INL/EXT-17-42404

Transient Simulation of the Multi-SERTTA Experiment with MAMMOTH

Javier Ortensi[1], Benjamin Baker[1], Yaqi Wang[2], Sebastian Schunert[2], Mark DeHart[1]

[1]Reactor Physics Design and Analysis
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3840

[2]Nuclear Engineering Methods and Development
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3840

June 2017

INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance

INL/EXT-17-42404

Transient Simulation of the Multi-SERTTA Experiment with MAMMOTH

Javier Ortensi, Benjamin Baker, Yaqi Wang, Sebastian Schunert, Mark DeHart

June 2017

Idaho National Laboratory
Nuclear Systems Design and Analysis
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Contents

Abstract

This work details the MAMMOTH reactor physics simulations of the Static Environment Rodlet Transient Test Apparatus (SERTTA) conducted at Idaho National Laboratory in FY-2017. TREAT static-environment experiment vehicles are being developed to enable transient testing of Pressurized Water Reactor (PWR) type fuel specimens, including fuel concepts with enhanced accident tolerance (Accident Tolerant Fuels, ATF). The MAMMOTH simulations include point reactor kinetics as well as spatial dynamics for a temperature-limited transient. The strongly coupled multi-physics solutions of the neutron flux and temperature fields are second order accurate both in the spatial and temporal domains. MAMMOTH produces pellet stack powers that are within 1.5% of the Monte Carlo reference solutions. Some discrepancies between the MCNP model used in the design of the flux collars and the Serpent/MAMMOTH models lead to higher power and energy deposition values in Multi-SERTTA unit 1. The TREAT core results compare well with the safety case computed with point reactor kinetics in RELAP5-3D. The reactor period is 44 msec, which corresponds to a reactivity insertion of 2.685% $\Delta k/k$. The peak core power in the spatial dynamics simulation is 431 MW, which the point kinetics model over-predicts by 12%. The pulse width at half the maximum power is 0.177 sec. Subtle transient effects are apparent at the beginning of the reactivity insertion in the experimental samples due to the control rod removal. Additional differences due to transient effects are observed in the sample powers and enthalpy. The time dependence of the power coupling factor (PCF) is calculated for the various fuel stacks of the Multi-SERTTA vehicle. Sample temperatures in excess of 3100 K, the melting point UO_2, are computed with the adiabatic heat transfer model. The planned shaped-transient might introduce additional effects that cannot be predicted with PRK models. Future modeling will be focused on the shaped-transient by improving the control rod models in MAMMOTH and adding the BISON thermo-elastic models and thermal-fluids heat transfer.

List of Figures

List of Tables

1 Introduction

TREAT static-environment experiment vehicles are being developed to enable transient testing of Pressurized Water Reactor (PWR) type fuel specimens, including fuel concepts with enhanced accident tolerance (Accident Tolerant Fuels, ATF), as well as other types of compatible specimens. These types of experiment vehicles have been termed Static Environment Rodlet Transient Test Apparatus (SERTTA). The first of these SERTTA devices has been designed in detail by the INL transient experiment team and is currently undergoing final prototype testing and design qualification for use in TREAT. This design can accommodate four rodlets, each within its own hermetic boundary, and is known as the Multi-SERTTA. The Multi-SERTTA irradiation vehicle provides support for specimens and instrumentation, affords desired boundary conditions, and safely contains the hazards associated with transient testing of nuclear fuel.

The objectives of the test require the precise energy deposition in the targets to demonstrate that ATF fuel designs can withstand current regulatory limits. Each of the four SERTTA vessels is shrouded with neutron-absorbing flux collars to tailor the reactors axial flux profile and enforce uniform specimen PCFs in each specimen. Leveling the Power Coupling Factors (PCFs) across the core axial profile in this manner has not been attempted historically in TREAT. The core energy release is currently being predicted by point kinetics using historic temperature feedback tables. The core-to-specimen PCF values are predicted during steady state solutions using the MCNP5 Monte Carlo code [1]. Point reactor kinetics (PRK) predictions can provide reasonable estimates of the core average behavior with respect to time, but lack the ability to predict 3-D spectral or spatial shifts. Estimates of 3-D behavior are particularly important for the Multi-SERTTAs' intricate design, since none of its geometry can be adequately represented by simplified models (e.g. axisymmetric, two-dimensional approximations, etc.). Therefore, Monte Carlo methods are used to predict 3-D steady state behavior, but lack the ability for full coupling to implicitly predict time-dependent effects.

The Multi-SERTTA provides a significant challenge/opportunity for advanced modeling and simulation. This work initiates the study of the behavior of the experimental samples during the transient. Both the energy deposition rates and accurate fuel temperatures are necessary to accurately predict the performance of the test fuel. Comparisons of the performance of the PRK and the full spatial dynamics models should unveil any transient effects. A temperature-limited transient with a 2.6% $\Delta k/k$ reactivity insertion is included in this work due to time constraints and the need to improve the current control rod models.

2 Methods

The neutron cross sections were prepared with Serpent [2] and are based on ENDF/B-VII.1 data. Serpent is a three-dimensional continuous-energy Monte Carlo reactor physics code developed at VTT Technical Research Centre of Finland. Serpent was selected for this project because it offers spatial homogenization and multi-group constant generation for deterministic reactor simulations from a Monte Carlo simulation. At the same time Serpent provides a detailed reference calculation without energy, angular, or spatial discretization error.

The MAMMOTH [3] reactor multi-physics analysis application is being developed as the primary tool for M&S of TREAT at Idaho National Laboratory. MAMMOTH is based on the MOOSE framework [4], a finite element method development environment that focuses on multi-physics simulations with strong or tight coupled physics applications. MAMMOTH is in fact a control application that inherently and seamlessly interfaces with several other MOOSE applications including Rattlesnake [5] for solving the Boltzmann transport equation, BISON [6] for heat transfer and fuel performance modeling and RELAP-7 [7] for thermal fluids calculations. The Rattlesnake neutron transport solver incorporates a variety of spatial and angular discretizations including diffusion, P_N and S_N (both 1^{st} and 2^{nd} formulations). In this report the primary solvers used are the first order formulation of the S_N equation and diffusion.

Currently Serpent does not include the capability to generate anisotropic diffusion coefficients to allow better modeling of neutron streaming effects with a diffusion solver. A Tensor Diffusion Coefficient (TDC) calculation method was implemented in the Rattlesnake transport solver [8, 9] to address the streaming effects through air regions in the slotted elements and the hodoscope penetration. In addition, an SPH equivalence procedure [9, 10] is employed to ensure preservation of the reaction rates between the reference Monte Carlo model and the cross section set used in the MAMMOTH models. The final data preparation sequence presented in Figure 1 entails the generation of various cross section tabulations with Serpent, followed by the calculation of anisotropic diffusion coefficients and, finally, the equivalence correction of the cross sections with SPH. Rattlesnake automatically generates a new tabulation after each step without the need to use multiple datasets or other cumbersome data processing.

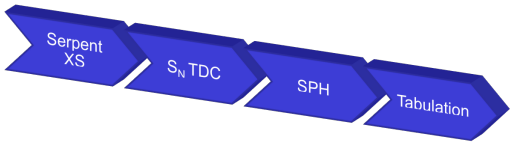


Figure 1: Cross section preparation sequence for TREAT.

3 Models

The Multi-SERTTA experiment adds a level of complexity that is unprecedented in TREAT modeling at Idaho National Laboratory. The level of resolution has to be increased and optimized in order to obtain a viable simulation that produces meaningful results. The MAMMOTH modeling process entails three fundamental tasks: 1) the preparation of cross sections, 2) the development of the analysis mesh and 3) the assembly of the MAMMOTH input. The preparation of the cross sections is introduced in Section 3.1. The improvements in mesh generation are presented in Section 3.2. Additional physics models and the description of other MAMMOTH inputs for this analysis are discussed Section 3.3.

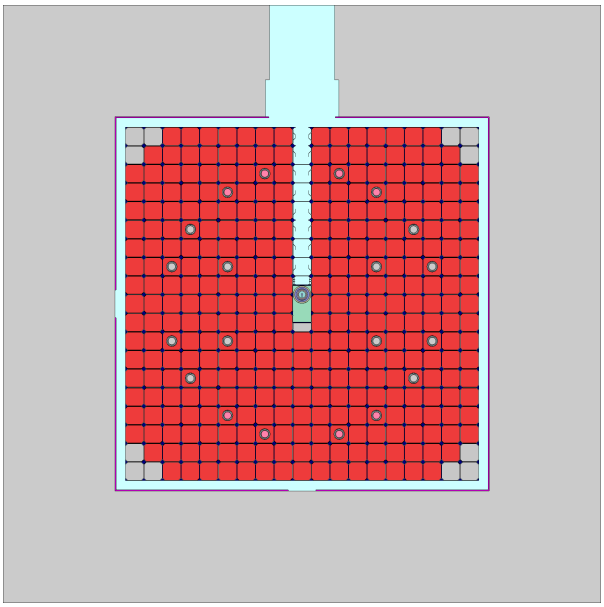
3.1 Cross Section Preparation

The Serpent cross sections are spatially-homogenized over various material regions and energy-condensed to 11 energy groups. This spatial homogenization comprises various fuel elements in the 19x19 array of the active core and it is mainly driven by the equivalence procedure. The various homogenized regions in the 19x19 array are depicted in Figure 2. Each color represents one separate cross section library for the element. The elements in the 19x19 array contain 13 to 19 axial zones. The radial reflector includes 3 evenly distributed radial regions and 4 axial regions bounded in incremental order by: bottom of geometry, bottom of active core, transient control rod position, top of active core and top of geometry.

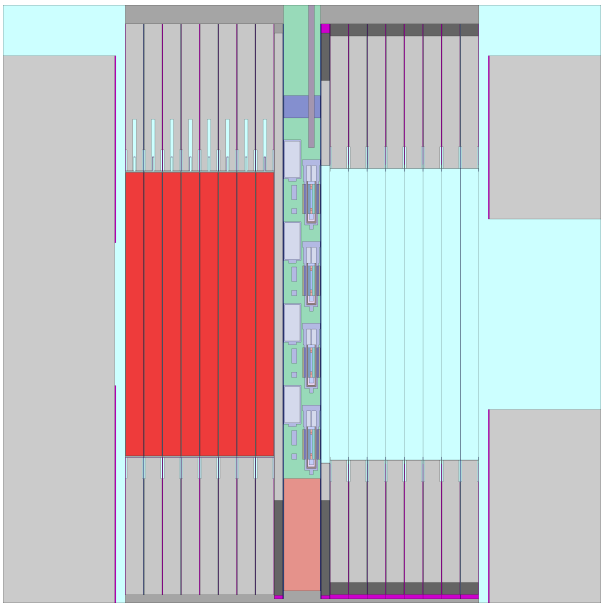
The Multi-SERTTA vehicle was added to the full core Serpent model as shown in Figure 3. The TREAT core cross sections are tabulated as a function of core average fuel temperature and control rod position. The fuel temperature points include 300, 400, 500 and 600 K, whereas the two control rod configurations are the pre and post-transient control rod positions. To generate the cross sections for the experiment regions a set of Monte Carlo source files are written at the experiment vehicle surface in each of these full core runs. These source files are then used in Monte Carlo source calculations for the experiment region in isolation to speed-up the computation of the cross sections. The MSERTTA cross sections are tabulated as a function of fuel pellet average temperature, control rod position and core average fuel temperature. The fuel pellet tabulations include average temperature points from 553.15 to 2500 K at every 200 K.

6000	6000	12800	12800	12800	12800	13400	12500	11700	5450	11700	12500	13400	12800	12800	12800	12800	6000	6000
6000	12800	12900	12900	12900	12900	11200	13500	11500	5400	11500	13500	11200	12900	12900	12900	12900	12800	6000
12800	12900	12900	12900	11200	11200	11200	1100	11500	5350	11500	1100	11200	11200	11200	12900	12900	12900	12800
12800	12900	12900	12900	11200	1200	11200	13600	11500	5300	11500	13600	11200	1200	11200	12900	12900	12900	12800
12800	12900	11300	11300	11200	11200	11200	13700	11600	5250	11600	13700	11200	11200	11200	11300	11300	12900	12800
12800	12900	11300	1700	11300	12300	12300	13700	11600	5200	11600	13700	12300	12300	11300	1700	11300	12900	12800
12800	11300	11300	11300	11300	11900	11900	13800	14600	5150	14600	13800	11900	11900	11300	11300	11300	11300	12800
12800	11300	1800	11300	11900	1500	11900	13800	14500	5100	14500	13800	11900	1500	11900	11300	1800	11300	12800
12800	11300	11300	11300	11900	11900	11900	13900	14400	5900	14400	13900	11900	11900	11900	11300	11300	11300	12800
12800	13100	13100	13000	13000	13000	13000	13900	14300	9900	14300	13900	13000	13000	13000	13000	13100	13100	12800
12700	11400	11400	11400	12000	12000	12000	12100	14200	6100	14200	12100	12000	12000	12000	11400	11400	11400	12700
12700	11400	2000	11400	12000	1600	12000	12100	14100	14000	14100	12100	12000	1600	12000	11400	2000	11400	12700
12700	11400	11400	11400	11400	12000	12000	12100	12100	12100	12100	12100	12000	12000	11400	11400	11400	11400	12700
12700	12400	11400	1900	11400	12200	12200	12100	12100	12100	12100	12100	12200	12200	11400	1900	11400	12400	12700
12700	12400	11400	11400	11100	11100	11100	12100	12100	12100	12100	12100	11100	11100	11100	11400	11400	12400	12700
12700	12400	12400	12400	11100	1400	11100	11100	11100	11800	11100	11100	11100	1400	11100	12400	12400	12400	12700
12700	12400	12400	12400	11100	11100	11100	1300	11100	11800	11100	1300	11100	11100	11100	12400	12400	12400	12700
6000	12700	12400	12400	12400	12400	11100	11100	11100	11800	11100	11100	11100	12400	12400	12400	12400	12700	6000
6000	6000	12700	12700	12700	12700	12600	12600	12600	12600	12600	12600	12600	12700	12700	12700	12700	6000	6000

Figure 2: Serpent radial cross section region assignment in the 19x19 TREAT array.



