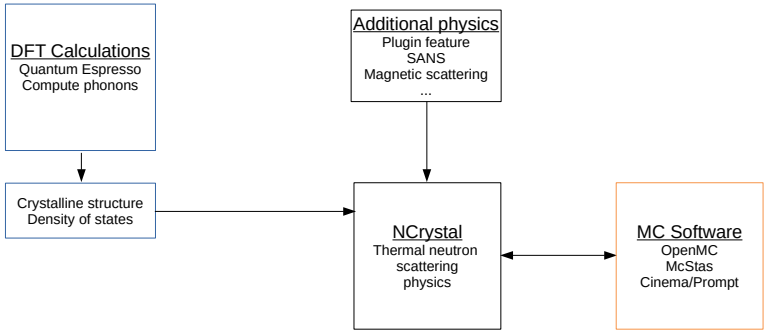
EUROPEAN
SPALLATION
SOURCE

An Introduction to OpenMC

HighNESS Thermal Scattering Kernel School

JOSÉ IGNACIO MÁRQUEZ DAMIÁN,
SPALLATION PHYSICS GROUP, ESS

The big picture



Introduction to radiation transport.

Introduction

In this course we will study the transport of neutrons using the Monte Carlo method, as implemented in OpenMC.
We will start with a review of fundamental concepts related to radiation transport, which are the base to understand the practical aspects of the code.

Fundamental concepts

Radiation transport analysis is usually done by studying a continuous quantity: particle density.

Neutron density:

$$n\left(\vec{r}, \vec{v}, t\right) \mathrm{d}\vec{r}\, \mathrm{d}\vec{v} = n\left(\vec{r}, E, \hat{\Omega}, t\right) \mathrm{d}\vec{r}\, \mathrm{d}E\, \mathrm{d}\hat{\Omega}$$

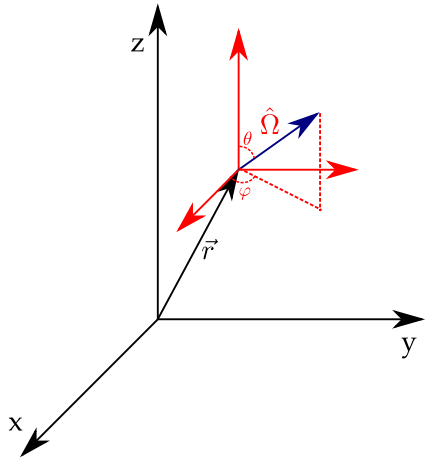
number of neutrons at time t in the volume $\mathrm{d}\vec{r}$ around point $\vec{r}$ with:

- velocities $\mathrm{d}\vec{v}$ around $\vec{v}$, or
- energies $\mathrm{d}E$ around E and directions $\mathrm{d}\hat{\Omega}$ around $\hat{\Omega}$.

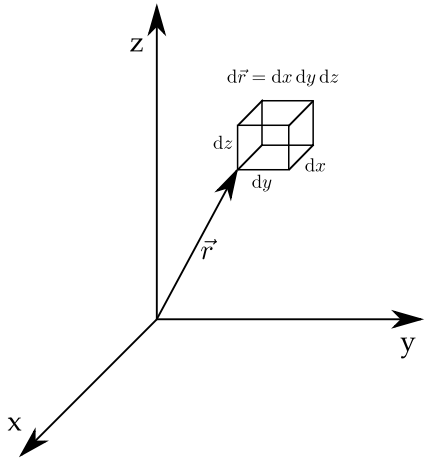
(both descriptions are equivalent).
Direction $\hat{\Omega}$ is given by the velocity, so:

$$\hat{\Omega} = \frac{\vec{v}}{v}$$

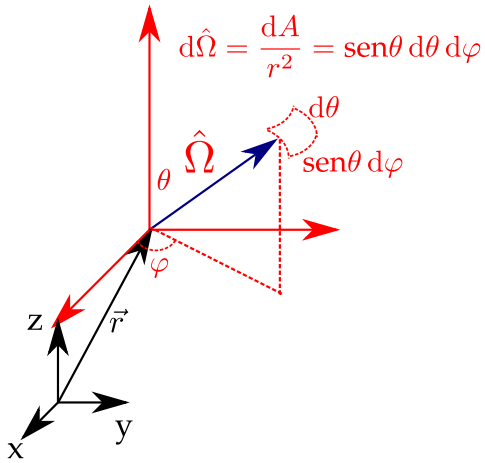
Fundamental concepts (cont.)



