# UPDATES ON OPEN-SOURCE MULTIPHYSICS ENDEAVORS WITH OPENMC

SHRIWISE, P.1 ROMANO, P.K.1 NOVAK, A.1 RAHAMAN, R.1

P. Shriwise

Argonne National Laboratory

Lemont, IL USA

Email: pshriwise@anl.gov

1Argonne National Laboratory, 9700 S Cass Ave, Lemont, IL 60439, United States

Email contact of corresponding author: pshriwise@anl.gov

OpenMC [1] is currently used in a number of projects aimed at multiphysics modeling of nuclear systems. This paper reviews recent progress in this area for open-source applications involving OpenMC, recently added or planned features relevant to multiphysics modeling, and challenges faced during development of open-source multiphysics projects along the way.

# Multiphysics Projects Updates

OpenMC is part of many multiphysics efforts. Two fully open-source projects will be discussed in this section: The Exascale Nuclear Reactor Investigative COde (ENRICO) and A Unified Resource for OpenMC (fusion) Reactor Applications (AURORA). Each code currently has the ability to perform in-memory coupling of physics kernels relevant to reactor simulation.

ENRICO is part of the Exascale Computing Project under the U.S. DOE. ENRICO features an abstract interface for passing information between the neutronics and thermal-hydraulics codes [2]. The value of an abstract interface has been demonstrated by support for both OpenMC, a state-of-the-art Monte Carlo neutronics code primarily developed at Argonne National Laboratory, as well as Shift [3], a high performance GPU-enabled Monte Carlo code developed at Oak Ridge National Laboratory. This value is also demonstrated for thermal-hydraulics solvers by support of both Nek5000 [4], an open-source and highly scalable CFD solver, along with a thermal hydraulic surrogate subchannel code. The coupling in ENRICO performs a mapping from a neutronics CSG-cell to thermal hydraulic mesh element during initialization to determine how neutron heating values will influence the CFD simulation as well as how temperatures will be fed back to the neutronics model. Control of the coupling scheme is available in the form of controls for the temperature convergence criterion; the norm used to determine convergence; maximum timesteps; maximum picard iterations; and underrelaxation parameters for the heat source, temperature, and density. The parallel data transfers are designed to allow for optimal allocation of computing resources for the codes in use, allowing for separate topologies of processes for the neutronics and thermal hydraulic simulations. Since the initial publication cited above, several improvements have been made to ENRICO. One of which is the update from Nek5000 to NekRS, a GPU-enabled version of Nek5000 which shows unparalleled performance in CFD simulations. Another is the addition of a thermal-hydraulics driver that allows ENRICO to perform in-memory coupling with a modified version of OpenFOAM’s conjugate heat transfer solver [5]. When employing additional physics kernels with OpenFOAM, however, ENRICO will have no knowledge of the convergence criterion for the additional kernels and the coupling methods being used in OpenFOAM as ENRICO is designed to be the main driver of all subsequent codes.

Another effort is that of AURORA [6], an open-source project oriented toward holistic modeling of fusion energy systems using the MOOSE framework [7]. While it is largely intended for modeling of tokamak fusion systems, the coupling mechanisms employed can also be applied to fission reactors. AURORA couples native MOOSE solvers for heat conduction and tensor mechanics to OpenMC on a unified unstructured mesh. AURORA relies on MOOSE to transfer solutions for coupled fields on a transfer mesh in MOOSE. Discussing only the physics kernels equivalent to those demonstrated in ENRICO to date, results from the neutronics code are mapped onto the tetrahedral solution mesh using unstructured mesh tallies in OpenMC. Temperature regions are defined and updated between iterations in OpenMC using the boundary triangles of mesh elements whose temperature values lie inside specified intervals to create tessellated Direct Accelerated Monte Carlo (DAGMC) Toolkit cells in OpenMC for transport [8]. The MOOSE framework provides a variety of options for underrelaxation for field transfers, convergence criterion, and the order of execution for coupled physics. Control of parallel communication topology is limited, however, requiring that each physics kernel occupy the same number of processes.

In general, ENRICO is designed for high performance on the exascale machines currently under construction in the US DOE complex, focusing on performance of the transfers and coupling methodologies for light water reactors. The addition of OpenFOAM does open opportunities for incorporation of additional physics kernels, but with limited control over the code execution order and iteration. AURORA is designed to be a generic coupling tool with OpenMC and other MOOSE physics kernels. AURORA provides more flexibility for order of code execution and solver sub-stepping, but its coupling procedure and OpenMC run-mode is more computationally expensive than that of ENRICO.