(a) XY view



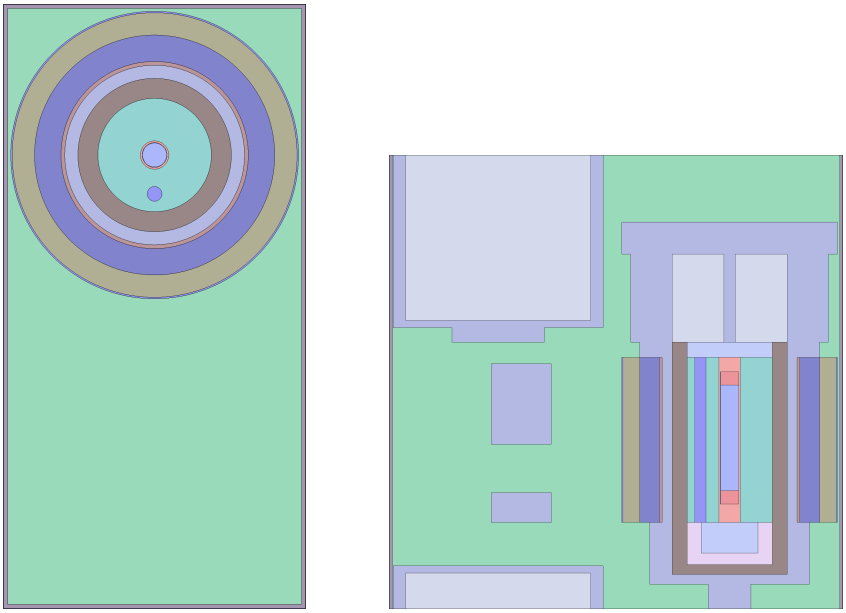
(b) YZ view

Figure 3: Full Core Serpent Model with the Multi-SERTTA Vehicle.

Several geometric features are simplified in the MSERTTA experiment to expedite the modeling. Those include:

- the approximation of the vehicle walls with a rectangular shape,
- the approximation of the expansion tank with a rectangular cylinder,
- the omission of the micro fission chamber and horse shoes,
- the omission of additional small metal clips, screws, light pipes, thermocouples, wires, etc.,
- the homogenization of the support rodlet with the water volume in the lower (south) azimuthal sector, and
- the atom number density in the water is adjusted to preserve the total number of atoms.

The cross section regions defined in each SERTTA unit include various concentric rings to model the pin, clad, water, pressure vessel internals as well as neutron absorption collars and external heater. Views of the simplified Serpent model are shown in Figure 4. Two azimuthal divisions were added to the pin and moderator in order to better capture the expected flux asymmetry introduced by the slotted elements facing the experiment.



(a) XY view **(b)** YZ view

Figure 4: Serpent Model of the SERTTA Unit

3.2 Mesh Development

The mesh development for TREAT geometries is accomplished through the Cubit [11] application interface with a Python toolkit developed at INL. The python scripts were updated to allow the use of an unstructured mesh in the reflector regions and to add the hodoscope and detector penetrations. To better model the SERTTA unit, the python scripts now allow multiple concentric and non-concentric cylinders as well as the ability to generate azimuthal regions in fuel pins. MAMMOTH employs an automatic mapping of cross sections using a material identification variable that is directly written into the mesh file.

The mesh generated for the full core is shown in Figure 5. The TREAT fuel, control rod, dummy, and slotted elements are meshed with an extruded, structured mesh. An unstructured mesh is employed in the radial reflector, hodoscope hole and experiment vehicle. There are 943,776 elements in the mesh with 826,149 nodes.

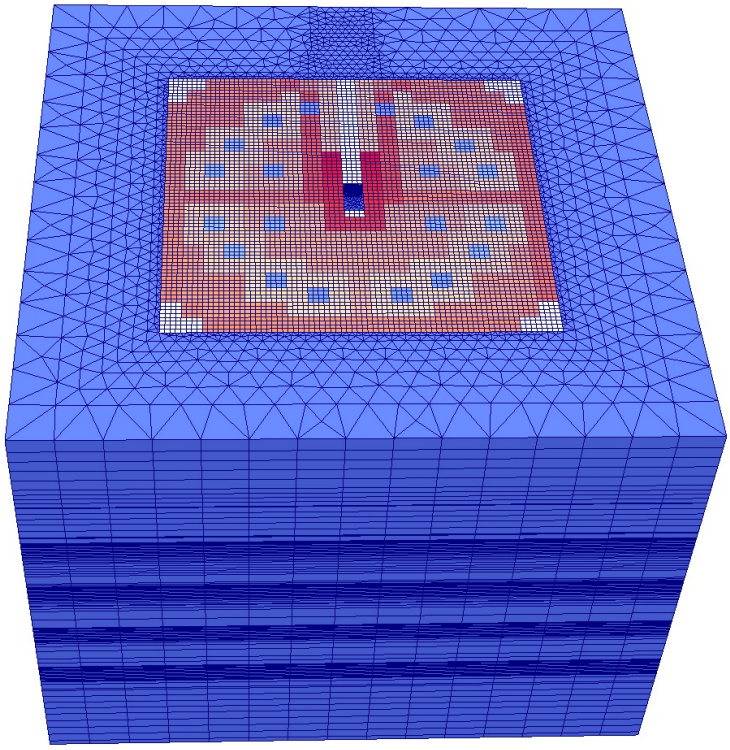


Figure 5: TREAT core mesh

Additional detailes of the mesh for the SERTTA unit and the fuel pellet are shown in Figures 6 and 7, respectively. The SERTTA unit is comprised of the various extruded concentric rings to model the pin, clad, water, pressure vessel internals, neutron absorption collars, and heater. The neutron absorption collars can be observed in the finer mesh located radially outwards from the central pin. The azimuthal regions in the pellet are shown in Figure 7.

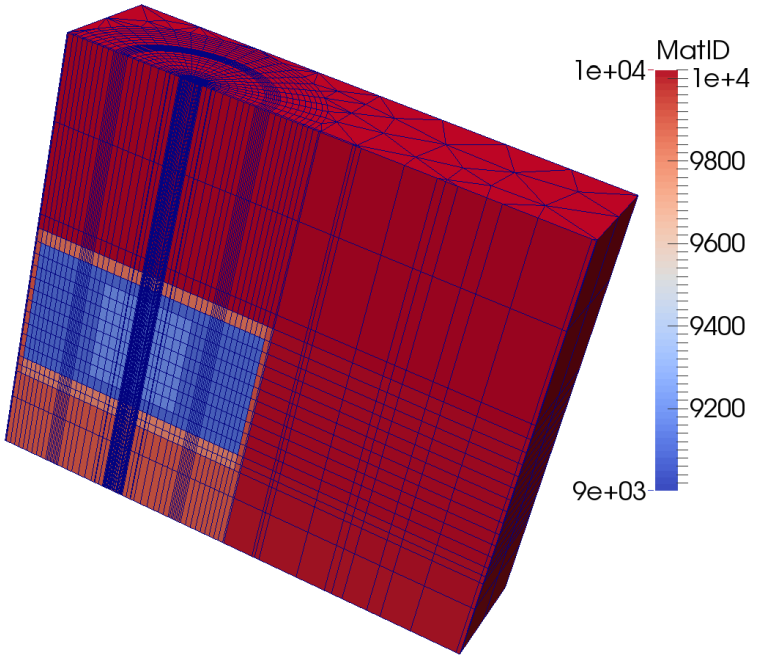


Figure 6: Mesh of the SERTTA unit

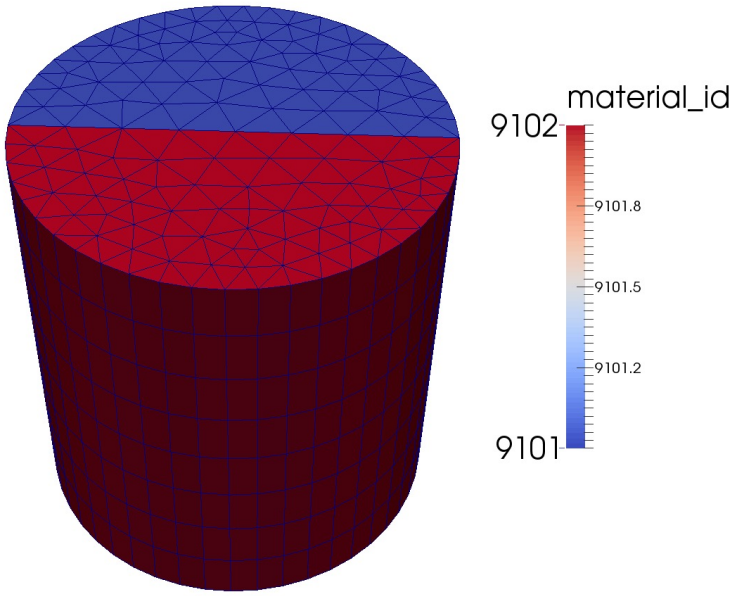


Figure 7: Enriched fuel pellet with two azimuthal regions

3.3 MAMMOTH model

The MAMMOTH input incorporates the neutron and thermal physics models used in the calculations. The details for the PRK and SD neutronics model are discussed in Section 3.3.1, whereas the thermal model is included in Section 3.3.2. All of the physics are solved in a single system of strongly coupled equations. All primal variables, group fluxes and temperature, are expanded with first order Lagrange shape functions yielding second order spatially accurate solutions. The BDF2 time integration scheme ensures second order accuracy in the temporal domain.

3.3.1 Neutronics model

The MAMMOTH point reactor kinetics (PRK) model parameters are based on the Serpent runs and included in Table 1. The feedback reactivity model is functionalized in Eq. 1. Note that this PRK nomenclature is used interchangeably with Point Kinetics Equation (PKE).

Table 1: PRK parameters.

group	λ [sec^{-1}]	β [unitless]
1	1.3336E-02	2.4124E-04
2	3.2739E-02	1.2406E-03
3	1.2078E-01	1.1861E-03
4	3.0278E-01	2.6451E-03
5	8.4949E-01	1.0871E-03
6	2.8530E+00	4.5556E-04

$\Gamma = 8.9767x10^{-4}$ seconds

$$reactivity_{feedback} = -5.279189x10^{-10}T^2 + 6.728639x10^{-7}T - 4.086887x10^{-4} \quad (1)$$

where $reactivity_{feedback}$ is in $\Delta\rho$ and T is in Kelvin.

The spatial dynamics model employs the Rattlesnake diffusion solver with 11 energy groups, 6 delayed neutron groups and tensor diffusion coefficients. Vacuum boundary conditions are imposed on all boundaries. The initial conditions for the core are based on the steady state calculation without feedback with a core thermal power at 1000W. The cross section interpolation of the experiment is based on the temperature of surrounding fuel elements within 40 cm of each Multi-SERTTA unit and the control rod position. The reflectors and other non-heated materials in the core depend on the core average temperature and control rod position. The control rod model uses a piece-wise linear function to effectively simulate the ejection of the transient control rods from the core between 0 to 0.1156 seconds.

3.3.2 Thermal Model

An adiabatic temperature model for both the TREAT fuel and the experiment region use the basic capabilities from BISON. The initial conditions are at 553.15 K for the internals of each SERTTA unit and 293.6 K for the TREAT fuel elements. The density and heat capacity of the graphite-urania fuel are based on the values in the TREAT Design Summary Report [12], Appendix B. The density is assumed to be 1.72 g/cc and the temperature dependence

of the heat capacity in J/g/K is modeled with Equation 2.

$$C_p(T) = -5.8219x10^{-10}T^3 - 4.3694x10^{-7}T^2 + 2.8369x10^{-3}T - 1.009x10^{-2} \quad (2)$$

The thermo-physical properties for the test fuel are based on [13] for unirradiated values with a density of 10.4215 g/cc and the heat capacity in J/g/K

$$C_p(t) = 0.193218 + 0.325711t - 0.311972t^2 + 0.116810t^3 - 9.75233x10^{-3}t^4 - 2.64384x10^{-3}t^{-2} \quad (3)$$

where t = T/1000 with T in K.

4 Results

4.1 Steady State Calculations

The Serpent and MAMMOTH eigenvalues for the various TREAT fuel temperatures and transient control rod positions used in the cross section tabulation are compared in Table 2. The TDC-SPH corrected diffusion results show significant differences in the multiplication factor compared to Serpent ($\sim$2000 pcm), but it is important to note that the results without the corrections are in the 6,000 to 10,000 pcm range. The multiplication factor is very sensitive to the diffusion coefficients in the void regions in the active core liner, between the 19x19 array and the radial reflector, and in the hodoscope hole. As an example a model with larger (50-70%) tensor diffusion coefficients in these regions is also included for comparison. The relevant integral reaction rates that affect the eigenvalue are shown in Table 3 at 293.6K. These indicate that most of the eigenvalue differences arise form the poor predictions of the source rates. Leakage rates are less important for these calculations, since their magnitude is $\sim$7 times lower. The table also shows that the TDC-SPH corrections provide a significant improvement to the reaction rates.

Table 2: Full core TREAT eigenvalues calculated with MAMMOTH.

Temperature $[K]$	CR	Serpent	TDC-SPH Diffusion	TDC-SPH Diffusion*	pcm difference	pcm difference*
293.6	in	0.98779	1.00686	0.99140	1930.5	365.0
293.6	out	1.01309	1.03336	1.01443	2000.4	132.6
400.0	out	0.98925	1.01000	0.99051	2097.3	127.5
500.0	out	0.96886	0.98979	0.96601	2160.4	-294.2
600.0	out	0.95059	0.97550	0.95144	2620.8	89.1

* with modified diffusion coefficients in the active core liner and hodoscope hole.

Table 3: Comparison of integral reaction rates between Serpent and MAMMOTH (293.6 K).

	Source rate	Absorption rate	Leakage rate	Integral flux
Serpent	8.4602E+16	7.3712E+16	1.0892E+16	5.3900E+19
Diffusion	8.9687E+16	7.5645E+16	1.4041E+16	5.5245E+19
TDC-SPH	8.3007E+16	7.3840E+16	9.1664E+15	5.3852E+19
TDC-SPH*	8.4864E+16	7.3704E+16	1.1160E+16	5.3911E+19
Diffusion	6.01%	2.62%	28.92%	2.50%
TDC-SPH	-1.89%	0.17%	-15.84%	-0.09%
TDC-SPH*	-0.37%	0.04%	-3.14%	-0.02%

* with modified diffusion coefficients in the active core liner and hodoscope hole.