Fundamental concepts (cont.)



Fundamental concepts (cont.)



Fundamental concepts (cont.)

The product of neutron density by the magnitude of the velocity is defined as *angular flux*:

$$\psi\left(\vec{r}, E, \hat{\Omega}, t\right) \equiv vn\left(\vec{r}, E, \hat{\Omega}, t\right)$$

The integral of the angular flux over all directions is the *scalar flux*:

$$\phi\left(\vec{r}, E, t\right) = \int_{4\pi} \mathrm{d}\hat{\Omega}\psi\left(\vec{r}, E, \hat{\Omega}, t\right)$$

Fundamental concepts (cont.)

A neutron flux $\psi\left(\vec{r}, E, t\right)$ interacting with matter generates a reaction rate R:

$$R\left(\vec{r}, E, \hat{\Omega}, t\right) = \Sigma\psi\left(\vec{r}, E, \hat{\Omega}, t\right)$$

where the proportionality constant Σ is the *macroscopic cross section* and has dimensions $[L]^{-1}$, usually cm^1. The macroscopic cross section is the product of the number density of atoms times the *microscopic cross section*:

$$\Sigma = N\sigma$$

The microscopic cross section is a property of each nuclide, and can depend on the energy of the neutron and other properties. Its dimension are $[L^2]$, usually $\mathrm{b} = 10^{-24}\ \mathrm{cm}^2$.

Fundamental concepts (cont.)

If the cross section does not depend on the incident direction of neutrons (*):

$$R\left(\vec{r}, E, t\right) = \Sigma\phi\left(\vec{r}, E, t\right)$$

as the neutrons hitting the target from one direction do not interfere with neutrons coming from a different direction.

(*) This is an assumption used in OpenMC and other Monte Carlo codes because simplifies the simulation, but it is not generally true and in particular when oriented NCrystal materials are used.

Fundamental concepts (cont.)

Neutron flux as defined previously is useful to describe the density of neutrons in a point and calculate reaction rates, but it does not have the properties of "flux" as used in electromagnetism or fluid mechanics: it is not a vector quantity and it is not related to the flow of particles from one place to another.
The quantity with these properties is the *current*. The definition of *angular current* is:

$$\vec{j}\left(\vec{r}, E, \hat{\Omega}, t\right) \equiv \hat{\Omega}\psi\left(\vec{r}, E, \hat{\Omega}, t\right)$$

and its integral over all direction is the *net current*:

$$\vec{J}\left(\vec{r}, E, t\right) = \int_{4\pi} \mathrm{d}\hat{\Omega}\, \vec{j}\left(\vec{r}, E, \hat{\Omega}, t\right)$$

The net current does have the properties usually of "flux" used in other disciplines, so:

$$\vec{J}\left(\vec{r}, E, t\right) \cdot \hat{n}\, \mathrm{d}A$$

is the amount of neutrons with energy E going through a surface with normal $\hat{n}$ through a differential area $\mathrm{d}A$.

Fundamental concepts (cont.)

For a surface with normal $\hat{n}$, the *incoming and outgoing currents* are the angular currents integrated for directions for which $\hat{\Omega} \cdot \hat{n}$ greater or smaller than zero respectively:

$$J_+ \left(\vec{r}, E, t\right) = \int_{2\pi^+} \mathrm{d}\hat{\Omega}\ \hat{n} \cdot \vec{j}\left(\vec{r}, E, \hat{\Omega}, t\right)$$

$$J_- \left(\vec{r}, E, t\right) = -\int_{2\pi^-} \mathrm{d}\hat{\Omega}\ \hat{n} \cdot \vec{j}\left(\vec{r}, E, \hat{\Omega}, t\right)$$

and the net number of neutrons going through a surface can be computed as the difference of the partical currents:

$$\vec{J} \cdot \hat{n} = J_+ - J_-$$

Boltzmann transport equation

The neutron flux ψ is the solution to an integro-differential equation which describes the balance between the local variations of the neutron density and the net production of neutrons by fission, emission by external sources or scattering from other energies or directions:

$$\frac{\mathrm{D}n}{\mathrm{D}t} = \frac{1}{v}\frac{\partial \psi}{\partial t} + \hat{\Omega}\nabla\psi = \text{sources} - \text{sinks}$$

Boltzmann transport equation (cont.)

