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

Verification of MOCkingbird, an Unstructured-Mesh, Method of Characteristics Implementation Using the MOOSE Multiphysics Framework

Derek R. Gaston,* Benoit Forget,† Kord S. Smith,† Richard C. Martineau,*

*NS&T Modeling and Simulation, Idaho National Laboratory, PO Box 1625-3835, Idaho Falls, ID 83415
†Massachusetts Institute of Technology, Department of Nuclear Science and Engineering, 77 Massachusetts Avenue, Cambridge, MA 02139
derek.gaston@inl.gov

Abstract - *During operation of a nuclear reactor heat-conduction, fluid-flow, solid-mechanics, neutronics and material science are intricately linked. Therefore, accurate simulation of a reactor requires the ability to find solutions to systems of equations describing all of these phenomena simultaneously. In the past this has often been accomplished by mapping fields between disparate solvers. An alternative technique is to solve all of the physics using the same geometrical representation. In this work, neutronics is solved using the method of characteristics (MOC) on unstructured, finite-element mesh. This allows for directly linking neutronics together with other physics solved by the finite-element method (such as heat-conduction and solid-mechanics). The current study is a verification of the new* `MOCkingbird` *code developed using the* `MOOSE` *framework. Verification analysis is carried out by convergence studies using both 2D and 3D C5G7 benchmarks. The method is shown to be both accurate and efficient.*

I. INTRODUCTION

High-fidelity, multiphysics simulation of nuclear reactors is critical for ensuring the safety and performance of both existing reactor designs as they age and new, novel reactor designs. Inside a nuclear reactor, heat conduction, solid mechanics, nuclear fission, fluid-flow and many other physical processes interact in complex ways over its lifetime. Proper simulation of these processes would involve the simultaneous solution of all of these physics.

While many tools have been developed to allow for multiphysics simulation of nuclear reactors [1, 2, 3, 4], one common difficulty they all face is in the transfer of data and solution fields between disparate physical representations of the geometry. In particular, neutron transport is often solved using computational solid geometry (CSG) representations of the reactor [5, 6]. In contrast, heat conduction[7], solid mechanics[8] and fluid flow often employ unstructured mesh geometrical representations. This rift in spatial discretization has given rise to large efforts in development of efficient and accurate data transfers [2, 9, 10].

Another option is to utilize the same geometrical discretization for all physics. One popular choice is S_N or P_N related methods for solution of neutron transport on unstructured mesh [11, 12, 13, 14, 12, 15]. Monte-Carlo (MC) methods have also been implemented on unstructured mesh for multiphysics coupling [16, 17].

In comparison, relatively few explorations have been made into using the Method of Characteristics (MOC) on unstructured, finite-element mesh. `NEWT` [18] used a discrete ordinate method on a non-uniform grid, though it wasn't a finite-element mesh. More recently, `PROTEUS-MOC` [19] has shown that unstructured mesh MOC is viable, although [19] used an operator form of MOC and a linear solver in addition to some assumptions about axially extruded geometry. In addition, `MOCUM` [20] has demonstrated two-dimensional capability.

This paper introduces `MOCkingbird`, a new implementation of unstructured-mesh MOC built using the `MOOSE` [21] finite-element, multiphysics framework. `MOCkingbird` utilizes long-characteristics and a traditional sweeping method similar to CSG-based MOC codes such as `OpenMOC` [5]. By building on top of an existing multiphysics framework it is possible to directly link to physics solved using the finite-element method (such as heat conduction and solid mechanics), enabling high-fidelity, multiphysics reactor simulation. This implementation also allows for fully-heterogeneous 2D and 3D geometries, including the ability to solve on a deforming (moving) mesh. Finally, special attention has been paid to domain decomposed parallelization allowing for whole-core solutions.

This paper will proceed as follows: first, an explanation of the mathematics behind the MOC method and details on the implementation of that method using unstructured mesh will be provided. Next, the domain decomposed, parallel ray-tracing algorithm will be described. Finally, results of both 2D and 3D C5G7 will be used to show the accuracy of the method.

