M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea, April 16-20, 2017, on USB (2017)

Delta-tracking in the GPU-accelerated WARP Monte Carlo Neutron Transport Code

Kelly L. Rowland*, Ryan M. Bergmann†, R. N. Slaybaugh*, Jasmina L. Vujić*

*Department of Nuclear Engineering; University of California, Berkeley; Berkeley, CA, USA
†Paul Scherrer Institute, Villigen, Switzerland
krowland@berkeley.edu, ryanmbergmann@gmail.com, slaybaugh@berkeley.edu, vujic@nuc.berkeley.edu

Abstract - *Graphics Processing Units (GPUs) have increased in computational power, offering a higher aggregate memory bandwidth and many more floating-point operations per second (FLOPS) than traditional central processing units (CPUs). As such, many new supercomputing platforms are being built to incorporate GPUs to increase computational capacity, and software must be adapted to or developed for these emerging architectures. WARP ("Weaving All the Random Particles") is a new code that efficiently performs three-dimensional (3D) continuous energy Monte Carlo neutron transport code on a GPU. Like traditional Monte Carlo neutron transport codes, WARP uses ray tracing and distance-to-collision calculations to follow the random walks of neutrons in a system. Specifically, WARP uses the OptiX ray tracing framework, a highly-optimized library developed by NVIDIA, to handle the system geometry. This work discusses the addition of delta-tracking to WARP as an alternative to the distance-to-collision calculation to explore the effects of the method when used in a Monte Carlo neutron transport code executed on a GPU. The delta-tracking method allows neutron random walks to continue over different material regions without stopping the particle at each boundary surface, and the geometry routine is reduced to determining the material at each collision point. It was found that the delta-tracking version of WARP incurs significantly longer runtimes than the original version. The figures-of-merit of the calculations performed by the different physics routine range from one order of magnitude greater than that of the original version of the code for the simplest tested geometry configuration to one order of magnitude worse than that of the original version of the code for more complex configurations.*

I. INTRODUCTION

Graphics processing units (GPUs) have gradually increased in computational power from small, job-specific boards to programmable powerhouses. Compared to more common central processing units (CPUs), GPUs have a higher aggregate memory bandwidth, much higher floating-point operations per second (FLOPs), and lower energy consumption per FLOP [1].

WARP, which can stand for "Weaving All the Random Particles", is a three-dimensional (3D), continuous energy, Monte Carlo neutron transport code originally developed by Dr. Ryan M. Bergmann at UC Berkeley to efficiently execute on a CPU/GPU platform [1]. The main function of the code is executed on a CPU, with "kernels" offloaded onto the GPU. WARP uses a boundary representation geometry based on surfaces. The impetus for developing GPU-based codes such as WARP is that many emerging supercomputing platforms are including GPUs as accelerators. To make effective use of these platforms, nuclear engineering researchers must either adapt existing codes to these new platforms or begin developing new Monte Carlo neutron transport software for efficient use on GPUs. Both routes require substantial effort; this work discusses a particular venture in developing a new code specifically for use on GPUs.

In developing codes for new architectures, it is imperative that we explore algorithms alternative to those traditionally used in codes written for CPU-based machines. What may work well on a CPU may not work well on a GPU, and vice versa. The delta-tracking physics routine has yet to be implemented and tested in a GPU-based Monte Carlo neutron transport code; the work presented here discusses the implementation and results of implementing delta-tracking in the GPU-accelerated WARP code.

In this paper, we start by presenting a foundation for testing the delta-tracking routine on the GPU architecture, the theory behind delta-tracking, and a brief history of the algorithm in past and present Monte Carlo neutron transport codes. Results of the algorithm's implementation are compared against the base version of the WARP code in addition to results from MCNP 6.1 [2] and Serpent 2.1.21 [3], with concluding remarks following.

II. THEORY

1. Vector Computing as a Basis for GPU Algorithms

WARP is based on research done in the 1980s and 1990s for mapping the Monte Carlo and collision probability methods onto vector computers [1]. These methods on those machines used an "event-based" algorithm in which neutrons are organized and processed according to their required operation. If a neutron is about to undergo inelastic scattering, its data are put into the inelastic scattering buffer. The same process is done for all reaction types: neutrons inducing fission are placed in the fission buffer, and so on. Vector processing calculations such as these are more generally referred to as "single instruction multiple data", or SIMD, processes. Since GPUs are massively parallel and rely on SIMD, WARP uses a modified event-based algorithm for GPU-accelerated Monte

Carlo neutron transport.

The RACER and VMONT codes were both vectorized Monte Carlo codes that used delta-tracking for the purpose of reducing runtime and processing [4, 5]. It follows logically that, since delta-tracking has been shown to be efficient in these vector machine codes, it has the potential to be efficient when used on a GPU-accelerated, vectorized Monte Carlo code such as WARP.

2. Delta-tracking

In Monte Carlo neutron transport, we track anything that can happen to neutrons in a system: collisions and their outcomes, surface crossings, etc. To do that, we need to track how these particles move in the system. This starts with determining the distance covered between events. Neutron path lengths, ℓ, in random walks of Monte Carlo simulations are sampled from exponential distributions using

$$\ell = \frac{-\ln \xi}{\Sigma_{\text{tot,m}}(E)}, \tag{1}$$

where ξ is a uniformly-distributed random variable on the unit interval and $\Sigma_{\text{tot,m}}$ is the macroscopic total cross section of the material m in which the neutron is located [6].

The sampled path length is not statistically valid if the neutron crosses a material boundary; the flight is stopped at the boundary surface and a new path length is sampled with the new material total cross section. This is the main principle of conventional ray tracing [6] and is what is done in WARP using the OptiX framework.

The idea behind delta-tracking is to effectively homogenize the total cross sections such that sampled path lengths are valid over the entire geometry. Consider the concept of the "virtual collision", a scattering reaction that preserves neutron energy and direction. Since virtual collisions have no impact on the final outcome, any number of them can "happen" in the random walk. Thus, the total cross section of a material can be adjusted by adding an arbitrary virtual collision cross section, $\Sigma_{0,\text{m}}(E)$ [6]:

$$\Sigma'_{\text{tot,m}}(E) = \Sigma_{\text{tot,m}}(E) + \Sigma_{0,\text{m}}(E). \tag{2}$$

Because all material cross sections can now be freely and arbitrarily increased, a *majorant* cross section can be assigned to represent the effective total (physical + virtual) collision probability in all materials:

$$\begin{aligned}\Sigma_{\text{maj}}(E) &= \Sigma'_{\text{tot,1}}(E) = \Sigma'_{\text{tot,2}}(E) = \ldots = \Sigma'_{\text{tot,M}}(E) \\ &= \max\{\Sigma_{\text{tot,1}}(E), \Sigma_{\text{tot,2}}(E), \ldots, \Sigma_{\text{tot,M}}(E)\}\,.\end{aligned} \tag{3}$$

Path lengths sampled using the global majorant are statistically valid in all materials, effectively homogenizing the material total cross sections and eliminating the need to calculate surface intersection distances [6]. Instead, an additional step is included in the tracking routine for handling virtual collisions. Rejection sampling is carried out for each collision, and the collision point is accepted with probability

$$P_{\text{col}}(E) = \frac{\Sigma_{\text{tot,col}}(E)}{\Sigma_{\text{maj}}(E)}. \tag{4}$$

If the point is rejected, the collision is considered virtual and the random walk continues by sampling a new path length.