Sources:

External source:

$$S(\vec{r}, E, \hat{\Omega}, t)$$

Scattering:

$$\int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\hat{\Omega}' \, \Sigma_s \left(\vec{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega}\right) \psi \left(\vec{r}, E', \hat{\Omega}', t\right)$$

Fission:

$$\frac{\chi(E)}{4\pi} \int_0^\infty \mathrm{d}E' \, \nu(E') \Sigma_f \left(\vec{r}, E'\right) \phi \left(\vec{r}, E', t\right)$$

Sinks:

All reactions:

$$\Sigma_t \psi \left(\vec{r}, E, \hat{\Omega}, t\right)$$

Boltzmann transport equation (cont.)

$$\frac{1}{v}\frac{\partial \psi}{\partial t} + \hat{\Omega}\nabla\psi = -\Sigma_t \psi\left(\vec{r}, E, \hat{\Omega}, t\right)$$
$$+ \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\hat{\Omega}'\, \Sigma_s\left(\vec{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega}\right) \psi\left(\vec{r}, E', \hat{\Omega}', t\right)$$
$$+ \frac{\chi(E)}{4\pi} \int_0^\infty \mathrm{d}E'\, \nu(E')\Sigma_f\left(\vec{r}, E'\right) \phi\left(\vec{r}, E', t\right)$$
$$+ S(\vec{r}, E, \hat{\Omega}, t)$$

+ initial condition $\phi\left(\vec{r}, E', 0\right)$,

+ boundary conditions.

Boltzmann transport equation (cont.)

$$\Sigma(v' \to v; \boldsymbol{\Omega}' \to \boldsymbol{\Omega}) = v\Sigma(E' \to E; \boldsymbol{\Omega}' \to \boldsymbol{\Omega}). \qquad \text{(A3-8)}$$

Clearly a quantity of some fundamental interest is the scattering function $\Sigma(E' \to E; \boldsymbol{\Omega}' \to \boldsymbol{\Omega})$. The definition of the scattering function which arises is essentially a classical one, this is because the Boltzmann equation itself was derived from classical phase space balance conditions. It was demonstrated many years ago by Uehling and Uhlenbeck [1933] that to a very good approximation it is valid to use the classical Boltzmann equation with the quantum mechanical value of the scattering function. In the original quantum mechanical derivation of the Boltzmann equation the effect of statistics and degeneracy were included. Thus, it was assumed that the deflection of a particle m_1 by a scatterer m_2 with relative velocity $\boldsymbol{g}$, is much the same as if, on the classical theory, each molecule were surrounded by a wave field whose linear extension is of the order of the wavelength $\hbar/\mu g$ where μ is the reduced mass. Using a mean value for g we have that the increase in size is appreciable if $\hbar/(\sigma\mu kT)^{1/2}$ is not small compared with unity. Effects such as these are accounted for by using the correct quantum mechanical scattering laws.

M.M.R Williams, *The Slowing Down and Thermalization of Neutrons*, 1966 - Appendix III

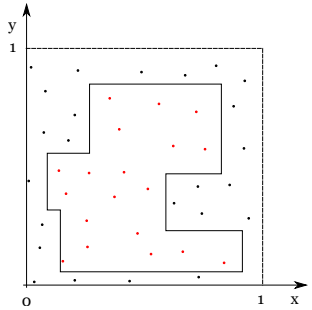
Bibliography and references

- Duderstadt, Martin. *Transport theory*. Wiley, 1979.
- Duderstadt, Hamilton. *Nuclear Reactor Analysis*. Wiley, 1976.
- Lewis, Miller. *Computational Methods of Neutron Transport*. Wiley, 1984.

The Monte Carlo method.

