International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)
Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009)

ACCELERATION TECHNIQUES FOR DIRECT USE OF CAD-BASED GEOMETRIES IN MONTE CARLO RADIATION TRANSPORT

Timothy J. Tautges
Mathematics and Computer Science Division
Argonne National Laboratory
425 ERB, 1500 Engineering Dr, Madison, WI 53706
tautges@mcs.anl.gov

Paul P.H. Wilson, Jason A. Kraftcheck, Brandon M. Smith, Douglass L. Henderson
Fusion Technology Institute, University of Wisconsin-Madison
419 ERB, 1500 Engineering Dr, Madison, WI 53706
wilsonp@engr.wisc.edu, kraftche@cae.wisc.edu, henderson@engr.wisc.edu,
bmsmith6@wisc.edu

ABSTRACT

The Direct Accelerated Geometry for Monte Carlo (DAGMC) software library makes use of a variety of acceleration techniques in order to accomplish efficient ray tracing directly on CAD-based solid models, without translation to the native Monte Carlo applications input language. These techniques include high-fidelity faceting combined with hierarchical trees of oriented bounding boxes for those facets. The resulting methods are both efficient and enable the modeling of very complex geometries including those with high-order surfaces. This work describes the acceleration techniques and provides examples of detailed simulations that demonstrate capabilities and performance.

Key Words: Monte Carlo, CAD, acceleration

1. INTRODUCTION

Monte Carlo radiation transport codes typically provide a native input language for specifying the spatial domain model or geometry in terms of various boolean combinations of geometric surface or volume primitives. This has long been recognized as a difficult and error-prone process, especially as model complexity increases. Two primary approaches have been taken to address this difficulty, both starting from a geometric model designed in a CAD system. The first approach translates CAD surfaces into their equivalent representation in the Monte Carlo code language, while the second approach uses the CAD model directly for Monte Carlo analysis. This paper reports on our experiences with this latter approach.

Several efforts have been reported on the translation approach described above. Tsige-Tamarat described the translation of Open.Cascade-based CAD models into MCNP input [1], while Wu describes the MCAM system for doing the same for ACIS-based CAD models [2]. Since the target Monte Carlo code cannot represent higher-order surfaces in its native language, both efforts require conversion of such surfaces to 2nd-order surfaces before translation. Also, the boolean-based construction process in MCNP typically results in much greater numbers of cells

in the translated models than in the original CAD models. Although in theory the translation-based process should result in lower runtimes for the Monte Carlo analysis (because evaluations in the native geometry representation in those codes is usually very fast), in practice this effect is diluted by the increased number of cells.

In a direct geometry approach, the CAD model is read and evaluated directly by the Monte Carlo code. Franke et. al observed that using the CAD engine itself for evaluations resulted in a 50-100x increase in overall execution time for the Monte Carlo analysis [3]. This paper reports efforts to accelerate these CAD-based evaluations, with the goal of the resulting Monte Carlo code being at least competitive with (e.g. having 2-5x) the native code in execution time. We also provide the capability in component form, in the Direct Accelerated Geometry for Monte Carlo (DAGMC) software library. DAGMC is designed to be integrated with any Monte Carlo radiation transport tool, replacing a handful of the built-in particle tracking methods with those specifically designed for the CAD-generated geometry. This approach provides a number of advantages including the ability to represent surfaces beyond the limitations of the native geometry and the introduction of a common domain representation that facilitates coupling to other physics simulations.

DAGMC is implemented as part of the Mesh-Oriented datABase (MOAB) library [4], and makes calls to the Common Geometry Module (CGM) library[5]. MOAB and CGM are developed as part of the ITAPS project [6], whose goal is to deliver interoperable interfaces, services, and tools for mesh and related simulation data.

The remainder of this paper is organized as follows. Section 2 describes the workflow for using this tool, that is, the steps required to perform a Monte Carlo analysis from start to finish. Other tools used in that workflow are also described. Section 3 describes specific geometric modeling and acceleration techniques critical to our direct geometry approach. Section 4 describes implementation features of DAGMC, which are useful for understanding how to integrate this capability into other Monte Carlo codes. Section 5 reports timing on several example problems and discusses performance issues of the code, and Section 6 concludes this paper.