The inherent advantage of delta-tracking is that the neutron random walk can be continued over material boundaries without stopping the walk at each surface crossing. The geometry routine reduces to determining the material in which the collision point resides, which is computationally less expensive than calculating the surface distances when running a simulation on traditional CPU hardware [6]. The simplified geometry routine is advantageous in that delta-tracking is often faster than using distance-to-collision calculations in complex geometries, and complicated objects and surface types are easier to handle with a delta-tracking algorithm [6].

One shortcoming of the delta-tracking method arises when material total cross sections differ greatly. A representative example of this is a light water reactor (LWR) fuel assembly that contains localized heavy absorbers (such as control rods or burnable poison rods) [6]. Although the absorber itself occupies a relatively small space physically, its large cross section dominates the majorant at low neutron energies. This causes the neutron random walk to be cut into many short tracks, wasting computing time by continually incurring virtual collisions and resampling.

Additionally, using the delta-tracking method necessitates the use of the collision flux estimator rather than the track-length flux estimator, which is generally considered a drawback for implementation in traditional Monte Carlo codes [6]. The track-length estimation of the reaction rate integral can be written in simplest analog form as

$$R_{\text{tle}} = \sum_{i=1}^{I} \ell_i f_i, \tag{5}$$

where ℓ_i is the i^{th} neutron track length, f_i is the i^{th} response function, and the summation is over all tracks in the region of interest. This estimator cannot be used when employing the delta-tracking method because the sampled neutron paths may extend over several material regions and the discontinuity points are not known [6].

An alternative to the track-length flux estimator is the collision flux estimator:

$$R_{\text{cfe}} = \sum_{i=1}^{I} \frac{f_i}{\Sigma_{\text{tot,i}}}, \tag{6}$$

where $\Sigma_{\text{tot,i}}$ is the material total macroscopic cross section at the i^{th} collision site and the summation is over all collisions within the region of interest [6]. This flux estimator is problematic in that it often results in poor efficiency for tallies scored in regions of low volume or with low collision density. However, the use of the collision flux estimator should not be considered a disadvantage for the implementation of the delta-tracking method in WARP; this estimator is already the code's only flux tally method [1].

3. Delta-tracking in Existing Monte Carlo Neutron Transport Codes

Over the years, many codes have had delta-tracking available as at least an option for neutron tracking. We will point

out, however, that none of these codes were implemented on GPUs. For an extended review, please see UCB-NE-5154 [7].

The Serpent Monte Carlo calculation code employs a unique combination of both the standard distance-to-collision calculation and delta-tracking in its geometry routine [6]. Serpent switches to using the distance-to-collision calculation when collision sampling efficiency is low. Selection between the two methods is done by comparing the neutron mean free path resulting from the majorant to the physical value of the mean free path given by the material total cross section at the neutron's location [6]. If

$$\frac{\ell_{\text{maj}}(E)}{\ell_m(E)} = \frac{\Sigma_{\text{tot,m}}(E)}{\Sigma_{\text{maj}}(E)} > 1 - c, \tag{7}$$

where ℓ_m is the material-calculated path length and ℓ_{maj} is the majorant-calculated path length, the neutron path length is sampled using the majorant cross section and rejection sampling is subsequently carried out at the collision point. The constant c is the predefined cutoff criterion valued between 0 (no delta-tracking) and 1 (no distance-to-collision calculation) and set to a default fixed value of 0.90 [6], which can be changed by the user.

Serpent parametric studies compared the flux estimates in two LWR lattice configurations to demonstrate that the decrease in runtime from the use of the delta-tracking method is not outweighed by poor statistics. Comparisons were done with the figure-of-merit (FOM) metric, defined in Eqn. (8), which incorporates both the calculation time T and the relative statistical error $\Delta x/x$ of the result [6].

$$\text{FOM} = \frac{1}{T(\Delta x/x)^2} \tag{8}$$

In these studies, it was found that, for parameters integrated over the entire geometry, the accuracy of the two methods is comparable.

III. DELTA-TRACKING IN WARP

To understand what is happening in WARP, what follows is an explanation of the general delta-tracking algorithm, a comparison to the distance-to-collision calculation algorithm, discussion of the specific implementation in WARP, and some comments about the performance potential of using delta-tracking on GPUs for the first time.

In WARP, the delta tracking algorithm executes these steps:

- get current material using OptiX trace;
- calculate majorant cross section, sample path lengths, update neutron coordinates;
- update material information using OptiX trace;
- calculate material total cross section and perform rejection sampling;
- if collision is real (not "virtual"), determine with which isotope neutron interacts;
- determine neutron interaction.

For direct comparison, tracking particles via distance-to-collision calculation only in WARP is executed as:

- get current material using OptiX trace;
- calculate macroscopic total cross section and sample path length;
- determine with which isotope neutron interacts;
- move neutron to collision site or cell boundary, whichever is closer;
- determine neutron interaction.

From there, with either tracking scheme, neutrons are grouped based on the interactions that they are about to undergo and the vectors are subsequently processed.

1. Implementation

In WARP and other Monte Carlo neutron transport codes, particle positions are updated as:

$$x \mathrel{+}= \ell\hat{x}, \quad y \mathrel{+}= \ell\hat{y}, \quad z \mathrel{+}= \ell\hat{z}; \tag{9}$$

where ℓ, the sampled travel distance is defined as:

$$\ell = \begin{cases} -\ln\xi \,/\, \Sigma_{\text{tot}} & \text{with distance-to-collision calc.,} \\ -\ln\xi \,/\, \Sigma_{\text{maj}} & \text{with delta-tracking.} \end{cases} \tag{10}$$

If the distance-to-collision calculation is being used, a check is performed between the sampled neutron flight distance and the distance to the nearest boundary crossing on that trajectory. If the boundary crossing is closer than the sampled distance, the particle position is instead updated as

$$x \mathrel{+}= s\hat{x} + \varepsilon x_{\text{norm}},\ y \mathrel{+}= s\hat{y} + \varepsilon y_{\text{norm}},\ z \mathrel{+}= s\hat{z} + \varepsilon z_{\text{norm}}; \tag{11}$$

where s is the distance to the surface to be crossed. The epsilon term added to the position is required by the OptiX framework; the inclusion of this term ensures that the neutron is sufficiently far into the new cell such that it can be detected by OptiX and the corresponding material updated accordingly. The second term uses vector norms rather than the vector directions in Eqn. (9) to move particles away from the surface in a perpendicular manner rather than potentially along a surface boundary (which could cause issues with OptiX detection). In this case of a particle crossing a boundary with the distance-to-collision calculation, the particle is stopped at the updated location and the flight path distance must be resampled using the total macroscopic cross section of the new material.

Although delta-tracking circumvents the above resampling in the case of surface crossings, there is a potential need to resample at each collision site. The collision site is accepted as an actual reaction with probability

$$P_{\text{col}}(E) = \frac{\Sigma_{\text{tot,col}}(E)}{\Sigma_{\text{maj}}(E)} \tag{12}$$

and rejected as a "virtual" collision otherwise. It is important to note that this rejection sampling must use the total macroscopic cross section of the material in which the collision is being considered and not the material in which the neutron was originally located. If the collision is considered virtual, a new path length is sampled with the majorant cross section again, and the transport process continues.

