DataFrame` from tally data with `Tally.get_pandas_dataframe()`



Massachusetts Institute of Technology

OpenMC + Pandas (Cont.)



In [34]:
```
# Get a pandas dataframe for the distribcell tally data
df = tally.get_pandas_dataframe(summary=su, nuclides=False)

# Print the last twenty rows in the dataframe
df.tail(20)
```

Out[34]:

	level 1		level 2				level 3		distribcell	score	mean	std. dev.
	cell	univ	lat				cell	univ				
	id	id	id	x	y	z	id	id				
558	10003	0	10001	16	9	0	10002	10000	279	absorption	8.19e-05	7.82e-06
559	10003	0	10001	16	9	0	10002	10000	279	scatter	1.33e-02	6.19e-04
560	10003	0	10001	16	8	0	10002	10000	280	absorption	1.00e-04	7.93e-06
561	10003	0	10001	16	8	0	10002	10000	280	scatter	1.40e-02	5.61e-04
562	10003	0	10001	16	7	0	10002	10000	281	absorption	9.52e-05	7.08e-06
563	10003	0	10001	16	7	0	10002	10000	281	scatter	1.51e-02	6.50e-04
564	10003	0	10001	16	6	0	10002	10000	282	absorption	9.85e-05	9.47e-06
565	10003	0	10001	16	6	0	10002	10000	282	scatter	1.53e-02	4.63e-04
566	10003	0	10001	16	5	0	10002	10000	283	absorption	1.08e-04	1.34e-05
567	10003	0	10001	16	5	0	10002	10000	283	scatter	1.65e-02	7.04e-04
568	10003	0	10001	16	4	0	10002	10000	284	absorption	1.13e-04	7.91e-06
569	10003	0	10001	16	4	0	10002	10000	284	scatter	1.67e-02	5.51e-04
570	10003	0	10001	16	3	0	10002	10000	285	absorption	1.23e-04	9.53e-06
571	10003	0	10001	16	3	0	10002	10000	285	scatter	1.88e-02	7.25e-04
572	10003	0	10001	16	2	0	10002	10000	286	absorption	1.44e-04	1.34e-05
573	10003	0	10001	16	2	0	10002	10000	286	scatter	1.90e-02	7.07e-04
574	10003	0	10001	16	1	0	10002	10000	287	absorption	1.26e-04	8.66e-06
575	10003	0	10001	16	1	0	10002	10000	287	scatter	1.97e-02	7.23e-04
576	10003	0	10001	16	0	0	10002	10000	288	absorption	1.25e-04	9.59e-06
577	10003	0	10001	16	0	0	10002	10000	288	scatter	2.01e-02	6.75e-04

In [35]:
```
# Show summary statistics for absorption distribcell tally data
absorption = df[df['score'] == 'absorption']
```

Massachusetts Institute of Technology

Manipulating Tallies - Tally Slicing

- A `Tally` may contain more data than is needed for some downstream calculations
- **Tally "slicing"** extracts a subset of data from a `Tally` object
- The `Tally.get_slice(...)` method is designed to arbitrarily slice `Tally` objects by `Filter` bin(s), nuclide(s) or score(s)
- Slice out data for U-235 from a `Tally` with multiple nuclides:
  - `slice = tally.get_slice(nuclides=['U-235'])`
- The "sliced" tally data is encapsulated within a fully expressive `Tally`, while the original `Tally` remains intact

Massachusetts Institute of Technology

Manipulating Tallies - Tally Summation

- It may be useful to sum across a `Tally` for some downstream calculations (*i.e.*, nuclides, scores, etc.)
- **Tally "summation"** sums a subset of data in a `Tally` object
- The `Tally.summation(...)` method is designed to arbitrarily sum `Tally` objects across `Filter` bin(s), nuclide(s) or score(s)
- Sum data for U-235 and U-238 from a `Tally`:
  - `sum = tally.summation(nuclides=['U-235', 'U-238'])`
- The "summed" tally data is encapsulated within a fully expressive `Tally`, while the original `Tally` remains intact

