TOWARDS CAD-BASED GEOMETRY MODELLING WITH THE RANDOM RAY METHOD

Patrick C. Shriwise[1], John R. Tramm[1], Andrew Davis[2], Paul K. Romano[1]

[1]Argonne National Laboratory
9700 S. Cass Avenue Lemont, IL, USA

[2]United Kingdom Atomic Energy Agency
Culham Science Centre, Abingdon OX14 3DB, United Kingdom

{pshriwise,jtramm,promano}@anl.gov, andrew.davis@ukaea.uk

ABSTRACT

The Advanced Random Ray Code (ARRC) is a high performance computing application capable of high-fidelity simulations of full core nuclear reactor models. ARRC leverages a recently developed stochastic method for neutron transport, known as The Random Ray Method (TRRM), which offers a variety of computational and numerical advantages as compared to existing methods. In particular, TRRM has been shown to be capable of efficient simulation of explicit three dimensional geometry representations without assumptions about axial homogeneity. To date, ARRC has utilized Constructive Solid Geometry (CSG) combined with a nested lattice geometry which works well for typical pressurized water reactors, but is not performant for the general case featuring arbitrary geometries.

To facilitate simulation of arbitrarily complex geometries in ARRC efficiently, we propose performing transport directly on Computer-Aided Design (CAD) models of the geometry. In this study, we utilize the Direct-Accelerated Geometry Monte Carlo (DAGMC) toolkit which tracks particles on tessellated CAD geometries using a bounding volume hierarchy to accelerate the process, as a replacement for ARRC's current lattice-based accelerations. Additionally, we present a method for automatically subdividing the large CAD regions in the DAGMC model into smaller mesh cells required by random ray to achieve high accuracy. We test the new DAGMC geometry implementation in ARRC on several test problems, including a 3D pincells, 3D assemblies, and an axial section of the Advanced Test Reactor. We show that DAGMC allows for simulation of complex geometries in ARRC that would otherwise not be possible using the traditional approach while maintaining solution accuracy.

1. Introduction

The Advanced Random Ray Code (ARRC) is a recently developed neutron transport code that is capable of simulating nuclear reactors in high-fidelity. ARRC is built upon the Random Ray Method (TRRM) of neutral particle transport, which is a stochastic method that shares many

fundamentals with the Method of Characteristics (MOC), though with a continuous treatment of angle and space rather than a discretized one as with traditional deterministic MOC. Both TRRM and ARRC have been documented in a number of publications already [1,2,3,4], which include more information on the theory behind TRRM.

In the present study we introduce new geometry modeling capabilities into ARRC with the following goals:

1. Application of computer-aided design (CAD) models in ARRC as a user-friendly and portable geometry representation as opposed to ARRC's application-specific constructive solid geometry (CSG) model
2. Efficient modeling of arbitrarily complex geometries

The first goal above is highly valuable, as it can allow for problem specifications to be more easily produced and to potentially be portable between simulation applications. ARRC's traditional CSG input scheme relies on a lattice based hierarchy for efficient performance to be achieved. While this means that ARRC's approach to geometry is not efficient for the general case, this method is at least a good choice for pressurized water reactor (PWR) geometries that feature reactors composed of a lattice of fuel assemblies, which themselves are composed of a lattice of fuel pins. For such geometries, a user can easily define a handful of pincells and guide tube cell geometries, combine them into lattices of various composition, and then combine those fuel assembly lattices into a full reactor. These nested lattices can then be leveraged as an acceleration structure that greatly speeds up the location of rays in the geometry and limits the number of surfaces that must be checked for intersection as rays traverse the simulation geometry.

The second goal above is even more important, however. ARRC's nested lattice approach does not offer efficient performance for complex or irregular geometries. For instance, the Advanced Test Reactor (ATR) [5] geometry (shown in Figure 2a) features serpentine fuel elements and a high number of asymmetrical features that preclude the use of any sort of nested lattice acceleration. Thus, to attempt to model the geometry using ARRC's existing geometry capabilities would require that each CSG cell be individually defined by the user and that those cells then be subdivided into appropriately sized Flat Source Regions (FSRs). As the code exists now, without nested lattice acceleration at runtime as rays traverse the geometry all surfaces in the model would need to be ray-traced to determine the nearest surface along the ray's trajectory and in turn what FSR the ray will enter next. Overall, this scheme would be prohibitively expensive both from a model preparation perspective as well as a computational efficiency perspective for all but the smallest of problems.