II. METHOD OF CHARACTERISTICS

What follows is a brief introduction to the MOC method utilized in `MOCkingbird`. To begin, a simplified form of the Boltzmann transport equation is considered. In particular, steady state is assumed and an eigenvalue problem is formed:

$$\begin{aligned}\hat{\Omega}\cdot\nabla\psi(\boldsymbol{r},\hat{\Omega},E)+\Sigma_t\psi(\boldsymbol{r},\hat{\Omega},E)=\\ \int_0^\infty\int_{4\pi}\Sigma_s(\boldsymbol{r},\hat{\Omega}\cdot\Omega',E'\rightarrow E)\psi(\boldsymbol{r},\Omega',E')d\Omega'dE'\\ +\frac{\chi(E)}{4\pi k}\int_0^\infty\nu\Sigma_f(\boldsymbol{r},E')\phi(\boldsymbol{r},E')dE'.\end{aligned}\tag{1}$$

Equation (1) can be solved to find the angular flux (ψ) and principle eigenvalue (k) that balance the system. Σ_t, Σ_s and

Σ_f are the total, scatter and fission cross sections respectively which represent collision probabilities. $\boldsymbol{r}$ denotes a position in three dimensional (3D) space. $\hat{\Omega}$ represents direction with E being energy. ϕ is the "scalar flux" and is computed as $\phi = \int_{4\pi} \psi(\boldsymbol{r}, \Omega, E) d\Omega$. χ represents the fission spectrum and ν is the neutron multiplicity.

To further simplify (1) a substitution is made:

$$Q(\boldsymbol{r}, \hat{\Omega}, E) = \int_0^\infty \int_{4\pi} \Sigma_s(\boldsymbol{r}, \hat{\Omega} \cdot \Omega', E' \rightarrow E) \psi(\boldsymbol{r}, \Omega', E') d\Omega' dE' + \frac{\chi(E)}{4\pi k} \int_0^\infty \nu \Sigma_f(\boldsymbol{r}, E') \phi(\boldsymbol{r}, E') dE'. \tag{2}$$

If the equation is considered only along a single path $\boldsymbol{\Omega}$ (a "characteristic"), with s as the distance along the path beyond s_0, (1) can be rewritten using (2) as:

$$\frac{\partial}{\partial s} \psi(s, \boldsymbol{\Omega}, E) + \Sigma_t(s, E) \psi(s, \boldsymbol{\Omega}, E) = Q(s, \boldsymbol{\Omega}, E). \tag{3}$$

Solving (3) and discretizing energy into discrete energy groups g leads to:

$$\psi_g(s, \boldsymbol{\Omega}) = \psi_g(s_0, \boldsymbol{\Omega}) e^{-\int_{s_0}^{s} \Sigma_{tg}(s') ds'} + \int_{s_0}^{s} Q_g(s', \boldsymbol{\Omega}) e^{-\int_{s'}^{s} \Sigma_{tg}(s'') ds''} ds', \tag{4}$$

Finally, if cross sections and scalar fluxes are considered to be piecewise constant over regions of the domain and multiple lines (called "tracks", denoted with subscript k) through the domain are considered:

$$\psi_{k,g}(s) = \psi_{k,g}(s_0) e^{-\tau_{k,i,g}} + \frac{Q_{i,g}}{\Sigma_{t,i,g}} (1 - e^{-\tau_{k,i,g}}) \tag{5}$$

$$\tau_{k,i,g} = \Sigma_{t,i,g} (s - s_0), \tag{6}$$

where τ is the "optical depth" and i represents intersections of the k tracks with individual regions within the geometry. These i intersections are called "segments".

Equation (5) is conducive to solution via computational methods. A source iteration method is formed for solution of the eigenvalue problem where each iteration involves iteration across the k tracks and i segments, taking incoming angular flux ($\psi_{k,g}(s_0)$) on each segment and producing $\psi_{k,g}(s)$ on the other side.

Within `MOCkingbird` MOC "sweeps" (integrations across tracks) are performed in a similar manner to most other MOC based codes, such as `OpenMOC` [5]. It should also be noted that `MOCkingbird` currently lacks advanced acceleration methodology; the implementation of which will be the focus of future research. The main distinguishing MOC features of `MOCkingbird` is the handling of geometry, track intersections and parallelization.

III. RAY TRACING

To utilize (5) for solution of the eigenvalue problem it's required to trace tracks through the domain, intersecting them with the geometry. This can be thought of as being similar to traditional ray tracing applications. While traditional MOC codes often employ CSG for geometrical representation, `MOCkingbird` directly utilizes the unstructured mesh native to `MOOSE`. Using unstructured mesh has both pros and cons. As mentioned earlier, working on unstructured mesh allows `MOCkingbird` to directly couple to other physics solved using `MOOSE` and allows for a large amount of geometrical flexibility (anything that can be meshed can be used as the domain). However, it also presents difficulties in the form of mesh movement, non-axis aligned boundaries, gaps in the mesh, etc. In addition, distributed memory parallelization is not straightforward with unstructured mesh. What follows is a brief overview of the ray-tracing capabilities currently contained within `MOCkingbird`.