The effects of the active core liner and the hodoscope hole are less significant on the power distribution. Several Figures of Merit (FoMs) are compared in Table 4 and show the dramatic improvement in the results upon the application of the TDC and SPH corrections. The root mean squared (rms) improves by an order of magnitude when TDC-SPH is applied. The extrema in the power distribution experience the same enhancement.

Table 4: Comparison of FoMs for the power distribution (293.6 K).

	Diffusion	TDC-SPH Diffusion	TDC-SPH* Diffusion
rms	8.853	0.899	1.255
max	36.531	4.49	3.328
min	-23.249	-2.636	-6.079

* with modified diffusion coefficients in the active core liner and hodoscope hole.

The steady state power distribution normalized to 1MW (thermal power) is shown in Figure 8. The difference between the Serpent and MAMMOTH power distribution is shown in Figure 9. These differences are mainly driven by the grouping of element types to conduct the SPH correction and can be corroborated with the element types assigned in Figure 2. These differences can be ameliorated with small modifications to the determination and application of SPH factors.

The steady state pellet powers for the various SERTTA units obtained from MCNP [1]

and Serpent simulations are shown in Figure 10. Unit 1 is located at the top of the active core, while unit 4 is at the bottom. The gamma contribution from the MCNP simulation was added to the Serpent results, since it currently does not include gamma heating. The Serpent values have also been corrected to match the energy deposition assumed in the design calculation (199 MeV/fission). The simulations are normalized to a core thermal power of 1 MW. The flux collar design for the SERTTA units were performed with the MCNP model and are based on the power in unit 1. The simplified Serpent model yields different results for unit 1, but confirms the consistency of the collar design for units 3-4, since unit 1 contained no poison filters. The Power Coupling Factors (PCFs) computed with MCNP and Serpent for this dataset are include in Table 5.

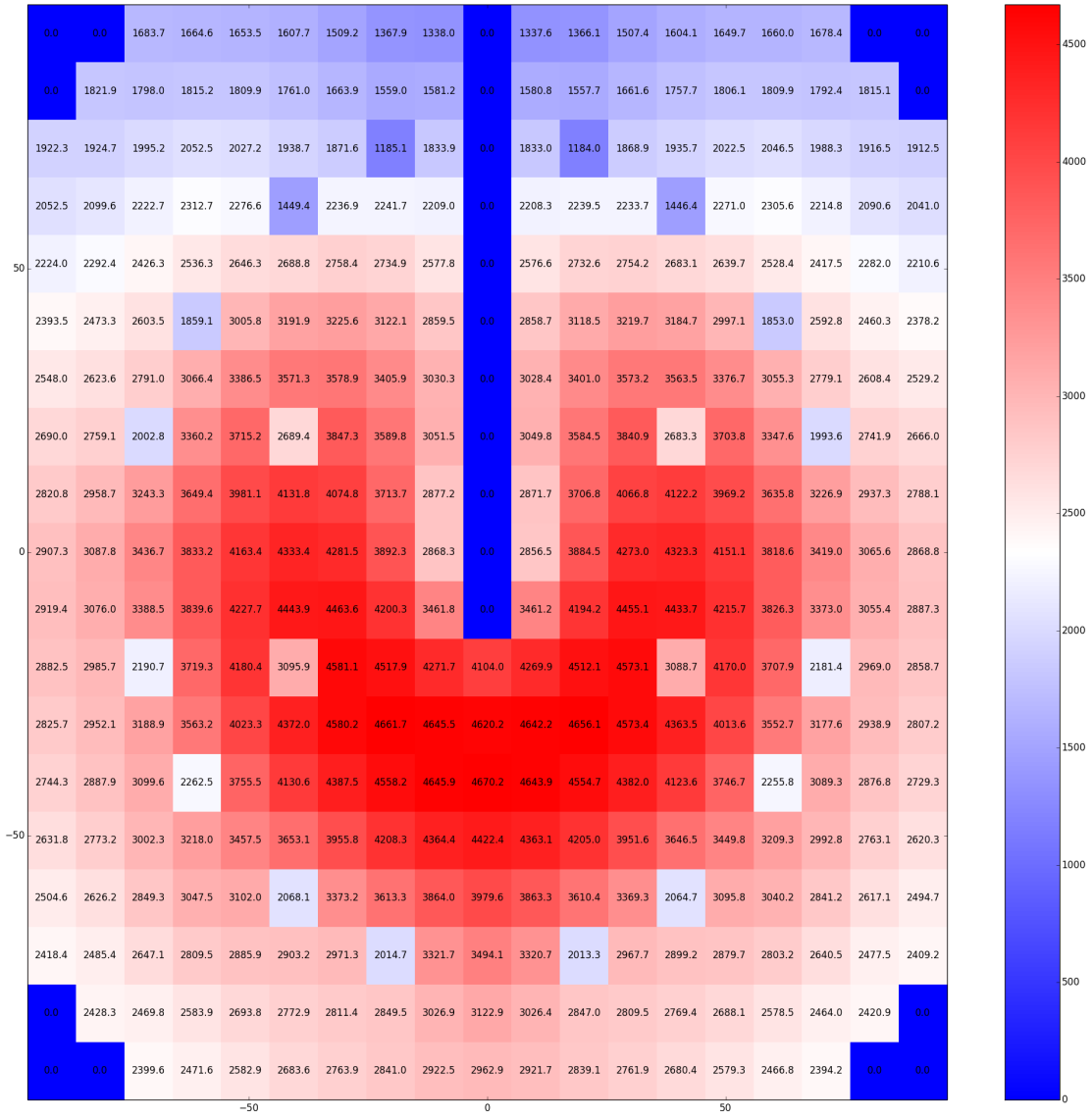


Figure 8: Serpent axially integrated radial power distribution at steady state

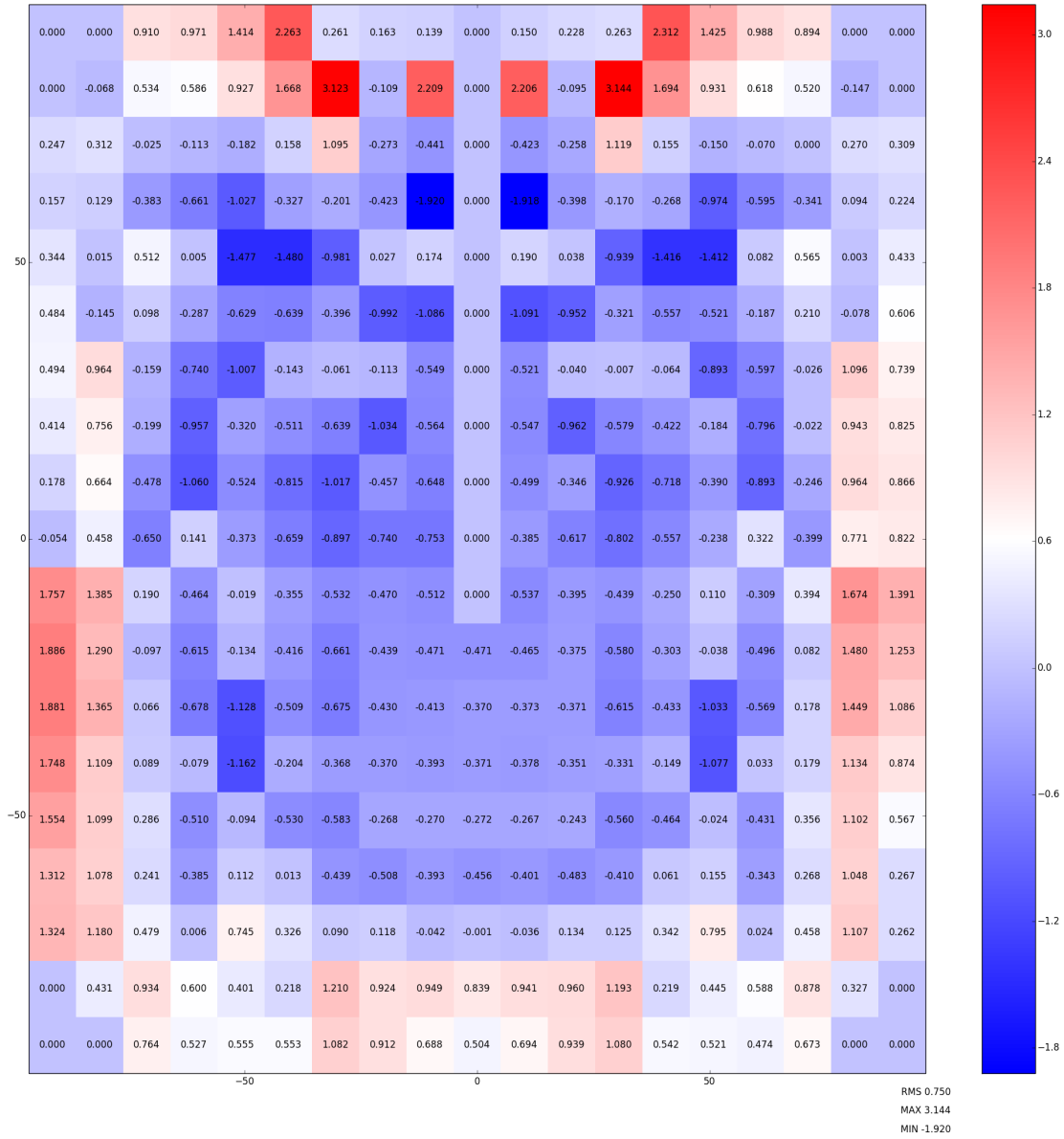
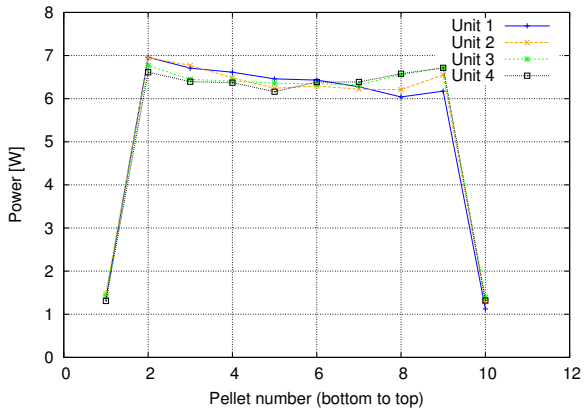
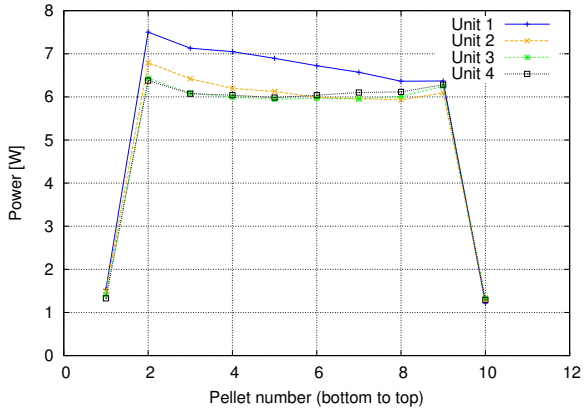


Figure 9: Differences between the Serpent and MAMMOTH axially integrated power distributions at steady state.



(a) MCNP



(b) Serpent

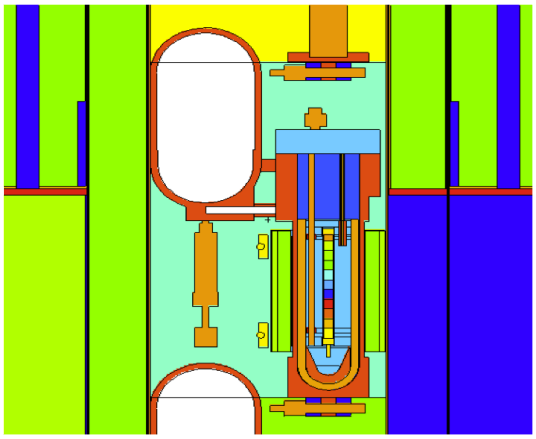
Figure 10: Pellet powers at 1MW thermal power with gamma contribution.

The differences in the unit 1 results appear to stem primarily from the modeling disparities between the MCNP model [1] and the Serpent model for the TREAT core and the Multi-SERTTA units. The Serpent model approximates several of the internal features of the SERTTA unit, as shown in Figure 11, but the Serpent model also includes more accurate slotted and half slotted elements as shown in Figure 11(b). The modeling of the slotted elements might have a significant effect on the top unit due to its closer proximity to the top reflector in the MCNP model and the location of the control rods. Finally, the Serpent model of the upper radial reflector region includes the large voids that are physically there, Figure 12. These latter modeling differences remove a substantial amount of moderator material, which also might explain the differences in the fundamental mode eigenvalue,

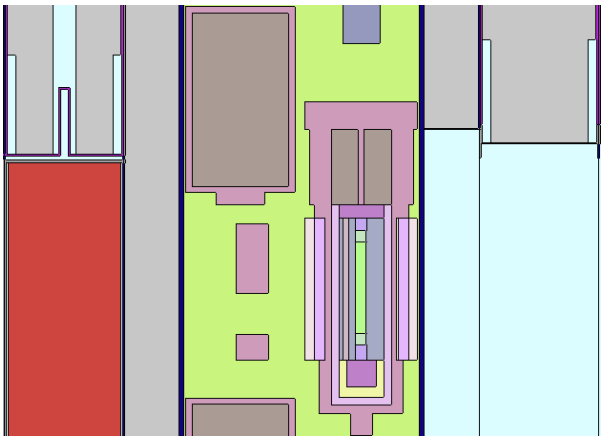
Table 5: Steady state power coupling factors with gamma heating (199 MeV/fission).

Code	Unit 1	Unit 2	Unit 3	Unit 4
MCNP	1.1381	1.1394	1.1438	1.1369
Serpent	1.2029	1.0907	1.0715	1.0799

with the transient rods out, between MCNP (1.0261) and Serpent (1.01309).

