An Open-source Monte Carlo code for neutron criticality simulations

Benoit Forget

September 22, 2017

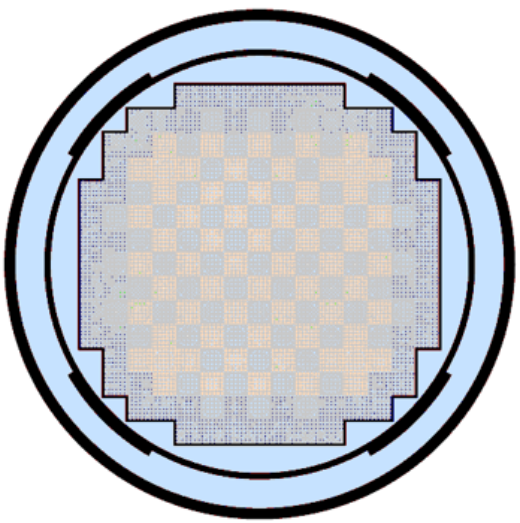
OpenMC

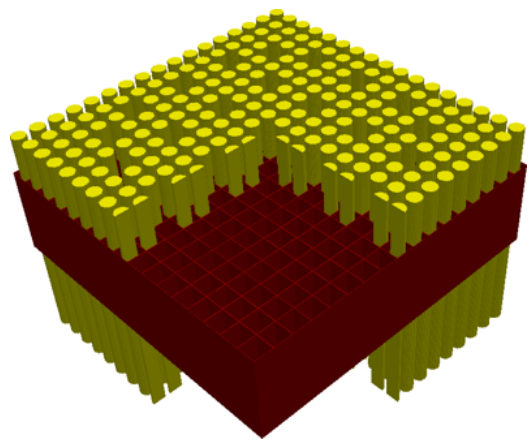
OpenMC is an open source Monte Carlo code that has been developed at MIT by the CRPG group since 2010.

- Developed as part of a PhD thesis in order to resolve scalability issues on leadership class computing platforms.
- Features were added to truly test parallel algorithm to a point where results became realistic.
- Serves as an essential research element of the current CPRG research

Development Team

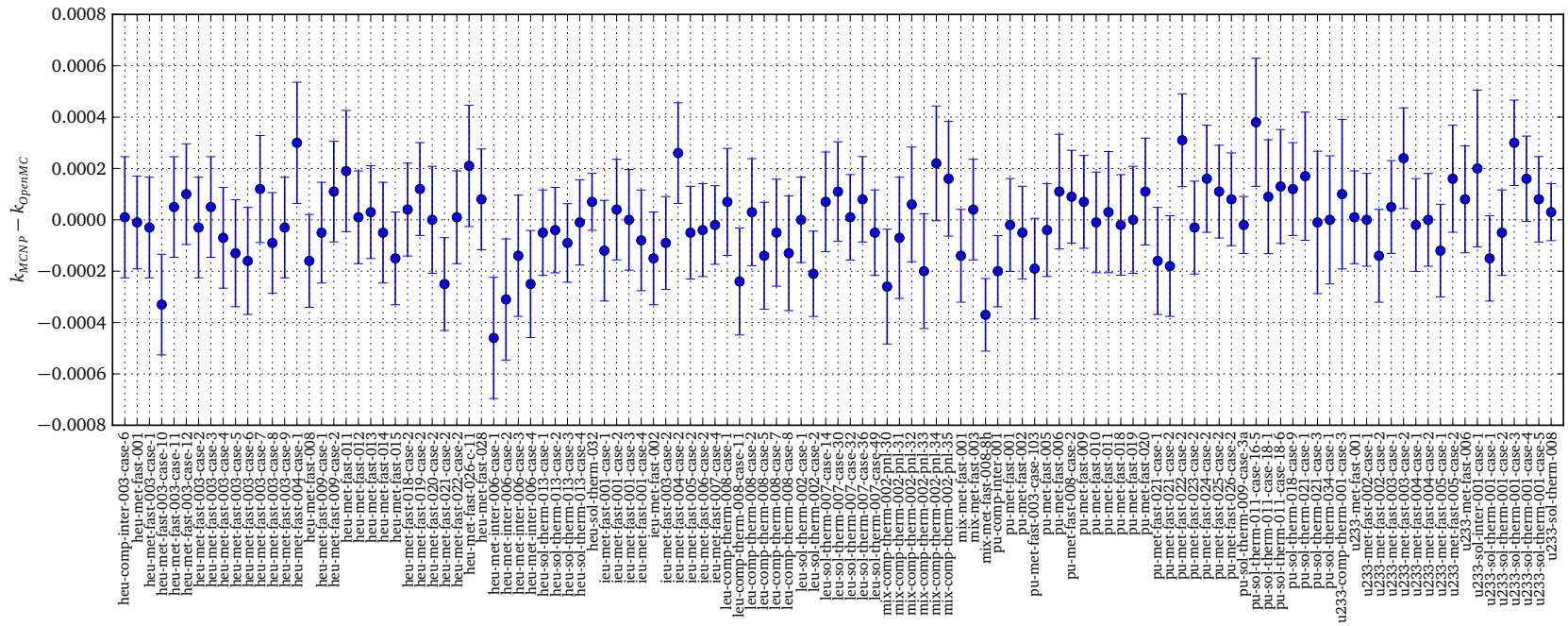
- Paul Romano (MIT grad, Lead developer, ANL)
- Nicholas Horelik (MIT grad, startup)
- Bryan Herman (MIT grad, KAPL)
- Adam Nelson (UM grad, Naval Reactors)
- Jon Walsh (MIT grad, LLNL)
- Will Boyd (MIT grad, BCG)
- Lulu Li (MIT grad, Google)
- Sterling Harper (MIT)
- Matt Ellis (MIT grad, startup)
- Sam Shaner (MIT grad, startup)
- Colin Josey (MIT grad, LANL)
- Travis Labossiere-Hickman (MIT)
- Jingang Liang (Tsinghua grad, MIT postdoc)
- Xingjie Peng (Tsinghua grad, MIT postdoc)