It should be noted that calculating the majorant cross section in WARP is done naïvely:

$$\Sigma_{\text{maj}}(E) = \max\{\Sigma_{\text{tot},1}(E), \Sigma_{\text{tot},2}(E), \ldots, \Sigma_{\text{tot,M}}(E)\}\,, \qquad (13)$$

where M is the number of materials in the entire system. One potential area of improvement would be to calculate the majorant cross section along the neutron's trajectory rather than this "global" majorant cross section.

With either tracking method, once the neutron has been determined to undergo an actual collision, the reaction it will experience is chosen using the microscopic cross sections of the material in which the particle is located. Following this, the reaction is processed, and transport continues as above.

2. Performance Potential of Delta-tracking in WARP

Because the OptiX trace is used to determine the properties of a neutron's current cell, and this must be done twice in each iteration of the delta-tracking process, the delta tracking version uses many more calls to the OptiX ray tracing routine than the original distance-to-collision calculation method. For this reason, this implementation of delta-tracking may not be faster than the standard method for simple geometry and material configurations. It has been seen in various other Monte Carlo neutron transport codes that the bulk of the runtime is involved with ray tracing, and WARP is not too different from other codes in that regard. This issue could potentially be circumvented if a different framework were to be used rather than OptiX, but this would require a sizable overhaul of the code and was not considered for this work. Further, the OptiX framework is optimized for NVIDIA GPUs. A handwritten hybrid system of acceleration structures for cell boundaries and constructive solid geometry for cell surface sense calculation may perform better for delta-tracking since it would eliminate the iterative ray tracing.

However, it is expected that, like in other codes, delta-tracking may be more efficient in WARP when considering certain geometry configurations. Like Serpent, WARP is intended for calculations in nuclear reactor analysis. Many reactor configurations consist of lattices composed of many geometry and material boundaries, causing simulations that use ray tracing to run slowly because neutron flights stop at each boundary crossing. Thus, we hope that the time saved in not stopping neutron flights at boundary crossings may compensate for having to call an additional OptiX trace in the delta-tracking method implemented in WARP.

IV. COMPUTING PLATFORMS AND TESTS

In this section, we will describe the computing platforms and various geometry and material configurations used in testing the delta-tracking version of WARP against other codes. The CPU and GPU platforms used in testing WARP are those available on the UC Berkeley Savio computing cluster [8]. For the CPU platform, a single Savio2 Lenovo NeXtScale nx360 M5 node (equipped with two Intel Xeon 12-core Haswell processors) was used. For the GPU platform, an NVIDIA Tesla K80 graphics card was used; the GPU nodes on the machine each include 2 Intel Xeon 4-core 3.0 GHz Haswell processors. Specifications of the two platforms are listed in Table I.

TABLE I: Specifications of computing platforms used in the benchmark cases [8].

Platform	**Physical Processors**	**Processor Frequency**
Lenovo NeXtScale nx360 M5	24	2.3 GHz
NVIDIA Tesla K80	13	823.5 MHz
Platform	**Local Memory**	**Memory Frequency**
Lenovo NeXtScale nx360 M5	64 GB	2.133 GHz
NVIDIA Tesla K80	12 GB	2.505 GHz

The benchmark cases were run with 6.5×10^6 neutrons per criticality batch with 20 initial batches discarded and 40 batches with statistics accumulated (60 batches total). The CPU cases were run with the optimal number of MPI processes and threads per process for the specific node to fairly measure the performance of the unit as a whole: 4 MPI processes each running 6 threads were used. The MCNP runs were launched with one additional MPI process since the master process does not perform any neutron tracking in MCNP. All codes were set to not use thermal scattering or unresolved resonance tables and to use analog capture only; this is because WARP does not yet include thermal scattering, unresolved resonance tables, or non-analog capture and the comparisons are intended to be one-to-one between the codes.

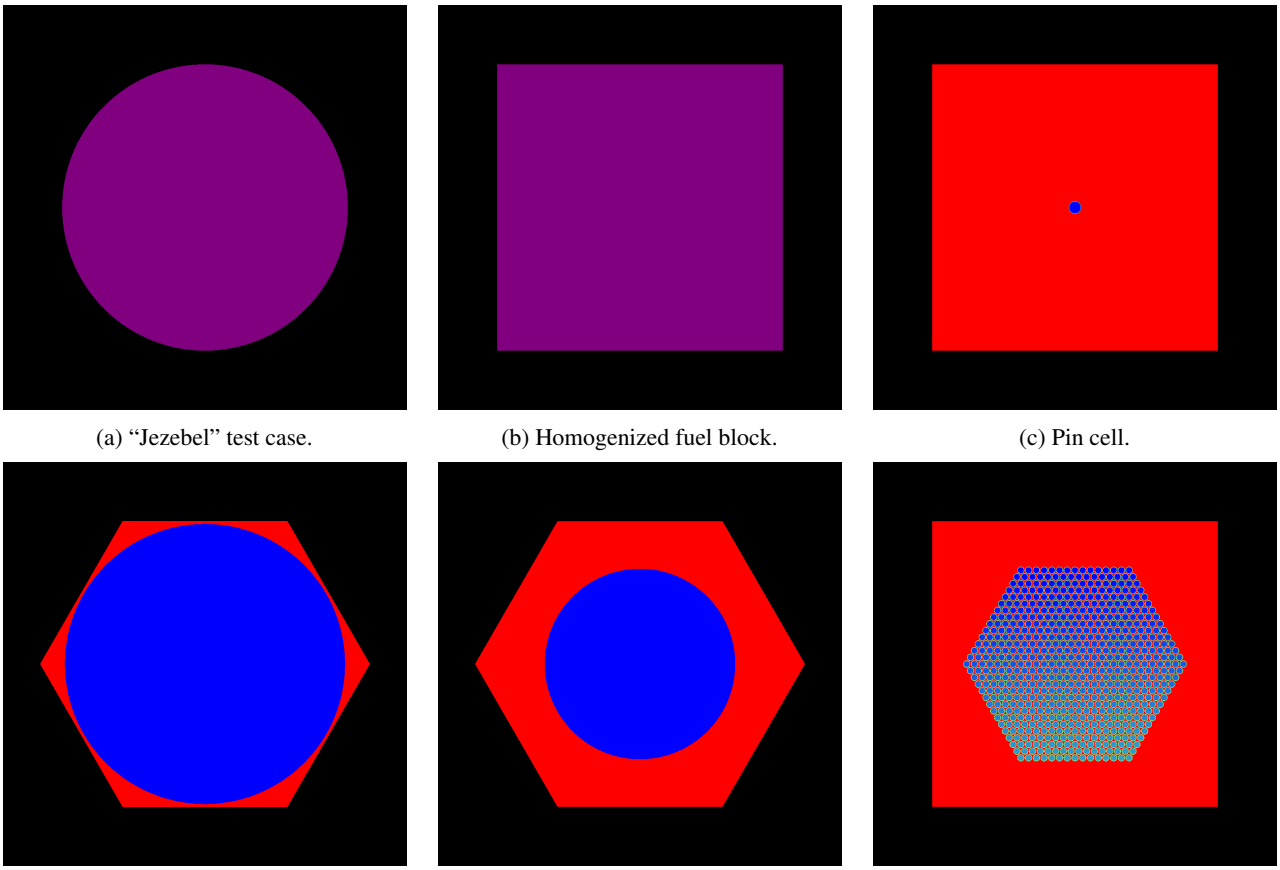
All tests use ENDF/B-VII cross sections that are distributed with the Serpent 1.1.7 release from RSICC. These cross sections contain fewer energy grid points than the ENDF/B-VII.1 cross sections that are distributed with the MCNP 6.1 release from RSICC. Since the Serpent 1.1.7 data require less storage space, using them allows more isotopes to be used by WARP. Using larger datasets such as those required for depletion problems is an area of future investigation for WARP.