Introduction

The Monte Carlo method is a way of solving mathematical problems through the simulation of random processes.
Example:



$$A \simeq \frac{N'}{N + N'}$$

Monte Carlo method for radiation transport

The processes involved in neutron transport are intrinsically probabilistic: $\Sigma \, \mathrm{d}x$ defines the probability of interaction in the differential $\mathrm{d}x$, and Σ_x / Σ_t is the probability of a reaction being of type x. On the other hand, the interaction probability depends on the state of the particles at a given time (position, energy, direction) but not on their previous properties. These two characteristics allows us to model the system as a *Markov chain*.

Monte Carlo method for radiation transport (cont.)

The Monte Carlo method for radiation transport is composed mainly of the following steps:

1) Sample the source.

2) Calculate of the position of the next interaction.

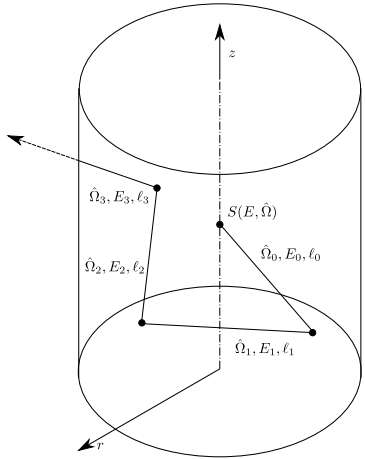
3) Analyze the interaction type (change of direction and/or energy in the case of scattering, generation of new particles in case of fission, disappearance in case of absorption, etc).

$\rightarrow$ steps 2) and 3) are repeated until the particle is absorbed or escapes from the system.

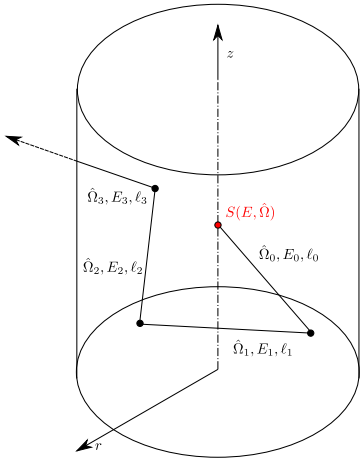
These steps constitute a Markov chain. It can be proved that these steps, combined with utilization of an appropriate statistical estimator for the neutron flux constitute a solution to the transport equation in its integral form [*].

[*]Spanier, Gelbard. "Monte Carlo Principles and Neutron Transport Problems", cap. 2.

Monte Carlo method for radiation transport (cont.)

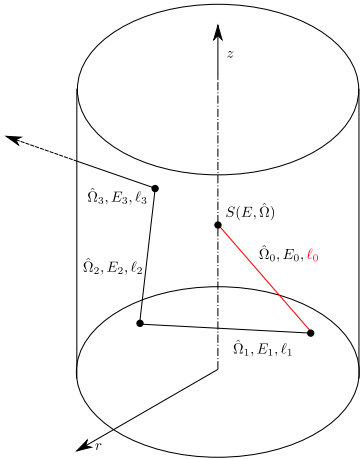


Monte Carlo method for radiation transport (cont.)



Sampling the source:
Emission processes are stochastic and are determined by distributions for energy (spectrum), direction and position. The source sampling process implies obtaining values for the variables that define the particle that is being emitted (energy, direction and position) from the distributions that describe the source.

Monte Carlo method for radiation transport (cont.)

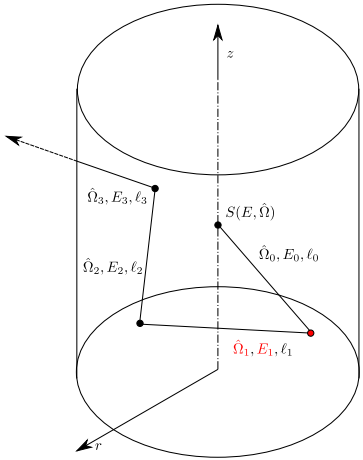


Calculation of the position of the next interaction:
In absence of forces that could affect the trajectory, the neutron will travel in a straight line in direction $\hat{\Omega}_0$. The cumulative probability is given by:

$$P(\ell) = 1 - \exp(-\Sigma_t \ell)$$

The distance to next collision is obtained by sampling this distribution.

Monte Carlo method for radiation transport (cont.)