2. WORKFLOW & RELATED TOOLS

Performing a Monte Carlo analysis requires the execution of a series of steps, using a number of tools besides the Monte Carlo radiation transport code itself. For the direct geometry approach reported here, these steps are described below, with particular reference to DAG-MCNP5, the result of integrating DAGMC with MCNP5 [7]

Construct geometric model in a CAD-based system: The spatial domain model sometimes starts with an existing CAD model and sometimes does not. In the latter case, it is still helpful to construct the model in a CAD system, which allows parametric editing and visual inspection of the results. We typically use the CUBIT mesh generation toolkit [8] for this task when model construction is required; we have also used models from the Catia CAD tool.

Annotate geometric model with material, tally, and other boundary condition information: In addition to the actual geometric description of the spatial domain, Monte Carlo codes also require material types, densities, and other non-geometric characteristics of the geometry. These

characteristics are most easily specified in the same interactive system in which the geometry was constructed. Hence, we use the CUBIT tool for this purpose as well. Material type and density are specified by constructing a group containing the relevant geometric volumes and embedding the material and density in a name assigned to the group. Tally and boundary condition groupings are specified in a similar manner. These data can all be overridden with data in the DAG-MCNP5 input file.

Assemble cross section data and other analysis parameters: Monte Carlo radiation transport codes like MCNP require construction of atomic density-averaged cross sections, based on material types and isotopic concentrations in a given model. These codes also require other types of input specific to the analysis, for example, the number of particle histories, and variance reduction parameters. In DAG-MCNP5, this input is put in the normal MCNP5 input file. Material types and densities can also be specified in this file, overriding those specified on the geometric model itself.

Read model & initialize search tree: One of the methods used by DAGMC to accelerate ray-tracing is to construct a tree-based decomposition of the facets defining CAD surfaces in the geometric model (this process is described in detail in the next section). This can be done either during the startup process of DAG-MCNP5 itself, or in a pre-processing step with the cgm2moab tool provided with DAGMC. The latter method eliminates the need to regenerate this structure every time the code is run on the same geometric model.

Run analysis: DAG-MCNP5 is run similarly to MCNP5, with command-line parameters specifying the location of the geometry and input files.

Analyze results, apply variance reduction techniques and repeat analysis: These steps are executed similarly to those used for the native MCNP5 code.

Several issues exist with the workflow described above:

- The translation of CAD models between CAD systems is a difficult process, and often introduces small features and even errors in the resulting model. DAGMC uses the Common Geometry Module (CGM) to read CAD data[5]. CGM has been ported to a variety of CAD engines, including ACIS, Catia, and Open.Cascade. We mitigate translation problems by generating the faceted representation directly from the native CAD system, where possible. Where this is not possible, translating through the STEP exchange format is also somewhat reliable.
- The annotation of materials and other data on the geometric model can be somewhat cumbersome. Because the CUBIT code cannot be modified directly for our purposes, we are forced to embed these data in names on geometric entities or groups. Embedding other Monte Carlo code input data in this way is impractical. We have started exploring the construction of a separate tool for this purpose [9].
- Constructing models for Monte Carlo analysis requires the addition of volumes representing void space and the outside boundary of the entire domain. This is discussed in the following section.

3. ACCELERATION TECHNIQUES

The overwhelming majority of execution time in a direct geometry approach to Monte Carlo analysis is the ray tracing used to track particles as they travel through and interact with the domain. Although solid modeling engines provide ray-tracing functions, previous efforts have shown this to be far too expensive for Monte Carlo analysis. DAGMC uses several methods for reducing overall runtime. Some of these methods involve the preparation of the geometric model, some the actual ray tracing process, and some the use of ray tracing in conjunction with radiation interactions with the domain. These accelerations are described below.