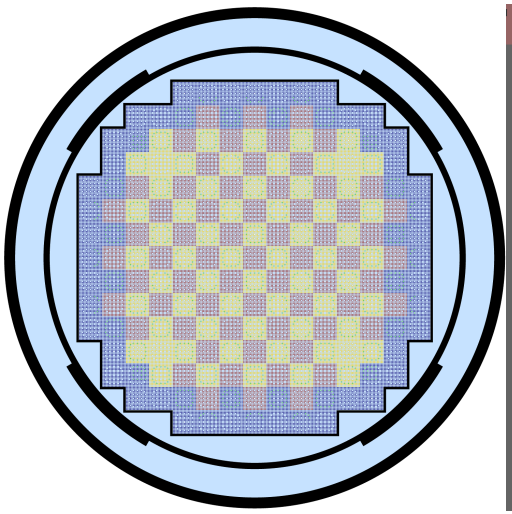


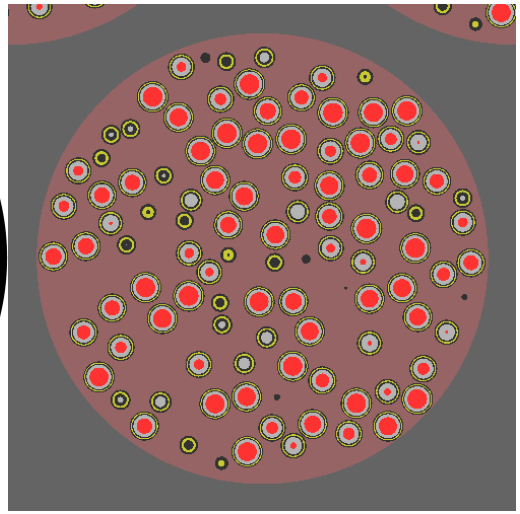


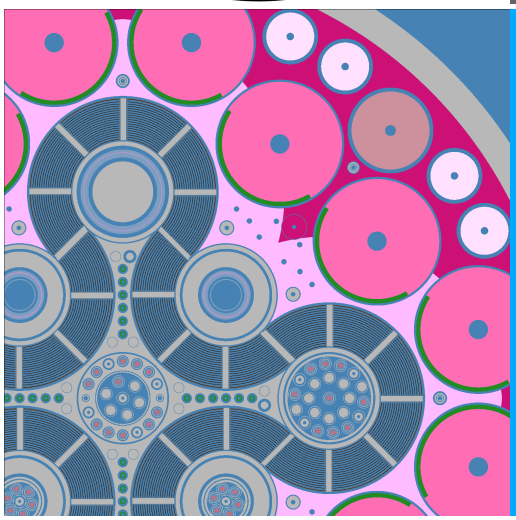
OpenMC is validated against MCNP Criticality Benchmark Suite

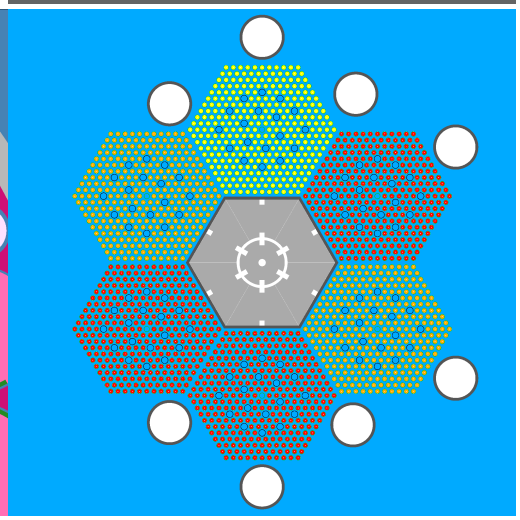
117 configurations with different spectra, materials, enrichment











OpenMC can model a wide variety of geometries, for example:

Full-core PWR

TRISO particles

ATR

What makes OpenMC special is its Python-powered input generation and post-processing


```
>>> import openmc
```

OpenMC is 46,000 lines of F90 and 46,000 lines of Python

The OpenMC workflow:

1. Write Python code describing the problem.
2. Use `.export_to_xml()` to create XML files.
3. Run OpenMC (using Python or shell). This creates `tallies.out` and `statepoint.h5` output files.
4. Read `tallies.out` with a text editor or read `statepoint.h5` with Python.

OpenMC Features

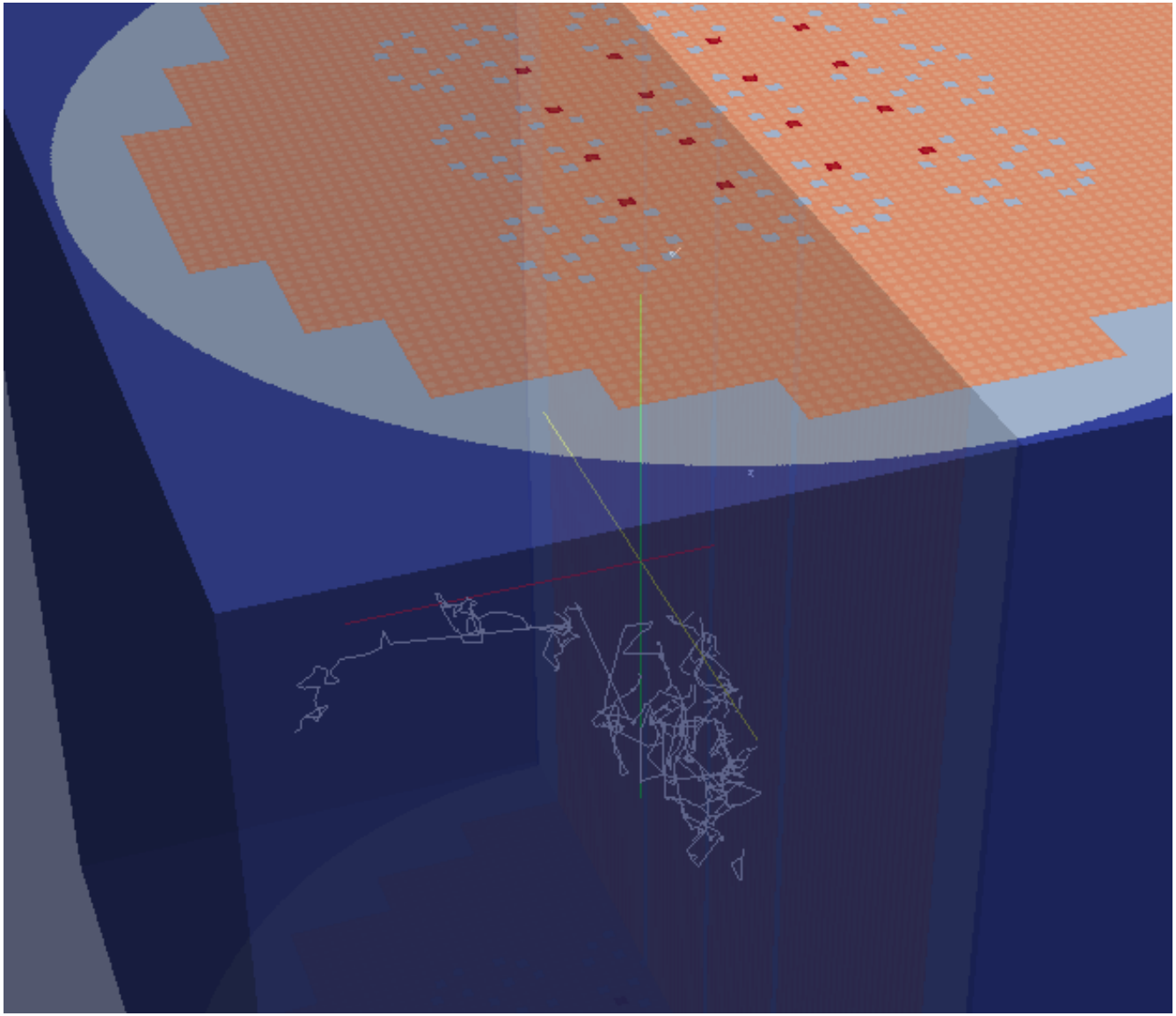
Public Release

- **Modes:** Fixed source and k-eigenvalue
- **Geometry:** CSG second order, universes, translations, rotations, rectangular and hexagonal lattices
- **Cross Sections:** HDF5-ACE, MG, WMP-beta
- **Physics:** neutron transport, S(α, β) tables, URR probability tables, free gas scattering, resonance upscattering, ...
- **Acceleration:** CMFD
- **Parallelism:** Distributed/shared memory via MPI/OpenMP, replication
- **Input:** XML or Python API
- **Output:** HDF5
- **Diagnostics:** Shannon entropy, iso-in-lab scattering, particle tracking files