1. Ray Tracing Algorithm

Ray tracing within `MOCkingbird` heavily relies on the connectivity structure present within the unstructured mesh of `MOOSE`. As shown in Algorithm 1, each ray begins life within a particular element. Each side of that element is tested for intersection with the ray. The connectivity of the elements is then utilized to find the next element the ray will move to and then the algorithm repeats. By testing each side and moving to neighboring elements the ray can be traced completely across the domain from one domain boundary to another.

```
current_elem ← starting element;
current_point ← starting point;
while not at domain boundary do
    for each side of current_elem do
        test intersection
    end
    if no intersection then
        if very near boundary then
            apply boundary condition
            break loop
        end
        if near element corner then
            for all elements at corner do
                look for longest path out
            end
        end
    end
    current_point ← optimal intersection;
    current_elem ← neighboring element;
end
```

Algorithm 1: Ray Tracing Algorithm

The most basic component of a ray tracing capability is to find intersections of lines (rays) with the geometry. `MOCkingbird` currently supports geometry containing triangles and quadrilaterals in 2D as well hexahedrals in 3D. For

two dimensional ray tracing a straightforward line to line-segment intersection algorithm is utilized. In 3D a modified form of the algorithm described in [22] is used for finding intersections of rays with the quadrilateral sides that form a hexahedral.

A. Corner Cases

Unstructured mesh provides many opportunities for both figurative and literal corner cases that must be carefully considered. For instance, a ray directly striking a junction between four quadrilateral elements will have difficulty traversing through the connectivity structure to emerge on the other side. As shown in Algorithm 1, these cases are explicitly handled within the ray tracing capability of `MOCkingbird` with specialized code that searches for optimal pathways away from the current intersection point.

In the case of a ray striking an internal (to the domain) corner, each element connected to the corner of the current element will be inspected to see if it is a candidate for the ray to leave through. Longer path lengths out of a corner are preferred, otherwise a ray may infinitely cycle through all of the elements meeting at the corner due to floating point tolerances in the ray tracing algorithm. Therefore, each element connected to the corner is taken in turn and searched for an intersection point, with the one providing the longest path out of the corner being chosen.

A ray intersecting an internal (to the domain) element edge *very* near a boundary (within floating point tolerance) but not actually an element edge that lies on the boundary may not be able to continue to the boundary. In this case, as shown in Algorithm 1, `MOCkingbird` will just consider the ray to have actually met the domain boundary. `MOCkingbird` will then search for a side in the current element that is actually on a boundary and (within floating point tolerance) could contain the current intersection point. The intersection point is then assigned to that side, ultimately causing the boundary condition to be applied to the ray and the ray tracing to end.

IV. PARALLELIZATION

The geometric representation and accuracy needed in reactor physics requires a massive amount of computational capability. Therefore, parallelization is essential for any neutron transport tool. Modern clusters and supercomputers typically employ two types of parallelism: shared memory and distributed memory. Shared memory parallelism is utilized within one computational "node" of the cluster, allowing multiple processors to share data within one memory hierarchy. Distributed memory parallelism typically involves communication between nodes over a network, often employing the Message Passing Interface (MPI) [23]. These two capabilities are complementary and both are used within `MOCkingbird` and can be used simultaneously.

1. Shared Memory

Shared memory parallelism within `MOCkingbird` is achieved using a "task-based" interface to UNIX threads. The backend for those tasks can be served by either pthreads [24], OpenMP [25] or Intel TBB [26] depending on the configuration of the cluster. All major compute loops within `MOCkingbird` and `MOOSE` are threaded. In particular, the ray tracing / transport sweep loops within `MOCkingbird` are completely threaded. Each thread receives a "packet" of tracks that need to be traced and all threads carry out that tracing simultaneously. Instead of using thread locking, duplicate memory banks are used for the source contributions from the integration process, allowing each thread to work completely independently.

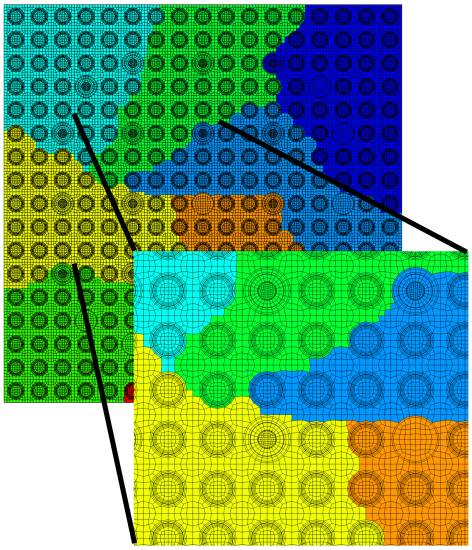


Fig. 1: Assembly geometry partitioned for use with 8 MPI processes.