To accomplish the two goals stated above we utilized the Direct-Accelerated Monte Carlo (DAGMC) [6] toolkit to represent CAD model surfaces as tessellations of triangles in ARRC and to build a bounding volume hierarchy (BVH) acceleration data structure for performant ray tracing as an alternative option to ARRC's current lattice-based accelerations. DAGMC has been implemented in a variety of Monte Carlo codes; including MCNP [7], Geant4 [8], OpenMC [9], and Shift [10]; to great effect, enabling geometric representations otherwise unavailable in terms of higher order surface representation and/or the human time necessary to describe the model in the code's native geometric representation (typically a form of text-based CSG).

Model	Implementation	Rays/Iteration	k-eff	Std. Dev. (pcm)	Runtime (s)
CASL Pincell	ARRC	300	1.18298	10.301	81.4
CASL Pincell	DAG-ARRC	300	1.18305	10.488	1621.7
CASL 3×3 Assembly	ARRC	500	1.17557	20.5	399
CASL 3×3 Assembly	DAG-ARRC	500	1.17544	19.8	5600

Table 1: Comparison of the initial application of DAGMC geometry (with each DAGMC volume representing a FSR in ARRC) to ARRC's native CSG geometry using an Intel i7-8650U CPU with one thread. The simulated pincell is based on the CASL VERA Core Physics Benchmark Problem number 1A [11]. The 3x3 assembly problem includes 20 cm axial reflector regions above and below the fuel cells.

In the following sections, we will give some details on our implementation of DAGMC in ARRC and present a series of studies on example problems to determine the performance and accuracy of the new geometry treatment in ARRC.

2. Initial Implementation

In our initial implementation, we began with the simplest possible approach where ray tracing was performed directly on the DAGMC geometry to determine distances through volumes and reflect rays at domain boundaries. In this model, each DAGMC volume corresponded to a single FSR in ARRC. We then tested this approach on a 3D PWR pincell simulation problem featuring explicit fuel, gap, clad, and moderator regions, as well as a small 3D PWR assembly test case. We then compared simulation results in ARRC using both the DAGMC geometry model as well as ARRC's traditional native CSG representation. The pincell and assembly meshes used as input to DAGMC featured the same FSR subdivisions used in ARRC's native representation, meaning that both methods used the same number of FSRs and with the same approximate shapes. As ARRC has already been validated against Monte Carlo reference solutions extensively [2], we limit the scope of our comparison to differences between ARRC's native CSG and the DAGMC model.

The simulation results in Table 1 using DAGMC geometries agree closely with those using ARRC's native geometry, demonstrating that DAGMC's tessellated surfaces do not introduce additional error into as compared to the native CSG model in ARRC. While this was encouraging, the simulations took up to 20 times longer than those using native ARRC CSG. To match the exact models used in ARRC, the radial and axial sections of the pincell and assembly models were generated manually in Trelis [12]. While creating these sections manually was useful to verify the implementation, manually sectioning DAGMC volumes into FSRs is not realistic for models of higher complexity. To incorporate more complex models in to ARRC with reduced manual effort, an alternative implementation was explored.

3. A Generalized Approach to CAD-Based Models in ARRC

Having verified an initial implementation of DAGMC in ARRC, an 1 cm axial section at the midplane of an existing Advanced Test Reactor (ATR) CAD model (see Figure 2a) was selected as a target for high-fidelity modeling of a complex reactor geometry, one that would be highly labor intensive to represent in CSG. Little work was needed to convert an existing DAGMC ATR model from previous Monte Carlo studies for use with our code. As ARRC currently requires that users input their own multi-group cross sections, cross section data for this model was generated with the OpenMC Monte Carlo code using the 70 group CASMO [13] format. OpenMC is also capable of representing CAD models using DAGMC; thus the exact same tessellated geometry was used as in the ARRC simulations. A reference solution was also generated using OpenMC with resulting multi-group cross sections and isotropic scattering. No transport correction was used, as generation of optimal multi-group cross sections was not the focus of this work, but could be achieved for this problem by applying the P_0 transport correction factor [14]. Additionally, deterministic methods for multi-group cross section generation could be used, such as FEDS [15]. While the geometry is a single, 1 cm axial section of the reactor modeled in a 2D fashion with reflecting boundary conditions on the top and bottom border surfaces, all ray tracing and geometry operations were performed in 3D.