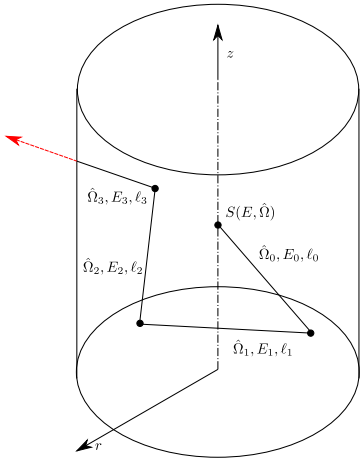


Analysis of the collision:
The probability for each type of interaction is given by:

$$p_i = \frac{\Sigma_i}{\Sigma_t}$$

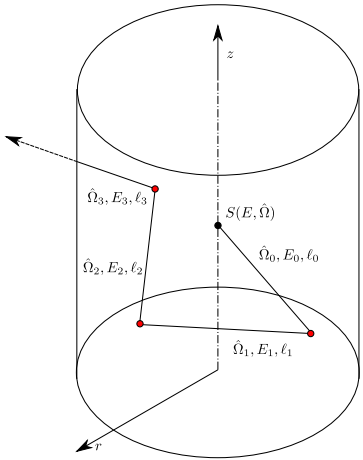
If the collision is an scattering event, we need to sample the outgoing energy and direction distribution to obtain values for $\hat{\Omega}_1, E_1$.

Monte Carlo method for radiation transport (cont.)



End of the simulation for this particle:
The simulation is ended when the particle is absorbed or escapes the system. The simulation continues by sampling a new particle from the source, until we reach the total number of particles to simulate (*N*).

Monte Carlo method for radiation transport (cont.)



Flux estimators: collisions

The number of collisions c_i for each simulated particle inside of a volume of interest allows us to calculate the collision rate:

$$\tilde{c} = \frac{1}{N} \sum_i c_i$$

The average value of the collision rate is related to the neutron flux as:

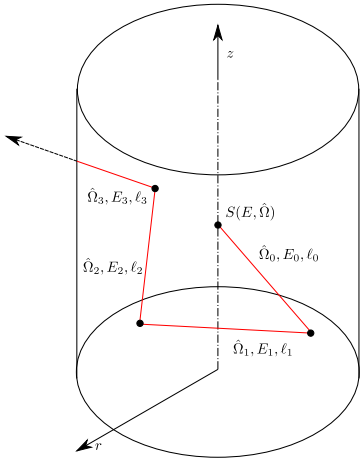
$$\bar{c} = \bar{\Sigma}_t \bar{\phi} V$$

Therefore, the neutron flux can be estimated as:

$$\tilde{\phi} = \frac{1}{V} \frac{1}{\bar{\Sigma}_t} \frac{1}{N} \sum_i c_i$$

but what happens in void?

Monte Carlo method for radiation transport (cont.)



Flux estimators: track length

We defined before:

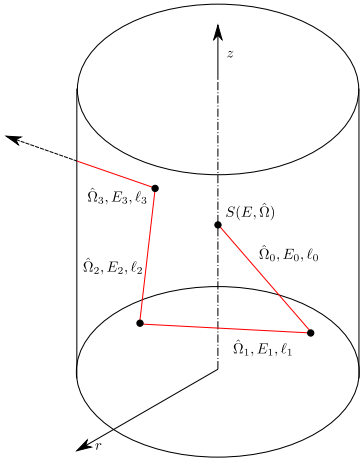
$$\psi\left(\vec{r}, E, \hat{\Omega}, t\right) \equiv vn\left(\vec{r}, E, \hat{\Omega}, t\right)$$

If we integrate the angular flux over a volume $\mathrm{d}V$ and knowing that $v\,\mathrm{d}t$ is the track length over a time $\mathrm{d}t$:

$$\psi\left(\vec{r}, E, \hat{\Omega}, t\right)\mathrm{d}V\,\mathrm{d}E\,\mathrm{d}\hat{\Omega}\,\mathrm{d}t =$$
$$n\left(\vec{r}, E, \hat{\Omega}, t\right)\mathrm{d}V\,\mathrm{d}E\,\mathrm{d}\hat{\Omega}\,v\,\mathrm{d}t$$

we see that the integral of the scalar flux in the volume is the sum of the track lengths of the particles contained in the phase space $\mathrm{d}\hat{\Omega}\,\mathrm{d}E$.