1. "Jezebel" Bare Pu Sphere

The "Jezebel" criticality test is a bare plutonium/gallium sphere with vacuum boundary conditions; it is a standard test used to validate neutron transport codes and is described in the International Handbook of Evaluated Criticality Safety Test Experiments under the name "Pu-MET-FAST-001" [10]. The geometry and materials are outlined in Table II and a cross-sectional view of the configuration is shown in Figure 1a. All cross sections used were processed at 273.5 K [9].

2. Homogenized Fuel Block

The homogenized block criticality test is a bare cuboid with vacuum boundary conditions. The geometry and materials are outlined in Table III and a cross-sectional view of the configuration is shown in Figure 1b. All cross sections used were processed at 273.5 K [9].



(a) "Jezebel" test case. (b) Homogenized fuel block. (c) Pin cell.

(d) Pebble in FLiBe. (e) Sodium pin cell. (f) Hexagonal pin lattice.

Fig. 1: Horizontal slices of the geometry configurations of the different test cases. Different colors represent different cell numbers; black represents space outside the outermost boundary [9].

TABLE II: Geometry and materials used in the "Jezebel" benchmark.

Cells	Isotopes (Atm. %)	Densities
1 sphere, r = 6.6595 cm	Pu-239 (0.7381) Pu-240 (0.1942) Pu-241 (0.0299) Pu-242 (0.0038) Ga-69 (0.0203) Ga-71 (0.0135)	15.73 g/cm^3

3. Zr-clad UO_2 Pin in Heavy Water

This criticality test consists of a UO_2 cylinder clad in zirconium surrounded by a large block of heavy water. The water block has vacuum boundary conditions and the geometry and materials are outlined in Table IV; a cross-sectional view of the configuration is shown in Figure 1c. All cross sections used were processed at 273.5 K [9].

TABLE III: Geometry and materials used in the homogenized fuel block test case.

Cells	Isotopes (Atm. %)	Densities
1 cuboid, 100×100×50 cm	U-235 (0.10) U-238 (0.90) O-16 (3.0) H-2 (2.0) Zr-90 (0.5145) Zr-91 (0.1122) Zr-92 (0.1715) Zr-94 (0.1738) Zr-96 (0.0280)	5.50 g/cm^3

4. Homogenized Fuel Pebble in FLiBe

This criticality test consists of a single sphere of homogenized UO_2 and C surrounded by molten FLiBe (Li_2BeF_4) salt. The outer cell is a right hexagonal prism with specular reflective boundary conditions. The geometry and materials are outlined in Table V, where "r" shown for the hexagonal prism is the length of the apothem. A cross-sectional view of the configuration is shown in Figure 1d. The FLiBe cross

sections used were processed at 900 K and the pebble cross sections at 1200 K [9].

TABLE IV: Geometry and materials used in the single UO_2 pin in D_2O test case.

Cells	Isotopes (Atm. %)	Densities
1 cylinder; r = 2 cm, z = ± 20 cm	U-238 (0.90) U-235 (0.10) O-16 (2.00)	10.97 g/cm^3
1 cylinder; r = 2.2 cm, z = ± 20.2 cm	Zr-90 (0.5145) Zr-91 (0.1122) Zr-92 (0.1715) Zr-94 (0.1738) Zr-96 (0.0280)	6.52 g/cm^3
1 box, $50 \times 50 \times 50$ cm	H-2 (2.0) O-16 (1.0)	1.11 g/cm^3

TABLE V: Geometry and materials in the pebble in FLiBe case.

Cells	Isotopes (Atm. %)	Densities
1 sphere, r = 5.0 cm	U-238 (0.90) U-235 (0.10) O-16 (2.00) C-12 (1.978) C-13 (0.022)	8.75 g/cm^3
1 right hex prism, r = 5.1 cm	Li-6 (0.15) Li-7 (1.85) Be-9 (1.00) F-19 (4.00)	1.94 g/cm^3

5. Stainless Steel-Clad Metallic Uranium Pin in Liquid Sodium

This test consists of a single cylinder of metallic uranium surrounded by molten sodium. The outer cell is a right hexagonal prism with specular reflective boundary conditions. The geometry and materials are outlined in Table VI and a cross-sectional view of the configuration is shown in Figure 1e. The fuel and clad cross sections used were processed at 900 K, and the sodium coolant cross sections were processed at 600 K. In order to keep the specification tenable, only the major components of 316 stainless steel were included in the clad material [9].

6. Zr-Clad Hexagonal UO_2 Pin Cell Lattice in Light Water

This criticality test consists of 631 Zr-clad UO_2 cylinders laid out in a hexagonal lattice surrounded by light water. The material compositions, densities, and cylinder dimensions are similar to the pin cell test case, but, since this test has two orders of magnitude more objects, it serves to highlight the effect of introducing many geometric objects into the problem. The lattice is in the x-y plane, has a pitch-to-diameter ratio of 1.164, and has 15 elements on a side. Geometry and materials are outlined in Table VII with a cross-sectional view of the configuration shown in Figure 1f; all cross sections used were processed at 273.5 K [9].

TABLE VI: Geometry and materials in the steel-clad metallic uranium single pin test case.

Cells	Isotopes (Atm. %)		Densities
1 cylinder, r = 1.0 cm	U-235 (0.10) U-238 (0.90)		19.1 g/cm^3
1 cylinder, r = 1.2 cm	Fe-54 (0.0435) Fe-56 (0.6879) Fe-57 (0.0165) Fe-58 (0.0021) Cr-50 (0.0065) Cr-52 (0.1257)	Cr-53 (0.0143) Cr-54 (0.0035) Ni-58 (0.0681) Ni-60 (0.0262) Ni-62 (0.0036) Ni-64 (0.0009)	7.99 g/cm^3
1 hex prism, r = 1.8 cm	Na-23 (1.00)		0.927 g/cm^3

TABLE VII: Geometry and materials used in the Zr-clad UO_2 pin hexagonal lattice in light water test case.

Cells	Isotopes (Atm. %)	Densities
631 cylinders; r = 1.0 cm, z = ± 20 cm	U-235 (0.10) U-238 (0.90) O-16 (2.00	10.97 g/cm^3
631 cylinders; r = 1.2 cm, z = ± 20.2 cm	Zr-90 (0.5145) Zr-91 (0.1122) Zr-92 (0.1715) Zr-94 (0.1738) Zr-96 (0.0280)	6.52 g/cm^3
1 box, $48 \times 48 \times 48$ cm	H-1 (2.0) O-16 (1.0)	1.00 g/cm^3

V. RESULTS AND ANALYSIS

In this section, we will compare the results of the delta-tracking version of WARP to those calculated by the original version of WARP in addition to results from MCNP 6.1 and Serpent 2.1.21.