# Multiphysics-Related Features in OpenMC

To support each of these projects, OpenMC has added a variety of new features. Some have been contributed out of convenience to external projects while others were added in expectation of future needs of these projects.

To support coupled field transfers for cells containing universes or lattices, the ability to set the temperature of all material-filled cells within a universe- or lattice-filled cell in OpenMC at any point during the simulation was added for convenience. This capability is useful for managing coupled field transfers on an entire pebble rather than for each individual TRISO in the pebble, for example.

Support for unstructured mesh tallies has been included in OpenMC for two mesh libraries: MOAB (the Mesh Oriented datABase) [9] and libMesh [10]. It is often possible to split CSG cells manually to increase the spatial resolution of tallies, but this quickly becomes onerous for complex geometries. The ability to tally on unstructured meshes allows one to provide neutron heating information on a resolution similar or equal to that of the CFD/TH mesh in multiphysics work. Support for MOAB mesh is limited to tetrahedral elements with support for both tracklength and collision based estimators. The libMesh implementation allows for any element type supported by the library but is limited to a collision based estimator.

Support for universe-based CAD geometry is imminent in OpenMC. Upon completion of this feature, Direct Accelerated Geometry Monte Carlo (DAGMC) tessellated CAD geometry can be used to fill CSG cells in OpenMC models. This allows users to represent regions as CAD that are either better suited for CAD modeling or are impossible to represent using CSG’s surface definitions (e.g., spacer grids, wire wraps, piping manifolds, etc.). This feature provides the benefit of allowing users to leave the bulk of the model as CSG, improving particle transport performance. Universe-based CAD components are also able to be duplicated via insertion of the CAD universe into more than one cell without an increased memory footprint. The feature’s relevance to multiphysics modeling, similar to the work performed by Talamo et al. [11], is the ability to maintain an OpenMC model that matches the geometric detail of CFD/TH models with minimal overhead while maintaining the axioms of the generic/automated modeling methods currently employed in the projects described above.

Finally, prototype development of delta tracking in OpenMC is underway. In addition to being beneficial for certain reactor geometries, delta tracking simplifies the complexity of particle tracking on unstructured meshes by requiring only point-containment queries instead of ray-tracing element-to-element. Particle tracking directly on unstructured meshes would greatly simplify the time consuming and sometimes imperfect CSG-to-mesh mapping currently employed in many multiphysics codes. A refactoring of the tracking algorithms in OpenMC to allow for an event-based mode of particle transport provided an elegant approach for incorporating the additional tracking loop, but the real crux of the problem involves an efficient formulation of the majorant cross section. OpenMC currently calculates this cross section from continuous energy point-wise data during initialization based on the set of nuclides present in the problem, including majorant values in the probability tables for the unresolved resonance region. One remaining item to tackle is the ability to calculate the majorant for windowed multipole cross section data across the set of temperature values present in the model.

# Challenges of Developing Open-Source Multiphysics Codes

Throughout the course of work on these various projects, several challenges have arisen. Some of these are related to the various approaches to code design of the solver codes involved and others apply to the maintainability and testing of the code base.

Not all of the individual solver codes in a multiphysics project rely on the same build system (most commonly Make or CMake). This typically results in a selection of one build system for the driver framework with adaptation of other build systems or scripts. This can lead to difficult-to-maintain builds due to small variations in behavior across different versions of the build systems found on HPC systems. Another challenge can be common dependencies that are relied on by upstream solver codes, such as output formatting or file IO libraries. Problems can arise when different versions of these dependencies are required by different solver codes, causing issues with discovery of incorrect versions when building.

Incorporating other solver codes is another challenge to consider. One advantage of relying entirely upon open-source codes is the ability to leverage features of version control such as git submodules or subtrees to ensure that compatible versions of dependencies are available when compiling. In contrast, closed source codes create a scenario similar to an independent build system in which developers must accommodate an independent distribution mechanism for obtaining the code. This can be an additional complexity for software stacks that are conditionally open-source, or can be built with fully open-source or some closed codes depending on the configuration at build time. Even the submodule/subtree approach has its challenges as updates to the dependencies and the multiphysics codebase can result in breaking changes when development is occurring simultaneously in both codes.