Monte Carlo method for radiation transport (cont.)



Flux estimators: track length

Integrating over direction and energy we obtain:

$$\int_V \phi = \text{total path length}$$

The total track length can be estimated by Monte Carlo as:

$$\tilde{\ell} = \frac{1}{N} \sum_i \sum_j \ell_j$$

and the neutron flux can be then estimated as:

$$\tilde{\phi} = \frac{1}{V} \frac{1}{N} \sum_i \sum_j \ell_j$$

This estimator is particularly useful for optically thin systems.

Pseudo-random number generators

The Monte Carlo method requires the generation of random numbers. Random numbers are real numbers between 0 and 1 with an uniform distribution.
Although it is possible to generate "true" random numbers in a computer (and they are necessary for some applications, like cryptography), in Monte Carlo simulations it is better to use pseudo-random numbers, this is, sequences of numbers with the same properties as random numbers but produced in a reproducible way.
One algorithm commonly used for the generation of pseudo-random numbers is the *linear congruential method*:

$$x_i = (a x_{i-1} + b) \,\%\, m$$
$$\xi_i = \frac{x_i}{m}$$
$$a = 2^7 + 1, \; b = 1, \; m = 2^{35}$$

where $\%$ is the modulo operator. For an initial value or *seed* x_0, the sequence is always the same.

`https://docs.openmc.org/en/stable/methods/random_numbers.html`

Pseudo-random number generators

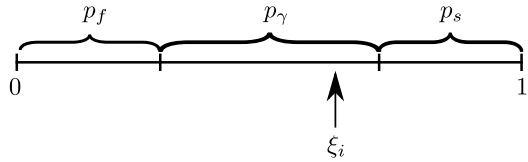
- When the generator produces again x_0, the sequence is repeated and this defines its *period*.
- To ensure the reproducibility of the simulation and allow to simulate particles in parallel, Monte Carlo simulations usually assign blocks of random numbers to each particles. These blocks are obtaining by advancing the random number generator a given number of steps known as *stride*.
- If we need n random numbers to simulate one particle (stride), and we want to simulate N particles, the random number generator should have a period bigger than $n \cdot N$. To solve this limitation we can use the random numbers from one algorithm as seeds for other algorithms.
- If the stride or period of the random number generator is reached we can compromise the quality of the simulation (and the Monte Carlo code will give a warning). But since the random numbers are used for different applications, this is not necessarily a problem.

Discrete probability distributions

- The Monte Carlo method requires to make decisions based on probability distributions.
- As an example, if for a given material Σ_f is the fission cross section, Σ_γ is the radiative capture cross section, and Σ_s is the scattering cross section (and if those are the only three possible reactions), they probabilities are:

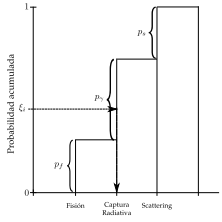
$$p_f = \frac{\Sigma_f}{\Sigma_t},\ p_\gamma = \frac{\Sigma_\gamma}{\Sigma_t},\ p_s = \frac{\Sigma_s}{\Sigma_t}$$

- Making a decision, or sampling this probability distribution is equivalent of finding the position of a random number ξ_i in the vector of cumulative probabilities:

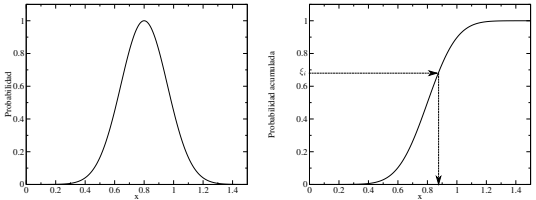


Continuous probability distributions

- The sampling process for a discrete distribution can be graphically described as:



- The equivalent for a continuous distribution is:



Continuous probability distributions

This can be done:

- Computing the cumulative probability distribution and inverting the function, or
- Applying a rejection method to the probability distribution. This is, generating random values of x with a uniform distribution and accepting the value if $\xi_i < p(x)$.

The first option is usually preferred, but not always possible (and can lead to numerical problems with the inversion). The second option is simpler and more robust, but usually comes with a higher computational cost.