2. Distributed Memory

Distributed memory parallelism is handled using MPI [23]. One defining feature of the distributed memory capability within `MOCkingbird` is the utilization of asynchronous MPI communication to reduce message passing overhead. This is an active area of research that may allow `MOCkingbird` to scale to full-core problems.

A. Mesh Partitioning

In order to achieve distributed memory parallelism, work has to be split up over the available nodes within the cluster. Within `MOCkingbird` the well-known `METIS` [27] library is utilized to split the mesh. `METIS` has been used for many years by finite-element tools employing distributed memory parallelism. Figure 1 shows an example of an assembly that has been partitioned by `METIS`. A custom `libMesh` tool takes the partitioning from `METIS` and generates individual mesh files for each MPI process to read when `MOCkingbird` is run. This ensures that each process only reads and stores the portion of the domain it is working with.

B. Track Claiming

Once the domain is split, decisions must be made on how to trace tracks across it. Other MOC codes often employ

"Spatial Domain Decomposition" (SDD) [28] where tracks begin and end at processor boundaries and boundary angular fluxes are exchanged after each iteration. This has a large downside of requiring the storage of angular fluxes at the intersection of every ray with every processor boundary. While this may work well in codes employing structured geometry that can easily be partitioned into equal pieces, unstructured mesh partitioning can create many processor boundaries which would then imply an excess amount of memory utilization.

To combat this, `MOCkingbird` utilizes a "long characteristic" approach. At each iteration, all tracks start at the domain boundary and are traced the full length of the domain until they intersect another domain boundary. This provides for the exact same execution behavior in both serial and parallel. In addition, there are no internal (to the domain) angular fluxes to store, greatly reducing the overall memory requirements of the application.

Of course, this capability is not completely free of issues. In particular a lot of messages need to be sent in a very unstructured pattern during the iteration. Further, processors working on the internal portion of the domain can remain idle while they wait for work to propagate to them. The aforementioned asynchronous MPI communication strategy has been developed to help combat these two issues. While still research in progress, the capability is operating smoothly and is able to provide good preliminary results as shown below.

Critical to this capability is the assignment of track starting positions. As mentioned earlier, `OpenMOC` is utilized to generate the long characteristic tracks on the fly in memory. `MOCkingbird` takes each track and uses a quad/oct-tree search mechanism to find where it will begin within the mesh (required by ray tracing Algorithm 1). Each MPI processes does this independently for all tracks that could start on that process's portion of the domain.

The next step is a "claiming" round where each process attempts to claim tracks which begin within its subdomain (note that a track starting directly on a processor boundary could be claimed by more than one MPI process). MPI communication is used to determine the ultimate "winner" for each track. That winner will always start that track at the beginning of every iteration. The parallel ray tracing capability within `MOCkingbird` will ensure that every track that is started from the boundary is propagated across the domain, achieving a transport sweep.

V. RESULTS

While there are many areas of active research within `MOCkingbird`, the current study is primarily concerned with accuracy of the deterministic transport scheme. In order to assess the efficacy of the solver both the 2D and 3D versions of the C5G7 benchmark were run. Those results are summarized in the next sections.

1. 2D C5G7

The well-known C5G7 benchmark [29] has often been used in the literature as a verification step for MOC-based, deterministic transport codes [5, 30, 31, 20]. As shown in Figure 2, the quarter core 2D benchmark includes four assemblies with 17x17 fuel pins and a water reflector. One of the defining features of C5G7 is the heterogeneous representation of the pin cell with the exception of the clad and gap homogenization.

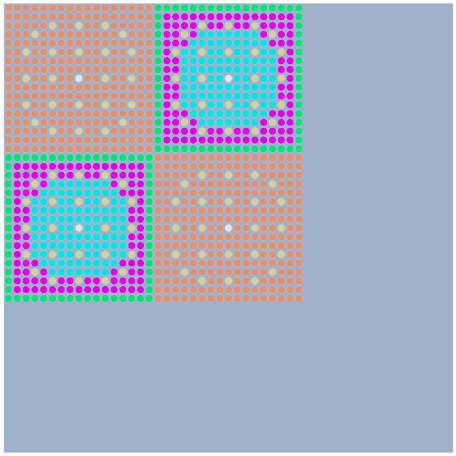
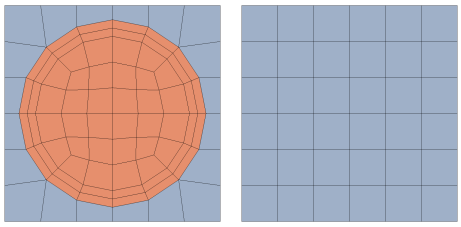


Fig. 2: Quarter-core C5G7 geometry. Colors represent sets of fuel mixtures and the moderator.



