OpenMC Course Introduction

Joint ICTP-IAEA Workshop on Open-Source Nuclear Codes for Reactor Analysis
August 8, 2023

Course Logistics

Logistics

Instructors: Jiwon Choe (KAERI) & Patrick Shriwise (ANL)

Asking Questions:

- In-person
- Chat on Zoom (during sessions)

Interactive Sessions

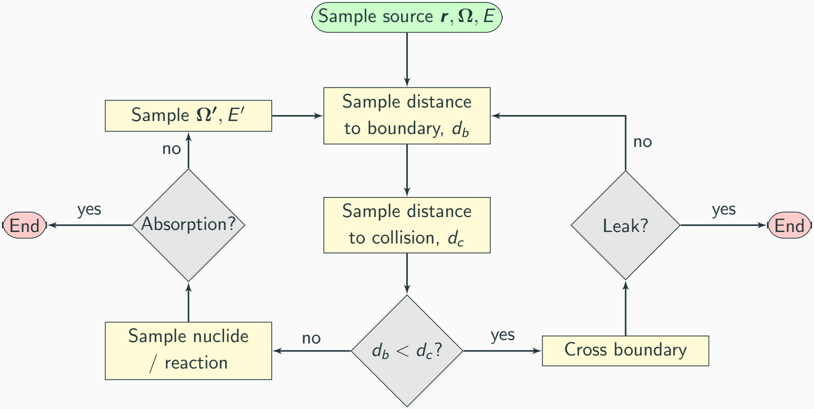
- You will be using **Jupyter Lab** for demonstrations and (optionally) take-home exercises
- Instructor will give live demo for each session, and you can follow along in your own Jupyter Lab instance (dual monitor / side-by-side)
- The URL provided to you will be available all week but will be shutdown at the end of the week — "notebooks" can be downloaded at anytime

Monte Carlo Basics

Monte Carlo Particle Transport

- Analysis of nuclear reactors, fusion devices, radiation shielding, and other problems relies on ability to solve particle transport equations
  - *Deterministic methods:* discrete ordinates, method of characteristics, collision probability method, diffusion theory
  - *Monte Carlo (MC) method:* directly simulate life of individual particles using known probability distributions
- MC method confers a number of benefits:
  - Use of continuous-energy interaction data (no grouping necessary)
  - No spatial approximations necessary
  - Parallelization is “simple” since particles do not interact with one another
  - Some classes of problems are very difficult to solve at all with deterministic methods (e.g., high-energy physics)
- Biggest impediment to wider use is *time to solution*

Neutral particle transport



Tallies

Monte Carlo is well-suited to calculating volume integral quantities of the form:

$$X = \int d\boldsymbol{r} \int d\boldsymbol{\Omega} \int dE \; f(\boldsymbol{r}, \boldsymbol{\Omega}, E)\psi(\boldsymbol{r}, \boldsymbol{\Omega}, E)$$

During a simulation, physical quantities of interest (called *tallies* or *detectors*) are accumulated as:

$$\hat{X} = \frac{1}{N} \sum_{i \in T} w_i \ell_i f_i$$

Statistics

At the end of a simulation, we have a set of realizations for each tally, $\hat{X}_1, \hat{X}_2, \dots, \hat{X}_N$. We can calculate mean and variance as

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} \hat{X}_i$$

$$s_X^2 = \frac{1}{N-1} \left(\frac{1}{N} \sum_{i=1}^{N} \hat{X}_i^2 - \bar{X}^2 \right)$$

Problem Types

- For **fixed source** problems, the source of particles is known *a priori*, e.g., 100 neutrons/sec from an isotropic point source
- When neutrons from fission are the primary source, the distribution of source sites is not known *a priori* because it depends on the flux, which is what we're solving for

k Eigenvalue Algorithm

Guess initial source distribution and k
for $i = 1 \to n_{generations}$ **do**
  for $j = 1 \to n_{particles}$ **do**
    Sample neutron from source bank
    Track neutron until death, at each **collision** storing

$$n = \left\lfloor \frac{\nu \Sigma_f}{\Sigma_t} + \xi \right\rfloor \quad \text{fission sites}$$

  Sample $N = n_{particles}$ neutrons from N' fission sites collected
  Calculate $k^{(i)} = N'/N$

Inactive generations

- Our goal is to estimate physical quantities (e.g., ^{235}U fission rate) resulting from a source
- In the generation algorithm, we have to wait until the spatial distribution of source sites converges (otherwise, our results would be biased by the arbitrary source guess)
- Simulation is broken up into *inactive* and *active* generations
- For problems with large dominance ratio, hundreds of generations may need to be discarded

OpenMC Intro

Objectives

The overarching objectives of the OpenMC project:

- Open source contribution model, freely available
- Extensible for research purposes
- Adopt best practices for software development
- Ease of installation, minimize third-party dependencies
- High performance, scalable on HPC resources
- Use best physics models when possible
- Fun to use, and thriving user and developer community!