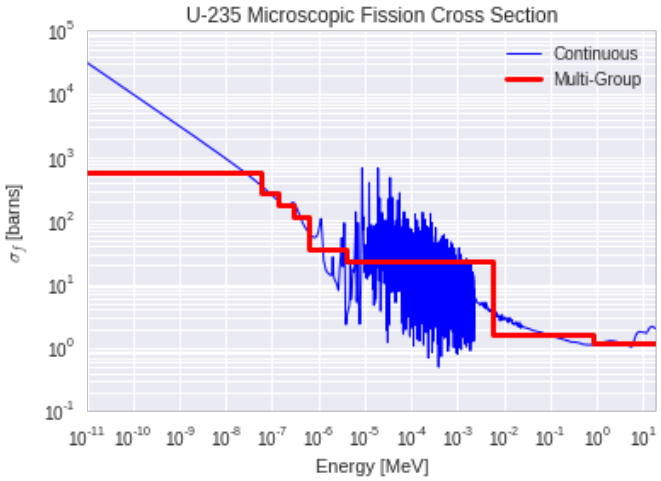
Massachusetts Institute of Technology

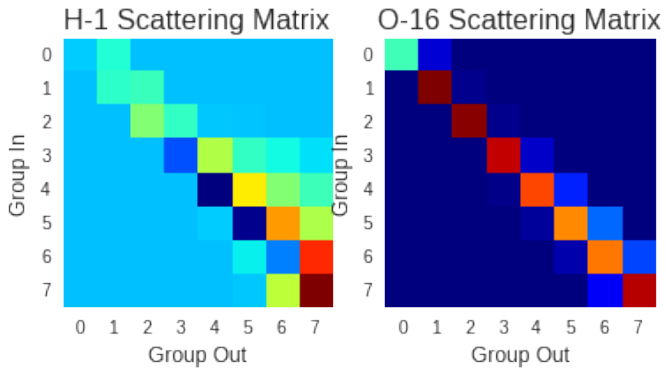
Manipulating Tallies - Tally Arithmetic

- Many data workflows require the arithmetic combination of multiple tallies
- **Tally "arithmetic"** applies binary operations to `Tally` objects
- Tally arithmetic automatically performs **uncertainty propagation** through each operation
- Add two `Tally` objects and divide by a third:
  - `tallyD = (tallyA + tallyB) / tallyC`
- The resultant tally data is encapsulated within a fully expressive `Tally`, while the original `Tally` operands remains intact
- **Fast** and **efficient** with **NumPy vectorization**

Massachusetts Institute of Technology

Multi-Group Cross Section Generation



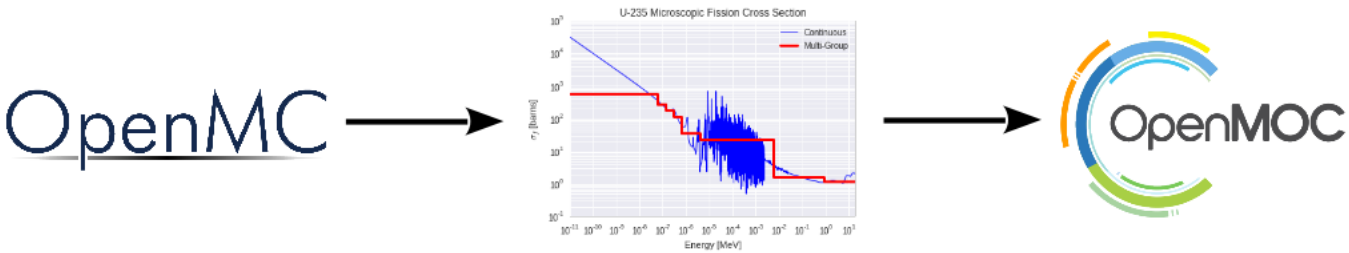


- New `openmc.mgxs` package to calculate **multi-group cross sections** (MGXS)
- Built atop of Python API using tally arithmetic, slicing, etc.
- Tight integration with custom Pandas `DataFrames`
- **Fast** and **efficient** with **NumPy vectorization**
- Designed for next-gen **high-fidelity neutron transport codes**