(a) Fuel pin-cell (b) Reflector cell

Fig. 3: Representative cells making up the C5G7 discretization.

The first step in producing solutions using `MOCkingbird` is to discretize the domain using a finite-element mesh. `MOCkingbird` is currently compatible with first-order quadrilateral and triangular elements in 2D, and tetrahedral and hexahedral elements in 3D. Python scripts were used to create the geometry and generate volume-preserving pin-cells using `Cubit` [32]. These pin-cells were then extruded and stitched together using `MOOSE` `MeshModifiers` to form the final mesh.

Figure 3a shows the unstructured mesh utilized by `MOCkingbird` for each of the pin-cells in the following results. Each pin-cell contains 84 elements, for a total of 97,104 elements within the assembly regions of the domain. As shown in Figure 3b, the water reflector mesh is quite a bit coarser, containing only 36 elements per cell for a total of 52,020 elements in the water reflector. In total, 149,124 elements are used to discretize the 2D domain. This total is directly in line with the level of spatial resolution used in the CSG-based `OpenMOC` results [5] for the same problem.

`MOCkingbird` employs two different angular flux integration strategies: deterministic using a fixed quadrature and stochastic using the Tramm Random Ray Method (TRRM)

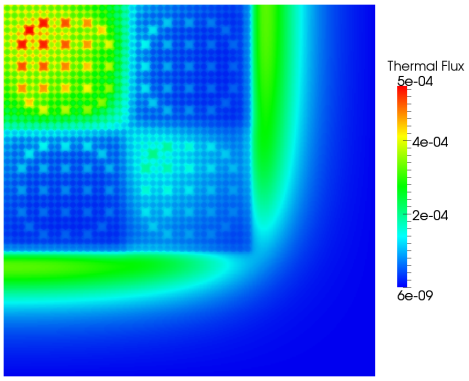


Fig. 4: Thermal flux using 40 azimuthal angles.

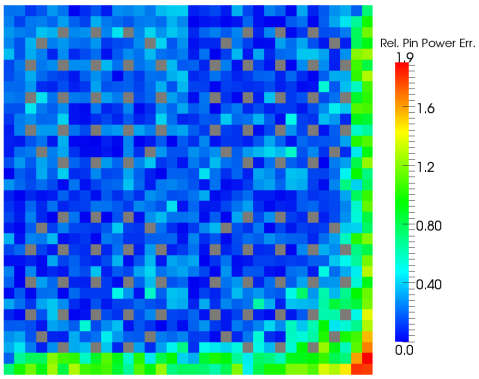


Fig. 5: Relative error in normalized pin powers.

[33]. For this verification study only the deterministic option will be used. Track starting positions and directions are read from a file or directly generated by `OpenMOC` [34]. `OpenMOC` provides many options for track generation, in this study only spacing and number of azimuthal angles will be modified. In the polar direction, TY [35] quadrature with three angles is used.

Figure 4 shows a solution using 64 azimuthal angles and a 0.01 azimuthal spacing. This result agrees well with other published results [5]. In addition, the normalized, relative pin power error is shown in Figure 5. This result is similar to those found in [31], showing larger error in the fuel pins near the reflector.

To assess the accuracy of the code an angular convergence study was conducted. A set azimuthal spacing of 0.01 was utilized with increasing azimuthal angles. Convergence was judged using a fission source RMS step tolerance:

$$\sqrt{\frac{\sum_q \left(\frac{\Sigma_f \phi_{new} - \Sigma_f \phi_{old}}{\Sigma_f \phi_{old}}\right)^2}{N_q}} \tag{7}$$

(where q is iterating over the fissionable elements) of 1e-5. The results can be found in Table I. The results were run using a single 12-core 2.6 GHz, Intel Core i7 processor. The twelve cores were all utilized via 12 MPI processes performing the domain-decomposed sweeps. This test setup was chosen to closely replicate the results in [5].

TABLE I: `MOCkingbird` converged eigenvalues for increasing azimuthal angles with an azimuthal spacing of 0.01.

# Angles	# Sweeps	k_{eff}	$\Delta\rho$ (pcm)	Time (s)
4	642	1.18553	-102	502
8	651	1.18486	-169	955
16	650	1.18516	-139	1836
32	649	1.18641	-14	3587
64	649	1.18666	11	7103
128	649	1.18678	23	14492

The results in Table I show excellent agreement with the results published in [5]. Both the trends and actual numbers for sweeps, k_{eff}, $\Delta\rho$ and even run time correlate well with the published results. The true benchmark reference k_{eff} is 1.18655 ± 9.5 [29]. `MOCkingbird` is able to converge to within a few 10's of pcm of this value.