Flat source random ray, like flat source MOC, requires that FSRs generally be significantly smaller than the mean free path of a neutron to obtain accurate results. Historically, in its use with Monte Carlo codes, DAGMC volumes represent entire components in reactor geometries, which are far larger than FSRs necessary for high-fidelity representations of the flux across those components. In the ATR model, there are many volumes whose average chord length can be many tens of cm. In Table 2, the first entry represents a simulation equivalent to using our initial implementation where FSRs are defined by the CAD mesh regions, which results in nearly 10,000 pcm error in the eigenvalue, which is an unacceptably large error even for a low-order model. Thus, to achieve better accuracy, it is necessary to subdivide large regions in the model into sufficiently small FSRs. There are a number of approaches we considered:

1. Have the user subdivide their mesh into smaller FSRs manually. For a large problem like the ATR, this would require a prohibitive amount of manual user work, and high a level of expertise as the user would need to have a sense of where the flux gradients are highest, so as to know where FSR subdivision is needed.
2. Automate the subdivision of CAD models before tessellation based on a Cartesian mesh, though this may be difficult for larger models due to practical limitations of the CAD software. For instance, very small FSRs that result from automated subdivision may be problematic for CAD operations.
3. Have ARRC overlay a virtual Cartesian mesh automatically and manage ray tracing through both the Cartesian mesh and the user input DAGMC geometry in a hybrid manner. This requires additional code complexity, but removes the burden on the user.

For our target problem, we found that option (1) above was not practical as it would potentially require hundreds of thousands of manual CAD subdivision operations to be performed by the user. Option (2) was more feasible, but we ran into a number of issues that slowed this process

down. Model manipulation became impractically slow due to limitations with the CAD software. This left us with option (3) above, which we feel will make it easiest to use DAGMC models in ARRC while also adding some performance benefits.

As a next step in our analysis, we implemented the hybrid ARRC Cartesian mesh and DAGMC model scheme shown in Algorithm 1. In this scheme, we add a secondary inner loop into the normal ray transport sweep process. In the outer loop, a ray will trace against the DAGMC model to determine the next intersection distance with that model. The inner loop will perform traversal of the Cartesian mesh in ARRC until the ray reaches its intersection with the DAGMC volume. For instance, in a large DAGMC volume, we may find that our ray would normally travel 15 cm before reaching the volume boundary. If we have a mesh overlaid with 1 cm cells, this means that in our inner loop we would then perform many steps moving the ray through the Cartesian grid FSRs (about 1 cm at a time) until the ray has travelled the 15 cm distance to the point of the DAGMC model intersection, at which point another DAGMC ray-tracing operation in the neighboring volume would be performed and the process repeated. An added benefit of this hybrid method is that traversal of the Cartesian mesh is much faster than DAGMC ray queries. This is another advantage of this approach over options (1) and (2). While DAGMC volumes would be appropriately sized for use as FSRs in those cases, simulation would still require ray traces on a triangle mesh. Point location and ray traversal costs on the virtual Cartesian mesh are negligible by comparison.