Massachusetts Institute of Technology

Pipelining OpenMC MGXS to OpenMOC

- MIT CRPG's OpenMOC deterministic neutron transport code is fully equipped to easily make use of `openmc.mgxs`
- Compute OpenMC MGXS `Library` with 5-10 lines of Python
- Inject MGXS `Library` into OpenMOC with 1 line of Python
- **Use a single constructive solid geometry for both OpenMC and OpenMOC calculations**



Massachusetts Institute of Technology

Outline

Massachusetts Institute of Technology

Installation on Ubuntu

For users with Ubuntu 15.04 or later, a binary package for OpenMC is available through a Personal Package Archive (PPA) and can be installed through the APT package manager.

```
sudo apt-add-repository ppa:paulromano/staging
```

```
sudo apt-get update
```

```
sudo apt-get install openmc
```

Massachusetts Institute of Technology

Building from Source

To compile from source you will need a Fortran compiler (e.g gfortran v4.6 or greater), CMake and an HDF5 library compiled with your Fortran compiler (and optionally MPI).

```
git clone https://github.com/mit-crpg/openmc.git
```

```
cd openmc && git checkout master
```

```
mkdir build && cd build && cmake .. && make
```

For more details, see `http://openmc.readthedocs.io/en/latest/`

Massachusetts Institute of Technology

Question and Bugs

Questions can be submitted to the OpenMC User group on Google Groups

- Search for OpenMC on https://groups.google.com/
- or https://groups.google.com/forum/#!forum/openmc-users

Bugs or interest in new features can be reported on the github page by creating a New Issue

- https://github.com/mit-crpg/openmc/issues
- For bugs, please include input file reproducing the error
- For new features, new developers are always welcome!

Massachusetts Institute of Technology

OpenMC Papers at PHYSOR 2016

- **Monday Afternoon, 4:25pm (Columbine)** J.Walsh, B.Forget, K.Smith, F.Brown: *On-The-Fly Doppler Broadening of Unresolved Resonance Region Cross Sections Via Probability Band Interpolation*
- **Monday Evening 6:00pm** P.Romano: *Efficacy of Hardware Threading for Monte Carlo Particle Transport Calculations on Multi- And Many-Core Systems*
- **Wednesday Morning, 11:50am (Columbine)** Z.Liu, K.Smith, B.Forget: *A Cumulative Migration Method for Computing Rigorous Transport Cross Sections and Diffusion Coefficients for LWR Lattices with Monte Carlo*
- **Wednesday Afternoon, 2:20pm (Sawtooth)** W.Boyd, P.Romano, S.Harper: *Equipping OpenMC for the Big Data Era*
- **Thursday Morning, 8:00am (Columbine)** M.Ellis, D.Gaston, B.Forget, K.Smith: *Continuous Temperature Representation in Coupled OpenMC/MOOSE Simulations*

Massachusetts Institute of Technology

Other CRPG Papers at PHYSOR 2016

- **Monday Afternoon, 4:25pm (Camas)** S.Shaner, G.Gunow, K.Smith, B.Forget: *Verification of the 3D Method of Characteristics Solver in OpenMOC*
- **Tuesday Morning, 9:15am (Columbine)** K.Smith: *Nodal Diffusion Methods: Understanding Numerous Unpublished Details*
- **Tuesday Afternoon, 1:30pm (Columbine)** P.Ducru, V.Sobes, B.Forget, K.Smith: *On Methods for Conversion to Multipole Formalism*
- **Thursday Morning, 8:25am (Boiler)** G.Gunow, S.Shaner, B.Forget, K.Smith: *Reducing 3D MOC Storage Requirements with Axial On-the-Fly Ray Tracing*