The time to solution in Table I is generally more than that found for `OpenMOC` in [5]. This is in line with the fact that `MOCkingbird` is using full double-precision and the intrinsic exponential whereas `OpenMOC` is typically run using single-precision and an optimized, table-based exponential evaluation. In addition, `MOCkingbird` incurs a penalty for performing all ray-tracing "on-the-fly" as opposed to the offline ray tracing and segmentation that `OpenMOC` uses.

While on-the-fly ray-tracing incurs a time penalty it also provides increased flexibility for heterogeneous 2D and 3D geometries with reduced memory overhead. In addition, on-the-fly ray-tracing also allows for domain modifications during the calculation such as when coupling to solid-mechanics for computation of mesh deformation. With these differences in mind, the time to solution matches well with expectations.

2. 3D C5G7

The C5G7 benchmark has been extended to a full suite of 3D benchmarks [36]. There are three primary configurations in the 3D benchmark: *Unrodded*, *Rodded A* and *Rodded B*. *Rodded B* offers a large challenge by having multiple banks of control rods partially inserted to different depths. As a consequence it is the focus of the current study.

The mesh for *Rodded B* was generated by taking the 2D mesh used previously and extruding it using `MOOSE` in four separate sections representing each of the unique radial layouts encountered axially in the problem. These four three dimensional meshes were then stitched together using `MOOSE` to form the final domain. In this way, the radial fidelity was automatically set by the 2D mesh, however the axial fidelity is left as a choice. The number of axially extruded elements needed to achieve a high-fidelity result will be the focus of this verification effort.

Three different meshes were created: *coarse*, *medium* and *fine* corresponding to using 48, 100 and 200 total axial segments and generating final domains with 7.5M, 15M and 30M elements respectively. After examining the results in Table I the number of azimuthal angles for track generation was chosen to be 32. An azimuthal spacing of 0.13cm was chosen after a small amount of experimentation trying to balance accuracy and problem size. In 3D the number of polar angles

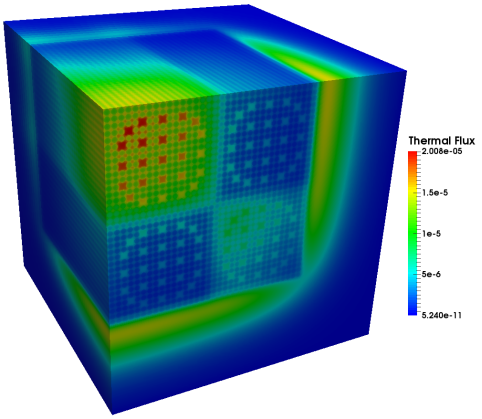


Fig. 6: Thermal flux for the *coarse* 3D mesh.

and axial spacing must also be chosen. The current study used 6 polar angles with 0.3cm axial spacing. Ultimately, these settings generated nearly 21M tracks within the 3D domain. It needs to be stressed that 3D deterministic track generation is a non-trivial problem that has been the focus of considerable research [34] and the current authors are grateful for the ability to reuse that work here.

Each of the three meshes were used in runs of `MOCkingbird` that utilized 960 processors on Idaho National Laboratory's Falcon supercomputer. Falcon consists of nodes containing 24 Intel Xeon cores interlinked by infiniband. The results of these runs can be found in Table II. Within Table II AVG and RMS are measures of the fission source error and are defined as:

$$AVG = \frac{\sum_n |e_n|}{N_n}, \tag{8}$$

$$RMS = \sqrt{\frac{\sum_n e_n^2}{N_n}}, \tag{9}$$

where n is iterating over fuel pins, N_n is the total number of fuel pins and e_n is the percent error as compared to the benchmark. It should be noted that timings and iteration counts for the 3D runs have been omitted due to the use of experimental acceleration techniques that are beyond the scope of this paper.

TABLE II: Eigenvalues and fission source errors for each mesh used for the 3D C5G7 *Rodded B* configuration.

Mesh	# Elem	k_{eff}	$\Delta\rho$ (pcm)	AVG	RMS
Coarse	7.5M	1.07455	322	0.60	0.82
Medium	15M	1.07687	90	0.53	0.76
Fine	30M	1.07752	25	0.45	0.67

The results show that `MOCkingbird` is able to achieve a k_{eff} within 25pcm of the benchmark solution (1.07777) on the finest grid. It is clear that axial fidelity does matter in this case as the coarse mesh result is off by 322pcm. The convergence of `MOCkingbird` toward the true eigenvalue and fission source as the mesh is refined is a good indicator that the 3D, parallel, unstructured mesh MOC solution algorithm is working properly.