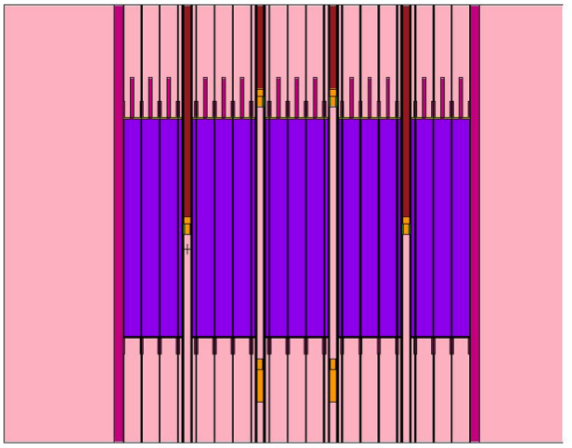


(a) MCNP

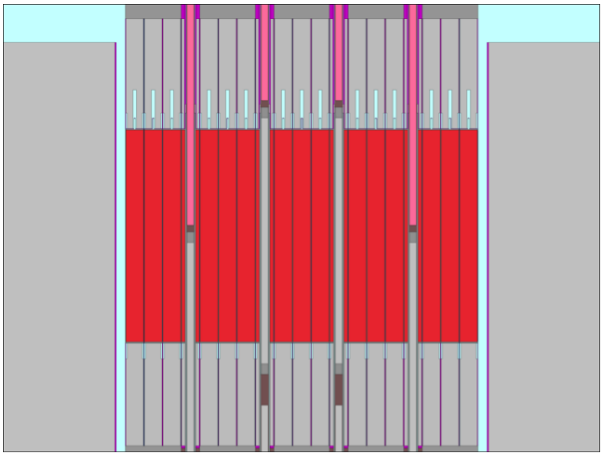


(b) Serpent

Figure 11: Differences in MCNP and Serpent models for the SERTTA unit and slotted elements.



(a) MCNP



(b) Serpent

Figure 12: Differences in MCNP and Serpent models for the radial reflector.

The transient simulations in Section 4.2 are based on an energy deposition of 182 MeV/fission determined with validation data from the M8 Calibration series [14]. A comparison of PCFs based on neutron heating alone and a value of 182 MeV/fission is included in Table 6. The coupling factors predicted in MAMMOTH are improved by almost 1% in units 2 and 3 with the modified diffusion coefficients in the active core liner and hodoscope hole.

Table 6: Steady state power coupling factors from neutron heating (182 MeV/fission).

Code	Unit 1	Unit 2	Unit 3	Unit 4
MCNP	1.0066	1.0030	1.0063	1.0040
Serpent	1.0627	0.9553	0.9371	0.9490
TDC-SPH	1.0540	0.9405	0.9218	0.9413
TDC-SPH*	1.0626	0.9515	0.9320	0.9467

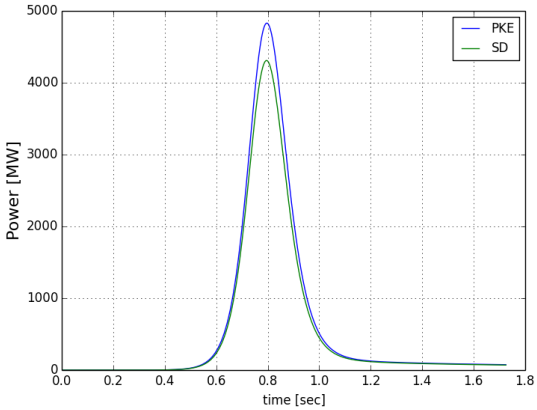
* with modified diffusion coefficients in the active core liner and hodoscope hole.

4.2 Transient Calculations

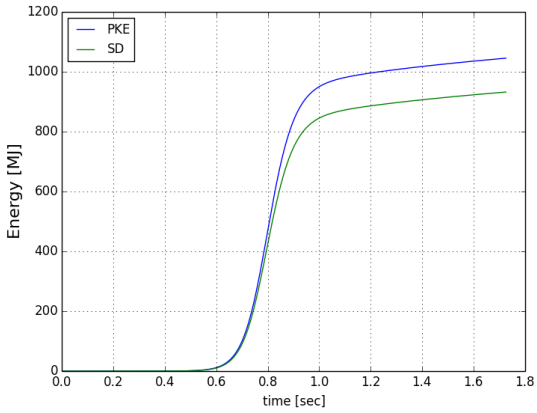
The transient behavior of the core power is shown in Figure 13 for both the PKE and the SD models. The results indicate a reactor period of approximately 44 milliseconds and corresponds to a reactivity insertion of 2.685% using the inhour equation. The peak powers are 483 and 431 MW and the integrated energies are 1045 and 932 MJ at 1.725 seconds for the PKE and SD, respectively. The pulse width at half the maximum power is 0.177 seconds for both simulations. The PRK model yields a higher peak, which is mainly due to the feedback model, since the average core temperature is actually higher, and all being equal, the power should turn around faster in the PKE. This can be verified in the future with the IQS capabilities in MAMMOTH, which would calculate the kinetics parameters as well as dynamic reactivity feedback. These results are consistent with the RELAP5-3D safety case for Multi-SERTTA [15] where the core peak power was approximately 440 MW for a step insertion of 2.634% and a pulse width of 0.195 seconds.

The average and maximum core temperatures are shown in Figure 14. The PKE simulation yields a peak core average temperature of 464 K, whereas the SD model is 18 K lower. The maximum core temperature predicted with the SD is 538 K at 1.275 seconds. The PKE model does not include a "hot channel" model at this point, so maximum fuel temperatures are not included.

The Multi-SERTTA results with PKE and SD are shown in Figures 15 and 16. Again, unit 1 is located at the top of the active core, while unit 4 is at the bottom. The PKE results overestimate the power and energy deposition in the Multi-SERTTA units by approximately 6% for this temperature limited transient. This is 6% lower than expected, since the core instantaneous and integral powers are 12% higher for the PKE simulation. The other 6% are due to transient effects. A shaped transient (with control rod clipping) might unveil more



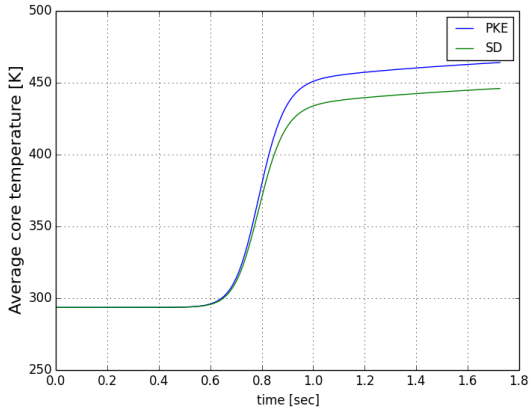
(a) Instantaneous



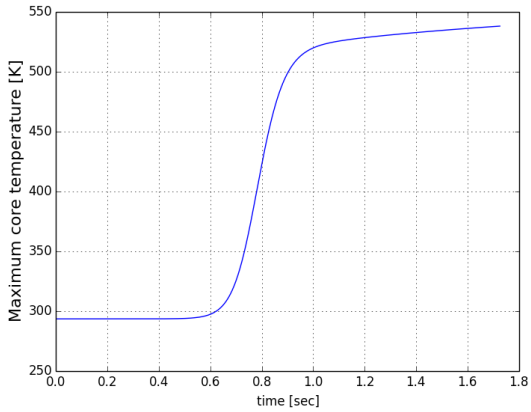
(b) Integrated

Figure 13: Core power profile during a 2.685% insertion.

of these transient effects, since the control rods will be inserted at the point of maximum energy deposition rate. Further studies will be conducted next year to determine the magnitude of these transient effects. Overall, there is good agreement between units 2-4, but unit 1 power and energy deposition is higher, which is expected based on the steady state results.

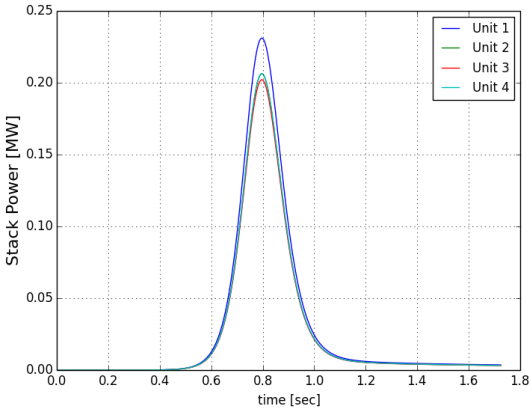


(a) Average

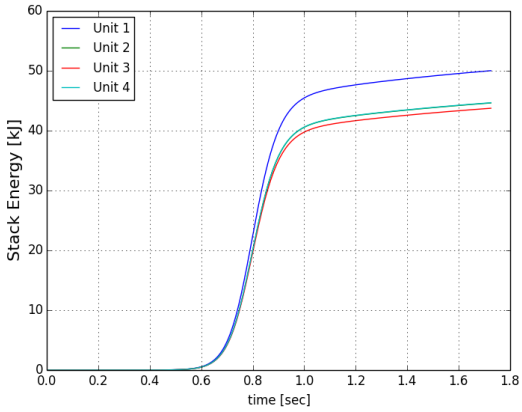


(b) Maximum

Figure 14: Core temperature profile during a 2.685% insertion.

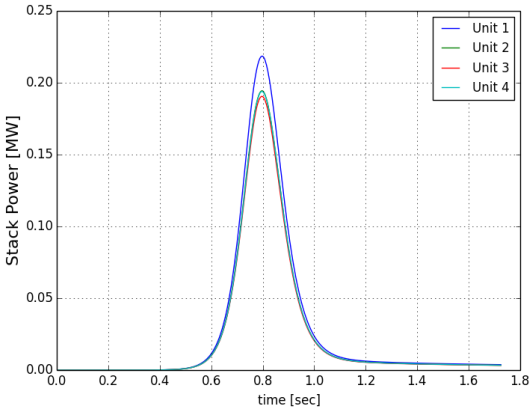


(a) Instantaneous

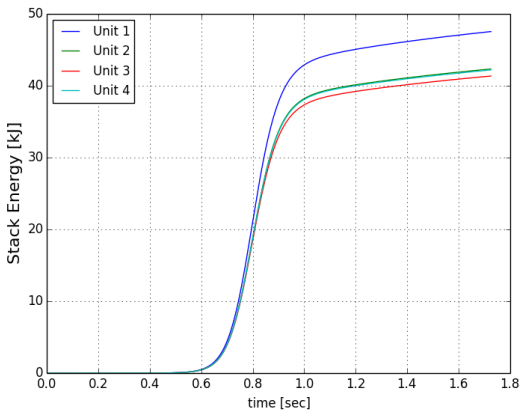


(b) Integrated

Figure 15: Multi-SERTTA enriched fuel power profile during a 2.685% insertion (PKE).



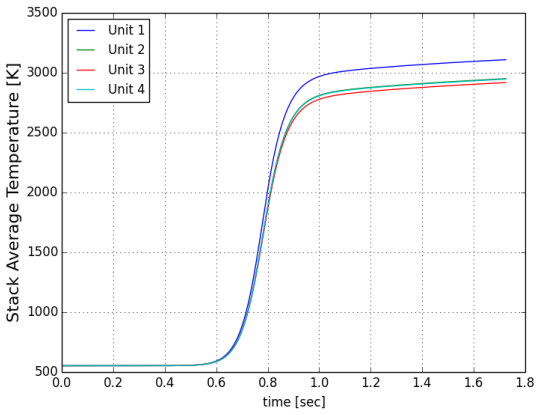
(a) Instantaneous



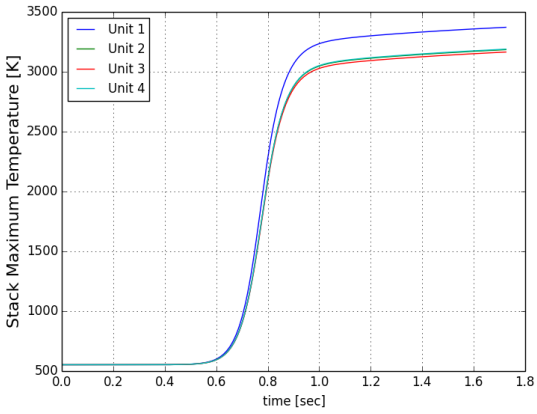
(b) Integrated

Figure 16: Multi-SERTTA enriched fuel power profile during a 2.685% insertion (SD).

The average and maximum enriched zone temperatures obtained from the SD simulation are included in Figure 17. The results show that the melting point of UO_2 (3,138 K) will be reached without clipping the transient, but the adiabatic condition will be lost before that occurs due to radiation heat transfer, swelling, etc. These effects can be investigated by adding more sophisticated BISON models to determine the thermo-elastic behavior of the samples. An average temperature of 2,100K is reached in all units at the point of maximum power, 0.8 seconds into the transient. At this point the maximum temperatures in the pellets are near 2,400 K.



(a) Average



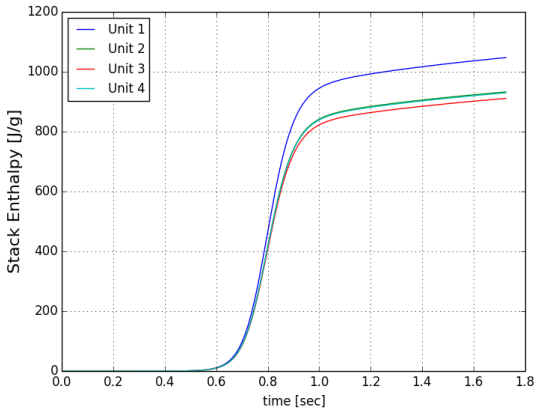
(b) Maximum

Figure 17: Multi-SERTTA enriched fuel temperature profiles during a 2.685% insertion.

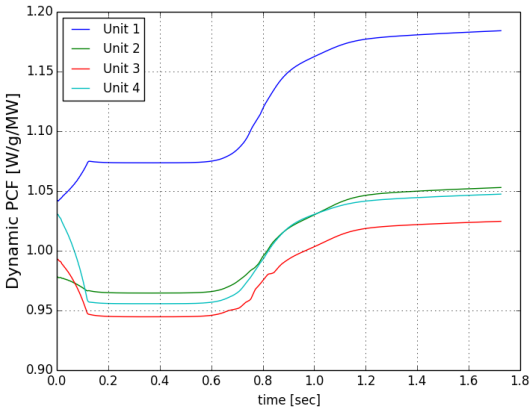
One of the great advantages of developing SD simulations is to better understand the transient effects on energy deposition in the samples. Two values are important for the simulations, one is the enthalpy of the fuel, shown in Figure 18(a) and the time dependence of the coupling factor, Figure 18(b). The planned energy injection has a maximum radially averaged enthalpy of 711 J/g. The calculated values reported here are based on the integral energy of the enriched stack. The more interesting plot is the latter, which shows some dynamics features of the transient. From 0 to 0.115 seconds the control rods are withdrawn from the core and the axial power distribution shifts towards the top of the core. Consequently, the PCF decreases in all units except unit 1. This is followed by a flat region during the positive reactivity insertion until the the temperature effects take place. At 0.7 seconds, the temperature feedback effects start and have similar magnitudes ($\sim$7%) for units 2-3 but have a higher impact on unit 1. Some of these effects could be significant during a shaped-transient where the control rods are re-inserted at the point of maximum energy deposition rate.