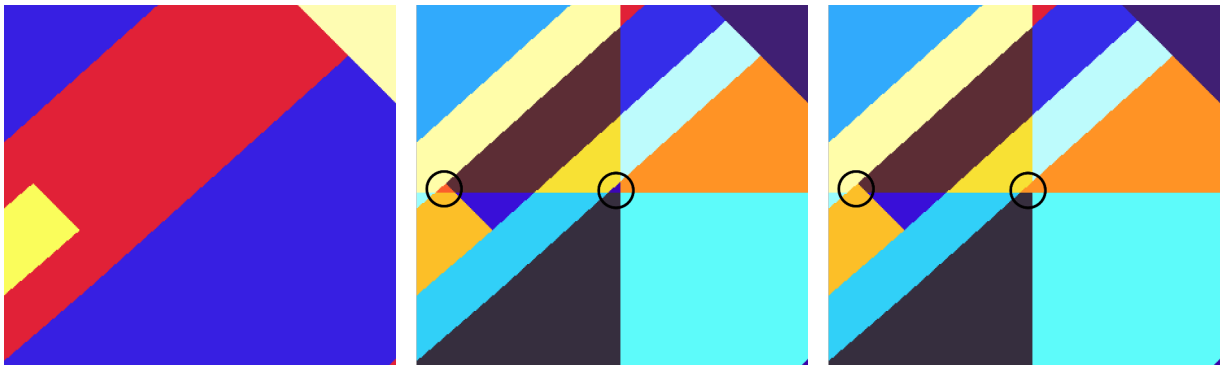
Algorithm 1 ARRC Transport Sweep with DAGMC and Cartesian Subgrid

for each ray **do**
  while ray is alive **do** ▷ Loop for DAGMC Surface Crossings
    $C =$ find Cartesian cell that ray is in
    $V =$ find DAGMC volume that ray is in
    $I =$ lookup FSR ID based on C and V
    $D_{DAGMC} =$ get next geometric intersection distance from DAGMC model
    $D_{Cartesian} =$ get next geometric intersection distance from Cartesian mesh
    while $D_{Cartesian} < D_{DAGMC}$ **do** ▷ Loop for Cartesian Mesh Surface Crossings
      Attenuate flux through FSR I for all energy groups
      Move ray forward $D_{Cartesian}$
      $D_{DAGMC} = D_{DAGMC} - D_{Cartesian}$
      $D_{Cartesian} =$ get next geometric intersection distance from Cartesian mesh
      $C =$ find Cartesian cell that ray is in
      $I =$ lookup FSR ID based on C and V
    end while
    Move ray into next DAGMC volume
  end while
end for

There are several important implementation details to note. The first detail is that this hybrid ray-tracing process requires a brief "mapping" phase before the simulation begins. In this phase, rays are launched and tracked through the hybrid geometry, and a map is built that indicates which DAGMC volumes are present in each Cartesian grid cell. The map is constructed by randomly firing rays through the geometry until new entries are no longer being added to the map. At this point, the map is considered complete and each unique combination of Cartesian grid cell

and DAGMC volume can be assigned a FSR ID, which allows for all of the standard bookkeeping operations in ARRC to be used without modification (e.g., construction of the neutron source, update of eigenvalue each iteration, etc). This mapping is then used in the simulation phase to associate transport data with the correct FSR given a ray's location (lines 5 and 14 of Algorithm 1).

One challenge we encountered is that the naive overlay of a Cartesian mesh tends to create a large quantity of vanishingly small FSRs, as shown in Figure 1b. This effect worsens as the resolution of the mesh increases. The problem with having very small FSRs in the model is that, as they are not hit by rays very often, they have high variances relative to larger FSRs in those regions. We observed that having many small, high-variance FSRs can cause the simulation to become unstable in some cases. The "brute-force" solution to this problem is to increase the number of rays run per power iteration, but we found that this requires an impractically high number of rays to stabilize the solution.



(a) Materials **(b) FSRs generated after Cartesian mesh overlay** **(c) FSRs after merging**

Figure 1: FSR merging process in a fuel plate region of the ATR model.

To remedy this issue we implemented a routine that merges very small FSRs together with larger, neighboring FSR of the same material type, as shown in Figure 1c. This routine utilizes the stochastic FSR volume, hit rate, and centroid data collected during the mapping initialization phase of the problem. For each FSR below a fixed volume and hit rate threshold, each neighboring Cartesian grid cell is checked to determine if it contains a region with the same DAGMC volume ID. Then, the neighbor with the closest centroid is chosen for merging. In some cases, this process may result in a non-contiguous FSR being created, but we were not able to find any examples of this happening, and in TRRM there is no reason FSRs must be contiguous. In this analysis, we tested various minimum volume threshold values for merging, and arrived empirically at minimum 2D area of 10^{-3} cm^2, though we note that performance and accuracy were not very sensitive to this choice – a wide range of minimum values produced similar results.