3.1 Geometric Model Preparation

Imprint & merge operations

By default, CAD tools generate manifold geometric models. In particular, if two distinct volumes meet at a surface, that surface will still be represented twice, once for each volume. This type of representation impacts the performance of Monte Carlo codes because it forces more than twice the number of ray tracing calls for particles crossing surfaces of the model, one to find where a particle leaves a volume and several (potentially many) others to find the next volume entered. If there is no void space between such volumes, the surface can be represented once and shared by each volume. With this type of representation, finding the volume entered after crossing a surface is a topological check, which is much more efficient than ray tracing.

Transforming a manifold solid model into a non-manifold model is accomplished using "imprinting" and "merging" operations. In the imprinting process, surfaces and edges in the geometric model which are partially coincident are split into wholly coincident and wholly disjoint parts. The merge operation takes two or more edges or two surfaces which are of similar topology and geometry, and removes all but one, substituting this one in other entities using the removed instances. So, for example, one of the two surfaces described above is removed, and one bounding volume is modified to use the retained surface. The imprint and merge process is used extensively in the mesh generation process as well [8]. The result of imprinting and merging is a series of contiguous volumes, separated from each other by surfaces each shared by two volumes, with remaining surfaces bounding a single volume, with void space on the other side. Note that this void space can be connected with the outer extent of the entire model, or it can be a void completely surrounded by non-void volumes.

Complement construction

Monte Carlo codes like MCNP5 require the explicit representation of the "universe" surrounding the non-void materials in a given analysis. In the trivial case of a single spherical volume, the universe is defined simply by the reverse of the surface bounding the sphere. For more complicated models, specification of the universe can be more difficult. Furthermore, in a direct geometry-based process, the universe volume must be constructed explicitly. Also, some models contain regions of void space not directly connected with the universe outside all non-void volumes. Monte Carlo codes typically require these types of voids to be separate from the universe volume and to be represented explicitly.

We handle void space in a model using one of two methods. In the first approach, the void is computed explicitly using boolean operations commonly found in CAD systems. Although straightforward, this can be a computationally expensive operation. A second approach,

developed more recently, is to represent the complement implicitly. In this case, the processing of ray tracing the results in the Monte Carlo code is modified to account for these implicit void volumes, some of which may be completely surrounded by non-void material.

3.2 Ray Tracing Accelerations

Surface Faceting

A routine capability of solid modeling tools and their underlying engines is the faceting or tessellation of surfaces, typically to enable efficient visualization. This faceted representation is typically viewed as an approximation to the real geometry, with a tolerance parameter providing guidance to the tessellation methods about the maximum allowable distance between a facet and the real surface at any point. If any facet is too distant from the actual surface, additional facets are generated to better resolve the surface.

DAGMC uses this faceted representation to enable the oriented bounding box acceleration discussed in the next section. When testing for the intersection of a ray with a surface, the method evaluates the facets for intersections first. The user has the option to use this intersection point as the actual surface crossing or as a first guess for finding the intersection point on the actual CAD surface. In the latter case, this operation may require a root-finding operation on a high-order geometric surface and the associated computational cost.

Oriented Bounding Box Accelerations

A surface can be completely surrounded with a box having right angles and parallel sides. A bounding box can be defined around a surface based on the maximum extents of that surface. A bounding box is useful because it provides a computationally simple and well defined geometry for testing ray-surface interactions. An axis-aligned bounding box has its surfaces oriented normal to the primary axes of the underlying coordinate system. In contrast, an oriented bounding box is rotated in space to achieve a box of minimum volume that still bounds the surface in question. The volume of an oriented bounding box is guaranteed to be no larger than that of an axis-aligned bounding box and is often much smaller.

Testing for intersection between a ray and the bounding box of an object is generally less expensive than testing for the intersection with that object itself. The least expensive test determines whether or not a ray intersects with a bounding sphere whose center is the same as that of the bounding box and which circumscribes the bounding box. This evaluation can be performed in the ray's frame of reference. Rays that do not intersect the sphere are guaranteed to miss the box and the surface; the others must be tested against each plane of the bounding box. For rays that intersect the bounding box, they are then tested against the object itself. The real advantage of these bounding box tests is amplified when combined with a hierarchical bounding box tree.