Massachusetts Institute of Technology

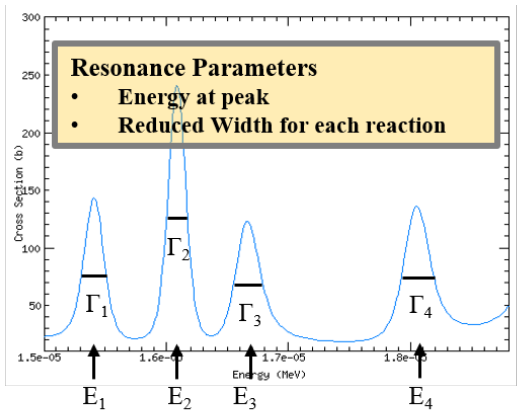
Upcoming Features

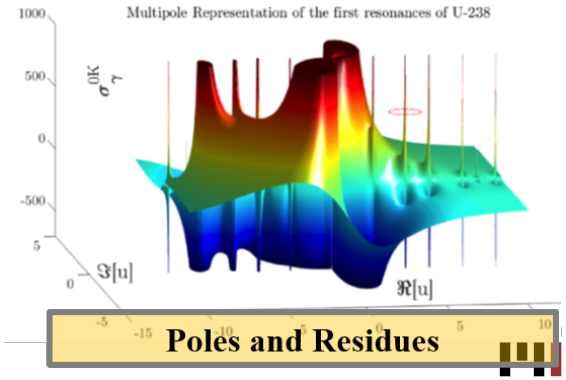
- On-the-fly Doppler Broadening
- Domain Decomposition
- Multiphysics
- Differential Tallies
- Depletion
- Transient Analysis

Massachusetts Institute of Technology

Doppler Broadening

- On-the-fly Doppler broadening based on the windowed multipole method (300 of the 423 nuclides in ENDF/BVII.1)
- On-the-fly URR sampling or equiprobable E-T surfaces
- Kernel Reconstruction for remaining 100 nuclides
- Currently developing an HDF5 library format to replace ACE files

