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

Preliminary Serpent–MOOSE Coupling and Implementation of Functional Expansion Tallies in Serpent

Leslie Kerby,[*,†] Aaron Tumulak,[†] Jaakko Leppänen,[‡] and Ville Valtavirta[‡]

[*] *Idaho State University, Nuclear Engineering, Pocatello, ID, USA*
[†] *Idaho National Laboratory, Idaho Falls, ID, USA*
[‡] *VTT Technical Research Centre of Finland, Espoo, Finland*
kerblesl@isu.edu

Abstract - *Modeling nuclear reactors is complex, requiring multiphysics solutions between neutronics, thermal hydraulics, and fuel behavior. A preliminary Serpent–MOOSE coupling has been created, with consistent results demonstrated for a single fuel element with simple diffusion. In parallel with this work, Functional Expansion Tallies (FETs) have been implemented in Serpent. These promise to ease mesh conversion for future coupling, as well as reduce error around bin boundaries and decrease memory requirements.*

I. INTRODUCTION

Modeling nuclear reactors is complex, requiring multiphysics solutions between neutronics, thermal hydraulics, and fuel behavior. Monte Carlo methods are becoming more desirable and feasible with advances in parallel computing technology, yet they are still impractical for full-core simulations. However, neutronics simulation could be performed with Monte Carlo codes to obtain accurate fission powers and cross sections, and then these solutions could be coupled with deterministic thermal hydraulic and/or fuel behavior codes.

Serpent 2, a three-dimensional continuous-energy Monte Carlo reactor physics burnup calculation code [1], has been tested and produces accurate cross sections for the Advanced Test Reactor (ATR) and shows potential for use in the Transient Reactor Test Facility (TREAT) [2], both at Idaho National Laboratory (INL). The Multiphysics Object-Oriented Simulation Environment (MOOSE) framework developed at INL provides a high-level interface for solving systems of coupled, nonlinear partial differential equations [3], lending itself useful for modeling multiphysics phenomena found in reactor physics problems. A preliminary Serpent–MOOSE coupling has been created, with consistent results demonstrated for a single fuel element with simple diffusion. In parallel with this work, Functional Expansion Tallies (FETs) have been implemented in Serpent. These promise to ease mesh conversion for future Serpent–BISON coupling, as well as reduce error around bin boundaries and decrease memory requirements.

II. COUPLING OF SERPENT AND MOOSE

A parallelizable, coupled Serpent 2.1.26–MOOSE code has been created and tested successfully on a single fuel element.

1. Implementation

A. UserObjects

The coupled code utilizes three MOOSE UserObjects: ElementTransfer, RunSerpent, and HeatToMoose. Figure 1 illustrates the flow of the coupled Serpent–MOOSE code. The UserObject HeatToMoose transfers the Serpent fission heat generated, per element volume, to the MOOSE mesh. It reads

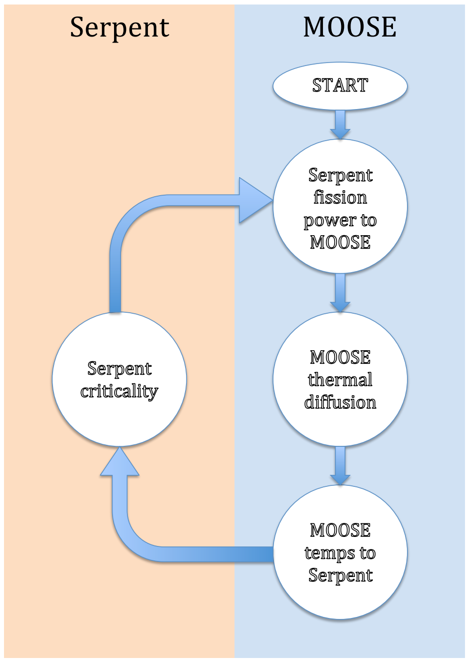

Fig. 1. Flow of the coupled Serpent–MOOSE code.