Hierarchical Bounding Box Tree

For each surface, S, represented by a set of facets, $\{F_S\}$, a bounding box, B_S, can be formed for that surface. To create a binary tree of bounding boxes, the set of facets, $\{F_S\}$, is divided geometrically into two subsets, $\{F_{S1}\}$ and $\{F_{S2}\}$, and a bounding box is formed for each subset of facets, B_{S1} and B_{S2}, and so on, until there is a single bounding box per facet. By maintaining a logical relationship between the parent bounding box, B_S, and the two child bounding boxes

formed from the subset of its facets, B_{S1} and B_{S2}, a binary tree for each surface can be created for testing ray-bounding box intersection.

When a ray fails to intersect with the bounding box, B_S, it is certain that it will not intersect any of the facets contained therein. When a ray does intersect B_S, that ray can then be tested against each of the child bounding boxes and is likely to hit only one of them. A binary search in the bounding box tree allows the determination of which facet the ray intersects, if any.

This binary search using bounding boxes is critical for allowing both an accurate representation of complex surfaces, *ie.* many facets, and an efficient method for determining ray-surface intersections. As an enhancement to the original implementation [10], bounding box trees for all the surfaces, $\{S\}$, that form a volume, V, can be merged into a larger binary tree for the whole volume. Testing intersection of a ray with an arbitrarily complex volume can then be constructed as an efficient traversal of a bounding box tree.

3.3 Physics-Based Acceleration: Collision Distance Limit

To further accelerate the traversal of the bounding box tree, the distance to the next collision is sampled prior to the geometric intersection. If the bounding box intersection is found to be at a distance beyond the point of the next collision, the binary search in that entire branch of the tree can cease since it is certain that the collision will occur prior to intersection with any facets in that branch of the tree.

4. IMPLEMENTATION AND FEATURES

The Direct Accelerated Geometry Monte Carlo (DAGMC) software library is a collection of C++ functions that provide geometry services typically required by Monte Carlo transport codes. Some of these services are implemented in DAGMC directly; DAGMC also calls MOAB, for more generic mesh-based functionality like tree construction and searching, and CGM, for importing and evaluating CAD-based models. A set of interface functions are typically needed to translate between the data structures of the Monte Carlo code and those of the DAGMC library. There are three primary geometry operations and a number of supporting functions. In a typical implementation, for each of the primary geometry functions the program flow is diverted to the interface function and the native capability is circumvented. The three primary functions are: 1) finding the intersection of a ray with the surfaces of a cell/volume, 2) determining whether a point is inside a cell/volume, and 3) determining the new cell/volume upon crossing a surface.

Because the geometry definition is provided by a CAD-based representation, the geometry portion of the standard Monte Carlo input file is not required. Modifications to the native input processing may be necessary to both avoid errors due to incomplete input files and initialize important variables that are normally defined by input file variables. Methods in DAGMC may be necessary to provide information to the native code for these initializations. This arrangement usually requires that the geometry be loaded prior to parsing the input file for the native application in order to provide relevant quantities and information to the initialization processes.

5. BENCHMARKING & APPLICATIONS

Although there has been much work done on tree-based searching and collision detection on faceted data, we did not find any related to ray tracing. We give timing data for several examples below. Timing data is also given for several real-world transport problems.

5.1 Ray Tracing Performance

Three models were used to measure ray tracing performance of the Hierarchical Oriented Bounding Box Tree code in DAGMC and MOAB; see Figure 1. The first model is a single volume bounded by a spherical surface; the second is a similar volume, but with slots cut out of the volume, each bounded by a pair of larger-radius spherical surfaces. The third model is a portion of the ITER model described in the next section; this corresponds to the complement of all material-filled volume, i.e. the void space between and surrounding ITER components. All models were faceted with varying tolerances, resulting in 10^4 to 10^7 facets. Timing measurements were made for rays starting at the origin and pointing in a random direction. Ray tracing times for these three problems, as a function of number of facets in the model, are given in Figure 2.