Private Branches

- **URR:** Equiprobable surfaces for temperature interpolation, on-the-fly URR
- **Parallelism:** Domain decomposition, tally servers
- **Physics:** Resonance upscattering with WMP, depletion, photon transport, continuous material tracking, multigrid time dependence
- **Tallies:** Functional expansion tallies
- **Acceleration:** Low order transport
- **Input:** VERAin converter (CASL)
- **UQ:** IFP, Clutch, Differential tallies
- **Diagnostics:** Center-of-mass variance

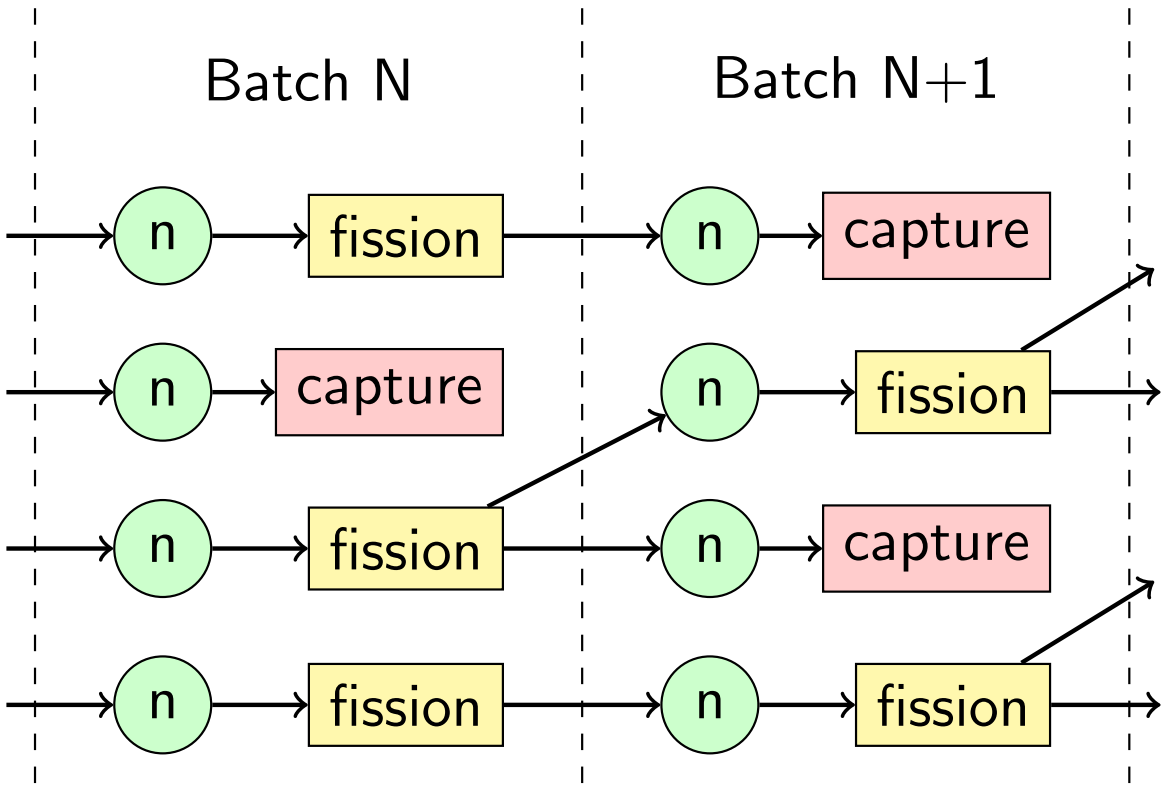
OpenMC can also store the individual tracks of a neutron history



OpenMC

- Fission sampling
- Nuclear data representation
- Uncertainty quantification
- Multiphysics coupling

Eigenvalue Mode



MC eigenvalue simulations track successive generations

Neutrons are born from fission sites in the last batch mimicking the concept of generations

- Sample number of fission sites at each collision
- Scaled by keff to avoid uncontrolled growth or destruction

```
! Determine expected number of neutrons produced
nu_t = p % wgt / keff *  micro_xs(i_nuclide) % nu_fission / &
       micro_xs(i_nuclide) % total

! Sample number of neutrons produced
if (prn() > nu_t - int(nu_t)) then
  nu = int(nu_t)
else
  nu = int(nu_t) + 1
end if
```

- Store fission neutrons in bank with sampled direction and energy

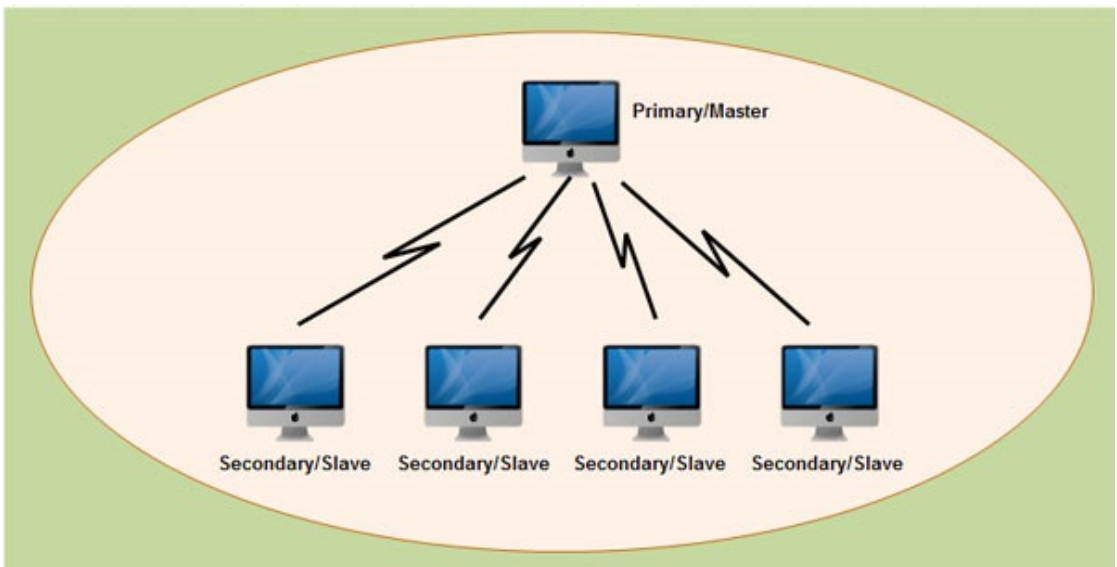
```
do i = size_bank + 1, size_bank + nu
  ! Bank source neutrons by copying particle data
  bank_array(i) % xyz = p % coord(1) % xyz

  ! Sample delayed group and angle/energy for fission reaction
  call sample_fission_neutron(nuc, nuc % reactions(i_reaction), &
       p % E, bank_array(i))

  ! Set delayed group on particle too
  p % delayed_group = bank_array(i) % delayed_group

  ! Increment the number of neutrons born delayed
  if (p % delayed_group > 0) then
    nu_d(p % delayed_group) = nu_d(p % delayed_group) + 1
  end if
end do
```