4. Results

In Table 2, we present the results of a mesh refinement study on the ATR problem, where the radial dimensions of ARRC's Cartesian mesh are refined with one grid cell in the axial dimension.

In theory, as the FSRs decrease in size, the ARRC simulation will approach an eigenvalue that is identical to the multi-group reference solution that was generated with OpenMC. We found that overlaying the Cartesian mesh was extremely effective, reducing the error compared to the OpenMC simulation to ∼300 pcm at the 200×200×1 Cartesian mesh resolution level (corresponding to Cartesian mesh cells that were 1 cm×1 cm in radial size). In theory, continual refinement of the Cartesian mesh should eventually produce an eigenvalue that matches the reference solution provided by OpenMC. Our simulation did not behave as expected — instead converging to about 350 pcm above the reference eigenvalue. Future work will be aimed at eliminating or explaining this unexpected bias. We also plan on quantifying the power distribution error as measured by fuel bundle rates in future work, but the plots of thermal and fast flux (shown in Figure 2) appear to be well converged and the relatively good agreement of the ARRC and reference eigenvalues at a preliminary stage is a highly encouraging result. It is worth noting that in this study we applied recent work leveraging the Embree ray tracing kernel within DAGMC [16,17,18] to achieve the performance numbers shown below. In terms of time per integration during transport, ARRC runs at a speed of 0.256 ns/integration [19] for a comparably sized 3D assembly problem when using its traditional CSG geometry on the same test system – making the DAGMC version only approximately 50% slower by this metric.

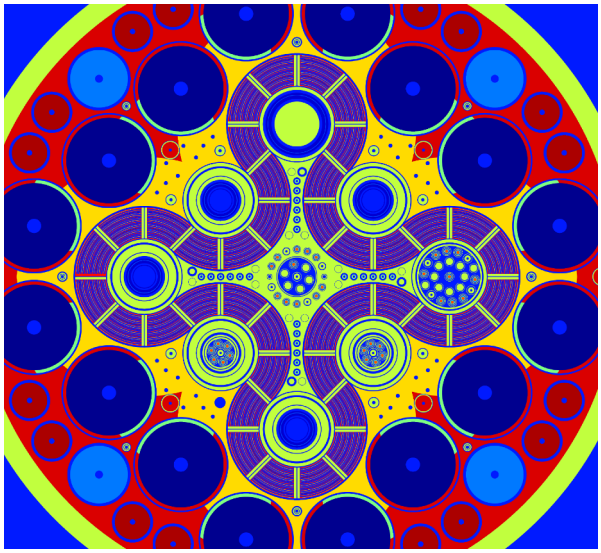
Mesh Resolution (X×Y×Z)	# of FSRs	k-eff	Std. Dev. (pcm)	Error vs. Reference (pcm)	Time per Integration (ns)	Simulation Time (s)
1×1×1	5,609	1.02626	52.6	9,899	12.92	745.2
10×10×1	7,236	1.07602	64.0	4,923	1.43	205.5
25×25×1	11,465	1.09693	55.4	2,832	1.33	347.1
50×50×1	18,293	1.11402	45.4	1,124	1.24	307.2
100×100×1	34,728	1.12416	38.5	109.4	1.04	365.3
200×200×1	79,763	1.12838	25.8	313.3	0.86	454.6
400×400×1	217,605	1.12859	19.5	333.0	0.62	832.4
800×800×1	682,114	1.12884	14.2	358.0	0.45	1368.8
Reference Solution Value: 1.12534 ± 16.7 pcm						

Table 2: Results of a the Cartesian mesh refinement study performed using a dual-socket Intel Xeon Platinum 8180M node with 56 total cores. Using 110 threads, all simulations used 17,000 rays per iteration with a distance of 250 cm per ray to meet the Source Averaged Relative Error (SARE) < 0.003 condition applied.