The distribution of the power density in Stack 4 at 0.8 seconds is included in Figure 19. A 40% decrease in the axial power density is obtained in the last enriched pellet near the natural uranium pellet. The power density profile is flat in the radial direction with a variation of 6%, except for the thin ring at the periphery of the sample, which appears to be a visualization issue, since the Paraview visualizer might exhibit issues with first order monomial functions near boundaries where the value of the variable goes to 0. This is confirmed in Figure 21 by examining the integrated power, which uses a constant monomial representation and behaves as expected. The temperature distribution in Figure 20 exhibits two axial peaks, which are expected from the additional moderation at the bottom and top of the unit. An azimuthal dependency on the temperature field is visible, with a variation of $\sim$20 degrees from peak to peak. A 100 degree temperature difference exists in the radial direction.

The neutron energy deposition in Joules is shown in Figure 21. The integrated power is an auxiliary variable in MAMMOTH with constant monomial representation, which explains the step changes in the distributions. Axial, radial and azimuthal effects of 12%, 7%, and 2%, respectively, are visible. A plot of the centerline energy distribution in each of the units is shown in Figure 22 where the effects of the axial reflectors can be seen in units 4 and 1. The axial profiles are almost symmetric about the centerline because the thermal flux shape in the core. The axial profile of the thermal flux in the experiment and the graphite element next to the experiment are shown in Figures 23 and 24, respectively. Finally, plots of the distribution in unit 1 for the first and last flux groups are included for completeness in Figures 25 and 26.

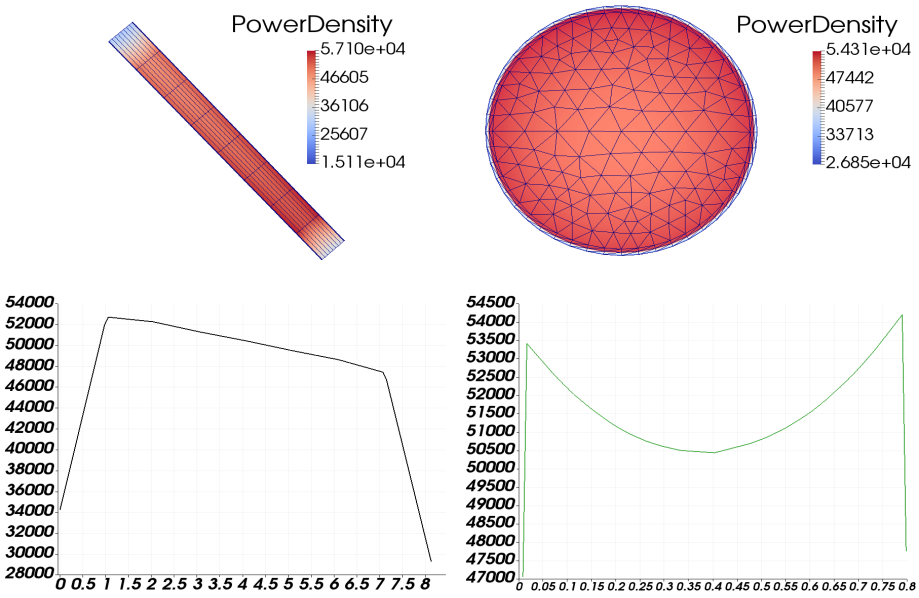


(a) Enthalpy



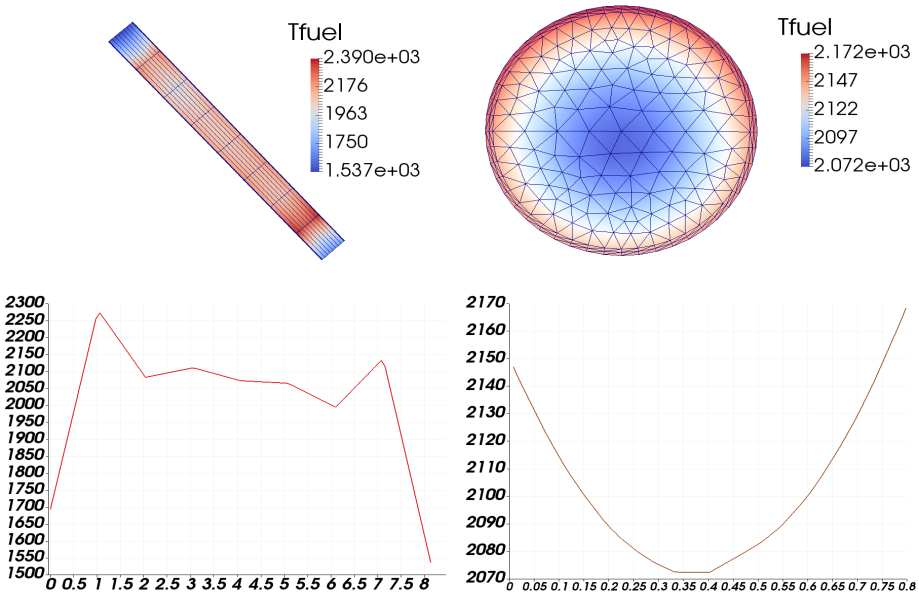
(b) Dynamic PCF

Figure 18: Multi-SERTTA Energy deposition metrics for a 2.685% insertion.



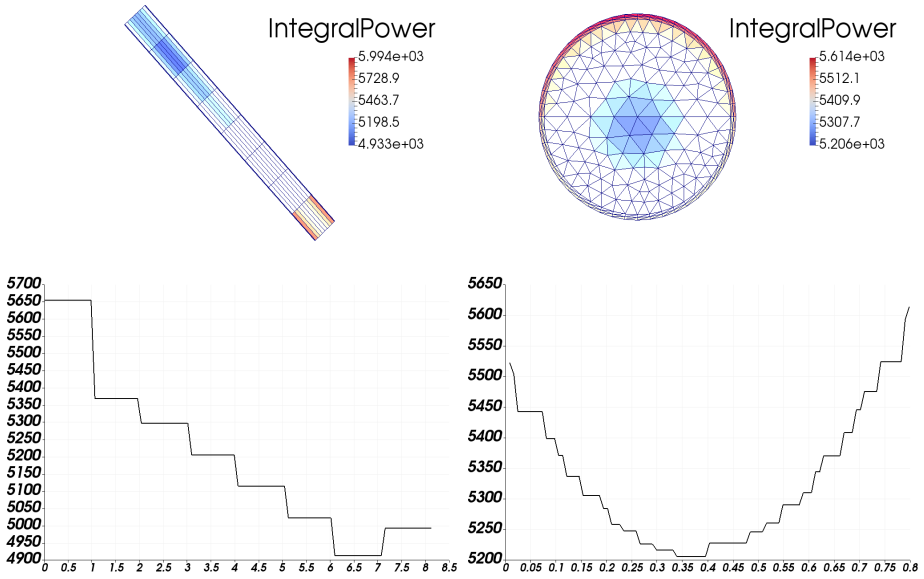
(a) Axial centerline **(b)** Radial centerline

Figure 19: Power density [W/cc] in Multi-SERTTA Unit 1 (0.8 sec).



(a) Axial centerline **(b)** Radial centerline

Figure 20: Temperature [K] in Multi-SERTTA Unit 1 (0.8 sec).



(a) Axial centerline **(b)** Radial centerline

Figure 21: Neutron energy deposition [J] in Multi-SERTTA Unit 1 (0.8 sec).

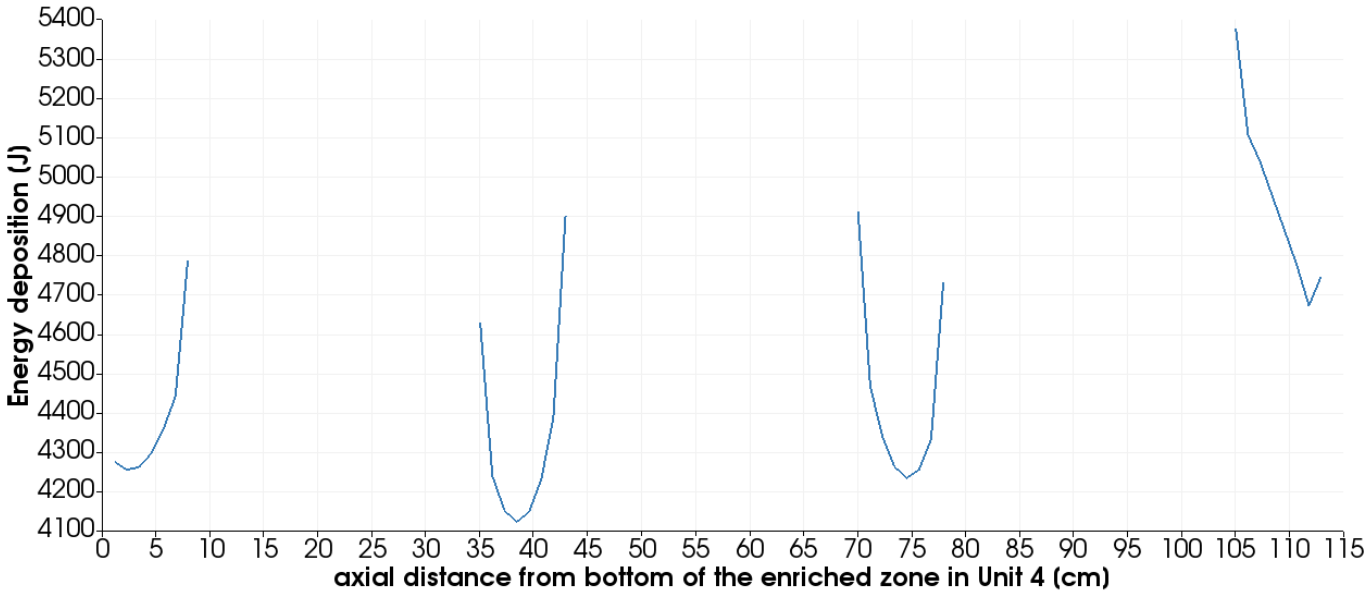


Figure 22: Axial neutron energy deposition [J] in Multi-SERTTA Units 4-1 (0.8 sec).

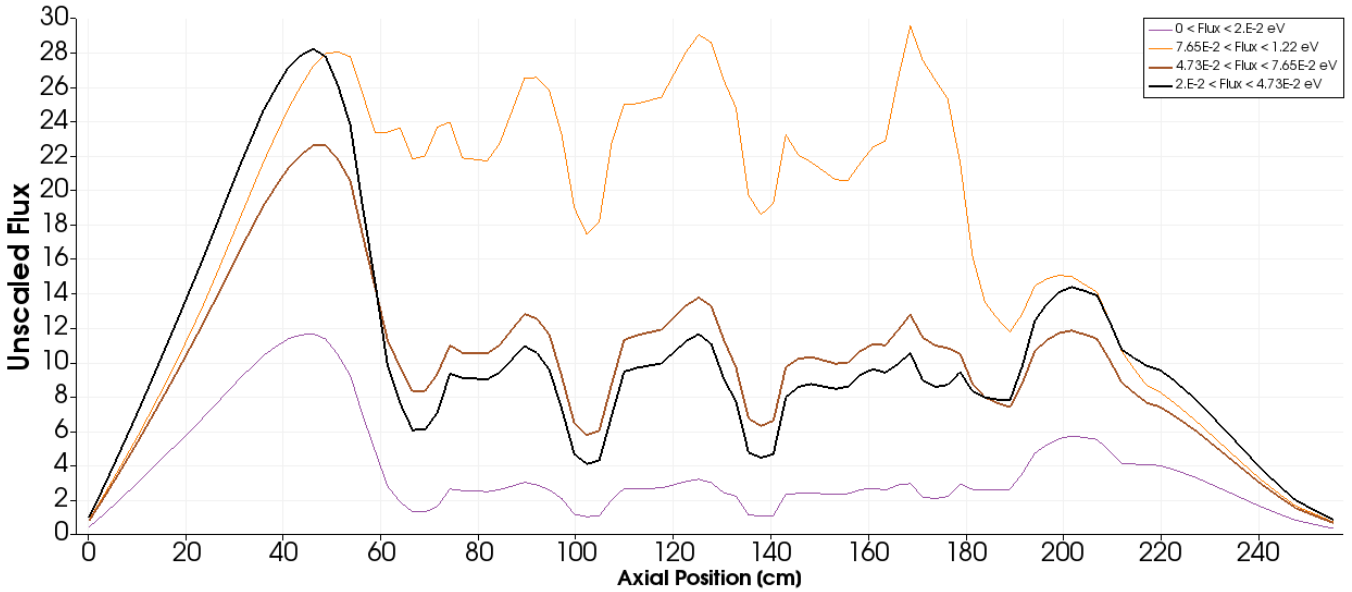


Figure 23: Axial thermal flux distributions through experiment centerline (0.8 sec).