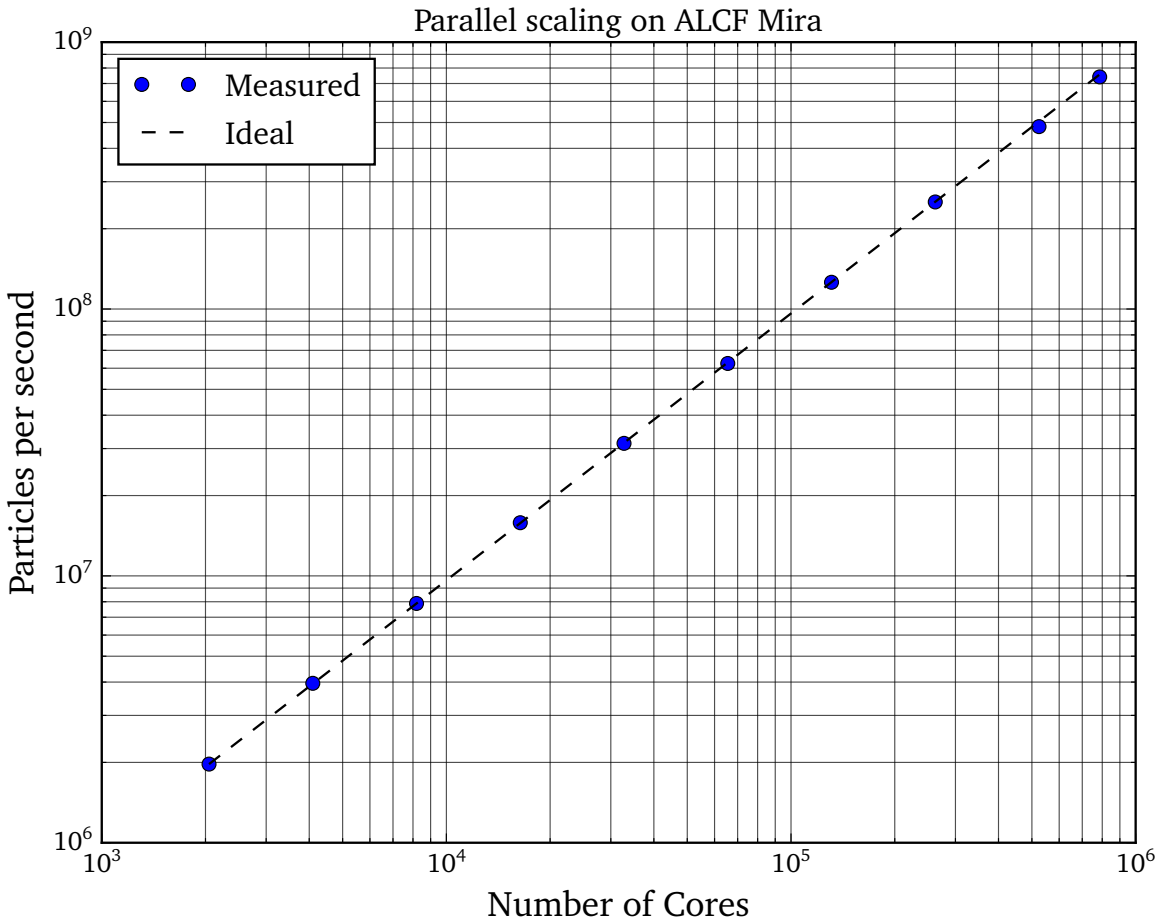
- Sample direction and energy

```
! Sample cosine of angle
mu = TWO * prn() - ONE

! Sample azimuthal angle uniformly in [0,2*pi)
phi = TWO*PI*prn()
site % uvw(1) = mu
site % uvw(2) = sqrt(ONE - mu*mu) * cos(phi)
site % uvw(3) = sqrt(ONE - mu*mu) * sin(phi)

! Determine total nu, delayed nu, and delayed neutron fraction
nu_t = nuc % nu(E_in, EMISSION_TOTAL)
nu_d = nuc % nu(E_in, EMISSION_DELAYED)
beta = nu_d / nu_t

if (prn() < beta) then
   sample delayed energy
else
   sample prompt energy
end if
```

Parallelization

- Development of OpenMC was initially started to address scalability issues on leadership class computers
- Traditional master/slave communication model was replaced by a slave-to-slave communication model that maintains reproducibility

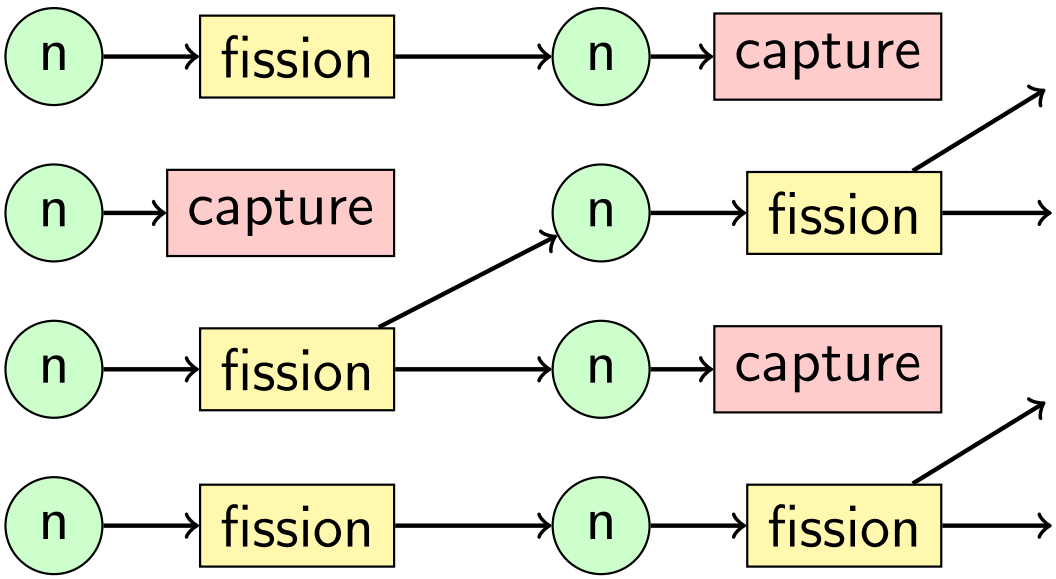




OpenMC scales linearly up to ∞ processors

(786,000 cores, 3,150,000 threads on Mira supercomputer)

Fixed Source Mode

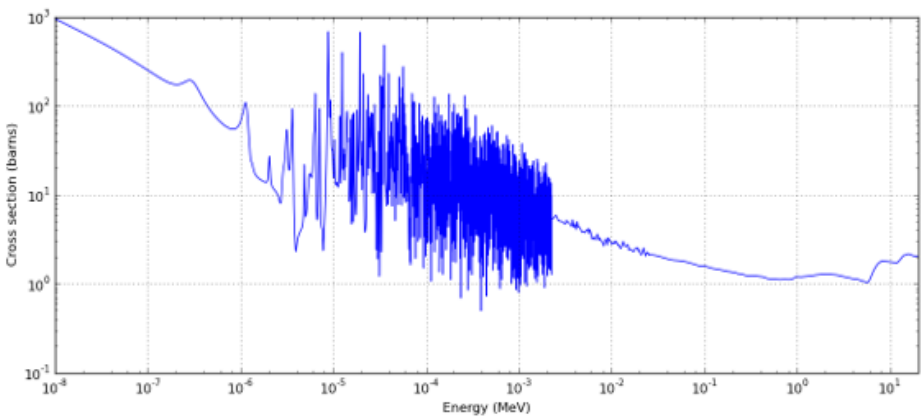


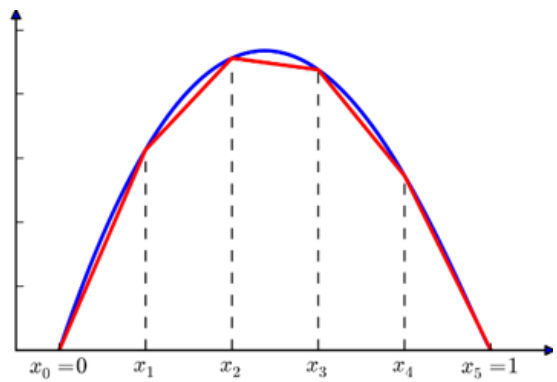
MC fixed source simulations track the full neutron history from source to death