Results are shown in Table VIII and Figures 2 – 7 [9]. Relative differences versus MCNP and Serpent are in red and blue for the original and delta-tracking versions of WARP, respectively. The grey shaded area in the error subplots shows the space within two standard deviations of the statistical uncertainty of the production code. It can be seen in Figures 2 – 7 that both versions of WARP compare well to MCNP but generally have a constant positive offset error compared to Serpent. Since this offset appears in multiple spectra, it may indicate a possible normalization discrepancy between MCNP and Serpent. More significant deviations between WARP and MCNP occur at large resonances, indicating that some reaction sampling may not be treated exactly the same way in WARP as in Serpent and MCNP. Despite these discrepancies in the flux spectra, the value of k_{eff} calculated by the delta-tracking version of WARP matches that of the other codes within statistical uncertainty.

Although the delta-tracking version of WARP incurs

TABLE VIII: Multiplication factors, runtimes, and figures-of-merit for the various benchmarks. All runtimes are in minutes.

	Jezebel			**Homogenized Block**		
Code	k_{eff}	Runtime	FOM	k_{eff}	Runtime	FOM
MCNP 6.1	$0.9999600 \pm 7.00\times10^{-5}$	4.95	4.12×10^{7}	$0.5933000 \pm 4.00\times10^{-5}$	44.03	1.42×10^{7}
Serpent 2.1.21	$0.9997840 \pm 9.50\times10^{-5}$	6.32	1.75×10^{7}	$0.5934090 \pm 1.20\times10^{-4}$	24.10	2.88×10^{6}
WARP (orig.)	$0.9999602 \pm 9.58\times10^{-5}$	1.18	9.23×10^{7}	$0.5932266 \pm 1.24\times10^{-4}$	5.79	1.12×10^{7}
WARP (DT)	$0.9999824 \pm 5.53\times10^{-5}$	1.43	2.29×10^{8}	$0.5935092 \pm 1.35\times10^{-4}$	9.27	5.92×10^{6}
	Pin Cell			**FLiBe**		
Code	k_{eff}	Runtime	FOM	k_{eff}	Runtime	FOM
MCNP 6.1	$0.2751700 \pm 7.00\times10^{-5}$	64.18	3.18×10^{6}	$0.8805100 \pm 6.00\times10^{-5}$	81.03	3.43×10^{6}
Serpent 2.1.21	$0.2750510 \pm 1.80\times10^{-4}$	76.51	4.03×10^{5}	$0.8804900 \pm 8.70\times10^{-5}$	45.08	2.93×10^{6}
WARP (orig.)	$0.2749884 \pm 1.56\times10^{-4}$	20.91	1.97×10^{6}	$0.8807546 \pm 9.58\times10^{-5}$	24.68	4.41×10^{6}
WARP (DT)	$0.2754006 \pm 1.75\times10^{-4}$	191.61	1.70×10^{5}	$0.8805805 \pm 5.53\times10^{-5}$	54.85	5.96×10^{6}
	Sodium Pin			**Hex Assembly**		
Code	k_{eff}	Runtime	FOM	k_{eff}	Runtime	FOM
MCNP 6.1	$1.0987700 \pm 6.00\times10^{-5}$	199.15	1.39×10^{6}	$1.0506500 \pm 9.00\times10^{-5}$	112.25	1.10×10^{6}
Serpent 2.1.21	$1.0987100 \pm 2.20\times10^{-4}$	68.46	3.02×10^{5}	$1.0503300 \pm 7.20\times10^{-5}$	44.32	4.35×10^{6}
WARP (orig.)	$1.0985898 \pm 5.53\times10^{-5}$	81.40	4.02×10^{6}	$1.0511035 \pm 1.24\times10^{-4}$	35.63	1.83×10^{6}
WARP (DT)	$1.0984879 \pm 7.90\times10^{-5}$	203.42	7.88×10^{5}	$1.0511572 \pm 9.58\times10^{-5}$	754.00	1.45×10^{5}

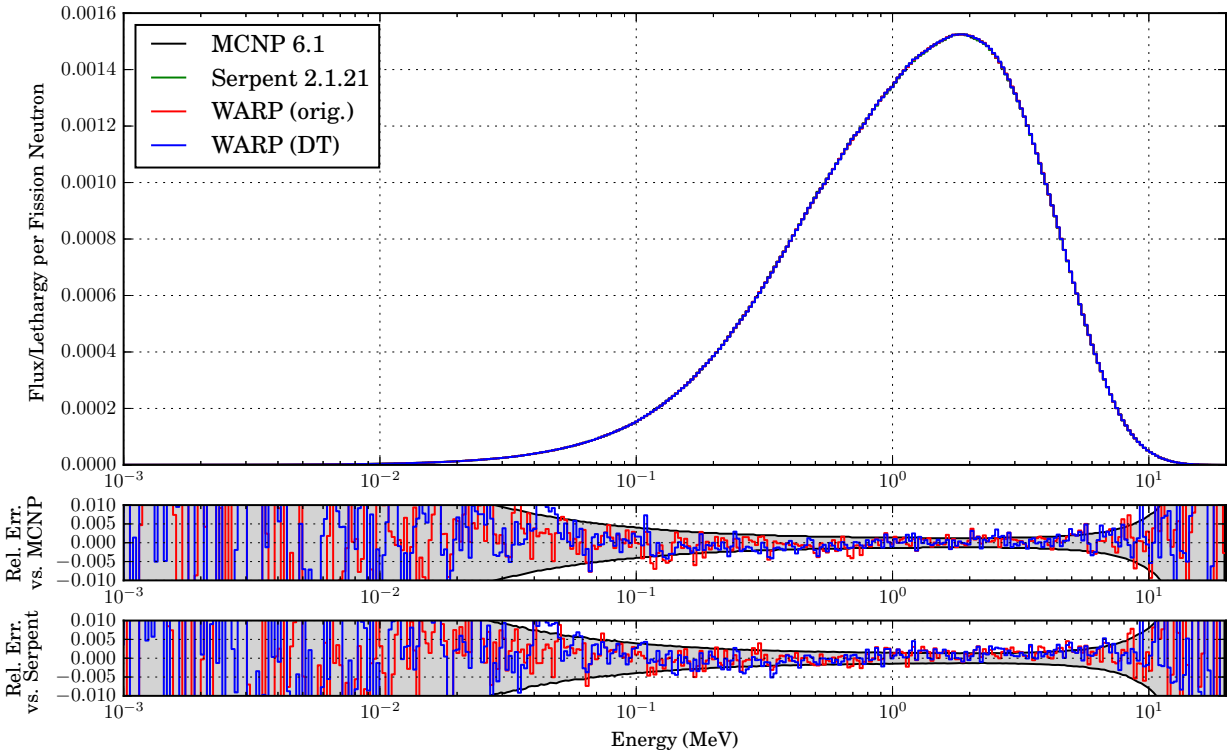


Fig. 2: Volume-averaged flux spectra for the "Jezebel" benchmark.