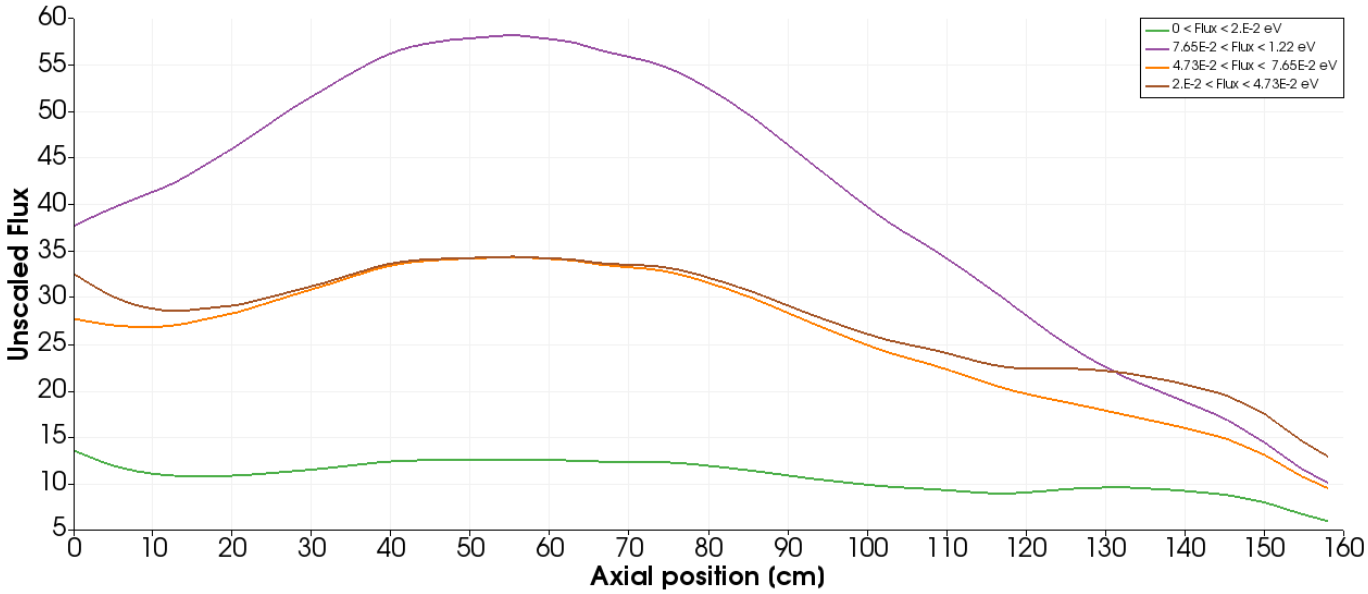
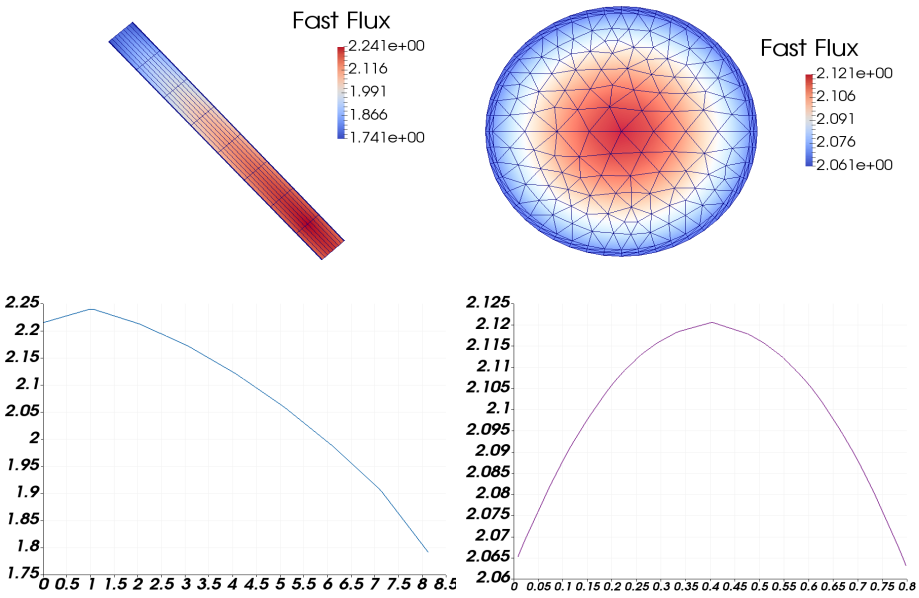
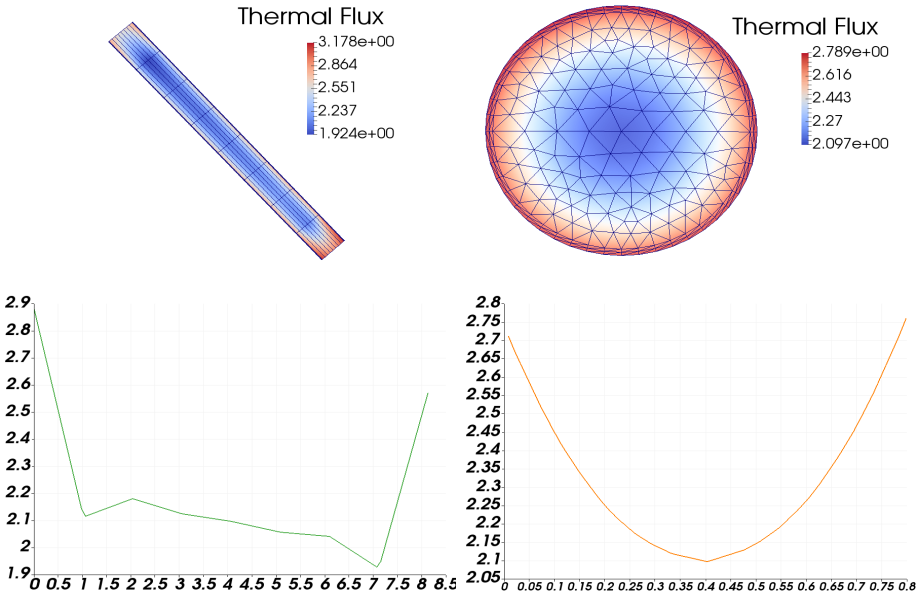


Figure 24: Axial thermal flux distributions in half graphite element near experiment (0.8 sec).



(a) Axial centerline **(b)** Radial centerline

Figure 25: Fast flux (unscaled) in Multi-SERTTA Unit 1 (0.8 sec).



(a) Axial centerline **(b)** Radial centerline

Figure 26: Thermal flux (unscaled) in Multi-SERTTA Unit 1 (0.8 sec).

5 Conclusion

High resolution modeling of the TREAT reactor with the Multi-SERTTA vehicle has been conducted using the MAMMOTH reactor physics application. MAMMOTH produces pellet stack powers that are within 1.5% of the Monte Carlo reference solutions. Some discrepancies between the MCNP model used in the design of the flux collars and the Serpent/MAMMOTH models lead to higher power and energy deposition values in Multi-SERTTA unit 1. The TREAT core results compare well with the safety case computed with point reactor kinetics in RELAP5-3D. The reactor period is 44 msec, which corresponds to a reactivity insertion of 2.685% $\Delta k/k$. The peak core power in the spatial dynamics simulation is 431 MW, which the point kinetics model over-predicts by 12%. The pulse width at half the maximum power is 0.177 sec. Subtle transient effects are apparent at the beginning of the reactivity insertion in the experimental samples due to the control rod removal. Additional differences due to transient effects are observed in the sample powers and enthalpy. The time dependence of the power coupling factor (PCF) is calculated for the various fuel stacks of the Multi-SERTTA vehicle. Sample temperatures in excess of 3100 K, the melting point UO_2, are computed with the adiabatic heat transfer model. This is not unexpected since the design calculations take fuel melting into consideration and the safety case is based on the ”un-clipped” transient.

Future work planned for FY2018 includes:

- improvement of the control rod models for clipping and shape transients,
- determination of the effects of clipping on the energy deposition in Multi-SERTTA,
- optimization of the core power distribution by improving the application of SPH factors,
- obtain better estimates of the diffusion coefficients in the various core regions,
- perform transport calculations on the Multi-SERTTA vehicle via multi-scale methods,
- add BISON thermo-elastics, contact and fluid solutions.

References

[1] J.D. Bess and C.M. Hill. Neutronics analysis of baseline configuration multi-SERTTA experiment vehicle in TREAT. Engineering Calculations and Analysis, ECAR-3372, February 2017.

[2] J. Leppänen,. *Development of a New Monte Carlo Reactor Physics Code*. PhD thesis, Helsinki University of Technology, 2007.

[3] F. N. Gleicher, J. Ortensi, et. al. The coupling of the neutron transport application RATTLESNAKE to the fuels performance application bison. In *International Conference on Reactor Physics (PHYSOR 2014)*, Kyoto, Japan, May 2014.

[4] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grand'e. Moose: A parallel computational framework for coupled systems of non-linear equations. *Nucl. Eng. Design*, 239(1768-1778), 2009.

[5] Y. Wang. Nonlinear diffusion acceleration for the multigroup transport equation discretized with sn and continuous fem with rattlesnake. In *Proceedings to the International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2013)*, Sun Valley, Idaho, USA, May 2013.

[6] R.L. Williamson et al. Multidimensional Multi-physics Simulation of Nuclear Fuel Behavior. *Jou. Nucl. Mat.*, 423(149–163), 2012.

[7] D. Andrs et al. RELAP-7 Level 2 Milestone Report: Demonstration of a Steady State Single Phase PWR Simulation with RELAP-7. Technical Report INL/EXT-12-25924, Idaho National Laboratory, 2012.

[8] T.J. Trahan. *An Asymptotic, Homogenized, Anisotropic, Multigroup Diffusion Approximation to the Neutron Transport Equation*. PhD thesis, University of Michigan, 2014.

[9] J. Ortensi, B. Baker, S. Schunert, Y. Wang, F.N. Gleicher, and M.D. DeHart. Updates to the Generation of Physics Data Inputs for MAMMOTH Simulations of the Transient Reactor Test Facility - FY2016. Technical Report INL/EXT-16-39120, Idaho National Laboratory, June 2016.

[10] A. Hébert, G. Mathonniere. Development of a third generation superhomogenization method for the homogenization of pressurized water reactor assembly. *Nuc. Sci. Eng.*, 2:115, 1993.

[11] Randy Morris. Cubit 15.0 user documentation. Technical report, ETI, UT, 2014.

[12] G.A. Freund, P. Elias, D.R. Macfarlane, J.D. Geier, and J.F. Boland. Design summary report on the transient reactor test facility (TREAT). Technical Report ANL-6034, Argonne National Laboratory, 1960.

[13] S.G. Popov, J.J. Carbajo, V.K. Ivanov, and G.L. Yoder. Thermophysical properties of MOX and UO2 fuels including the effects of irradiation. Technical Report ORNL/TM-2000/351, Oak Ridge National Laboratory, 2000.

[14] B. Baker, J. Ortensi, M.D. DeHart, Y. Wang, S. Schunert, and F.N. Gleicher. Analysis Methods and Validation Activities for MAMMOTH Using M8 Calibration Series Data. Technical Report INL/EXT-16-40023, Idaho National Laboratory, September 2016.

[15] C. Davis, D. Gerstner, and A. LaPorta. Application of RELAP5-3D point kinetics model to TREAT experiments. Engineering Calculations and Analysis, ECAR-3459, February 2017.