the power production from the multiphysics interface output file produced by Serpent into an array and transforms it into the MOOSE 3D mesh. This array can later be accessed by MOOSE to calculate heat production in a certain element. An initial guess for the fission power generation, which can be accomplished by running standalone Serpent, must be supplied. MOOSE then runs a heat conduction solution given the fission heat distribution dictated by Serpent. The UserObject ElementTransfer averages the MOOSE temperature solution field for each element, transforms this to an OpenFOAM mesh, and transfers this to the Serpent interface input file. Next, the UserObject RunSerpent runs either a full Serpent calculation, or if Serpent has already been run once and does not need to be initialized, a shorter transport cycle. The initial guess for fission heat is then overwritten by the coupled Serpent, and the process starts over again, iterating the number of times specified in the MOOSE input file.

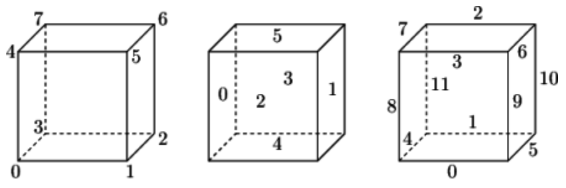

Fig. 2. OpenFOAM hexahedron vertices, faces, and edge numbering.

Fuel		Water	
Isotope	% mass	Isotope	%mass
^{235}U	2.9971	^{1}H	66.6667
^{238}U	85.153	^{16}O	33.3333
^{16}O	11.85		
5 *cm* x 5 *cm*		10 *cm* x 10 *cm*	

TABLE I. Fuel element properties.

B. Meshes

The current impementation utilizes an OpenFOAM mesh on the Serpent side. The OpenFOAM mesh uses a *polyMesh* object, which has five attributes: points, faces, owners, neighbors, and boundaries. These are separated into four separate files (boundaries are not used) in the Serpent multiphysics interface. The points file contains a list of vectors describing the cell vertices, where the first vector in the list represents vertex 0, the second vector represents vertex 1, etc. The faces file contains a list of faces, each face being a list of indices to vertices in the points list, where again, the first entry in the list represents face 0, etc. The owner file is a list of owner cell labels, the index of entry relating directly to the index of the face, so that the first entry in the list is the owner label for face 0, the second entry is the owner label for face 1, etc. And lastly, the neighbor file contains a list of neighbor cell labels. Fig. 2 shows an example of a hexahedron in the *cellShape* class, which is the specific type of *polyMesh* used on the Serpent side in this coupling.

In the present coupled code, MOOSE uses a *GeneratedMesh* with dim=3 and elem_type=PRISM6, or a rectangular-prism (ie, "box") mesh.

C. Code Modifications

Several modifications to the Serpent 2 code were necessary to create this coupled version. They are not detailed here for brevity, but may be found in [4]. In testing the single fuel element case we discovered a small bug in the TMS method, which was not apparent in the standalone Serpent 2. The fix for this was fairly simple. The corrections are implemented in Serpent 2.1.27.

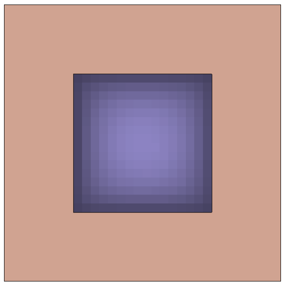

Fig. 3. Geometry of single fuel element.

2. Results

The coupled Serpent 2.1.26–MOOSE was tested with a single fuel element surrounded by water, as shown in Fig. 3 and detailed in Table I. Results for k_{eff} for standalone Serpent 2.1.26 and the coupled Serpent 2.1.26–MOOSE are displayed in Table II. The standalone was run with 50 inactive cycles and 1000 active cycles of 100,000 neutrons/cycle, and with OpenMP with 6 threads. The coupled Serpent–MOOSE was run with 50 inactive cycles and 500 active cycles of 1,000,000 neutrons, for 5 iterations of thermal feedback; only results from the final iteration are shown. The statistical error only accounts for the final iteration, and not the entire coupled calculation.