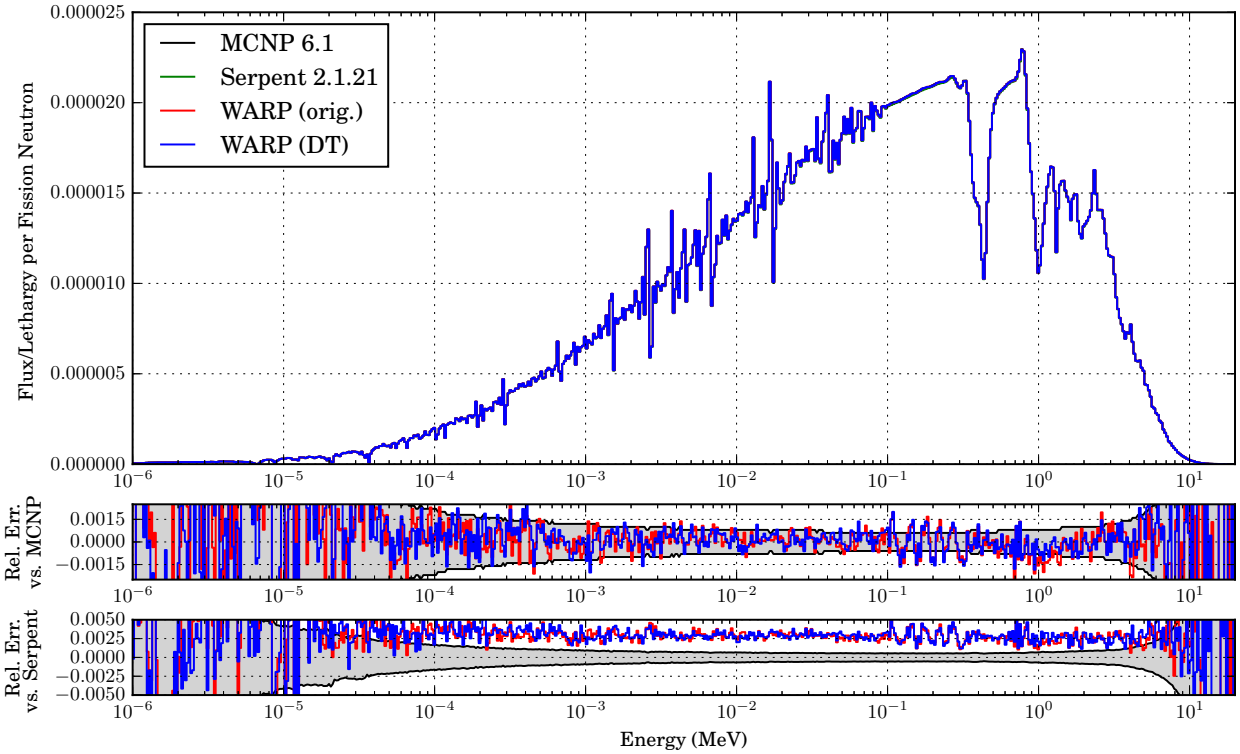


Fig. 3: Volume-averaged flux spectra inside the homogenized fuel test case.

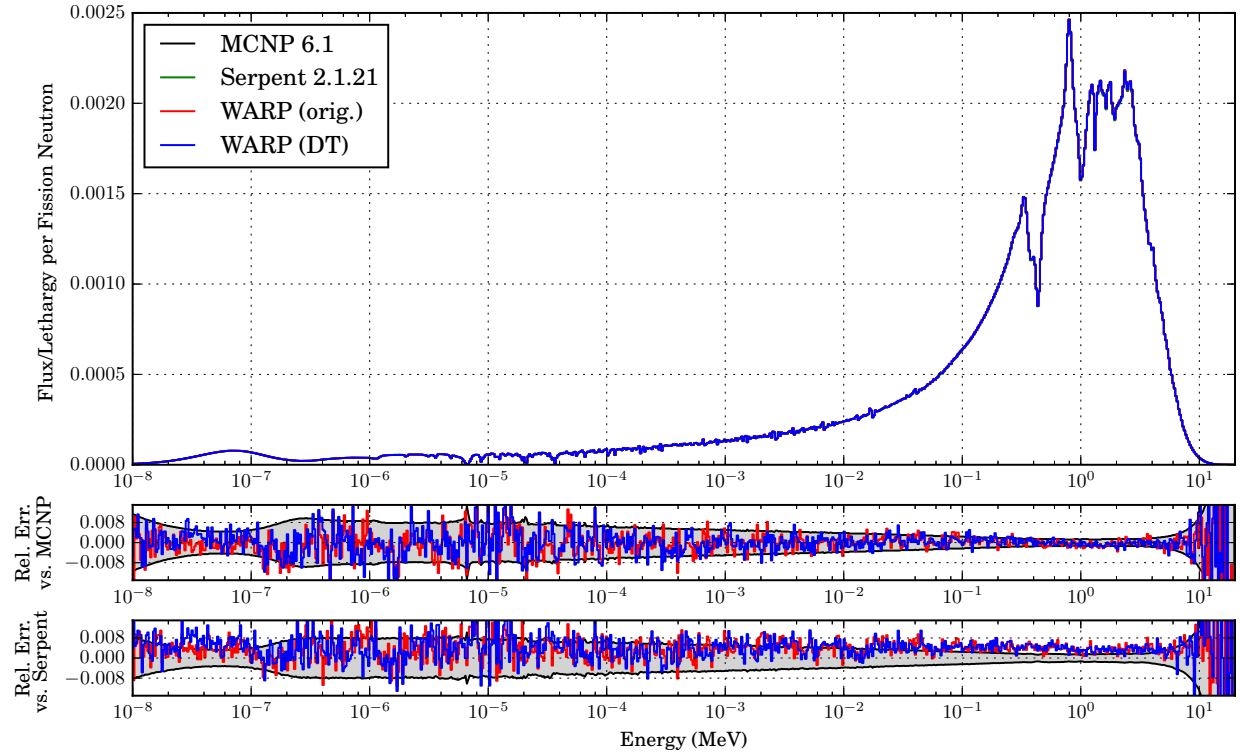


Fig. 4: Volume-averaged flux spectra inside the fuel pin in the pin cell test case.