Neutrons born from collision are banked and bank is emptied before starting next source particle

Nuclear Data Models

- Most common data model in Monte Carlo codes is the ACE format (A Compact ENDF format) stored in ASCII or binary files
- ENDF stores data in multiple formats from resonance models, log-log interpolation, lin-lin interpolation, ...
- ACE represents cross sections as point-wise data that can be linearly interpolated in energy and sometimes temperature





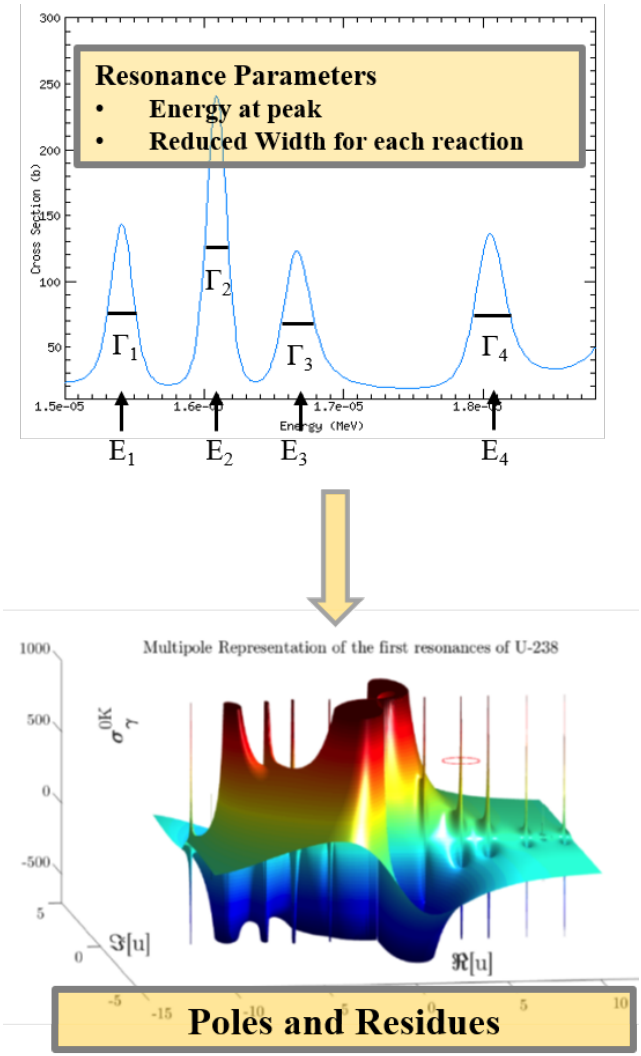
Doppler Broadening

$$v_n\overline{\sigma_x(v_n)} = \int d^3v_T P(v_T)|\vec{v_n} - \vec{v_T}|\sigma_x(|\vec{v_n} - \vec{v_T}|)$$

- The ACE format is used in NJOY since it can easily be Doppler broadened numerically
- The quest for fully coupled Monte Carlo simulations where each zone can have wildly different temperatures has led to the search of on-the-fly broadening techniques that are memory and computationally efficient.
- MCNP uses a fitting process where ACE files at multiple temperatures are fitted at each energy point using a high order polynomial
- Serpent uses a rejection sampling process where target velocities at the collision site are sampled randomly and compared to a "majorant" cross section

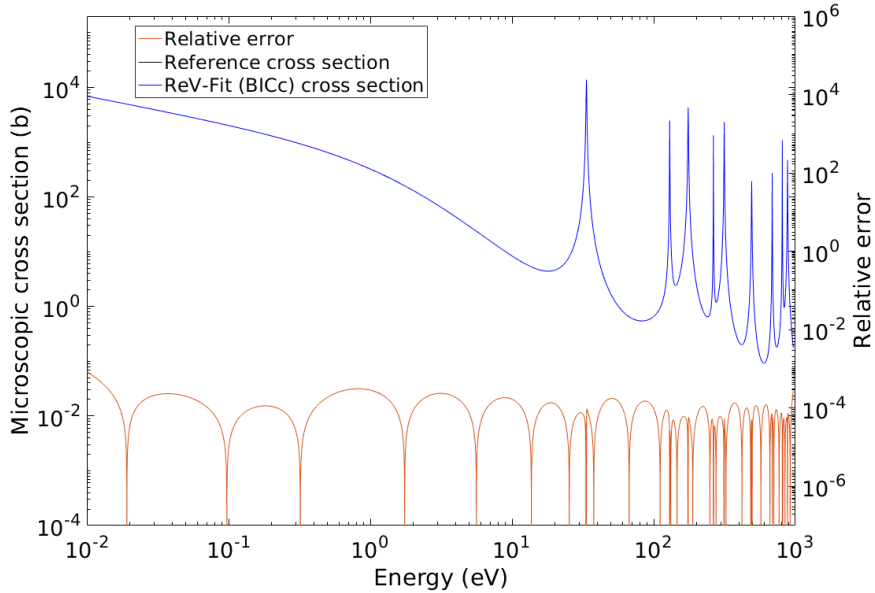
Multipole Formalism

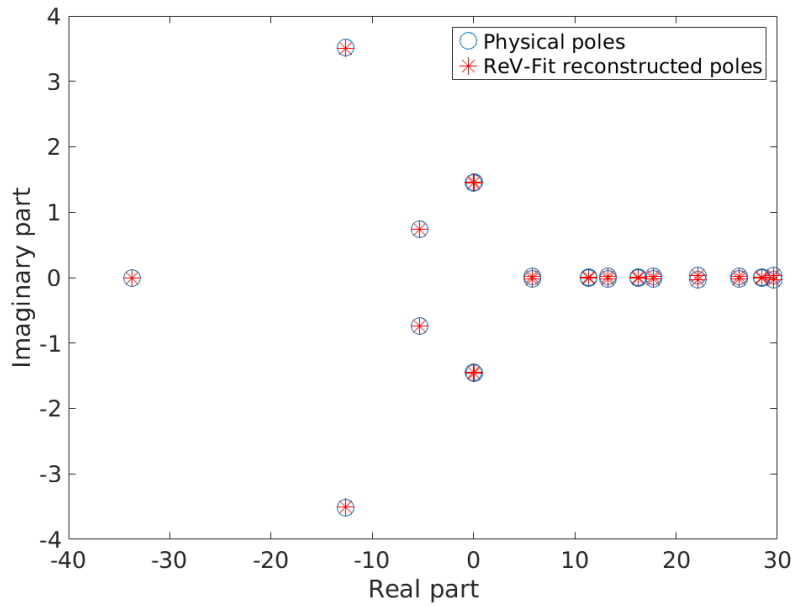
- One of the approaches proposed by CRPG was the partial fraction decomposition of the cross section which transforms resonance parameters to poles and residues.
- Hwang (1987) demonstrated that this conversion was possible and unique, and more importantly that the Doppler broadening operation was analytical.
- The main caveats are that the nuclides must be represented by resonance parameters in ENDF and that this works in the resolved resonance range only.