Bibliography and references

- Duderstadt, Martin. *Transport theory*. Wiley, 1979.
- Lewis, Miller. *Computational Methods of Neutron Transport*. Wiley, 1984.
- Spanier, Gelbard. *Monte Carlo Principles and Neutron Transport Problems*. Dover, 2008.
- Brown. *Fundamentals of Monte Carlo Particle Transport*. Tech. Rep. LA-UR- 05-4983, LANL.
- OpenMC Development Team, *Theory and Methodology*, `https://docs.openmc.org/en/stable/methods/index.html`.

Introduction to OpenMC.

Introduction

- OpenMC is a radiation (neutron and photon) transport Monte Carlo code that performs fixed source and criticality simulations, on arbitrary geometries, using continuous energy or multigroup cross sections.
- OpenMC is implemented in C++, using as input a set of XML formatted files. A Python API can be used for preparing the input files and postprocess the output

  Documentation:
  `https://docs.openmc.org/`

  Source code repository:
  `https://github.com/openmc-dev/openmc`

  Discussion board:
  `https://openmc.discourse.group/`

Introduction

From a conceptual point of view, a Monte Carlo radiation transport simulation requires the definition of the following parameters:

- Geometry: each point of space where particles can go has to be defined univocally, and has to be associated with vacuum or a material.
- Materials: the composition and properties of the materials used to fill the geometry have to be also well defined.
- Boundary conditions: the calculation code needs to know what to do with particles that reach the boundary of the simulation.
- Execution parameters: number of particles to simulate, external sources, etc.
- Tallies: we need to define which events need to be counted during the simulation.

In OpenMC these parameters are expressed as XML files generated with the Python API.

Introduction

Calling OpenMC without any parameters in an empty directory produces the following error:

```
> openmc
 ERROR: Settings XML file 'settings.xml' does not exist! In order to run OpenMC,
        you first need a set of input files; at a minimum, this includes
        settings.xml, geometry.xml, and materials.xml. Please consult the user's
        guide at https://docs.openmc.org for further information.
```

The name of the input files for OpenMC are:

`settings.xml`	Execution parameters
`geometry.xml`	Geometry definition
`materials.xml`	Material definition
`plots.xml`	Plots definition (optional, needed to call `openmc -p`)
`tallies.xml`	Tallies definition (optional, needed to define tallies)

Also, it is necesary to indicate the location of the file `cross_sections.xml`, either inside of the `materials.xml` file or through the environment variable `OPENMC_CROSS_SECTIONS`. The output is two binary files in HDF5 format (`summary.h5`, `statepoint.hdf5`) containing the results from the simulation.

Material definition

- To perform the simulation we need to define the composition of all materials to be used. This is done by specifying the amount of each nuclide that forms the material.
- OpenMC then loads a library of microscopic cross sections for each nuclide and reaction in the system. By combining the microscopic cross section with the user provided compositions, OpenMC calculates the macroscopic cross sections that are used in the simulation:

$$\Sigma_i\left(\vec{r}, E\right) = \sum_j N_j(\vec{r})\sigma_i(E)$$

  where the subindex j covers all nuclides in the material.
- These libraries have a continuous dependency in energy and are binary files in the HDF5 format (multigroup cross section can also be specified). The HDF5 can be generated from ACE formated nuclear data files (the same format used in MCNP, PHITS and Serpent).

Material definition: nuclear data

- The nuclear data required for the simulation needs to be complete (i.e. covering all energy ranges in the simulation) and unique (for a given energy, each nuclide should have only one value for each reaction).
- The process of producing a data library with recommended values is called *evaluation*, and the data is stored in *evaluated nuclear data libraries*:
  - ENDF/B, developed by the Cross Section Evaluation Work Group and maintained by Brookhaven National Laboratory.
  - JEFF, developed and maintained by the Nuclear Energy Agency from OECD.
  - JENDL, developed and maintained by the Japan Atomic Energy Agency.
  - ...
- These libraries are stored in a common format called ENDF-6, which contains the parameters to reconstruct the neutron interaction cross section.

Material definition: thermal nuclear data

- Nuclear data as defined above is given per nuclide. But, as we know, neutrons of low energy / long wavelength interact at the molecular and compound level.
- To account for this, evaluated nuclear data libraries contain a sublibrary for thermal neutrons, where the scattering cross section is stored as a function $S(\alpha, \beta)$, where α is the non-dimensional exchange in momentum and β is the non-dimensional exchange of energy.

