Accelerated Monte Carlo Simulation

Fuad Hasan, Jacob Merson

Department of Mechanical, Aerospace, and Nuclear Engineering Rensselaer Polytechnic Institute, 110 8th St. Troy, NY 12180

Abstract

This report presents the development of a GPU-accelerated library component of PUMI-Tally called `PhysicsKernel` for Monte Carlo neutral particle transport. PUMI-Tally is a GPU-accelerated unstructured mesh based track-length estimated tally library. By leveraging Kokkos - which is already being used in PUMI-Tally - for performance portability, the library implements basic Monte Carlo transport components on GPUs[1]. These include parallel random number generation for particle movement, efficient access to cross-section probability tables, and sampling of initial source distributions (uniform box and sphere), and the collision operator. The collision operator updates particle weights based on material cross sections, samples scattering angles, determines post-collision energy groups, and computes the path length to the next collision, updating particle positions in the mesh. This project aims to significantly accelerate Monte Carlo simulations, addressing a significant bottleneck in existing particle transport tools after unstructured mesh tallying. It builds on previous improvements in PUMI-Tally that exploited mesh adjacency for faster tallying. The `PhysicsKernel` is among the first GPU-based Monte Carlo physics kernels integrated with a mesh-aware tally system. To complete its full integration with PUMI-Tally and demonstrate, future work will include adding reflective boundary conditions, mesh-based source definitions, delta-tracking, fission reactions, and full integration testing with the tally system. Overall, the `PhysicsKernel` demonstrates a novel GPU Monte Carlo simulation code integrated with the PUMI-Tally system, paving the way for neutral particle transport simulations on GPUs.

Keywords: Monte Carlo, GPU, Neutronics, Fusion

PUMI-Pic Based Monte Carlo Neutron Transport:
Developer's repository link: `https://github.com/Fuad-HH/PumiUMTally/tree/physics_kernel`
Licensing provisions(please choose one): BSD 3-clause
Programming language: C++
Supplementary material: None
Nature of problem(approx. 50-250 words):
Efficient Monte Carlo simulations are crucial for fusion reactor simulation since neutrons play an important role in the fusion reactor dynamics, fuel generation, dose to structural materials and people. Due to the geometric complexity and size of the reactor's simulation domain, unstructured meshes and high number of particles are used. But the current solutions available are one of the major bottlenecks in a whole reactor simulation. It is found in one of our previous works that tallying on unstructured meshes and simulating the physics takes most of the time. In that previous work, tallying was addressed, and a significant amount of speedup was achieved

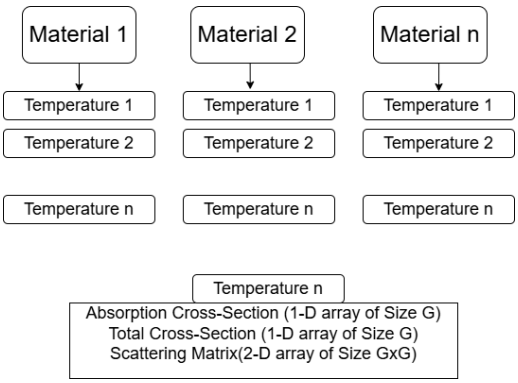


Figure 1: Structure of Multigroup Cross Sections

on both GPU and OpenMP based parallelization. It was also found that copying the particle properties from CPU to GPU for tallying was also taking a significant portion of the time of each iteration. Consequently, it is now imperative to simulate physics on GPU too, which should both cut the simulation time and copy time significantly.
Solution method(approx. 50-250 words):
The major steps in implementing the neutron physics, `PhysicsKernel`, were