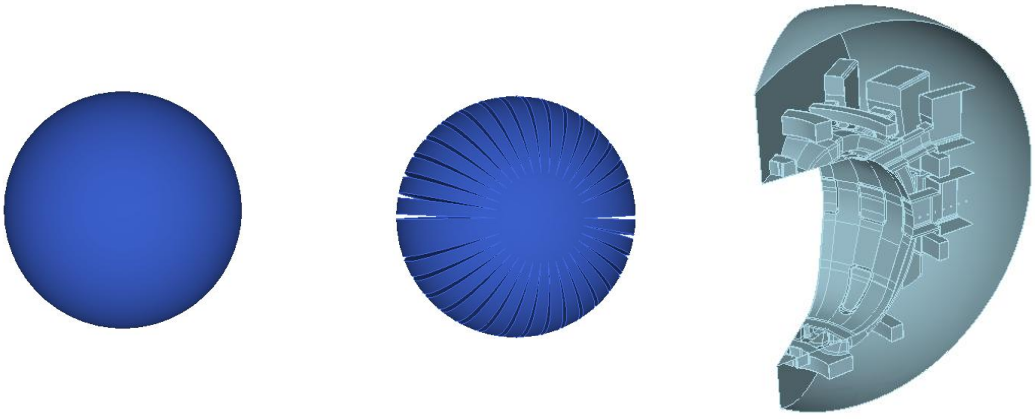


Figure 1: Models used for ray tracing performance test; sphere (left); "deathstar" (middle); ITER complement volume (right).

A number of interesting features are observed in these data. First, the ray tracing times appear to scale logarithmically for low and mid numbers of facets, but, for the deathstar and ITER examples, become linear for larger numbers of facets. Based on the fact that the sphere model does not show this behavior and also has only a single surface, we suspect this behavior is related somehow to the OBB tree structure for a volume. Namely, we require that each CAD surface is wholly represented by a single node in the tree, with nodes below that containing facets from only that surface. Nodes above that level contain facets from two or more surfaces. It appears that ray tracing at deeper levels of the OBB tree becomes dominant at larger numbers of facets. This issue will require further investigation.

Secondly, the memory data scales approximately linearly for all numbers of facets, with a final cost of approximately 100 bytes per facet.
Note