The analog and implicit k_{eff} calculated from the standalone Serpent and coupled Serpent–MOOSE are very similar and within a standard deviation of each other.

III. IMPLEMENTATION OF FETS

This research builds upon and utilizes work performed by OpenMC [5] in implementing Functional Expansion Tallies (FETs). OpenMC, a Monte Carlo particle transport simulation code focused on neutron criticality calculations, uses Functional Expansion Tallies (FETs) to allow for a more efficient passing of multiphysics data between OpenMC and MOOSE [6].

One of the pre-eminent issues in coupling two codes is accurately and efficiently transferring between their different meshes. The current Serpent–MOOSE coupling uses the unstructured OpenFOAM mesh on the Serpent side, and the structured hexahedral mesh on the MOOSE side. Implementing Functional Expansion Tallies (FETs) in Serpent enables us to have a mesh-free fission power distribution in Serpent which can more easily be transferred to any desired mesh within MOOSE (and eventually BISON and MAMMOTH).

An implementation of functional expansion tallies (FETs) to represent temperature, density, and local power in a single 3-D fuel pin is implemented in Serpent 2. Preliminary results show that the method is feasible and produces qualitatively

	Standalone Serpent 2.1.26	Coupled Serpent–MOOSE
k_{eff} (analog)	0.20577 +/- 0.00007	0.20573 +/- 0.00003
k_{eff} (implicit)	0.20578 +/- 0.00005	0.20573 +/- 0.00002

TABLE II. Fuel element k_{eff} values for the standalone Serpent and coupled Serpent–MOOSE.

acceptable results.

1. Theory

Fuel pins are usually cylindrical. In cylindrical geometry, a scalar-valued function $f(r,\theta,z)$ can be expanded as the sum of the product of Legendre and Zernike polynomials

$$f(r,\theta,z) = \sum_i \sum_j c_{ij} Z_j(r,\theta) P_i(z) \tag{1}$$

where the Zernike polynomials $Z_j = Z_n^m$ are defined for even $n-m$ and $n \geq m$ as

$$\begin{aligned} Z_n^m(r,\theta) &= \begin{cases} \sqrt{2(n+1)}R_n^m(r)\cos(m\theta) & \text{for } m > 0 \\ \sqrt{2(n+1)}R_n^{-m}(r)\sin(-m\theta) & \text{for } m < 0 \\ \sqrt{n+1}R_n^0(r) & \text{for } m = 0 \end{cases} \\ R_n^m(r) &= \sum_{k=0}^{\frac{n-m}{2}} (-1)^k \binom{n-k}{k} \binom{n-2k}{\frac{n-m}{2}-k} r^{n-2k} \end{aligned} \tag{2}$$

where the second and third factors in parenthesis are binomial coefficients.

For brevity in notation and convenience in code implementation, the radial and radial indices n and m are mapped to a single index j using Noll's indexing. The rules are as follows:

1. The first entry $(n = 0, m = 0)$ is $j = 1$.
2. (n,m) with greater n have greater j.
3. (n,m) with $m < 0$ have odd-numbered j.
4. (n,m) with $m > 0$ have even-numbered j.
5. Within a given n, (n,m) with greater $|m|$ have greater j.

The Legendre polynomials P_i are defined for integers $i \geq 0$ as

$$P_i(z) = \sqrt{\frac{2i+1}{2}} \sum_{k=0}^{i} \binom{i}{k} \binom{-i-1}{k} \left(\frac{1-z}{2}\right)^k \tag{3}$$

where the first two factors in parenthesis are binomial coefficients and the last factor is a real number.