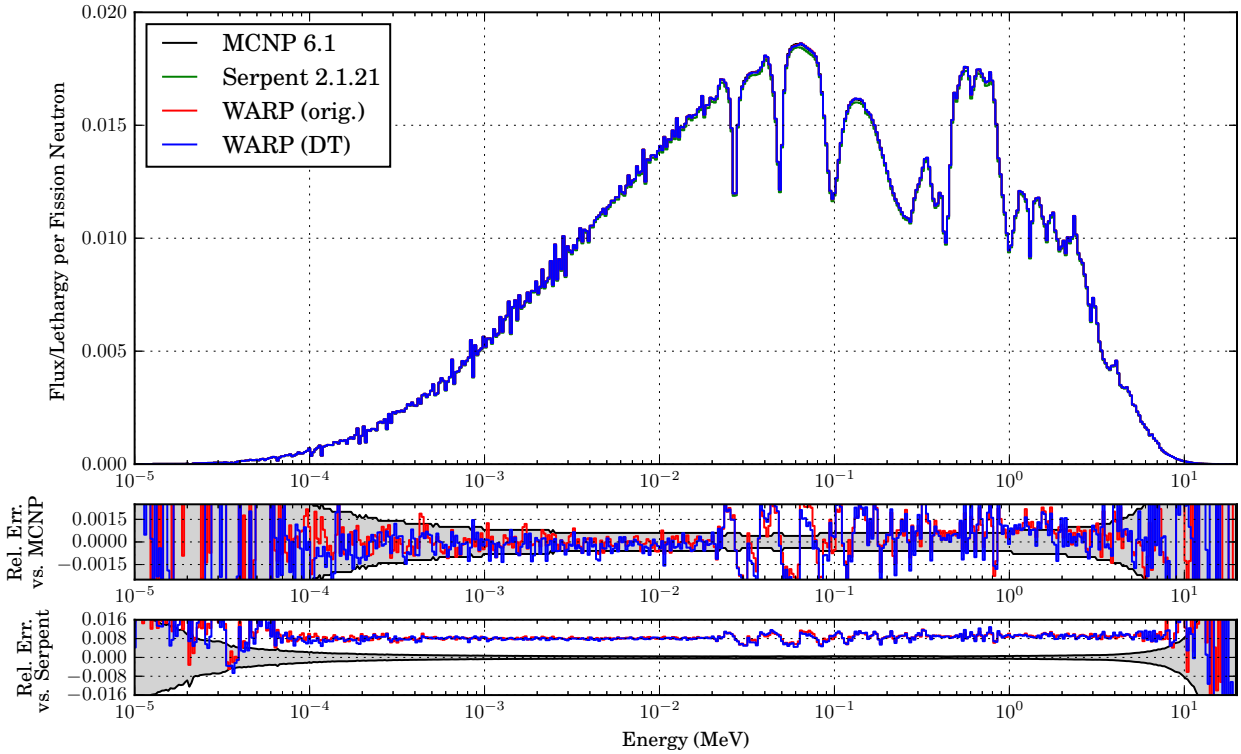


Fig. 5: Volume-averaged flux spectra inside the fuel pebble of the reflective FLiBe test case.

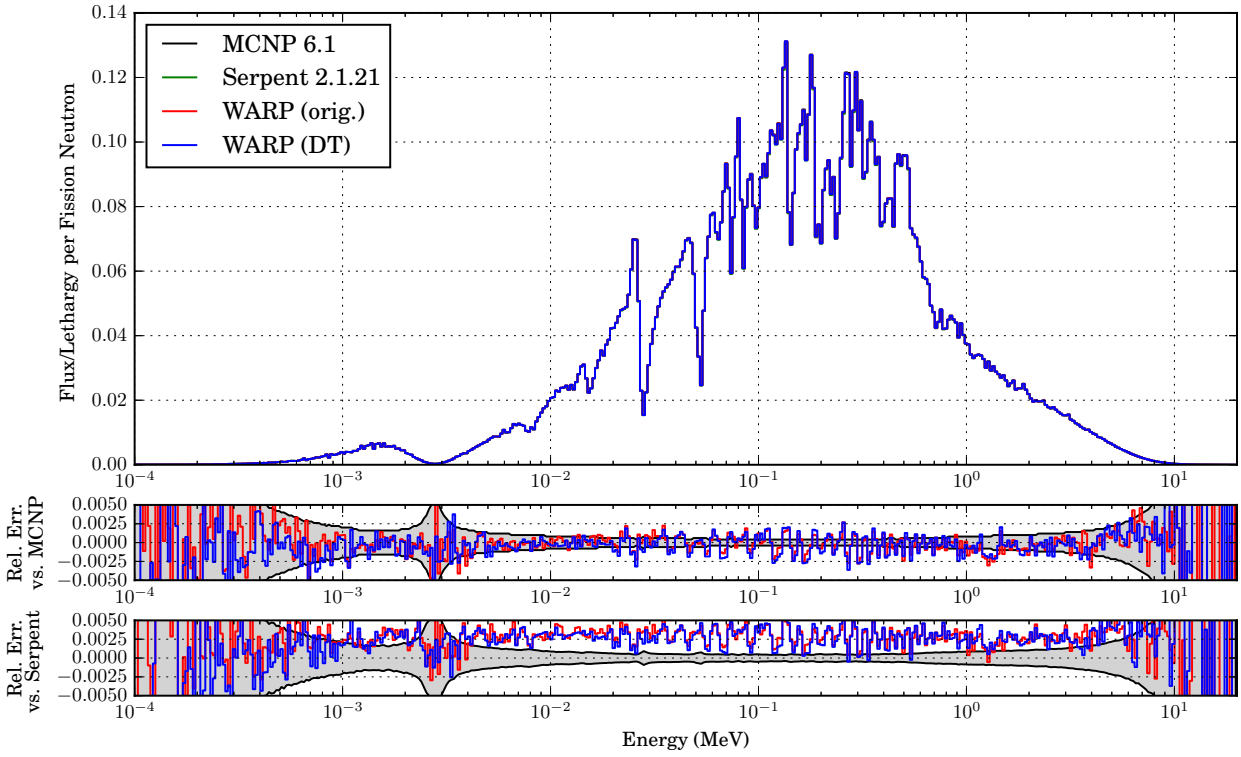


Fig. 6: Volume-averaged flux spectra inside the fuel pin of the steel-clad UO_2 pin in liquid sodium test case.

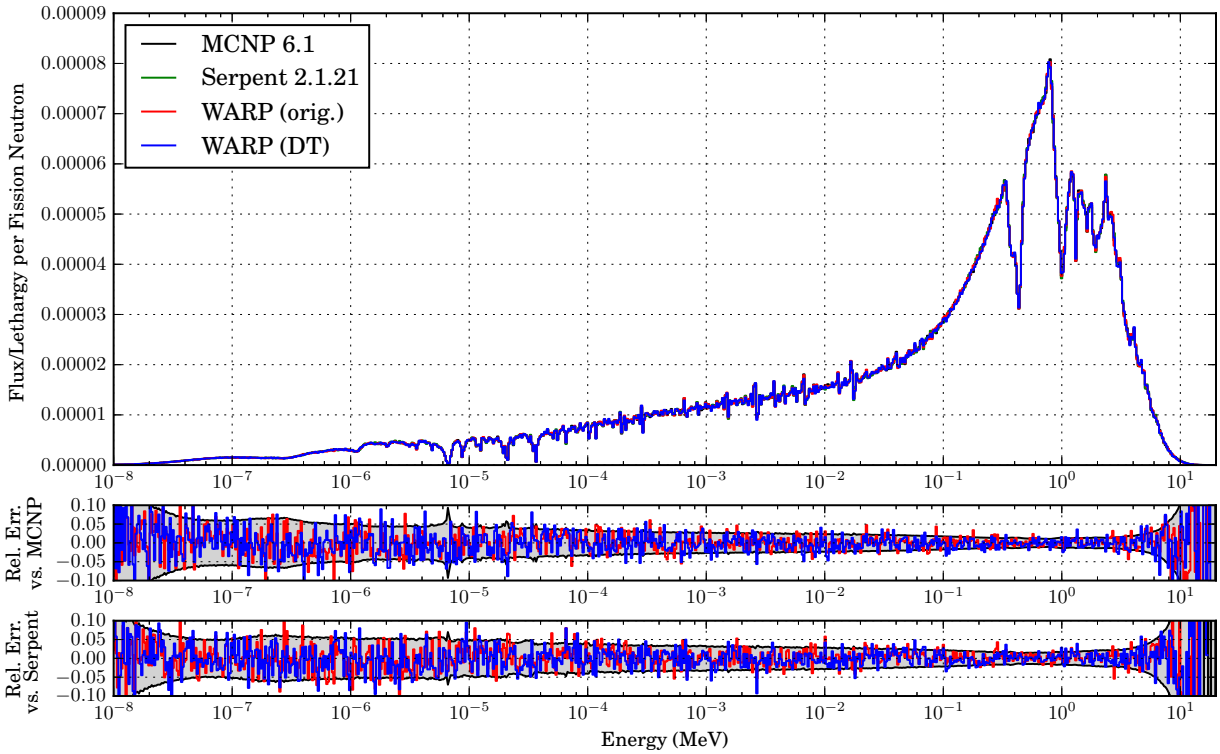


Fig. 7: Volume-averaged flux spectra inside the center fuel pin of the hexagonal assembly case.