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}E\mathrm{d}u} = \frac{\sigma_b}{2kT}\sqrt{\frac{E'}{E}}S(\alpha,\beta)$$

$$\alpha = \frac{E' + E - 2\mu\sqrt{E'E}}{AkT},\ \beta = \frac{E' - E}{kT}$$

$S(\alpha, \beta)$ is usually called the *thermal scattering law*. Thus, defining in a Monte Carlo code such as

MCNP consists in listing the nuclide compositions, assign the libraries that would provide their nuclear data and associate the correct thermal scattering laws.

Examples of material definition.

`TSL_School/openmc/Notebooks/Materials.ipynb`

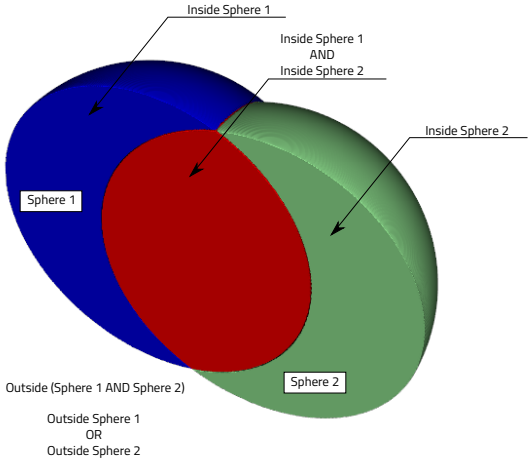
Geometry definition

- As in many radiation transport codes, in OpenMC geometry is defined using *constructive solid geometry (CSG)*. CSG is a technique where complex object is created by applying boolean operators to simple objects.
- The objective is subdivide the domain of the problem in a way that each point is associated univocally to a material or void. There cannot be superimposed regions or regions without assignment.
- The way to do it is using *surfaces*, which are combined using boolean operators to define *regions*. A region, in combination with an homogenous material and other properties form a *cell*.
- The dimensions used for geometry in OpenMC are cm.

Geometry definition

- A surface is defined as the set of points (x, y, z) such that $f(x, y, z) = 0$. As an example, a sphere centered in the origin with radius 1 is the set of points that satisfy $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$.
- Based on that definition, we can also define two regions, such that $f(x, y, z) > 0$ o $f(x, y, z) < 0$

Geometry definition



Examples of geometry definition.

`TSL_School/openmc/Notebooks/Geometry.ipynb`

Run settings and sources

- With the definition of the geometry and materials we have described the space, and the properties of the materials that fill the space.
- To complete the description of the simulation we need to specify the type of simulation we want to do, and describe the source of particles.

Run settings and sources

- OpenMC runs simulations in two modes, which solve different versions of the Boltzmann transport equation. For the purposes of this course we will restrict ourselves to the ”fixed source” mode, which solves the flux distribution for a source that is independent of the solution. The other run mode, ”eigenvalue”, obtains the fundamental eigenfunction of a multiplicative system and its associated eigenvalue. It is useful to analize nuclear reactors.
- The main parameter for the simulation will be the number of particles to run. This can be separated into batches, which is not necessary for fixed source simulations but helps to see its evolution.

Run settings and sources

- Within the execution settings we also need to define the external source. One or more sources can be defined in the system, with different intensities.
- Sources are defined by describing the probability distributions for their variables: energy, direction and position.
- The available distributions are imported using the `openmc.stats` submodule.

Examples of settings definition.

`Settings.ipynb`

Tallies

We need to define two things:

- Scores: i.e., *what* we want to count. This is associated with the quantity we want to estimate: flux, current, reaction rate.
- Filters: i.e., *where* (in the phase space) we want to count that. We can filter events spacially to happen on a given cell, or on a mesh in space; filter in energy to obtain spectra; filter in angle to get angular distributions.

This is explained in the OpenMC documentation as:

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

Examples of tally definition.

`TSL_School/openmc/Notebooks/Tallies.ipynb`

Examples of postprocessing.

`TSL_School/openmc/Notebooks/Postprocessing.ipynb`

Example simulations and exercises.

(see `TSL_School/openmc/Examples` subdirectory)