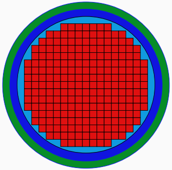
OpenMC: Overview of features

- **Modes**: Fixed source, k-eigenvalue calculations, volume calculations, geometry plotting
- **Geometry**: Constructive solid geometry, CAD-based, unstructured mesh (tallies only)
- **Solvers**: Neutron and photon transport, depletion, stochastic volume calculation
- **Data**: Continuous energy or multigroup cross sections, multipole for on-the-fly Doppler broadening

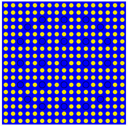
What makes OpenMC unique?

- Programming interfaces (C/C++ and Python)
- Nuclear data interfaces and representation
- Tally abstractions
- Parallel performance
- Development workflow and governance

Parallel Performance

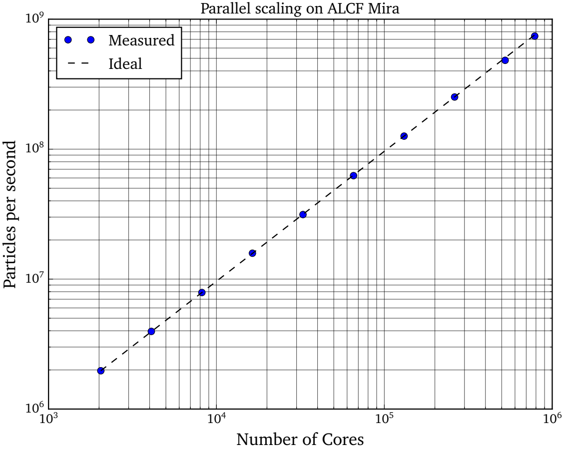


Core

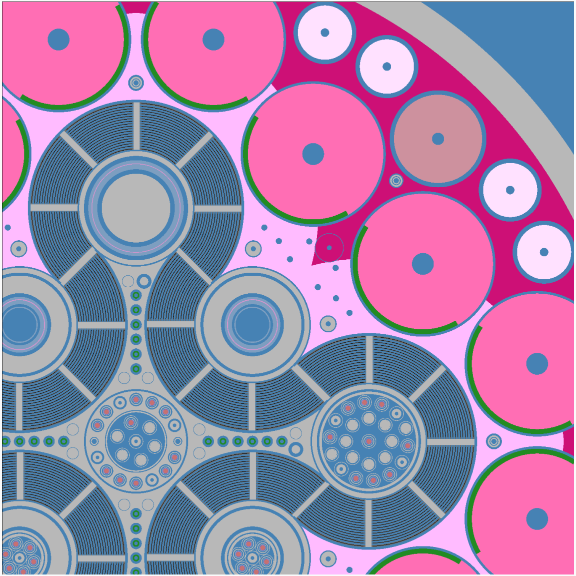


Assembly

- ALCF Mira supercomputer
- 49,152 nodes, 786,432 cores
- 4 hw threads/core = 3,145,728 threads



Example: Advanced Test Reactor



Software Architecture

- Mixed **C++** and **Python** codebase
- **CMake** build system for portability
- Distributed-memory parallelism via **MPI**
- Shared-memory parallelism via **OpenMP**
- Version control through **git**
- Code hosting, bug tracking through **GitHub**
- Regression/unit tests run on **GitHub Actions** CI platform

Upcoming developments

- GPU porting (Exascale Computing Project)
- Multiphysics coupling
- Fusion shutdown dose rate (SDR) calculations
- Unstructured mesh support
- Methods to support molten salt reactor design

Resources

- **Code:** https://github.com/openmc-dev/openmc
- **Docs:** https://docs.openmc.org
- **Nuclear Data:** https://openmc.org
- **Forum:** https://openmc.discourse.group