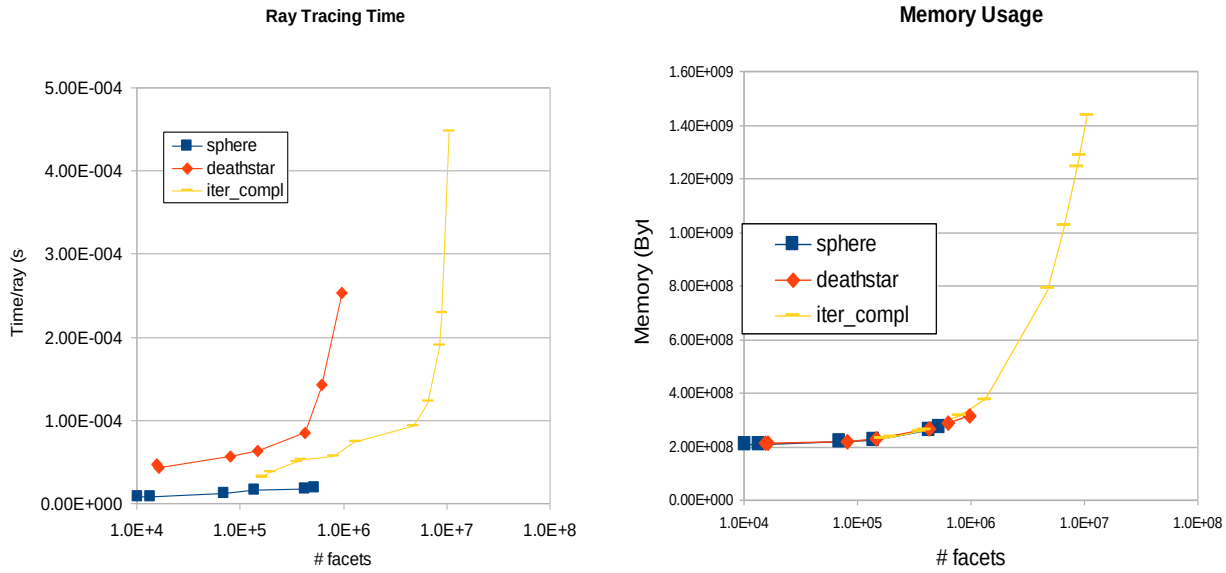
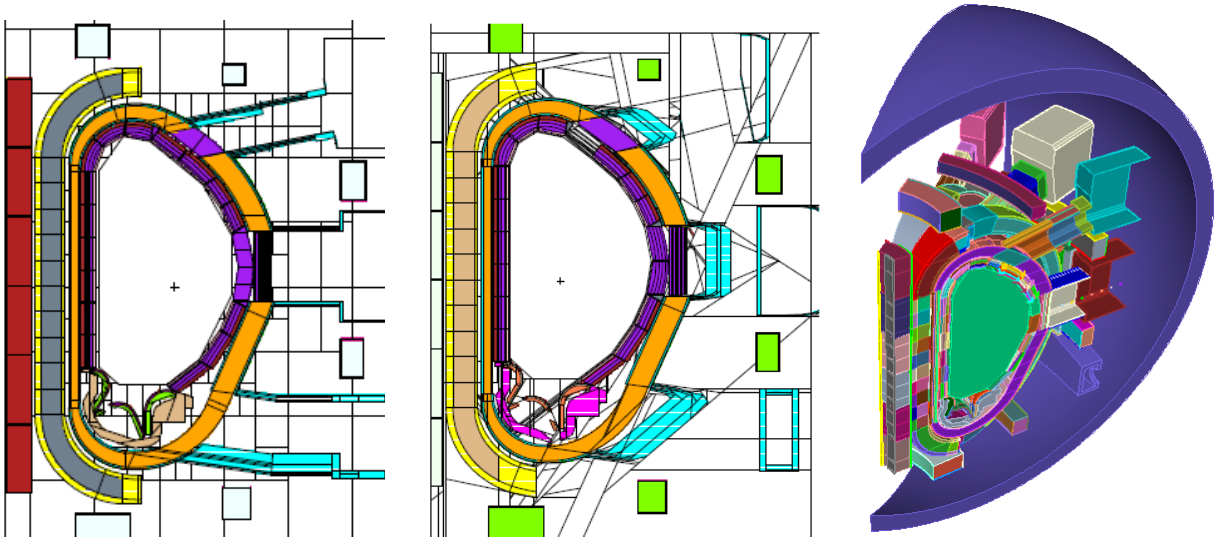


Figure 2: Performance information for standalone ray tracing in DAGMC; cpu time (left); memory (right).

5.2 ITER Benchmark Models

DAG-MCNP5 was compared to two translation-based approaches for performing transport on a complex model of the ITER experimental fusion device. The DAGMC-based approach used a CAD model which was close to the original model provided for the benchmark, with some modifications as described in section 3.1 above. The other approaches translated the CAD model into native MCNP5 geometry; each translation resulted in a different number of cells (volumes) and surfaces. The resulting models (Figure 3) were input to MCNP5 (or DAG-MCNP5). Physics results of the benchmark were reported elsewhere (ref); timing results for the three code runs are shown in Table I. These simulations include detailed material descriptions and a detailed source distribution, but no tallies or variance reduction. The McCad translation is slower than the MCAM translation, most likely because McCad creates more complex cell definitions with more surfaces. The void space in this DAGMC model (not shown in Figure 3) was explicitly generated and manually segmented.



(a) MCAM Translation (b) McCad Translation (c) DAGMC

Figure 3. MCNP5 and DAGMC geometries of the ITER Benchmark problem.

Table I. Geometry and Timing Data for ITER Benchmark Model Using a Variety of CAD-Based Tools

Model	Number of Volumes	Number of Surfaces	Relative Time
MCAM translation	4148	3192	1
McCad translation	6031	3800	1.63
DAGMC	802	9834	2.46

5.3 Applications to ITER Geometries

DAG-MCNP5 [7] is being used in production-scale calculations for ITER components. One example is the calculation of nuclear heating on a 3mm mesh in an ITER first wall and shield module . Around the ITER system are 18 rows of first wall and shield (FWS) modules that serve a variety of functions, including the removal of energy in the form of heat deposited by neutrons born in the fusion process. Module 13 is near the midplane of the outboard side of the Tokamak and sees nearly the largest current of 14.1 MeV neutrons. The U.S. is responsible for the design and analysis of this module, including nuclear, thermal, structural and electromagnetic analysis. To support this effort, we have performed combined neutron and photon transport calculations in the detailed geometry of Module 13 to determine the nuclear heating that is the source term for thermal analysis [11].