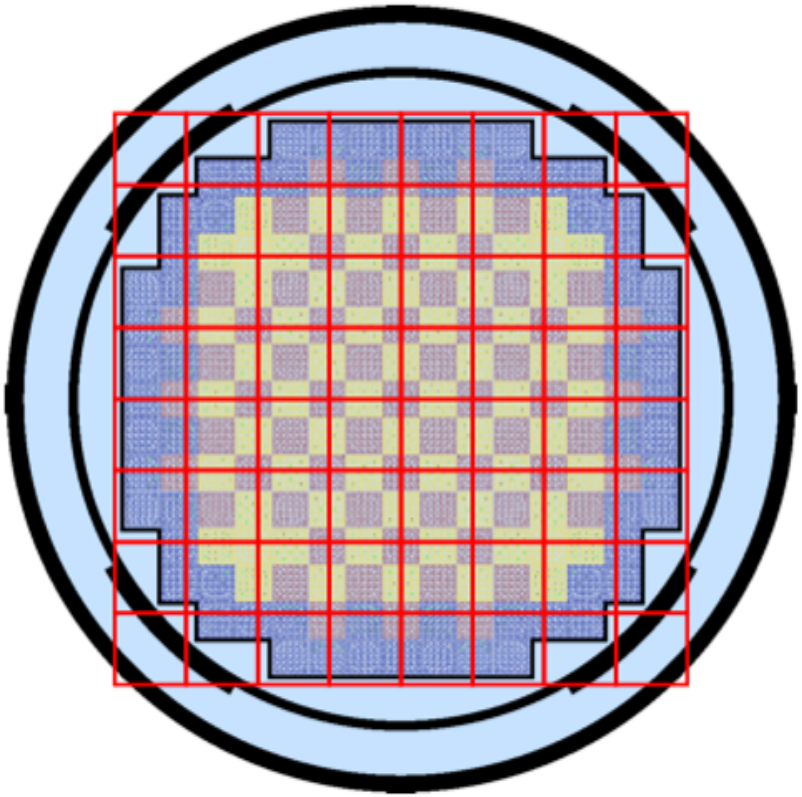




Massachusetts Institute of Technology

Domain Decomposition

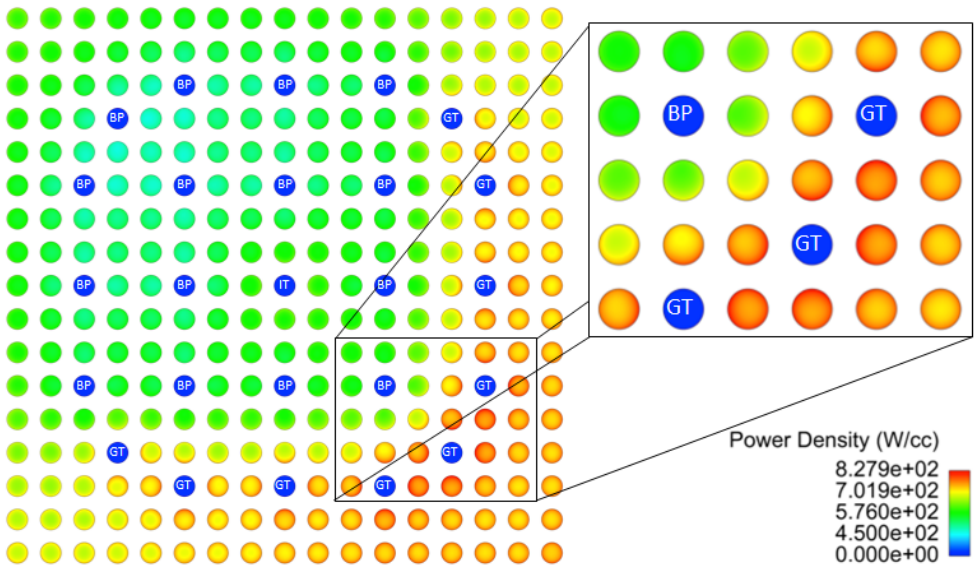
- Arbitrary mesh domain decomposition for distributing memory from tallies and material compositions
- Load balancing achieved by domain replication
- Currently under review



achusetts
ute of
ecnnology

Multiphysics

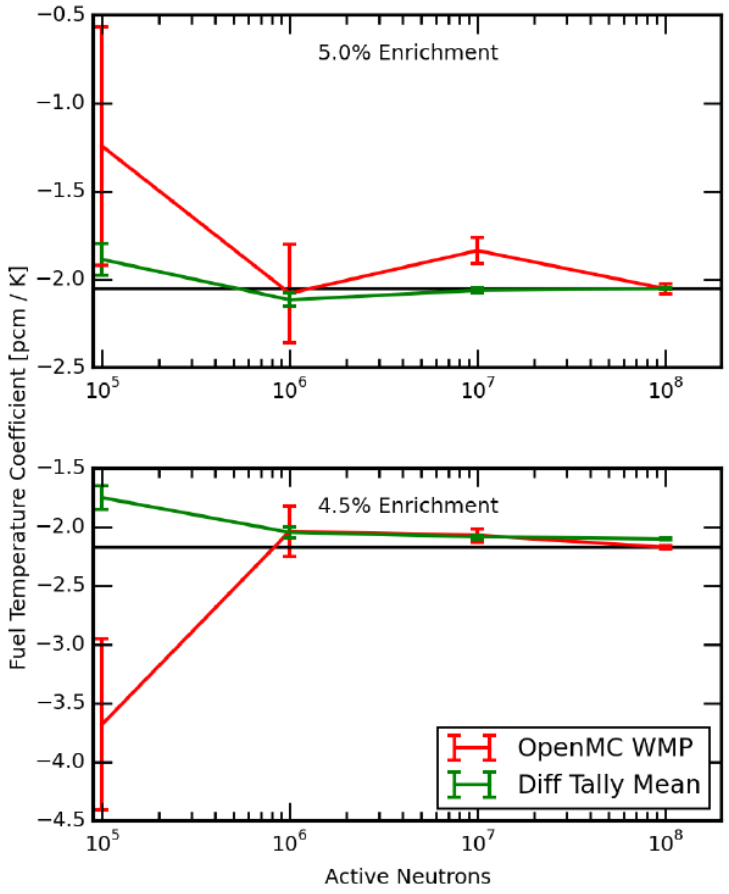
- OpenMC was built as a library and incorporated as a MultiApp in MOOSE
- Functional expansion tallies were implemented to transfer data between CSG and FEM
- Continuous material tracking was implemented to allow for continuously varying temperature and nuclide densities within a CSG cell