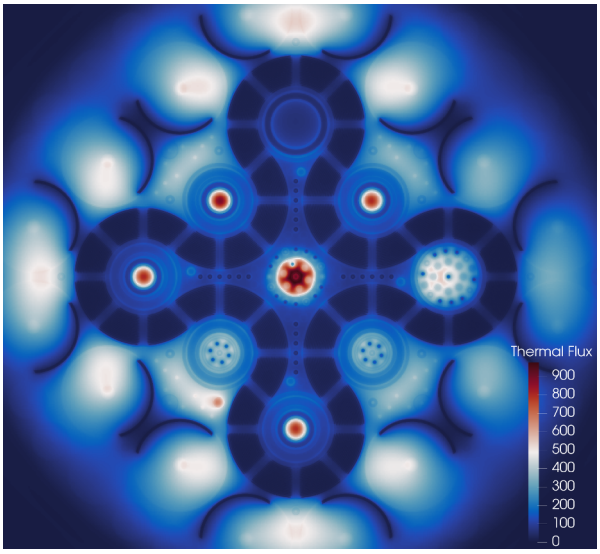
5. Conclusion

We have presented a new method for the use of CAD-based geometries in TRRM using the ray tracing capabilities of DAGMC in ARRC. DAGMC's geometry representation and particle tracking routines were shown to be reliable for use with ARRC. A method for representing arbitrarily complex 3D geometries in which a regular Cartesian mesh is superimposed over the DAGMC model was then applied to the ATR model with encouraging results in terms of the resulting eigenvalue in comparison with a Monte Carlo reference solution.

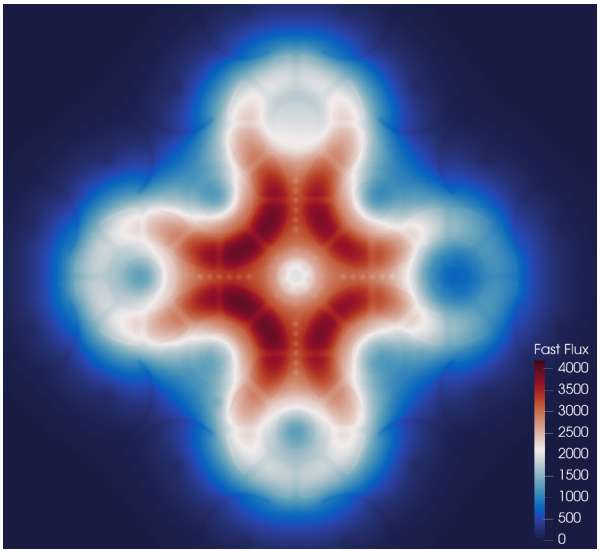
In future work, we plan to refine our method for using DAGMC in ARRC as rays are sometimes



(a) Material definition



(b) Thermal Flux (< 1 eV)



(c) Fast Flux (> 1 eV)

Figure 2: Material definition and flux distributions in the ATR using 800×800 Cartesian mesh resolution. These figures show a slightly zoomed in view towards the center of the simulation domain so as to better display the fine detail in the interior of the reactor.

lost in either the Cartesian or DAGMC geometries. New, and presumably small, FSRs are also sometimes not found in the mapping phase and encountered later, during the simulation phase. Methods for integrating these FSRs into other FSRs during simulation could be explored. We also plan to compare the per-bundle power distribution from ARRC for the full ATR model based on previous work [20].

ACKNOWLEDGEMENTS

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The material was based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory.

REFERENCES

[1] J. R. Tramm, K. S. Smith, B. Forget, and A. R. Siegel. "The Random Ray Method for neutral particle transport." *Journal of Computational Physics*, **volume 342**, pp. 229 – 252 (2017). URL http://www.sciencedirect.com/science/article/pii/S0021999117303170.

[2] J. R. Tramm, K. S. Smith, B. Forget, and A. R. Siegel. "ARRC: A random ray neutron transport code for nuclear reactor simulation." *Annals of Nuclear Energy*, **volume 112**(Supplement C), pp. 693–714 (2018). URL http://www.sciencedirect.com/science/article/pii/S0306454917303444.

[3] J. Tramm, B. Forget, and K. Smith. "Early experience in full core reactor simulation with The Random Ray Method." In *PHYSOR 2018: Reactor Physics Paving The Way Towards More Efficient Systems*, pp. 486–497. Cancun (2018).

[4] J. R. Tramm. "Development of The Random Ray Method of neutral particle transport for high-fidelity nuclear reactor simulation." Ph.D. thesis, Massachusetts Institute of Technology, Department of Nuclear Science and Engineering (2018). URL http://hdl.handle.net/1721.1/119038.