This analysis began with a simulation of the 40° ITER benchmark geometry using the detailed 3-D source distribution to accumulate an estimate of the neutron and photon current on the first wall of the module using the surface source write (SSW) capability of MCNP5. During this

simulation an approximation of the FWS module was used with 4 layers containing representative homogenizations of the materials. This surface source was then used for a detailed simulation of the FWS in a geometry (Figure 4) that included only Module 13 and an approximation of the vacuum vessel behind it, with reflecting boundaries on the top, bottom and sides. A typical MCNP5 mesh tally was used to determine the combined neutron and photon heating, with a 3mm mesh resolution over the approximately 137 x 85 x 43 cm (WxHxD) FWS module. This approach preserves the angular and energy distribution of radiation on the first walls while maximizing the efficiency of the detailed calculations.

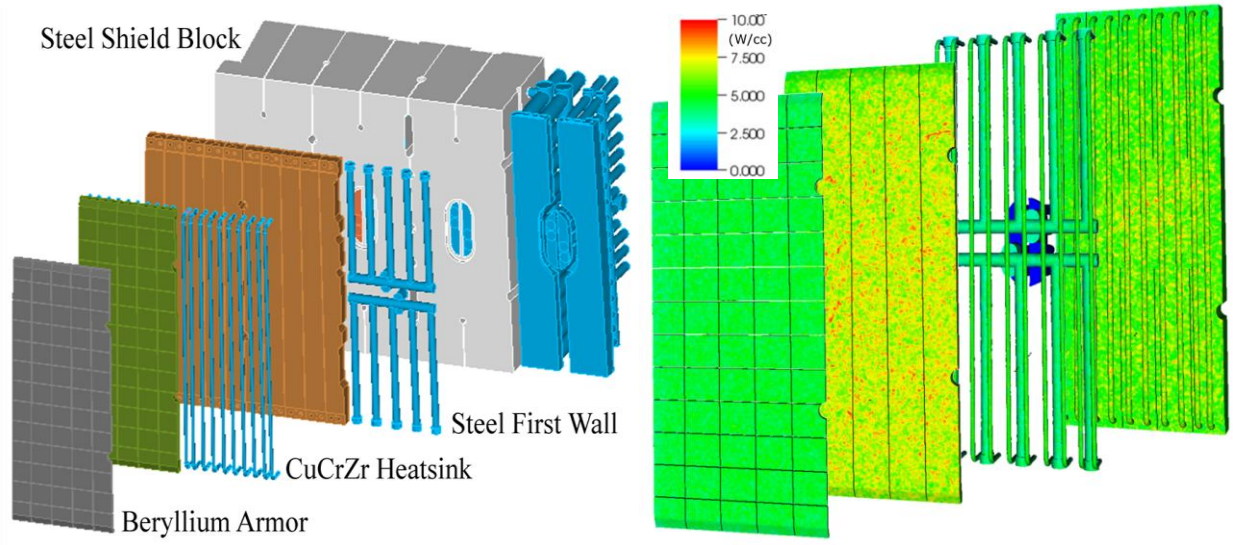


Figure 4. CAD Geometry (left) and first wall nuclear heating results (right) for ITER First Wall and Shield Module 13. The results have been interpolated onto a tetrahedral mesh suitable for thermal analysis.

The results of this mesh tally were interpolated onto a high-fidelity tetrahedral mesh for use in thermal analysis (and visualization). The results shown in Figure 4 show the detail of the geometry in the first wall by material. The beryllium tiles (first layer) are mounted on a CuCrZr substrate (second layer) and a stainless steel structural backing (fourth layer), both cooled by water (third layer). Included with the stainless steel results are the stainless steel coolant tubes that are embedded in the CuCrZr. DAG-MCNP5 is routinely used to perform such analyses of modified ITER FWS designs.