Appendices

A Mesh preparation script

```
import sys as _sys
import TREAT_assembly as cls_asmb
import TREAT_cyl_cluster as cls_cc
import TREAT_pr_box as cls_prbox
import TREAT_core as cls_core
import TREAT_pr as cls_pr
import TREAT_vehicle as cls_vh

################################################
# Some control parameters
################################################
write_cub = False  # saves the cubit file with the full geometry and mesh
dimension = 3      # dimention of the problem
write_material_id = True   # to write the material_id variable to the file
################################################
# Core Geometry specifications
################################################
# Core parameters
core = cls_core.core()  # calls the core class constructor
core.set_outfile('core_MSERTTA_TR_old')  # specify the ouput
# next we specify the core map with assembly number (these must match the assemblies that we build below in the Assembly specification)
core.set_map( [
                          [6000,  6000,  12800, 12800, 12800, 12800, 13400, 12500, 11700, 5450,  11700, 12500, 13400, 12800, 12800, 12800, 12800, 6000,  6000],
                          [6000,  12800, 12900, 12900, 12900, 12900, 11200, 13500, 11500, 5400,  11500, 13500, 11200, 12900, 12900, 12900, 12900, 12800, 6000],
                          [12800, 12900, 12900, 12900, 11200, 11200, 11200, 1100,  11500, 5350,  11500, 1100,  11200, 11200, 11200, 12900, 12900, 12900, 13300],
                          [12800, 12900, 12900, 12900, 11200, 1200,  11200, 13600, 11500, 5300,  11500, 13600, 11200, 1200,  11200, 12900, 12900, 12900, 13300],
                          [12800, 12900, 11300, 11300, 11200, 11200, 11200, 13700, 11600, 5250,  11600, 13700, 11200, 11200, 11200, 11300, 11300, 12900, 13300],
                          [12800, 12900, 11300, 1700,  11300, 12300, 12300, 13700, 11600, 5200,  11600, 13700, 12300, 12300, 11300, 1700,  11300, 12900, 13300],
                          [12800, 11300, 11300, 11300, 11300, 11900, 11900, 13800, 14600, 5150,  14600, 13800, 11900, 11900, 11300, 11300, 11300, 11300, 13300],
                          [12800, 11300, 1800,  11300, 11900, 1500,  11900, 13800, 14500, 5100,  14500, 13800, 11900, 1500,  11900, 11300, 1800,  11300, 13300],
                          [12800, 11300, 11300, 11300, 11900, 11900, 11900, 13900, 14400, 5900,  14400, 13900, 11900, 11900, 11900, 11300, 11300, 11300, 13300],
                          [12800, 13100, 13100, 13000, 13000, 13000, 13000, 13900, 14300, -1,    14300, 13900, 13000, 13000, 13000, 13000, 13100, 13100, 13300],
                          [12700, 11400, 11400, 11400, 12000, 12000, 12000, 12100, 14200, 6100,  14200, 12100, 12000, 12000, 12000, 11400, 11400, 11400, 13200],
                          [12700, 11400, 2000,  11400, 12000, 1600,  12000, 12100, 14100, 14000, 14100, 12100, 12000, 1600,  12000, 11400, 2000,  11400, 13200],
                          [12700, 11400, 11400, 11400, 11400, 12000, 12000, 12100, 12100, 12100, 12100, 12100, 12000, 12000, 11400, 11400, 11400, 11400, 13200],
                          [12700, 12400, 11400, 1900,  11400, 12200, 12200, 12100, 12100, 12100, 12100, 12100, 12200, 12200, 11400, 1900,  11400, 12400, 13200],
                          [12700, 12400, 11400, 11400, 11100, 11100, 11100, 12100, 12100, 12100, 12100, 12100, 11100, 11100, 11100, 11400, 11400, 12400, 13200],
                          [12700, 12400, 12400, 12400, 11100, 1400,  11100, 11100, 11100, 11800, 11100, 11100, 11100, 1400,  11100, 12400, 12400, 12400, 13200],
                          [12700, 12400, 12400, 12400, 11100, 11100, 11100, 1300,  11100, 11800, 11100, 1300,  11100, 11100, 11100, 12400, 12400, 12400, 13200],
                          [6000,  12700, 12400, 12400, 12400, 12400, 11100, 11100, 11100, 11800, 11100, 11100, 11100, 12400, 12400, 12400, 12400, 12700, 6000],
                          [6000,  6000,  12700, 12700, 12700, 12700, 12600, 12600, 12600, 12600, 12600, 12600, 12600, 12700, 12700, 12700, 12700, 6000,  6000] ])

core.set_assembly_pitch(10.16) # assembly pitch

my_dz = [3.486,27,22.68875,1.564,4.3,2.36,0.666,0.762,1.016,8.128,1.016,1.112,1.17229,7.55571,6.8385,4.3,2.36,1.428,1.016,8.128,1.016,1.11,1.17229,14.39421,
4.3,0.5485,1.8115,1.428,1.016,8.128,1.016,1.11,1.17229,2.59071,6.07975,5.72375,4.3,2.36,1.428,1.016,8.128,1.016,1.11,1.17229,3.74971,0.762,9.8825,12.18375,
0.51625,9.53,3.59375,27,8.02675]
my_int_z = [1,4,4,1,1,1,1,1,1,8,1,1,1,1,1,1,1,1,8,1,1,1,2,1,1,1,1,1,8,1,1,1,1,1,3,1,1,1,1,8,1,1,1,1,1,2,2,1,2,1,4,1]
core.set_dz(my_dz)
core.set_int_z(my_int_z)
core.set_isolate_experiment(False)
MatID_13R = [1,2,3,4,4,4,4,4,5,6,6,6,6,6,6,6,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,8,9,10,10,11,11,11,12,13]
MatID_19R = [1,2,3,4,4,4,4,4,5,6,6,6,6,6,6,6,7,7,7,7,7,7,7,7,7,7,7,8,8,8,9,9,9,10,10,11,11,11,12,13,13,14,14,14,14,14,14,14,14,15,16,16,17,17,17,18,19]

#########################################################################################################################################################
#  PERMANENT REFLECTOR SPECIFICATION
#########################################################################################################################################################
pr = cls_pr.pr(1)
pr.set_split(True) # cuts the PR layers at 45 deg angles generating 4 regions - block id and material id assignment have 4 entries each for each core axial layer
block_id = []
material_id = []
z = 0.0
for i,dz in enumerate(core.dz):
   z +=dz
   block_id.append( (400,400,400,400)  )
   if z <= 62.82675:
```

```
      material_id.append( (534,533,532,531) )
   elif z <= 147.711:
      material_id.append( (544,543,542,541) )
   elif z <= 183.7943:
      material_id.append( (554,553,552,551) )
   else:
      material_id.append( (564,563,562,561) )

pr.set_block_id(    tuple(block_id)    )
pr.set_material_id( tuple(material_id) )
pr.set_element_type(["WEDGE","WEDGE","WEDGE","WEDGE"])
pr.set_dr(5.714999995)
pr.set_int_long(30)
pr.set_int_short(2)
core.add_pr(pr)

pr = cls_pr.pr(2)
pr.set_split(True)
block_id = []
material_id = []
z = 0.0
for i,dz in enumerate(core.dz):
   z +=dz
   block_id.append( (500,500,500,500)  )
   if z <= 62.82675:
      material_id.append( (414,413,412,411) )
   elif z <= 147.711:
      material_id.append( (424,423,422,421) )
   elif z <= 183.7943:
      material_id.append( (434,433,432,431) )
   else:
      material_id.append( (444,443,442,441) )
pr.set_block_id(    tuple(block_id)    )
pr.set_material_id( tuple(material_id) )
pr.set_element_type(["WEDGE","WEDGE","WEDGE","WEDGE"])
pr.set_dr(20.0025)
pr.set_int_long(25)
pr.set_int_short(2)
core.add_pr(pr)

pr = cls_pr.pr(3)
pr.set_split(True)
block_id = []
material_id = []
z = 0.0
for i,dz in enumerate(core.dz):
   z +=dz
   block_id.append( (500,500,500,500)  )
   if z <= 62.82675:
      material_id.append( (454,453,452,451) )
   elif z <= 147.711:
      material_id.append( (464,463,462,461) )
   elif z <= 183.7943:
      material_id.append( (474,473,472,471) )
   else:
      material_id.append( (484,483,482,481) )
pr.set_block_id(    tuple(block_id)    )
pr.set_material_id( tuple(material_id) )
pr.set_element_type(["WEDGE","WEDGE","WEDGE","WEDGE"])
pr.set_dr(20.0025)
pr.set_int_long(20)
pr.set_int_short(2)
core.add_pr(pr)

pr = cls_pr.pr(4)
pr.set_split(True)
block_id = []
material_id = []
z = 0.0
for i,dz in enumerate(core.dz):
   z +=dz
   block_id.append( (500,500,500,500)  )
   if z <= 62.82675:
      material_id.append( (494,493,492,491) )
   elif z <= 147.711:
      material_id.append( (504,503,502,501) )
   elif z <= 183.7943:
      material_id.append( (514,513,512,511) )
```

```
   else:
      material_id.append( (524,523,522,521) )
pr.set_block_id(    tuple(block_id)    )
pr.set_material_id( tuple(material_id) )
pr.set_element_type(["WEDGE","WEDGE","WEDGE","WEDGE"])
pr.set_dr(20.955)
pr.set_int_long(15)
pr.set_int_short(2)
core.add_pr(pr)
#########################################################################################################################################################################
#  PERMANENT REFLECTOR BOX SPECIFICATION
#########################################################################################################################################################################

# add hodoscope Noth Wall boxes 1-7
# order is | row 1 left,right,center | row 2 left,right,center | row 3
prboxes_dims =      [ (40.005,20.0025,80.9875),(35.71875,20.0025,80.9875),(35.71875,20.955,80.9875)]
prboxes_centroids = [ (0,112.23625,123.3205),(0,132.23875,123.3205),(0,152.7175,123.3205)]
prboxes_long_int    = [9,9,6]
prboxes_z_int       = [8,8,8]
prboxes_block_id    = [600,600,600]
prboxes_material_id = [406,407,408]

for idx,dim in enumerate(prboxes_dims):
   pr_box = cls_prbox.pr_box(idx+1) # call pr_box constructor
   pr_box.set_dim(dim)
   pr_box.set_centroid(prboxes_centroids[idx])
   pr_box.set_element_type(["WEDGE"])
   pr_box.set_block_id( (prboxes_block_id[idx],) )
   pr_box.set_material_id( (prboxes_material_id[idx],) )
   pr_box.set_int_long(prboxes_long_int[idx])
   pr_box.set_int_z( (prboxes_z_int[idx],) )
   core.add_pr_box(pr_box)

#########################################################################################################################################################################
# ASSEMBLY SPECIFICATION
#########################################################################################################################################################################
####################  Build element types A ############################
for element in range(11100,14700,100): # loop through all type A elements
   # Define an assembly
   assembly = cls_asmb.assembly(element) # call assembly constructor
   assembly.set_int_short(2) # sets interval in short side of octagon
   assembly.set_int_long(4)  # sets interval in long side of octagon | side for FHnch geometry

   for i,val in enumerate(core.dz):
    if i < 8 or i > 44:
      assembly.add_block_id( [element+1] )
    else:
      assembly.add_block_id( [element] )
   for i in MatID_19R:
      assembly.add_material_id( [i] )

   assembly.set_element_type( ["HEX"] )   # set the Finite element type
   core.add_assembly(assembly)  # store assembly


####################  Build element types B ############################
for element in range(1100,2100,100): # loop through all type A elements
   # Define an assembly
   assembly = cls_asmb.assembly(element) # call assembly constructor
   assembly.set_int_short(2) # sets interval in short side of octagon
   assembly.set_int_long(4)  # sets interval in long side of octagon | side for FHnch geometry

   for i,val in enumerate(core.dz):
    if i < 8 or i > 44:
      assembly.add_block_id( [element+1] )
    else:
      assembly.add_block_id( [element] )
   for i in MatID_19R:
      assembly.add_material_id( [i] )

   assembly.set_element_type( ["HEX"] )   # set the Finite element type
   core.add_assembly(assembly)  # store assembly

####################  Build element types C ############################
for element in range(5100,5500,50): # loop through all type A elements
   # Define an assembly
   assembly = cls_asmb.assembly(element) # call assembly constructor
   assembly.set_int_short(2) # sets interval in short side of octagon
```

```
   assembly.set_int_long(4)  # sets interval in long side of octagon | side for FHnch geometry

   for i,val in enumerate(core.dz): # region 8-46 is slotted region | create a separate block for TDC cross sections
      if i < 7 or i > 45:
        assembly.add_block_id( [element+1] )
      else:
        assembly.add_block_id( [element] )

   for i in MatID_13R:
      assembly.add_material_id( [i] )

   assembly.set_element_type( ["HEX"] )   # set the Finite element type
   core.add_assembly(assembly)  # store assembly

#####################  Build element types D ############################
for element in range(6000,6100,100): # loop through all type A elements
   # Define an assembly
   assembly = cls_asmb.assembly(element) # call assembly constructor
   assembly.set_int_short(2) # sets interval in short side of octagon
   assembly.set_int_long(4)  # sets interval in long side of octagon | side for FHnch geometry

   for i,val in enumerate(core.dz):
      assembly.add_block_id( [element] )
   for i in MatID_13R:
      assembly.add_material_id( [i] )

   assembly.set_element_type( ["HEX"] )   # set the Finite element type
   core.add_assembly(assembly)  # store assembly

#########################################################################
###################  2 Half assemblies ##################################
#########################################################################
# Define an assembly
for element in [5900]:
   # Define an assembly
   assembly = cls_asmb.assembly(element) # call assembly constructor
   assembly.set_int_short(2) # sets interval in short side of octagon
   assembly.set_int_long(4)  # sets interval in long side of octagon | side for FHnch geometry
   assembly.is_half_element = True  # to set to halp element
   if element == 5900:
      assembly.shift ="top"         # to position the half element
   elif element == 6100:
      assembly.shift ="bottom"         # to position the half element

   for i,val in enumerate(core.dz): # region 8-46 is slotted region | create a separate block for TDC cross sections
      if i < 7 or i > 45:
        assembly.add_block_id( [element+1] )
      else:
        assembly.add_block_id( [element] )

   for i in MatID_13R:
      assembly.add_material_id( [i] )

   assembly.set_element_type( ["HEX"] )   # set the Finite element type

   core.add_assembly(assembly)  # store assembly

for element in [6100]:
   # Define an assembly
   assembly = cls_asmb.assembly(element) # call assembly constructor
   assembly.set_int_short(2) # sets interval in short side of octagon
   assembly.set_int_long(4)  # sets interval in long side of octagon | side for FHnch geometry
   assembly.is_half_element = True  # to set to halp element
   if element == 5900:
      assembly.shift ="top"         # to position the half element
   elif element == 6100:
      assembly.shift ="bottom"         # to position the half element

   for i,val in enumerate(core.dz):
      assembly.add_block_id( [element] )
   for i in MatID_13R:
      assembly.add_material_id( [i] )

   assembly.set_element_type( ["HEX"] )   # set the Finite element type

   core.add_assembly(assembly)  # store assembly
#########################################################################