[5] C. J. Stanley and F. M. Marshall. "Advanced Test Reactor – A national scientific user facility." In *16th International Conference on Nuclear Engineering (ICONE16)* (2008). URL https://inldigitallibrary.inl.gov/sites/sti/sti/3991950.pdf.

[6] T. J. Tautges, P. P. H. Wilson, J. Kraftcheck, B. M. Smith, and D. L. Henderson. "Acceleration Techniques for Direct Use of CAD-Based Geometries in Monte Carlo Radiation Transport." In *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)*. American Nuclear Society, Saratoga Springs, NY (2009).

[7] X.-. M. C. Team. "MCNP — A General Monte Carlo N-Particle Transport Code, Version 5." Technical Report LA-UR-03-1987, Los Alamos National Laboratory (2003). URL https://laws.lanl.gov/vhosts/mcnp.lanl.gov/pdf_files/la-ur-03-1987.pdf.

[8] S. Agostinelli et al. "Geant4—a simulation toolkit." *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, **volume 506**(3), pp. 250 – 303 (2003). URL http://www.sciencedirect.com/science/article/pii/S0168900203013688.

[9] P. K. Romano, N. E. Horelik, B. R. Herman, A. G. Nelson, and B. Forget. "OpenMC: A state-of-the-art Monte Carlo code for research and development." *Ann Nucl Energy*, **volume 82**, pp. 90–97 (2015). https://doi.org/10.1016/j.anucene.2014.07.048.

[10] S. P. Hamilton and T. M. Evans. "Continuous-energy Monte Carlo neutron transport on GPUs in the Shift code." *Annals of Nuclear Energy*, **volume 128**, pp. 236–247 (2019). URL http://www.sciencedirect.com/science/article/pii/S0306454919300167.

[11] A. T. Godfrey. "VERA core physics benchmark progression problem specifications, revision 4." Technical Report CASL-U-2012-0131-004, Consortium for Advanced Simulation of Light Water Reactors (CASL) (2014). URL https://www.casl.gov/sites/default/files/docs/CASL-U-2012-0131-004.pdf.

[12] csimsoft. "Trelis." URL http://csimsoft.com.

[13] J. Rhodes, K. Smith, and D. Lee. "CASMO-5 development and applications." In *ANS Topical Meeting on Reactor Physics (PHYSOR 2006)*, January 2006 (2006).

[14] B. R. Herman, B. Forget, K. Smith, and B. N. Aviles. "Improved diffusion coefficients generated from Monte Carlo codes." In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013)*, pp. 2947–2961 (2013).

[15] A. T. Till. "Finite Elements with Discontiguous Support for Energy Discretization in Particle Transport." Ph.D. thesis, Texas A&M University, College Station, TX (2015). URL https://oaktrust.library.tamu.edu/handle/1969.1/156427.

[16] I. Wald, S. Woop, C. Benthin, G. S. Johnson, and M. Ernst. "Embree: A Kernel Framework for Efficient CPU Ray Tracing." *ACM Trans Graph*, **volume 33**(4), pp. 143:1–143:8 (2014). URL http://doi.acm.org/10.1145/2601097.2601199.

[17] P. Shriwise, A. Davis, and P. P. Wilson. "Leveraging Intel's Embree Ray Tracing in the DAGMC Toolkit." In *Am. Nuc. Soc. Winter Meeting 2015*, volume 113. Washington, DC, USA (2015).

[18] P. Shriwise and P. P. Wilson. "Reduced Precision Ray Tracing Performance Enhancements in the DAGMC Toolkit." In *Am. Nuc. Soc. Winter Meeting 2018*, volume 119. Orlando, FL. USA (2018).

[19] J. R. Tramm, A. R. Siegel, and P. K. Romano. "An event-based algorithm for random ray neutral particle transport on GPUs." In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2019)*, pp. 2552–2563 (2019).

[20] J. Nielsen, D. Nigg, and A. LaPorta. "A fission matrix based validation protocol for computed power distributions in the advanced test reactor." *Nuclear Engineering and Design*, **volume 295** (2015).