Massachusetts Institute of Technology

Differential Tallies

- Implemented differential tallies to tally first order derivatives with density changes
- Extended to include temperature using first order derivatives of the multipole equations
- OpenMC will thus be able to provide accurate temperature and density reactivity coefficients in a single run
- Can also provide cross section temperature derivatives

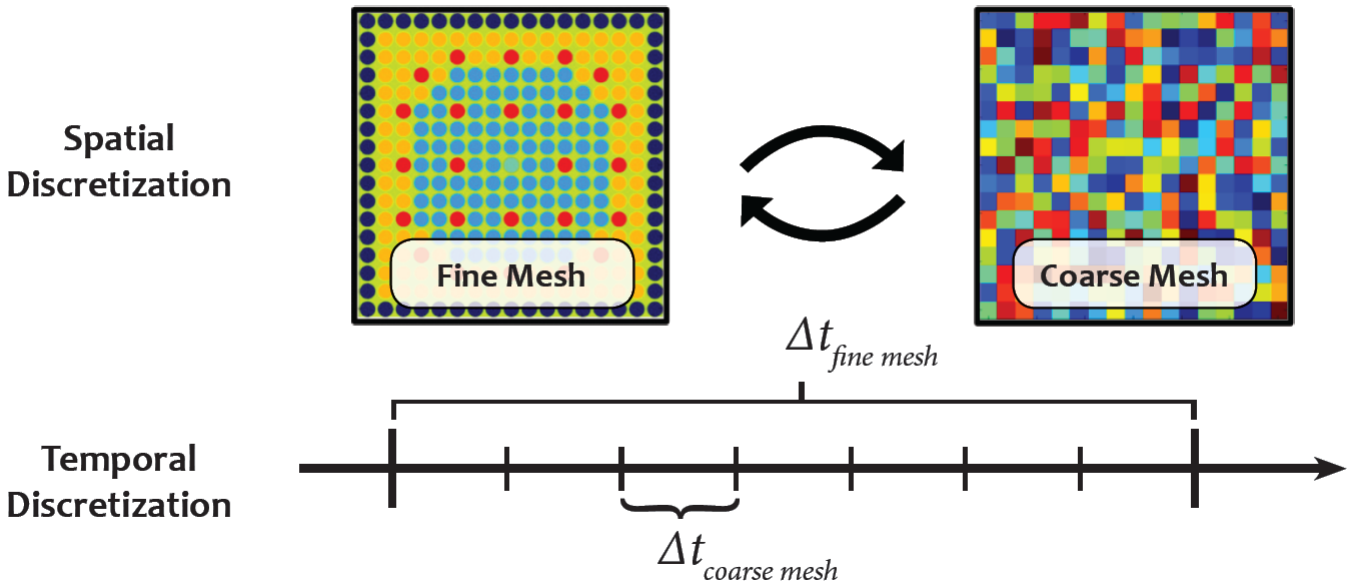


Depletion

When will OpenMC add depletion?