Another aspect of open-source codes for multiphysics applications is that dependencies rarely exist purely in support of the multiphysics application. Community-driven projects are well within their right to take the code in directions that are unfavorable for the downstream multiphysics project, add dependencies that complicate the build systems, and/or neglect maintenance and testing of multiphysics interfaces.

The division of responsibility and separation of functionality in testing should also be carefully approached in a multiphysics project. As codes are incorporated into a multiphysics project and interfaces are modified or redesigned in dependencies, the question of where testing occurs comes into question. Enforcing passing multiphysics coupling tests for merged code changes in dependencies creates a hidden barrier for new developers in solver codes and adds resource requirements to continuous integration services. A lack of these tests in the dependency leaves the multiphysics project blind to imminent breaking changes and new bugs. One approach taken in some projects is to create additional CI tests in the downstream multiphysics project which run with the most recent version of each dependency purely to notify developers of the multiphysics project that maintenance in one or more of the dependencies is needed before the next update.

# Conclusion

Generally, the multiphysics projects discussed above have moved from states which support coupling of the salient physics kernels—neutronics and TH—to the positions where additional physics (structural deformation/thermal expansion, mechanical contact, radiative heat transfer, etc.) can be explored. Support for several additional OpenMC features have been provided either in direct support of these projects or in expectation of their future needs. Issues surrounding the maintainability and testing of multiphysics projects were also discussed.

**References**

1. ROMANO, P.K., HORELIK, N.E., HERMAN, B.R., NELSON, A.G., FORGET, B., SMITH, K. OpenMC: A state-of-the-art Monte Carlo code for research and development. Ann. Nucl. Energy **82** (2015) 90–97.
2. ROMANO, P.K. HAMILTON, S.P. RAHAMAN, R.O. NOVAK, A. MERZARI, E. HARPER, S.M SHRIWISE, P. EVANS, T.M., A Code-Agnostic Driver Application for Coupled Neutronics and Thermal-Hydraulics Simulations, Nucl. Sci. Eng. **195** (2021) 391-411.
3. PEPLOW, D.E. BANERJEE, K. DAVIDSON, G.G. STEWART, I.R SWINNEY, M.W WAGNER, J.N. Preliminary Validation of the Shift Monte Carlo Code for Fixed-Source Radiation Transport Problems. Nucl. Tech. **206** (2020) 107-125.
4. NEK5000 Version 19.0 Dec. 28, 2019. Argonne National Laboratory, Illinois. Available: https://nek5000.mcs.anl.gov
5. WELLER H.G., TABOR, G. JASAK, H. FUREBY, C. A Tensorial Approach to Computational Continuum Mechanics Using Object-Oriented Techniques, Comput. Phys. **12** (1998) 620-631.
6. DAVIS, A. BROOKS, H. et. al. AURORA - A Unified Resource for OpenMC Reactor Applications. (2021) github repository: https://github.com/aurora-multiphysics/aurora
7. PERMANN, C.J GASTON, D.R ANDRS, D. CARLSEN, R.W. KONG, F. LINDSAY, A.D. MILLER, J.M. PETERSON, J.W. SLAUGHTER, A.E. STOGNER, R.H. MARTINEAU, R.C. MOOSE: Enabling massively parallel multiphysics simulation. SoftwareX **11** (2020) 100430.
8. WILSON, P.H. TAUTGES, T.J KRAFTCHECK, J.A. SMITH, B.M. HENDERSON, D.L. Acceleration Techniques for the Direct Use of CAD-Based Geometry in Fusion Neutronics Analysis. Fus. Eng. Des. **85** (2010) 1759-1765.
9. TAUTGES, T.J ERNST, C. STIMPSON, C. MEYERS, R.J. MERKLEY, K. MOAB: A Mesh-Oriented Database. Sandia National Laboratory Tech. Rep. (2004).
10. KIRK, B.S. PETERSON, J.W. STOGNER, R.H. CAREY, G.F. libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. Eng. with Comp. **22** (2006) 237-254.
11. TALAMO, A. GOHAR. Y. LEPPÄNEN, J. SERPENT Validation and Optimization with Mesh Adaptive Search on Stereolithography Geometry Models. Ann. Nucl. Energy **115** (2018) 619-632.