longer runtimes, the figures-of-merit of the calculations produced are comparable to those of the other codes (seen in Table VIII). The delta-tracking version of WARP outperforms all other codes when comparing figures-of-merit for the Jezebel test case (the simplest configuration). For the remaining scenarios, however, the figure-of-merit from the delta-tracking version of WARP is either on the same order of magnitude as the original version of WARP or one order of magnitude worse. This can be generally explained by the combination of the delta-tracking version of WARP taking much longer to run than the original version, with the error in the values of k_{eff} being on the same order of magnitude for both versions of the code. The only exception to this is the hexagonal assembly case; in this case the delta-tracking version of WARP produces an error one order of magnitude smaller than that of the original version of the code, but the dramatic increase in runtime causes the figure-of-merit for the delta-tracking version of WARP to be much worse.

It is not surprising that the delta-tracking version of WARP is slower than the original version; the delta-tracking version has an additional OptiX trace compared to the original code, and ray tracing comprises a large portion of runtime [1]. Unfortunately, any time saved in not stopping neutron flights at boundary crossings is not enough to compensate for the additional OptiX trace calls. In addition to this extra OptiX trace, the delta-tracking version of WARP incurs a tail effect caused by resampling. In configurations like the pin cell and the hexagonal assembly, the last few neutrons in a given cycle may spend an excess amount of time undergoing virtual collisions and transport until an actual event occurs.

A question that may arise at this point is the notion of implementing some sort of threshold similar to that in Serpent's algorithm. Without a major code overhaul, this is not possible for the delta-tracking algorithm in WARP. WARP hinges on the OptiX framework to handle the geometry, materials, neutron locations, and boundary condition. The two physics routines are incompatible in WARP in that they each have OptiX handle the boundary condition in a fundamentally different way. Using ray tracing and the distance-to-collision calculation algorithm "turns on" the boundary condition only when a neutron is in the outermost cell and has a trajectory pointing outward, while delta-tracking has the boundary condition constantly "on" because neutrons can leak out from any cell (not just the outermost one). Because of this difference, a combination of the two methods (like that done in Serpent) cannot be done without altering how the code relies on OptiX. Unfortunately, such a substantial overhaul is not the highest development priority at this time.

VI. CONCLUSIONS

In this work, delta-tracking was implemented as a neutron tracking routine in a GPU-accelerated Monte Carlo neutron transport code and the results were compared for accuracy and speed with respect to the original version of the code as well as two production level Monte Carlo transport codes.

The delta-tracking version of WARP performs calculations that are sufficiently accurate when compared to the original version of the code as well as production-level codes. We see that, when comparing figures-of-merit, the delta-tracking version of WARP outperforms all other codes for the simplest geometry configuration. For more complex scenarios, the figures-of-merit of the calculations from the delta-tracking version of WARP are on the same order of magnitude of those of the original version of WARP at best and one order of magnitude lower at worst. Though the errors achieved by both versions of the code are comparable, the figures-of-merit of the delta-tracking version of WARP are brought down by the excessive runtime incurred. This overall result is not surprising; the delta-tracking routine requires more OptiX traces than the original code, and the ray tracing comprises a significant part of runtime.

This first exploration of using delta-tracking in Monte Carlo neutron transport on GPUs has shown that it may not be worthwhile for the current version of WARP. There are some scenarios in which it might be helpful, such as domain decomposition when a problem is broken into subsets of simpler systems. In the long run, moving away from the OptiX framework may change these conclusions. Delta-tracking on GPUs may prove promising for other Monte Carlo implementations.

VII. ACKNOWLEDGMENTS

This material is based upon work supported by the Department of Energy National Nuclear Security Administration through the Nuclear Science and Security Consortium under Award Number(s) DE-NA0003180 and/or DE-NA0000979 as well as upon work supported under an Integrated University Program Graduate Fellowship. This research used the Savio computational cluster resource provided by the Berkeley Research Computing program at the University of California, Berkeley (supported by the UC Berkeley Chancellor, Vice Chancellor of Research, and Office of the CIO).

VIII. DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or limited, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

REFERENCES

1. R. M. BERGMANN and J. L. VUJIĆ, "Algorithmic choices in WARP–A framework for continuous energy Monte Carlo neutron transport in general 3D geometries on GPUs," *Annals of Nuclear Energy*, **77**, 176–193 (2015).
2. T. GOORLEY, M. JAMES, T. BOOTH, F. BROWN, J. BULL, L. COX, J. DURKEE, J. ELSON, M. FENSIN, R. FORSTER, ET AL., "Initial MCNP6 release overview," *Nuclear Technology*, **180**, *3*, 298–315 (2012).
3. J. LEPPÄNEN, "Serpent–a continuous-energy Monte Carlo reactor physics burnup calculation code," *VTT Technical Research Centre of Finland*, **4** (2013).
4. T. SUTTON, F. BROWN, F. BISCHOFF, D. MACMILLAN, C. ELLIS, J. WARD, C. BALLINGER, D. KELLY, and L. SCHINDLER, "The Physical Models and Statistical Procedures Used in the RACER Monte Carlo Code," Tech. rep. (July 1999).
5. Y. MORIMOTO, H. MARUYAMA, K. ISHII, and M. AOYAMA, "Neutronic Analysis Code for Fuel Assembly Using a Vectorized Monte Carlo Method," *Nuclear Science and Engineering*, **358**, 351–358 (1989).
6. J. LEPPÄNEN, "Performance of Woodcock delta-tracking in lattice physics applications using the Serpent Monte Carlo reactor physics burnup calculation code," *Annals of Nuclear Energy*, **37**, 715–722 (2010).
7. K. L. ROWLAND, "Delta-tracking in the GPU-accelerated WARP Monte Carlo Neutron Transport Code," Master's report; University of California, Berkeley; number UCB-NE-5154 (2015), https://github.com/kellyrowland/masters.
8. "Savio System Overview," http://research-it.berkeley.edu/services/high-performance-computing/system-overview.
9. R. M. BERGMANN, K. L. ROWLAND, N. RADNOVIC, R. N. SLAYBAUGH, and J. L. VUJIĆ, "Performance and Accuracy of Criticality Calculations Performed Using WARP - A Framework for Continuous Energy Monte Carlo Neutron Transport in General 3D Geometries on GPUs," *Annals of Nuclear Energy* (Pre-press).
10. O. N. E. AGENCY, *International Handbook of Evaluated Criticality Safety Benchmark Experiments*, Nuclear Energy Agency, OECD (1995).