VI. CONCLUSIONS

High-fidelity nuclear reactor simulations rely on the close coupling of many fundamental physics. One way to achieve that coupling is by solving all of the physics using the same geometrical representation. `MOCkingbird` is a new MOC-based neutron transport tool with the explicit goal of enabling straightforward coupling with finite-element based solution methods for multiphysics simulation. It has been shown to be accurate for both 2D and 3D on the C5G7 benchmark, with more testing to come in the future. The goal of this new tool is to help enable safety analysis and design studies on both existing and next generation designs.

VII. ACKNOWLEDGMENTS

The authors would like to thank John Tramm from MIT for his aid during the development of these algorithms. We would also like to acknowledge Samuel Shaner and Geoffrey Gunow from MIT for their contributions in track generation methodology and implementation. This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517. This work was funded by Idaho National Laboratory. This manuscript has been authored by Battelle Energy Alliance, LLC under Contract No. DE-AC07- 05ID14517 with the US Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

REFERENCES

1. D. R. GASTON, C. J. PERMANN, J. W. PETERSON, A. E. SLAUGHTER, D. ANDRŠ, Y. WANG, M. P. SHORT, D. M. PEREZ, M. R. TONKS, J. ORTENSI, ET AL., "Physics-based multiscale coupling for full core nuclear reactor simulation," *Annals of Nuclear Energy*, **84**, 45–54 (2015).
2. M. ELLIS, B. FORGET, K. SMITH, and D. GASTON, "Preliminary coupling of the Monte Carlo code OpenMC and the Multiphysics Object-Oriented Simulation Environment (MOOSE) for analyzing Doppler feedback in Monte Carlo simulations." in "ANS MC2015 - Joint International Conference on Mathematics and Computation (MC), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method," (2015).
3. K. CLARNO, T. EVANS, B. COLLINS, R. PAWLOWSKI, R. MONTGOMERY, B. KOCHUNAS, and D. GASTON, "Design of a High Fidelity Core Simulator for Analysis of Pellet Clad Interaction," Tech. rep., CASL Technical Report CASL-U-2015-0036-000 (2015).
4. J. LEPPÃĎNEN, V. HOVI, T. IKONEN, J. KURKI, M. PUSA, V. VALTAVIRTA, and T. VIITANEN, "The

Numerical Multi-Physics project (NUMPS) at VTT Technical Research Centre of Finland," *Annals of Nuclear Energy*, **84**, 55 – 62 (2015), multi-Physics Modelling of LWR Static and Transient Behaviour.