Vector Fitting

- To overcome these limitations, the idea of vector fitting was explored.
- Fits any "signal" into poles and residues but overfitting considerations must be taken.
- When done right, vector fitting is able to reproduce the true converted poles and residues starting from point-wise data!
- It also provides a pathway for dealing with threshold reactions and fast energy range.





Doppler Broadening

$$\sigma = \frac{1}{u^2} \sum_j \Re\left[\frac{r_j}{p_j - u}\right]$$

$$\sigma(u, T) = \frac{1}{u^2 2\sqrt{\xi}} \sum_j \Re\left[i r_j \sqrt{\pi} W(z_j)\right]$$

where

$$u = \sqrt{E}$$

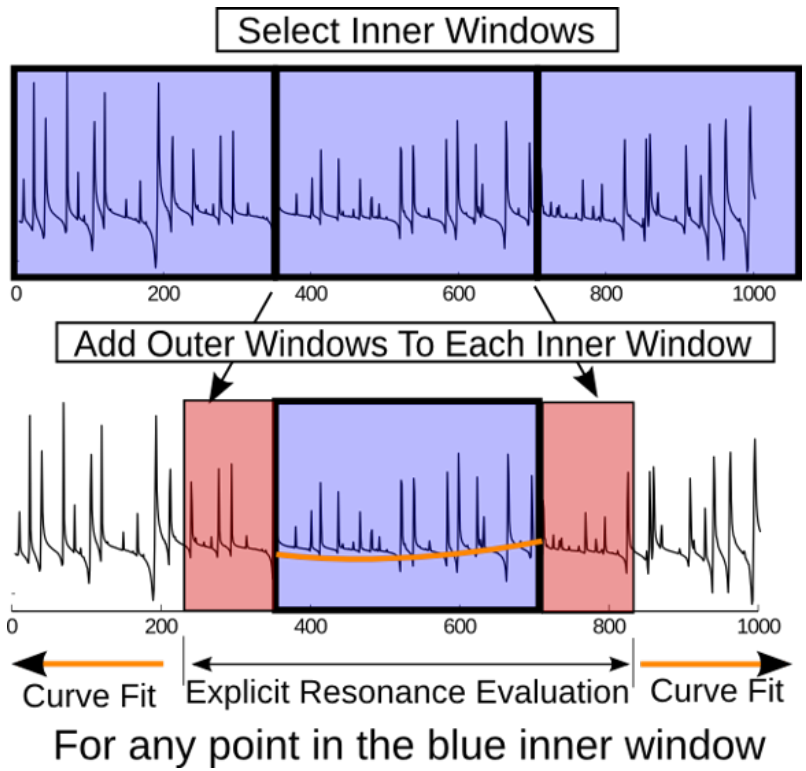
$$2\sqrt{\xi} = \sqrt{\frac{kT}{A}}$$

$$z_j = \frac{u - p_j}{2\sqrt{\xi}}$$

$$W(z) = e^{-z^2}\left(1 - \operatorname{erf}(-iz)\right)$$

Windowed Multipole (WMP) Method

- The number of poles from conversion or vector fitting is still too large for efficient use in analysis. Analytical broadening requires the evaluation of one Faddeeva function per pole.
- Windowing process was introduced where analytical integration is performed in the outer window (in red) and a low order curve fit (in blue inner window) is used to represent all far away resonances.
- Windowing process reduces the number of Faddeeva function evaluations from 1000's to 10's.

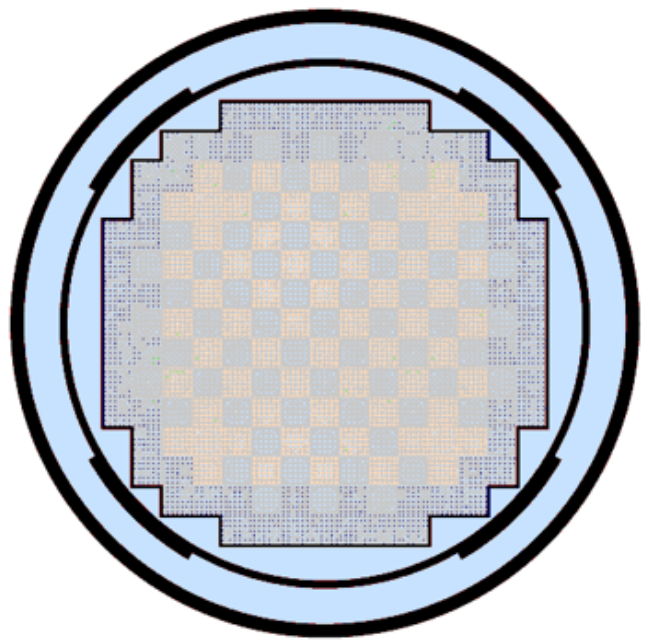


Performance on full core PWR model

- WMP requires 30 times less memory to represent the 70 nuclides in the resolved resonance range than the 2 ACE libraries used for interpolation.
- WMP accesses its data sequentially reducing large cache misses overhead of the binary searches.