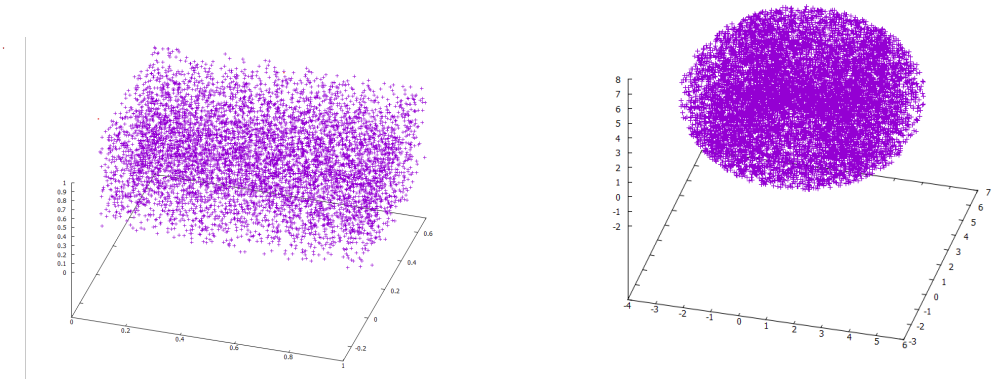
- Sampling Random Numbers
- Reading cross-section (Probability Tables for different reactions) efficiently
- Sampling Initial Source
- Transport Particles

Monte Carlo simulations are essentially random walks following the physics constraints of neutrons - making efficient random number generation crucial. Sampling random numbers in parallel is straightforwardly achieved using `Kokkos::Random_XorShift64_Pool` . In this project, multigroup cross-section are used, which is generally used fusion simulations. Multigroup cross-sections essentially means to have a table of probabilities for larger energy bins, in contrast to continuous cross-section, which contains probability against the continuous change in energy. Kokkos multidimensional arrays are used to store them for each material, temperature, and energy bins. The structure is shown in Figure 1. Each type of cross-sections are stored as below:

```
Kokkos::View<Omega_h::Real **> kTs_;     // mat, T
Kokkos::View<Omega_h::Real ***> sigma_t_;    // mat, T, g
Kokkos::View<Omega_h::Real ***> sigma_a_;    // mat, T, g
Kokkos::View<Omega_h::Real ****> scattering_matrix_; // mat, T, g, g
Kokkos::View<Omega_h::Real ***> sigma_s_;       // mat, T, g
```

which will allow efficient access to these arrays during transport. The `scattering_matrix` needs to be summed for each row for the last two indices, and then normalized. It is achieved using Hierarchical Parallelism of Kokkos.

There are two types of problems of neutron transport simulations: eigenvalue and fixed source. Only fixed source is solved here which is used in fusion reactor simulations. In this kind of simulation, a predefined source is sampled and then sampled until all the particles are either dead or have leaked out of the domain. Two kinds of geometries are now supported, box (given by corners) and sphere (given by radius and center).



(a) Uniform Box Source

(b) Uniform Spherical Source

Figure 2: Source Distribution of Fixed Source Simulationo

After creating the sources, transport is done. It involves four steps:

1. Calculate new particle weight based on cross-sections
2. Sample angle after scattering
3. Calculate energy group based on the angle
4. Sample distance to next collision and update position

They involve sampling random numbers and reading the multigroup cross-section data to do calculations. It is implemented in the `PumiTransport::nextCollision` method. It is not described here for brevity. After transport, the particle locations need to be sent to the `PUMI-Tally` library. To accommodate this new `PhysicsKernel` which stores the locations in Kokkos views, instead of vectors which is used before when `PUMI-Tally` was used in OpenMC, a new overload of `move_to_next_location` is created.

```
template <typename ViewType>
void move_to_next_location(ViewType particle_destinations, ViewType weights);
```

Additional comments including restrictions and unusual features (approx. 50-250 words):

Future Works

To complete the project and run demonstrations, the following needs to be done:

- Implement reflective boundary condition
- Mesh based source: source that can populate a specific volume in the geometry
- Implement delta-tracking
- Integration testing with tallies
- Fission reactions
- Profiling

Compile and Run

Follow this workflow to compile and run on Ubuntu:
`https://github.com/Fuad-HH/PumiUMTally/blob/physics_kernel/.github/workflows/ci.yml`.

References

[1] Fuad Hasan, Cameron W. Smith, Mark S. Shephard, R. Michael Churchill, George J. Wilkie, Paul K. Romano, Patrick C. Shriwise, and Jacob S. Merson. Gpu acceleration of monte carlo tallies on unstructured meshes in openmc with pumi-tally. Manuscript submitted for publication, 2025.