5. W. BOYD, S. SHANER, L. LI, B. FORGET, and K. SMITH, "The OpenMOC method of characteristics neutral particle transport code," *Annals of Nuclear Energy*, **68**, 43–52 (2014).
6. P. K. ROMANO and B. FORGET, "The OpenMC Monte Carlo Particle Transport Code," *Ann. Nucl. Energy*, **51**, 274–281 (2013).
7. C. NEWMAN, G. HANSEN, and D. GASTON, "Three dimensional coupled simulation of thermomechanics, heat, and oxygen diffusion in UO2 nuclear fuel rods," *Journal of Nuclear Materials*, **392**, *1*, 6–15 (2009).
8. J. HALES, S. NOVASCONE, R. WILLIAMSON, D. GASTON, and M. TONKS, "Solving nonlinear solid mechanics problems with the Jacobian-free Newton Krylov method," *Computer Modeling in Engineering & Sciences(CMES)*, **84**, *2*, 123–153 (2012).
9. S. SLATTERY, P. WILSON, and R. PAWLOWSKI, "The data transfer kit: a geometric rendezvous-based tool for multiphysics data transfer," in "International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)," (2013), pp. 5–9.
10. V. S. MAHADEVAN, "Coupled Physics Environment (CouPE) library-Design, Implementation, and Release," Tech. rep., Argonne National Laboratory (ANL), Argonne, IL (United States) (2014).
11. F. N. GLEICHER, J. ORTENSI, B. W. SPENCER, Y. WANG, S. R. NOVASCONE, J. D. HALES, D. R. GASTON, R. L. WILLIAMSON, and R. C. MARTINEAU, "The coupling of the neutron transport application RATTLESNAKE to the nuclear fuels performance application BISON under the MOOSE framework," in "JAEA Conference," (2015).
12. Y. WANG, "Nonlinear diffusion acceleration for the multigroup transport equation discretized with S {sub N} and continuous FEM with rattlesnake," Tech. rep., American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL 60526 (United States) (2013).
13. J. MOREL, J. MCGHEE, and E. W. LARSEN, "A three-dimensional time-dependent unstructured tetrahedral-mesh SP N method," *Nuclear science and engineering*, **123**, *3*, 319–327 (1996).
14. G. PALMIOTTI, M. SMITH, C. RABITI, M. LECLERE, D. KAUSHIK, A. SIEGEL, B. SMITH, E. LEWIS, ET AL., "UNIC: Ultimate neutronic investigation code," in "Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications, Monterey, California," (2007).
15. A. WATSON, R. GROVE, and M. SHEARER, "Effective Software Design for a Deterministic Transport System," in "International Conference on Advances in Mathetatics, Computational Methods, and Reactor Physics, Saratoga Springs, NY," (2009).
16. 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).
17. M. AUFIERO, C. FIORINA, A. LAUREAU, P. RUBIOLO, and V. VALTAVIRTA, "Serpent–openfoam coupling in transient mode: simulation of a godiva prompt critical burst," in "Proceedings of ANS MC2015–Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method, Nashville, Tennessee," (2015).
18. M. D. DEHART, "Advancements in generalized-geometry discrete ordinates transport for lattice physics calculations," *In Proc. Of PHYSOR-2006*, pp. 10–14 (2006).
19. A. MARIN-LAFLECHE, M. SMITH, and C. LEE, "Proteus-MOC: A 3D deterministic solver incorporating 2D method of characteristics," in "Proceedings of the 2013 International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering-M and C 2013," (2013).
20. X. YANG, R. BORSE, and N. SATVAT, "{MOCUM} solutions and sensitivity study for {C5G7} benchmark," *Annals of Nuclear Energy*, **96**, 242 – 248 (2016).
21. D. GASTON, C. NEWMAN, G. HANSEN, and D. LEBRUN-GRANDIE, "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering and Design*, **239**, *10*, 1768–1778 (2009).
22. A. LAGAE and P. DUTRÉ, "An efficient ray-quadrilateral intersection test," *journal of graphics, gpu, and game tools*, **10**, *4*, 23–32 (2005).
23. W. GROPP, E. LUSK, N. DOSS, and A. SKJELLUM, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel computing*, **22**, *6*, 789–828 (1996).
24. B. NICHOLS, D. BUTTLAR, and J. FARRELL, *Pthreads programming: A POSIX standard for better multiprocessing*, " O'Reilly Media, Inc." (1996).
25. L. DAGUM and R. MENON, "OpenMP: an industry standard API for shared-memory programming," *IEEE computational science and engineering*, **5**, *1*, 46–55 (1998).
26. A. KUKANOV and M. J. VOSS, "The Foundations for Scalable Multi-core Software in Intel Threading Building Blocks." *Intel Technology Journal*, **11**, *4* (2007).
27. G. KARYPIS and V. KUMAR, "Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs," *SC Conference*, **0**, 35 (1996).
28. B. KELLEY, ET AL., "CMFD Acceleration of Spatial Domain-decomposition Neutron Transport Problems," in "PHYSOR 2016," Knoxville, TN, USA (April 2012).
29. E. LEWIS, M. SMITH, N. TSOULFANIDIS, G. PALMIOTTI, T. TAIWO, and R. BLOMQUIST, "Benchmark specification for Deterministic 2-D/3-D MOX fuel assembly transport calculations without spatial homogenization (C5G7 MOX)," *NEA/NSC* (2001).
30. B. COLLINS, B. KOCHUNAS, and T. DOWNAR, "Assessment of the 2D MOC solver in MPACT: Michigan parallel characteristics transport code," Tech. rep., American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL 60526 (United States) (2013).

31. R. M. FERRER and J. D. RHODES III, "A Linear Source Approximation Scheme for the Method of Characteristics," *Nuclear Science and Engineering*, **182**, *2*, 151–165 (2016).
32. T. D. BLACKER, W. J. BOHNHOFF, and T. L. EDWARDS, "CUBIT mesh generation environment. Volume 1: Users manual," Tech. rep., Sandia National Labs., Albuquerque, NM (United States) (1994).
33. J. TRAMM, K. SMITH, B. FORGET, and A. SIEGEL, "The Random Ray Method – A computationally efficient algorithm for 2D and 3D neutron transport enabling high fidelity nuclear reactor simulation," (2017), manuscript submitted for publication.
34. S. SHANER ET AL., "Theoretical Analysis of Track Generation in 3D Method of Characteristics," in "Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method," Nashville, Tennessee (2015), Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method.
35. A. YAMAMOTO, M. TABUCHI, N. SUGIMURA, T. USHIO, and M. MORI, "Derivation of optimum polar angle quadrature set for the method of characteristics based on approximation error for the Bickley function," *Journal of Nuclear Science and Technology*, **44**, *2*, 129–136 (2007).
36. OECD, "Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation, MOX Fuel Assembly 3D Extension Case," *NEA* (2005).