Run	Clock Cycles
WMP	31587
Interpolation	38415

Table: Clock cycles per cross section lookup

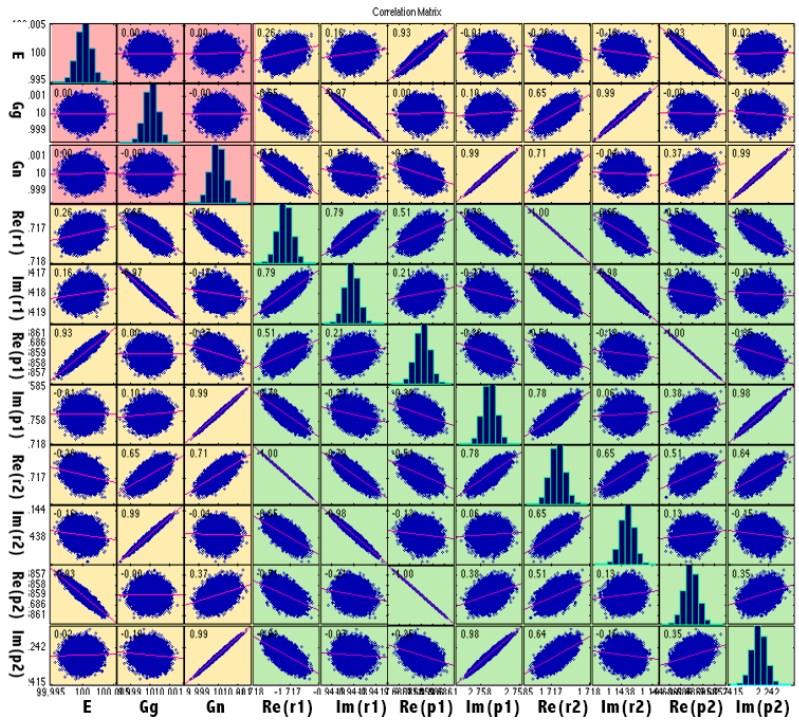


Run	L1 Misses	LL Misses
WMP	554	0.072
Interpolation	736	14.4

Table: Cache misses per cross section lookup

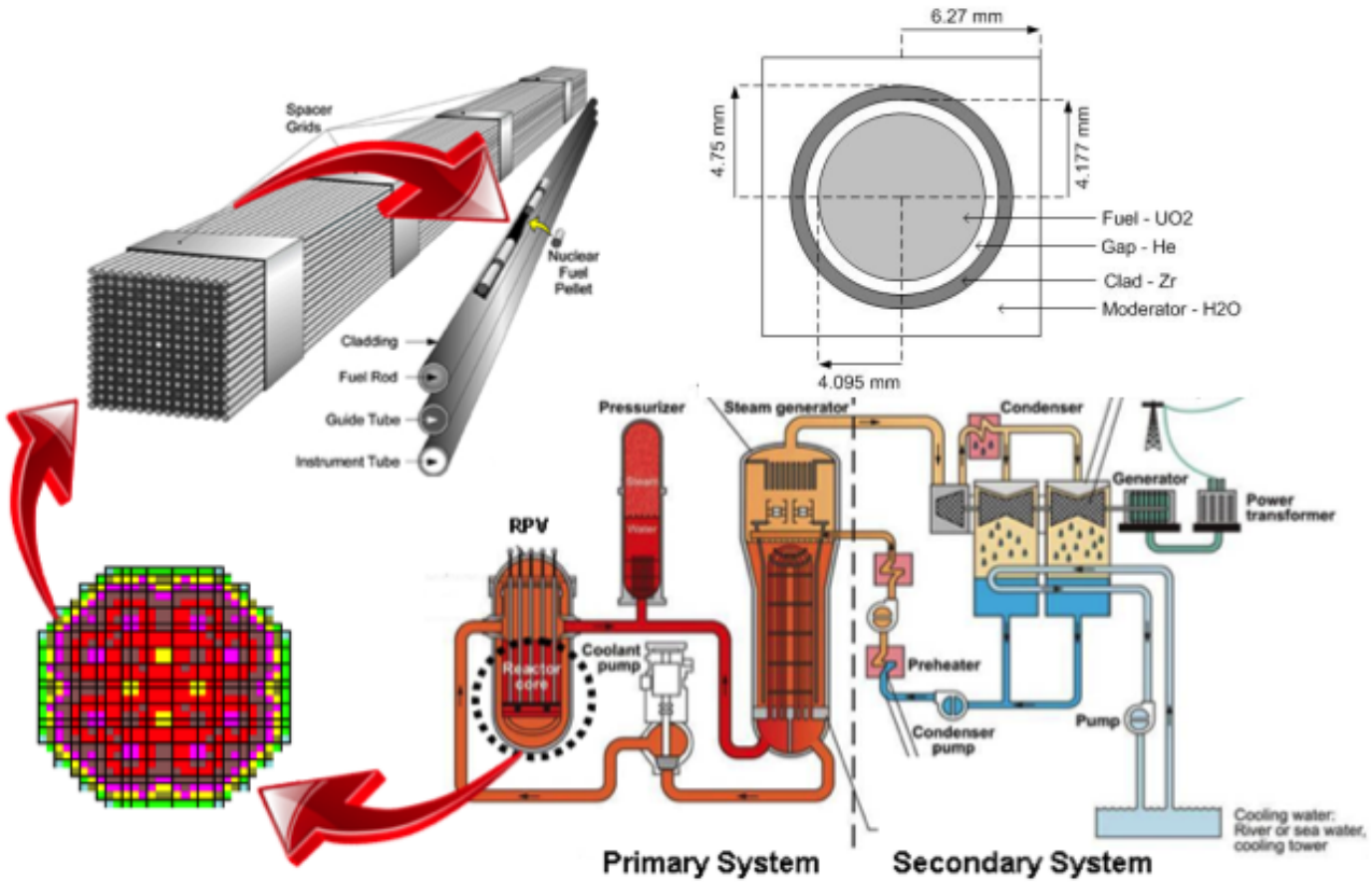
Sensitivity and UQ

- The pole representation of data also leads to an interesting avenue for uncertainty quantification where resonance parameters covariance data can be used directly instead of processing to an energy dependent form.
- CRPG demonstrated the equivalence between uncertainty propagation of resonance parameters and poles at 0K. Pole representation allows uncertainty to be propagated at any temperature.
- Future work is looking at embedding the resonance parameter uncertainty sampling during the random walk process.



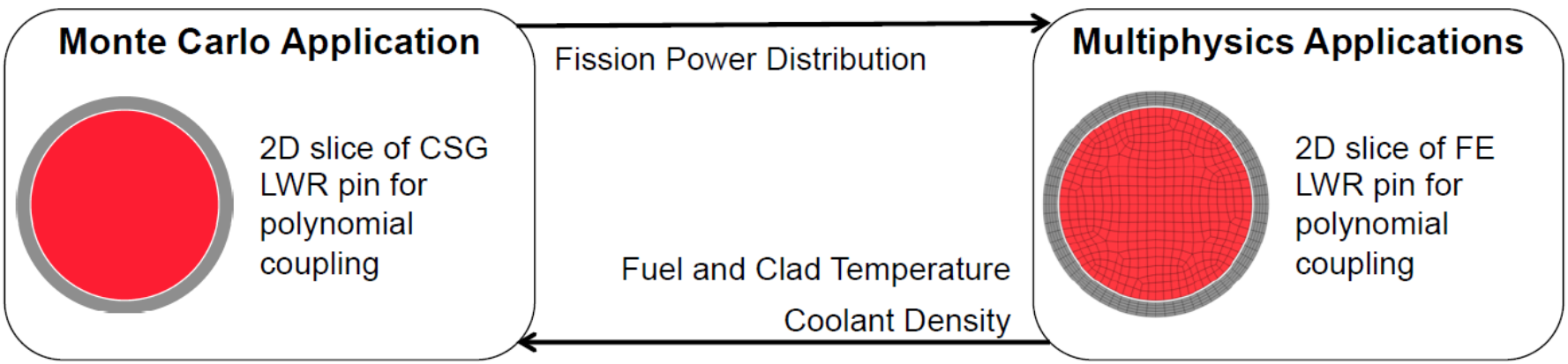
Coupling

- Many fields require coupling between different physics often represented by different codes and different meshes
- In nuclear reactors, fuel pins may require a FEM for heat transfer while the Monte Carlo method only requires a material mesh.



Coupling in OpenMC

- OpenMC proposes using Functional Expansion tallies (Griesheimer et al, 2006) to facilitate the data transfer from MC to FEM
- However, to be practical, a method to track neutrons in a continuously varying field (e.g. temperature, nuclide concentrations ...) is needed (Brown and Martin, 2003).