Transient Analysis

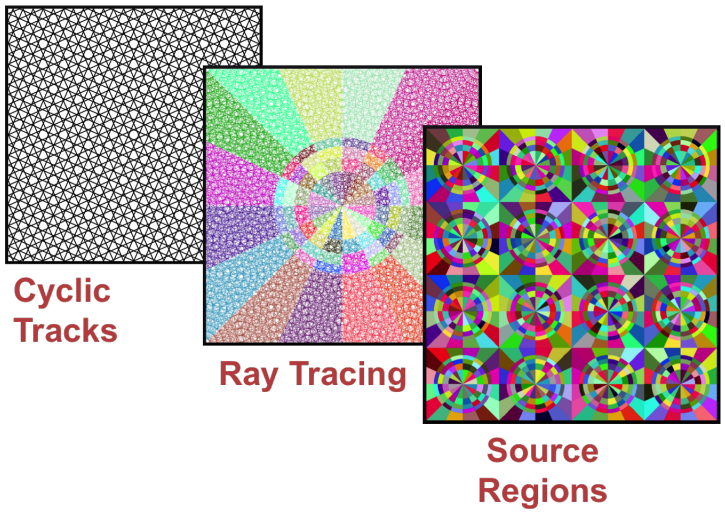
- Implementing multigrid amplitude function approach where the CMFD operator will be advanced on fine time steps with parameters updated from OpenMC on coarse time steps



Massachusetts Institute of Technology

Other MIT-CRPG code - OpenMOC

- A Method of Characteristics code platform for solving the multigroup 2D and 3D form of the neutron transport equation.
- MOC: "Monte Carlo on Rails"
- Fast and accurate deterministic transport simulations.
- Ongoing development of data pipeline between OpenMC and OpenMOC to utilize multigroup XS generated from OpenMC.

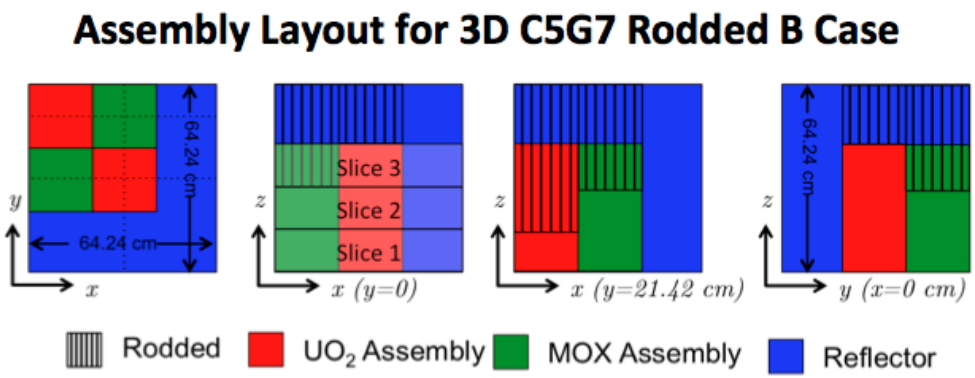


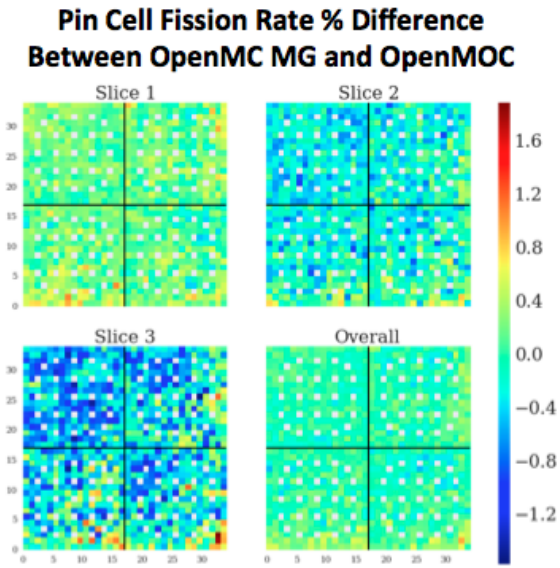
OpenMOC

Massachusetts Institute of Technology

Cross-code verification - OpenMC MG and OpenMOC

- Multigroup Monte Carlo solver recently added to OpenMC by Adam Nelson.
- Allows for cross-code comparison with deterministic transport codes, such as OpenMOC.





Massachusetts Institute of Technology