From the orthogonality of Zernike and Legendre polynomials, the product $Z_j(r,\theta)P_i(z)$ also satisfies

$$\begin{aligned} \int_{-1}^{1} dz \int_0^1 dr \int_0^{2\pi} d\theta \Big(Z_j(r,\theta)P_i(z)\Big)\Big(Z_{j'}(r,\theta)P_{i'}(z)\Big) \\ = \delta_{i,i'}\delta_{j,j'} \end{aligned} \tag{4}$$

where δ is the Kronecker delta function. Note that the forms of the polynomials defined in Eqs. [2] and [3] are normalized so that their inner products are one. The constants c_{ij} can then be defined as

$$c_{ij} = \int_{-1}^{1} dz \int_0^1 dr \int_0^{2\pi} d\theta\, f(r,\theta,z) Z_j(r,\theta) P_i(z). \tag{5}$$

For the purpose of tallying a score E with weight w at position (r,θ,z) using functional expansion tallies, the tally for coefficient c_{ij} can be incremented by a value

$$Z_j(r,\theta)P_i(z)Ew, \tag{6}$$

taking care to also increase the total weight W by an amount w.

Details of the implementation of FETs into Serpent can be provided in the full paper.

2. Results

To quantitatively measure the accuracy of functional expansion tallies in resolving the spatial distribution of the fission power, a benchmark PWR fuel assembly was developed. A radially infinite, axially finite assembly of PWR fuel pins 366 cm in height was used to examine how an axially linear density and temperature distribution in the coolant would affect the fission power distrubtion in the fuel.

Parameter	**Value**
Fuel	UO_2 (4.5% ^{235}U)
Clad	Zircalloy
Coolant	H_2O
Fuel Outer Radius	$4.095\,75 \times 10^{-1}$ cm
Void Outer Radius	$4.178\,30 \times 10^{-1}$ cm
Clad Outer Radius	$4.749\,80 \times 10^{-1}$ cm
Assembly Pitch	1.259 84 cm
Active Fuel Height	366 cm
Fuel Nominal Density	10.424 g cm^{-3}
Coolant Inlet Density	0.742 76 g cm^{-3}
Coolant Outlet Density	0.664 52 g cm^{-3}
Coolant Inlet Temperature	291.9 °C
Coolant Outlet Temperature	325.8 °C

TABLE III. Geometric and material parameters of fuel assembly. Note that the coolant temperature and density is linearly interpolated between the inlet and outlet values.

To compare results, the fuel pin was partitioned into 10 equally-spaced axial zones and 20 radial zones. The first five innermost radial zones have the same volume and the last fifteen outermost zones have the same (smaller) volume. This was done since the change in neutron flux tends to change significantly near the outer edge of the fuel pin. With 10 axial zones and 20 radial zones, the neutron neutron fission power in 200 zones was measured using Serpent detector cards. The functional expansion tally was carried out to 5th order Legendre and 4th order Zernike polynomials.

The fission power density was plotted along one-dimensional cross sections of the fuel pin. Figure [4] shows the axial fission power density along the centerline of the fuel pin. The fission power reaches a maximum closer to the coolant inlet ($z = -1.0$) due to increased moderation from denser water.

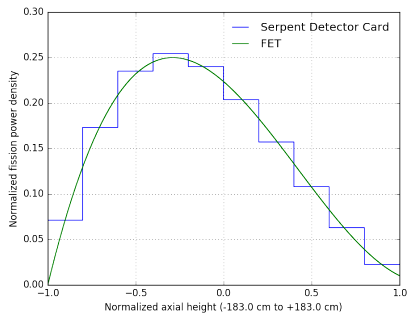

Fig. 4. Axial distribution of fission power density along centerline ($r = 0$).

Closer to the edge of the fuel pin there is a significant increase in fission power density due to the fuel's proximity to the moderator (this figure can be shown in the full paper).

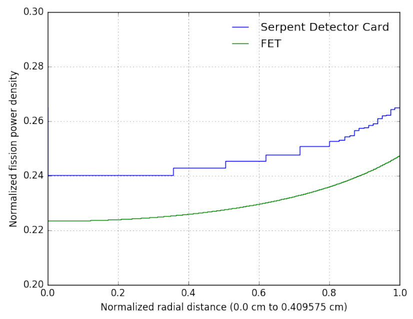

Fig. 5. Radial distribution of fission power density at axial point just below midpoint ($r \approx 0$).