Tracking in Monte Carlo

Traditional Monte Carlo transport for constant macroscopic cross section

$$g(s)\,ds = \Sigma_t e^{-\int_0^s \Sigma_t ds'} ds \xrightarrow[\text{sample CDF}]{\text{Integrate PDF and}} \xi = 1 - e^{-\Sigma_t \cdot s} \xrightarrow[\text{for flight path}]{\text{Analytically solve}} s = \frac{-\ln(\xi)}{\Sigma_t}$$

Monte Carlo transport for spatially varying macroscopic cross section

$$g(s)\,ds = \Sigma_t(s)\, e^{-\int_0^s \Sigma_t(s')ds'} ds \xrightarrow[\text{optical depth}]{\text{Integrate PDF and (if collision) sample}} \hat{\tau} = -\ln\left[1 - (1 - P_{\text{NC}}(s_b))\,\xi\right] \xrightarrow[\text{to calculate distance}]{\text{Solve nonlinear equation}} \hat{\tau} = \int_0^s \Sigma(s) ds$$

$\tau(s) \equiv$ Optical depth $= \int_0^s \Sigma(s')\, ds'$

$P_{\text{NC}}(s_b) \equiv$ Probability of no-collision to boundary $= e^{-\tau(s_b)}$

$s_b \equiv$ Distance to nearest boundary

$\xi \equiv$ Random number on $(0, 1)$

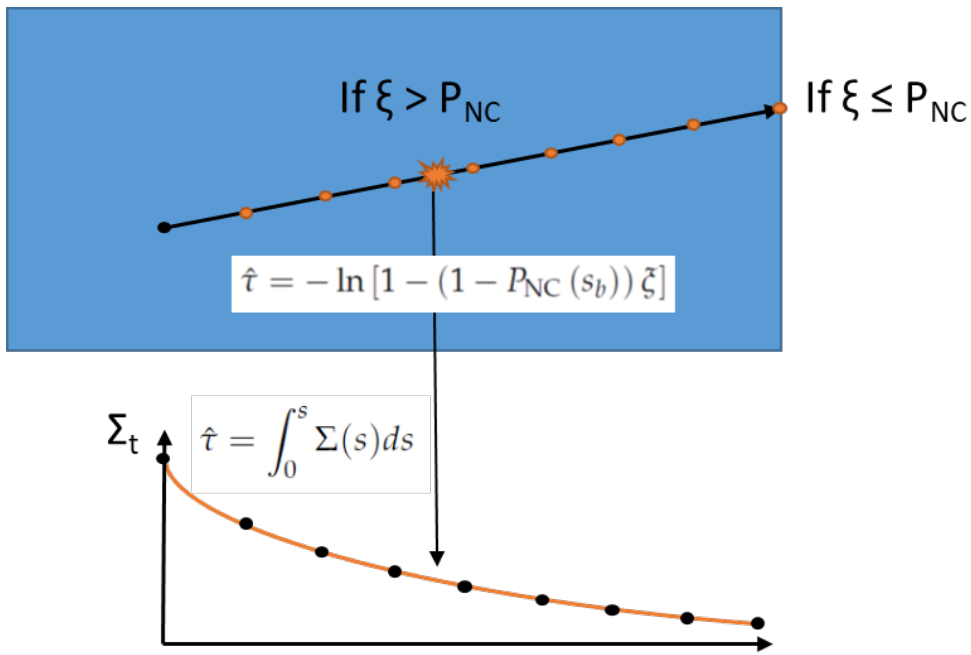
$\Sigma_t \equiv$ Total macroscopic cross section

$s \equiv$ Distance along flight path

CVMT Algorithm

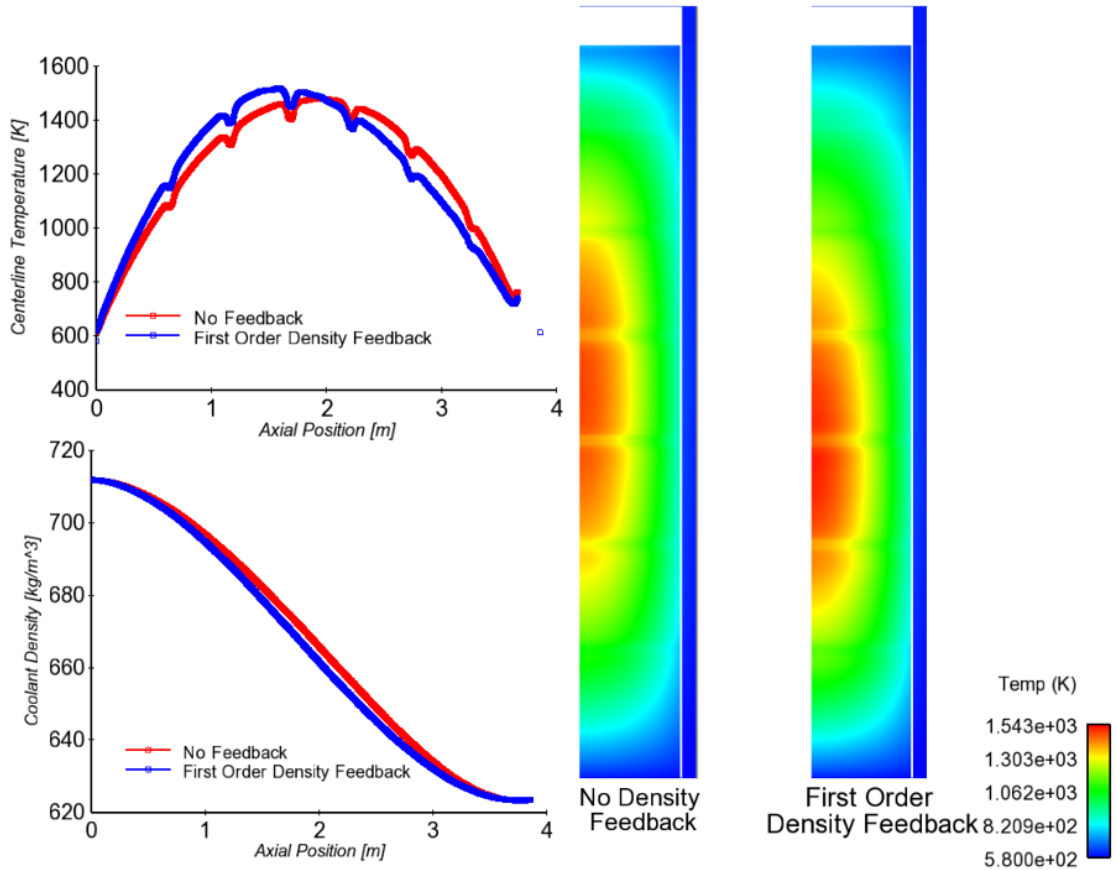
Algorithm 1 CVMT algorithm with numerical integration to determine optical depth and analytic inversion to determine neutron flight distance

1: Sample N equally spaced Σ_t values along neutron flight path
2: Compute $\tau\left(s_b\right)$ using three-point Newton-Cotes numerical integration
3: Compute $P_{\text{NC}} = exp[-\tau\left(s_b\right)]$
4: Sample ξ_1
5: **if** $\xi_1 \leq P_{\text{NC}}$ **then**
6: Move particle to cell boundary
7: **else**
8: Sample ξ_2
9: Compute $\hat{\tau} = -\ln\left[1-(1-P_{\text{NC}})\xi_2\right]$
10: Determine in which three-point Newton-Cotes numerical integral is the sampled $\hat{\tau}$
11: Calculate second-order polynomial coefficients a, b, c using $\Sigma_t(s_n), \Sigma_t(s_{n+1}), \Sigma_t(s_{n+2})$
12: Invert $\frac{1}{3}a\left(s'\right)^3 + \frac{1}{2}b\left(s'\right)^2 + c\left(s'\right) = \delta\tau$
13: Move particle calculated distance



Coupling Results - 3D Single Assembly PWR

- OpenMC/MOOSE coupling on 3D PWR assembly
- Zernike polynomials radially (4th to 6th order)
- Legendre polynomials axially (4th to 7th order)
- Power is passed to MOOSE and temperature and density profile returned to OpenMC
- 9 integration points in CVMT
- 1.5 to 3 times speedup over discretization



Conclusions

- OpenMC is a flexible open source research tool
- Most applications to date have centered on reactor simulations with the goal of resolving fully coupled transient simulations of full reactor cores
- Improvements in nuclear data representation plays a central role for OpenMC since they provide a natural path for vectorization and performance improvements