TOWARDS CAD-BASED GEOMETRY MODELLING
WITH THE RANDOM RAY METHOD

Patrick C. Shriwise¹, John R. Tramm¹, Andrew Davis², Paul K. Romano¹

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

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

{pshriwise,jtramm,ppromano}@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.

In TRRM, rays are sampled stochastically in space and angle, and followed through the problem geometry domain using on-the-fly ray tracing and characteristic integration techniques to solve for the angular flux along each ray’s path. As such, ray tracing usually accounts for about half of total runtime for a typical simulation – meaning efficient implementation of geometry and ray tracing procedures in ARRC is critical for achieving high computational performance. For performant ray tracing, an acceleration structure is required to enhance geometric operations like point location and next cell intersections – typically reducing the algorithmic complexity of these operations. To date, ARRC has utilized constructive solid geometry (CSG) combined with a nested lattice geometry approach as its acceleration structure, which works well for typical pressurized water reactors (PWRs) that are composed of nested lattices of different fuel assemblies and pincells. While this approach works well for most PWR geometries, it is not performant for the general case featuring arbitrary geometries.

To facilitate simulation of arbitrarily complex geometries (e.g., research reactors such as the Advanced Test Reactor) in ARRC, 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. We test the new DAGMC geometry implementation in ARRC on a variety of common reactor geometries, including pin-cells and 3D fuel assemblies to assess its accuracy and speed.
as compared to the traditional lattice-based CSG approach. We find that the DAGMC approach is able to provide high accuracy modelling, though at the cost of increased overall simulation runtimes in the range of 3 to 20x (depending on physical problem size and the number of neutron energy groups used). While this is a significant slowdown, DAGMC allows for simulation of complex geometries that would otherwise not be possible using the traditional approach. We also outline several potential avenues for optimizing the DAGMC and ARRC coupling routines and apply one of those methods to simulate an axial section of an Advanced Test Reactor (ATR) CAD model.

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 pincell 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 to limit the number of surfaces that must be ray traced against 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 geometries as tessellated surfaces 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).

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 faceted DAGMC geometry to determine distances through volumes in the tessellated CAD geometry and to 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.

The simulation results in Table 1 using DAGMC geometries agree closely with those using ARRC’s native geometry, demonstrating that DAGMC’s CAD-based geometries are a viable for use in TRRM. 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.

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

Having verified an initial implementation of DAGMC in ARRC, an 4 cm axial section at \( Z = 0 \) cm in 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 ARRC. 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 4 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, there are many volumes whose average chord length can be many tens of cm. In Table 2, the first entry represents a simulation equivalent using our initial implementation where FSRs are defined by the CAD mesh regions, which results in nearly 10,000 pcm error in 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 lot of 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 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 the easiest to use while also having
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 had 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.

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, leading to high variances in those regions. We observed that having

Proceedings of the PHYSOR 2020, Cambridge, United Kingdom
Algorithm 1 ARRC Transport Sweep with DAGMC and Cartesian Subgrid

1: for each ray do
2:     while ray is alive do ▷ Loop for DAGMC Surface Crossings
3:         \( C \) = find Cartesian cell that ray is in
4:         \( V \) = find DAGMC volume that ray is in
5:         \( I = \text{lookup FSR ID based on } C \text{ and } V \)
6:         \( D_{\text{DAGMC}} = \text{get next geometric intersection distance from DAGMC model} \)
7:         \( D_{\text{Cartesian}} = \text{get next geometric intersection distance from Cartesian mesh} \)
8:         while \( D_{\text{Cartesian}} < D_{\text{DAGMC}} \) do ▷ Loop for Cartesian Mesh Surface Crossings
9:             Attenuate flux through FSR \( I \) for all energy groups
10:            Move ray forward \( D_{\text{Cartesian}} \)
11:            \( D_{\text{DAGMC}} = D_{\text{DAGMC}} - D_{\text{Cartesian}} \)
12:            \( D_{\text{Cartesian}} = \text{get next geometric intersection distance from Cartesian mesh} \)
13:            \( C = \text{find Cartesian cell that ray is in} \)
14:            \( I = \text{lookup FSR ID based on } C \text{ and } V \)
15:        end while
16:        Move ray into next DAGMC volume
17:     end while
18: end for

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.

![Figure 1: FSR merging process in a fuel plate region of the ATR model.](image1)

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.

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 below 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). We 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 agreement of ARRC and reference eigenvalues at a relatively coarse mesh resolution 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, which are only 50% slower than the latest ARRC results using CSG geometry \[19\].

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

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’ve presented a generalized approach 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 the generated flux distributions and 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 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 plan to compare the per-bundle power distribution from ARRC for the full ATR model based on previous work [20].

Proceedings of the PHYSOR 2020, Cambridge, United Kingdom
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


Proceedings of the PHYSOR 2020, Cambridge, United Kingdom


The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan. [http://energy.gov/downloads/doe-public-access-plan.](http://energy.gov/downloads/doe-public-access-plan.)