6. FUTURE WORK

Development of DAGMC is ongoing in a number of areas. Improvements are underway to guarantee the robustness of the faceting and the associated ray-tracing through that faceting, including the ability to accommodate small imperfections in the geometry. At the same time, the MOAB mesh representation will facilitate the development of mesh tallies on arbitrary polyhedral meshes for optimizing the transfer of data between different physics simulations, *e.g.* nuclear heating source terms for complex CFD analysis. These high-fidelity meshes may also lead to advances in tracking the impact of isotopic changes including activation/dose and burnup. Finally, the intersection of these high-fidelity CAD-based methods and the increasingly common

use of deterministic methods to define variance reduction parameters requires some new analysis.

ACKNOWLEDGMENTS

This work supported in part by the U.S. Department of Energy through the U.S. ITER Project Office under subcontract #4000049451. Argonne National Laboratory is managed by U. Chicago, Argonne LLC under Contract DE-AC02-06CH11357.

REFERENCES

1. H. Tsige-Tamirat, U. Fischer, A. Serikov, and S. Stickel, "Use of McCad for the conversion of ITER CAD data to MCNP geometry," *Fusion Engineering and Design*, vol. 83, 2008, pp. 1771-3.
2. Y. Li, L. Lu, A. Ding, H. Hu, Q. Zeng, S. Zheng, and Y. Wu, "Benchmarking of MCAM 4.0 with the ITER 3D model," *Fusion Engineering and Design*, vol. 82, 2007, pp. 2861-6.
3. B.C. Franke, R.P. Kensek, H.K. Schriner, L.J. Lorence, F. Gelbard, and S. Warren, "Adjoint charge deposition and CAD transport in ITS," *ANS International Meeting on Mathematical Methods for Nuclear Applications*, La Grange Park, IL 60526, United States: American Nuclear Society, 2001, p. 13.
4. T.J. Tautges, R. Meyers, K. Merkley, C. Stimpson, and C. Ernst, *MOAB: A Mesh-Oriented Database*, Sandia National Laboratories, 2004.
5. T.J. Tautges, "CGM: A geometry interface for mesh generation, analysis and other applications," *Engineering with Computers*, vol. 17, 2001, pp. 299-314.
6. T.J. Tautges, P. Knupp, J.A. Kraftcheck, and H.J. Kim, "Interoperable geometry and mesh components for SciDAC applications," *Journal of Physics: Conference Series*, vol. 16, 2005, pp. 486-90.
7. J.F. Briesmeister, *MCNP -- a general Monte Carlo code for neutron and photon transport / Judith F. Briesmeister, editor ; contributors T.E. Booth ... [et al.]*, Los Alamos National Laboratory ; available from NTIS, [1986], 1986.
8. G.D. Sjaardema, T.J. Tautges, T.J. Wilson, S.J. Owen, T.D. Blacker, W.J. Bohnhoff, T.L. Edwards, J.R. Hipp, R.R. Lober, and S.A. Mitchell, *CUBIT mesh generation environment Volume 1: Users manual*, Sandia National Laboratories, May 1994, 1994.
9. A. Scholbrook, "Attribute Management in ACIS Based Geometry Files," University of Wisconsin-Madison, 2008.
10. M. Wang, T.J. Tautges, and D. Henderson, "Acceleration of direct-interface cad-based Monte Carlo radiation transport," *American Nuclear Society's 14th Biennial Topical Meeting of the Radiation Protection and Shielding Division*, La Grange Park, United States: American Nuclear Society, 2006, pp. 50-53.
11. B. Smith, P.P.H. Wilson, and M.E. Sawan, "Three dimensional neutronics analysis of the ITER first wall/shield module 13," *22nd IEEE/NPSS Symposium on Fusion Engineering - SOFE 07*, Piscataway, NJ 08855-1331, United States: Institute of Electrical and Electronics Engineers Inc, 2007, p. 4337910.