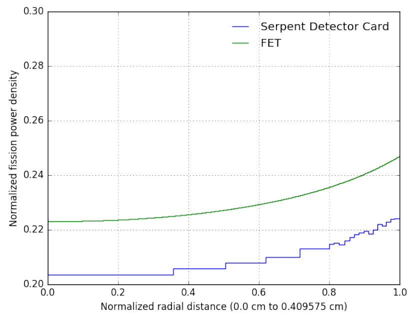

Fig. 6. Radial distribution of fission power density at axial point just above midpoint ($r \approx 0$).

Figures [5] and [6] show the radial fission power density along different axial cross sections of the fuel pin. They demonstrate the discontinuity of the traditional spatial tallying methods across the axial centerline of the fuel pin. The FET remains continuous across this boundary, but the traditional tallying results in significant overestimation of the radial fission power density (compared to the FET) immediately before the discontinuity (Fig. [5]) and an underestimation immediately after (Fig. [6]). Since the FET appears to match with the midpoints of the detectors, this indicates that smaller detector sizes would agree with FET results.

IV. CONCLUSION

A parallelizable coupled Serpent–MOOSE code has been created and Functional Expansion Tallies (FETs) have been implemented within Serpent. The coupled Serpent–MOOSE code simulated a slightly lower k_{eff} for the single fuel pin case than the standalone Serpent. At this point it is unclear if this is due to thermal feedback of the rising fuel temperatures or to statistical variance. The FETs implemented within Serpent provide a smooth, continuous function which has many advantages over the discontinuous tally system it replaces. The FETs were accurate representations of fission power in PWR fuel pin geometries. Both methods show promise of usefulness in reactor analysis at INL.

Future planned work includes implementing FETs into a coupled Serpent–BISON.

V. ACKNOWLEDGMENTS

Thank you to Javier Ortensi, Yaqi Wang, Frederick Gleicher, Matthew Ellis, Tuomas Viitanen, Cody Permann, Andrew Slaughter, John Peterson, and Benoit Forget, all of whom provided expert direction and discussion. This study was carried out under the auspices of the U.S. Department of Energy at Idaho National Laboratory under Contract No. DE- AC07-05ID14517.

REFERENCES

1. J. LEPPÄNEN, M. PUSA, T. VIITANEN, V. VALTAVIRTA, and T. KALTIAISENAHO, "The Serpent Monte Carlo code: Status, development and applications in 2013," *Ann. Nucl. Energy*, **82**, 142–150 (2015).
2. J. ORTENSI, M. D. DEHART, F. N. GLEICHER, Y. WANG, S. SCHUNERT, A. L. ALBERTI, and T. S. PALMER, "Full Core TREAT Kinetics Demonstration Using Rattlesnake/BISON Coupling Within MAMMOTH," INL Report, INL/EXT-15-36268 (2015).
3. D. GASTON, C. NEWMAN, G. HANSEN, and D. LEBRUN-GRANDIE, "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nucl. Engrg. Design*, **239**, 1768–1778 (2009).
4. L. KERBY, M. DEHART, and A. TUMULAK, "Integration of OpenMC methods into MAMMOTH and Serpent," INL Report, INL/EXT-16-39874 (2016).
5. P. K. ROMANO, N. E. HORELIK, B. R. HERMAN, A. G. NELSON, B. FORGET, and K. SMITH, "OpenMC: A State-of-the-Art Monte Carlo Code for Research and Development," *Ann. Nucl. Energy*, **82**, 90–97 (2015).
6. M. ELLIS, B. FORGET, and K. SMITH, "Continuous Temperature Representation in Coupled OpenMC/MOOSE Simulations," PHYSOR 2016, Unifying Theory and Experiments in the 21st Century, Sun Valley, Idaho, May 1–5, 2016.