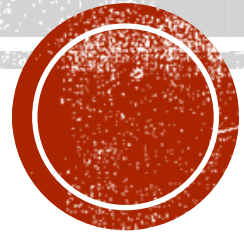
# OPENMC IS COOL

Day 2 – Syntax and Examples



# GOALS FOR TODAY

**Minimum Things To Take Away:**

- Creating Materials (composition (element, vs nuclide), temperature, and density)
  - Enrichment of U235
- Specifying Geometry (planes, cylinders)
  - Surfaces / Halfspaces
  - Cells
- Using Universe.plot()
- Boundary Conditions (mirror, vacuum)
- Sources (point, box)
- Settings
- Tallies (Fission Rate)
- Visualization

# ONE OF YOUR BEST RESOURCES WILL BE THE OPENMC DOCUMENTATION FOR BOTH METHODOLOGY AND SYNTAX QUESTIONS:

HTTPS://DOCS.OPENMC.ORG/EN/STABLE/

# MATERIALS

- Creating a material that is a combination of isotopes that can be applied to regions we specify in the code.
- Can specify composition (element, vs nuclide), temperature, density
- There is a built in function for enrichment of U235

# SURFACES/BOUNDARY CONDITIONS

- The way that we create regions in space in the code
- Surfaces are 2nd order 3D surfaces. Functions of x, y, and z
  - Spherical Surface Equation:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - R^2 = 0$$

  - General Surface Equation:

$$Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz + Gx + Hy + Jz + K = 0$$

- A **boundary condition** instructs a particle what to do when it reaches the edge of a surface
  - Most surface boundary conditions are simply "transmission" which allows the particle through
  - Outside boundary conditions my be "vacuum" or "reflective" depending on what behavior you want to simulate

# HALF SPACES

- Each surface creates 2 half spaces
  - Ie. Spheres create a half space "inside" and "outside"
  - The + half space is where the surface equation evaluates to a + number (and the opposite is true for a – half space)

# CELLS

- An region defined in space (either a half space or a combination of half spaces) that also has a defined material fill

# SOURCES

- To begin the first cycle, we must set a starting location, energy, direction, etc. for the particles in the simulation
- We guess where that source will be and specify that with a source in OpenMC
- You can specify point, box, or other kinds of sources

# UNIVERSES

- All cells (to be run) must be put in a universe for the code to run
- A sub-universe can be created with parameters, and that universe could be applied as a fill to a cell (we will not be doing this for now)

# USING UNIVERSE.PLOT()

# EXPORT TO XML

- OpenMC reads your code and creates xml files
- Required xml files:
  - Geometry
  - Materials
  - Settings
- Optional xml files:
  - Tallies
  - Plots
  - (a couple others we won't use)
- You specify what gets put in the XML file with Python, then OpenMC creates those files
- When OpenMC runs, it runs based on what's in the XMLs

# SETTINGS

- Required components of settings
  - run_mode ( “eigenvalue” calculates keff)
  - particles
  - batches (cycles)
  - inactive cycles
  - source

# RUNNING OPENMC

- In python just use the function openmc.run()

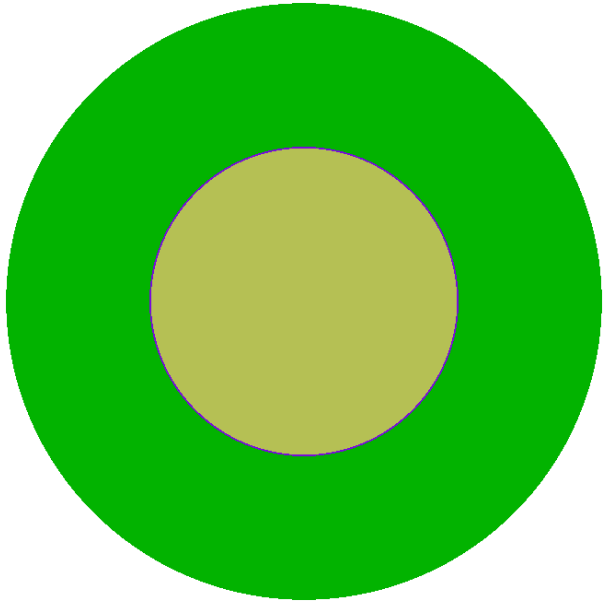
- If running from the terminal, there should also be an OpenMC executable that you can call to run the xml files in your directory

# TALLIES

- Specify Filters
  - Energy Filter
  - Material Filter
- Specify Score
  - Flux
  - Total Reaction Rate
  - Fission Rate
  - Absorption Rate
  - etc. (many more)
- Create tallies object and export to .xml
- Post Process Tallies

# VISUALIZATION – CREATING 2D PLOTS

- Create a plot and set the following parameters:
  - basis = what plane (xy, zy, yz)
  - Origin
  - width
  - pixels = resolution of image
  - color_by = color by ‘cell’ or ‘material’ (or a couple others)
  - filename

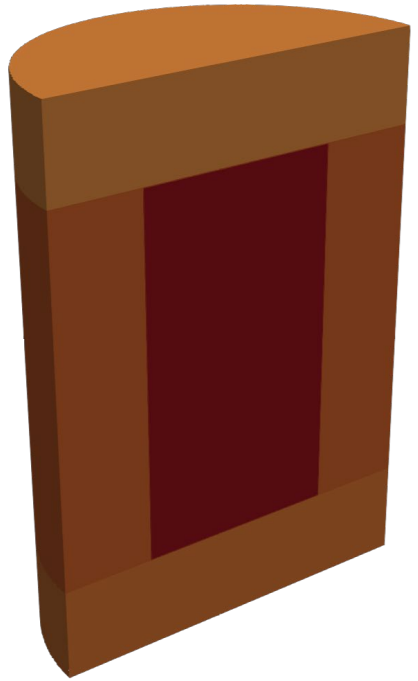


# VISUALIZATION – CREATING 3D PLOTS

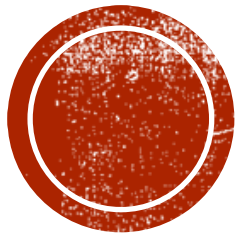
A Voxel plot is a way to visualize your project in 3D

You will create a voxel plot and specify

- type = 'voxel'
- Origin → (x, y, z)
- width → (x_width, y_width, z_width)
- pixels → (x_resolution, y_resolution, z_resolution)
- Filename



- Note: You will need to download ParaView (or something similar) in order to view these



# DEPLETION

- Specify
  - Reactor power
  - Timestep units (MWd/kg = MegaWatt Day / kg of Heavy Metal)
  - Timesteps
  - The volume of all depletable material
- Pass your model into a model object
- Specify your model object and depletion data in the operator object
- Chose an integrator
  - The integrator is pretty much which numerical method should be used. CECM is often the best operator. The simplest/fastest is usually Predictor Integrator.
- Specify the operator and other settings in the integrator object and then run the depletion calculation