############################  Vehicle ###################################
```

```
########################################################################
# Define an assembly
vehicle = cls_vh.vehicle(-1)
#vehicle.center_point = ()
vehicle.set_box_length( (10.16,20.32) ) #x,y lengths of inner most box
vehicle.set_box_x_thickness([]) # list of concentric thicknesses moving outwards in x direction
vehicle.set_box_y_thickness([]) # list of concentric thicknesses moving outwards in y direction
vehicle.set_int_short([4]) # list of intervals for concentric thicknesses
vehicle.set_int_long([8])  # list of intervals for concentric thicknesses

# Block ID assignment for each concentric box (start inside and move out)
# two regions
vehicle.set_split_box(True) # adds 1 more block for each layer
                  #  South | North | Box

vehicle.add_block_id( [ 1, 1] ) # BFF & ref 1
vehicle.add_block_id( [ 1, 1] ) # BFF & ref 1
vehicle.add_block_id( [ 1, 1] ) # ref 2
vehicle.add_block_id( [ 1, 1] ) # ref 2

vehicle.add_block_id( [ 2, 2] ) # tang & botttom RPV 1
vehicle.add_block_id( [ 2, 2] ) # cup water level to heater bottom 1
vehicle.add_block_id( [ 2, 2] ) # heater bottom to bottom of fuel stack 1
vehicle.add_block_id( [ 2, 2] ) # heater bottom to bottom of fuel stack 1
vehicle.add_block_id( [ 2, 2] ) # Nat U 1
vehicle.add_block_id( [ 2, 2] ) # Enr U 1
vehicle.add_block_id( [ 2, 2] ) # Nat U 1
vehicle.add_block_id( [ 2, 2] ) # top cap to top of heater 1
vehicle.add_block_id( [ 2, 2] ) # water and top crucible 1
vehicle.add_block_id( [ 2, 2] ) # upper assembly w tank 1
vehicle.add_block_id( [ 2, 2] ) # upper assembly w tank 1

vehicle.add_block_id( [ 3, 3] ) # tang & botttom RPV 2
vehicle.add_block_id( [ 3, 3] ) # cup water level to heater bottom 2
vehicle.add_block_id( [ 3, 3] ) # heater bottom to bottom of fuel stack 2
vehicle.add_block_id( [ 3, 3] ) # Nat U 2
vehicle.add_block_id( [ 3, 3] ) # Enr U 2
vehicle.add_block_id( [ 3, 3] ) # Nat U 2
vehicle.add_block_id( [ 3, 3] ) # top cap to top of heater 2
vehicle.add_block_id( [ 3, 3] ) # water and top crucible 2
vehicle.add_block_id( [ 3, 3] ) # upper assembly w tank 2

vehicle.add_block_id( [ 4,4] ) # tang & botttom RPV 3
vehicle.add_block_id( [ 4,4] ) # cup water level to heater bottom 3
vehicle.add_block_id( [ 4,4] ) # cup water level to heater bottom 3
vehicle.add_block_id( [ 4,4] ) # heater bottom to bottom of fuel stack 3
vehicle.add_block_id( [ 4,4] ) # Nat U 3
vehicle.add_block_id( [ 4,4] ) # Enr U 3
vehicle.add_block_id( [ 4,4] ) # Nat U 3
vehicle.add_block_id( [ 4,4] ) # top cap to top of heater 3
vehicle.add_block_id( [ 4,4] ) # water and top crucible 3
vehicle.add_block_id( [ 4,4] ) # upper assembly w tank 3
vehicle.add_block_id( [ 4,4] ) # upper assembly w tank 3
vehicle.add_block_id( [ 4,4] ) # upper assembly w tank 3

vehicle.add_block_id( [ 5,5] ) # tang & botttom RPV 4
vehicle.add_block_id( [ 5,5] ) # cup water level to heater bottom 4
vehicle.add_block_id( [ 5,5] ) # heater bottom to bottom of fuel stack 4
vehicle.add_block_id( [ 5,5] ) # Nat U 4
vehicle.add_block_id( [ 5,5] ) # Enr U 4
vehicle.add_block_id( [ 5,5] ) # Nat U 4
vehicle.add_block_id( [ 5,5] ) # top cap to top of heater 4
vehicle.add_block_id( [ 5,5] ) # water and top crucible 4
vehicle.add_block_id( [ 5,5] ) # upper assembly w tank 4
vehicle.add_block_id( [ 5,5] ) # upper assembly w tank 4
vehicle.add_block_id( [ 5,5] ) # upper assembly w tank 4

vehicle.add_block_id( [ 6, 6] ) # hanger rod to bottom tungsten alloy block
vehicle.add_block_id( [ 6, 6] ) # hanger rod to bottom tungsten alloy block
vehicle.add_block_id( [ 6, 6] ) # tungsten alloy block (or lead)
vehicle.add_block_id( [ 6, 6] ) # hanger rod ends
vehicle.add_block_id( [ 6, 6] ) # hanger rod ends
vehicle.add_block_id( [ 6, 6] ) # hanger rod ends

vehicle.set_element_type( ["WEDGE","WEDGE"] )

# material_ID assignment (this should match the library for each )
#                          South  North
```

```
vehicle.add_material_id( [ 10001, 10001] ) # BFF & ref 1
vehicle.add_material_id( [ 10001, 10001] ) # BFF & ref 1
vehicle.add_material_id( [ 10002, 10002] ) # ref 2
vehicle.add_material_id( [ 10002, 10002] ) # ref 2

vehicle.add_material_id( [ 10008,  9904] ) # tang & botttom RPV 1
vehicle.add_material_id( [ 10010,  9907] ) # cup water level to heater bottom 1
vehicle.add_material_id( [ 10012,  9909] ) # heater bottom to bottom of fuel stack 1
vehicle.add_material_id( [ 10012,  9909] ) # heater bottom to bottom of fuel stack 1
vehicle.add_material_id( [ 10014,  9912] ) # Nat U 1
vehicle.add_material_id( [ 10016,  9912] ) # Enr U 1
vehicle.add_material_id( [ 10016,  9912] ) # Nat U 1
vehicle.add_material_id( [ 10016,  9909] ) # top cap to top of heater 1
vehicle.add_material_id( [ 10018, 10017] ) # water and top crucible 1
vehicle.add_material_id( [ 10020, 10019] ) # upper assembly w tank 1
vehicle.add_material_id( [ 10020, 10019] ) # upper assembly w tank 1

vehicle.add_material_id( [ 10022,  9920] ) # tang & botttom RPV 2
vehicle.add_material_id( [ 10024,  9922] ) # cup water level to heater bottom 2
vehicle.add_material_id( [ 10026,  9923] ) # heater bottom to bottom of fuel stack 2
vehicle.add_material_id( [ 10028,  9926] ) # Nat U 2
vehicle.add_material_id( [ 10030,  9926] ) # Enr U 2
vehicle.add_material_id( [ 10030,  9926] ) # Nat U 2
vehicle.add_material_id( [ 10030,  9923] ) # top cap to top of heater 2
vehicle.add_material_id( [ 10032, 10031] ) # water and top crucible 2
vehicle.add_material_id( [ 10034, 10033] ) # upper assembly w tank 2

vehicle.add_material_id( [ 10036,  9932] ) # tang & botttom RPV 3
vehicle.add_material_id( [ 10038,  9934] ) # cup water level to heater bottom 3
vehicle.add_material_id( [ 10038,  9934] ) # cup water level to heater bottom 3
vehicle.add_material_id( [ 10040,  9935] ) # heater bottom to bottom of fuel stack 3
vehicle.add_material_id( [ 10042,  9938] ) # Nat U 3
vehicle.add_material_id( [ 10044,  9938] ) # Enr U 3
vehicle.add_material_id( [ 10044,  9938] ) # Nat U 3
vehicle.add_material_id( [ 10044,  9935] ) # top cap to top of heater 3
vehicle.add_material_id( [ 10046, 10045] ) # water and top crucible 3
vehicle.add_material_id( [ 10048, 10047] ) # upper assembly w tank 3
vehicle.add_material_id( [ 10048, 10047] ) # upper assembly w tank 3
vehicle.add_material_id( [ 10048, 10047] ) # upper assembly w tank 3

vehicle.add_material_id( [ 10050,  9944] ) # tang & botttom RPV 4
vehicle.add_material_id( [ 10052,  9946] ) # cup water level to heater bottom 4
vehicle.add_material_id( [ 10054,  9947] ) # heater bottom to bottom of fuel stack 4
vehicle.add_material_id( [ 10056,  9950] ) # Nat U 4
vehicle.add_material_id( [ 10058,  9950] ) # Enr U 4
vehicle.add_material_id( [ 10058,  9950] ) # Nat U 4
vehicle.add_material_id( [ 10058,  9947] ) # top cap to top of heater 4
vehicle.add_material_id( [ 10060, 10059] ) # water and top crucible 4
vehicle.add_material_id( [ 10062, 10061] ) # upper assembly w tank 4
vehicle.add_material_id( [ 10062, 10061] ) # upper assembly w tank 4
vehicle.add_material_id( [ 10062, 10061] ) # upper assembly w tank 4

vehicle.add_material_id( [ 10064, 10063] ) # hanger rod to bottom tungsten alloy block
vehicle.add_material_id( [ 10064, 10063] ) # hanger rod to bottom tungsten alloy block
vehicle.add_material_id( [ 10066, 10065] ) # tungsten alloy block (or lead)
vehicle.add_material_id( [ 10068, 10067] ) # hanger rod ends
vehicle.add_material_id( [ 10068, 10067] ) # hanger rod ends
vehicle.add_material_id( [ 10068, 10067] ) # hanger rod ends

##################################################
# build cylindrical cluster for the vehicle
##################################################
cyl_cluster = cls_cc.cyl_cluster(1)
cyl_cluster.set_center_point( (0.0,5.0845)  )       # x y global position

#          material    FR1    FR2 Gap/clad  W1      W2    ZrO2    RPV/Filter  Heater   SS
#  Serpent Surface     9004   9006   9008   9018    9009   9010     9012       9013    9015
cyl_cluster.set_cyl( [0.3818,0.403075,0.4750,   1.0, 1.9050, 2.5750, 3.1400,    4.0300, 4.8300] )
cyl_cluster.set_cyl_interval( [12,14,16,18,20,22,24,26,28] )
cyl_cluster.set_sectorize(2)
                                # FR1    FR2  Gap/clad  W1      W2    ZrO2   RPV/Filter  Heater   SS
cyl_cluster.set_sector_interval_az([ 24,    24,      24,  24,     24,    24,       24,      24,  24]) # curve side
cyl_cluster.set_sector_interval_r( [  8,     2,       2,   3,     3,     3,         8,      4,      4]) # stright side
cyl_cluster.set_sector_matid_at_r([True, True, False,   True,   True, False,    False, False, False]) # bool flag
cyl_cluster.set_sector_matid_at_dz([False,False,False,False,True,True,True,True,True,True,True,True,False,False,False,True,True,True,True,
True,True,True,False,False,True,True,True,True,True,True,True,True,False,False,False,False,True,True,True,True,True,True,True,False,False,False,
False,False,False,False,False,False,False]) # bool flag
```

```
# the centermost ring with sectors must have a different block number since we will use a different element type
                       # FR1 FR2  Gap/clad  W1      W2    ZrO2   RPV/Filter  Heater   SS
cyl_cluster.add_block_id( [1,7,7,7,7,7,7,7,7] ) # BFF & ref 1
cyl_cluster.add_block_id( [1,7,7,7,7,7,7,7,7] ) # BFF & ref 1
cyl_cluster.add_block_id( [1,7,7,7,7,7,7,7,7] ) # ref 2
cyl_cluster.add_block_id( [1,7,7,7,7,7,7,7,7] ) # ref 2

cyl_cluster.add_block_id( [2,8,8,8,8,8,8,8,8] ) # tang & botttom RPV 1
cyl_cluster.add_block_id( [2,8,8,8,8,8,8,8,8] ) # cup water level to heater bottom
cyl_cluster.add_block_id( [2,8,8,8,8,8,8,8,8] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_block_id( [2,8,8,8,8,8,8,8,8] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_block_id( [9,10,11,11,11,11,11,11,11] ) # Nat U 1
cyl_cluster.add_block_id( [9,10,11,11,11,11,11,11,11] ) # Enr U 1
cyl_cluster.add_block_id( [9,10,11,11,11,11,11,11,11] ) # Nat U 1
cyl_cluster.add_block_id( [2,8,8,8,8,8,8,8,8] ) # top cap to top of heater
cyl_cluster.add_block_id( [2,8,8,8,8,8,8,8,8] ) # water and top crucible
cyl_cluster.add_block_id( [2,8,8,8,8,8,8,8,8] ) # upper assembly w tank 1
cyl_cluster.add_block_id( [2,8,8,8,8,8,8,8,8] ) # upper assembly w tank 1

cyl_cluster.add_block_id( [3,12,12,12,12,12,12,12,12] ) # tang & botttom RPV 2
cyl_cluster.add_block_id( [3,12,12,12,12,12,12,12,12] ) # cup water level to heater bottom
cyl_cluster.add_block_id( [3,12,12,12,12,12,12,12,12] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_block_id( [13,14,15,15,15,15,15,15,15] ) # Nat U 2
cyl_cluster.add_block_id( [13,14,15,15,15,15,15,15,15] ) # Enr U 2
cyl_cluster.add_block_id( [13,14,15,15,15,15,15,15,15] ) # Nat U 2
cyl_cluster.add_block_id( [3,12,12,12,12,12,12,12,12] ) # top cap to top of heater
cyl_cluster.add_block_id( [3,12,12,12,12,12,12,12,12] ) # water and top crucible
cyl_cluster.add_block_id( [3,12,12,12,12,12,12,12,12] ) # upper assembly w tank 2

cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # tang & botttom RPV 3
cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # cup water level to heater bottom
cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # cup water level to heater bottom
cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_block_id( [17,18,19,19,19,19,19,19,19] ) # Nat U 3
cyl_cluster.add_block_id( [17,18,19,19,19,19,19,19,19] ) # Enr U 3
cyl_cluster.add_block_id( [17,18,19,19,19,19,19,19,19] ) # Nat U 3
cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # top cap to top of heater
cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # water and top crucible
cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # upper assembly w tank 3
cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # upper assembly w tank 3
cyl_cluster.add_block_id( [4,16,16,16,16,16,16,16,16] ) # upper assembly w tank 3

cyl_cluster.add_block_id( [5,20,20,20,20,20,20,20,20] ) # tang & botttom RPV 4
cyl_cluster.add_block_id( [5,20,20,20,20,20,20,20,20] ) # cup water level to heater bottom
cyl_cluster.add_block_id( [5,20,20,20,20,20,20,20,20] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_block_id( [21,22,23,23,23,23,23,23,23] ) # Nat U 4
cyl_cluster.add_block_id( [21,22,23,23,23,23,23,23,23] ) # Enr U 4
cyl_cluster.add_block_id( [21,22,23,23,23,23,23,23,23] ) # Nat U 4
cyl_cluster.add_block_id( [5,20,20,20,20,20,20,20,20] ) # top cap to top of heater
cyl_cluster.add_block_id( [5,20,20,20,20,20,20,20,20] ) # water and top crucible
cyl_cluster.add_block_id( [5,20,20,20,20,20,20,20,20] ) # upper assembly w tank 4
cyl_cluster.add_block_id( [5,20,20,20,20,20,20,20,20] ) # upper assembly w tank 4
cyl_cluster.add_block_id( [5,20,20,20,20,20,20,20,20] ) # upper assembly w tank 4

cyl_cluster.add_block_id( [6,24,24,24,24,24,24,24,24] ) # hanger rod to bottom tungsten alloy block
cyl_cluster.add_block_id( [6,24,24,24,24,24,24,24,24] ) # hanger rod to bottom tungsten alloy block
cyl_cluster.add_block_id( [6,24,24,24,24,24,24,24,24] ) # tungsten alloy block (or lead)
cyl_cluster.add_block_id( [6,24,24,24,24,24,24,24,24] ) # hanger rod ends
cyl_cluster.add_block_id( [6,24,24,24,24,24,24,24,24] ) # hanger rod ends
cyl_cluster.add_block_id( [6,24,24,24,24,24,24,24,24] ) # hanger rod ends

# material_ID assignment (this should match the set_sector_matid_at_dz,set_sector_matid_at_r)
                              # FR1          FR1    Gap/clad      W1         W2          ZrO2  RPV/Filter Heater  SS
          #  Serpent Surface     9004          9006     9008       9018        9009         9010    9012    9013   9015
                              #True,          True,     False,    True,     True,        False,   False,   False, False]) # bool flag
cyl_cluster.add_material_id( [10001] ) # BFF & ref 1
cyl_cluster.add_material_id( [10001] ) # BFF & ref 1
cyl_cluster.add_material_id( [10002] ) # ref 2
cyl_cluster.add_material_id( [10002] ) # ref 2

cyl_cluster.add_material_id( [ [9903,9903], [9903,9903], 9903, [9903,9903], [9903,9903],   9903,    9903,  9903, 9904 ] ) # tang & botttom RPV
cyl_cluster.add_material_id( [ [9906,9906], [9906,9906], 9906, [9906,9906], [9906,9906],   9906,    9906,  9906, 9907 ] ) # cup water level to heater bottom
cyl_cluster.add_material_id( [ [9800,9800], [9800,9800], 9800, [9801,9801], [9801,9801],   9802,    9803,  9804, 9804 ] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_material_id( [ [9800,9800], [9800,9800], 9800, [9801,9801], [9801,9801],   9802,    9803,  9804, 9804 ] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_material_id( [ [9001,9002], [9001,9002], 9060, [9071,9072], [9073,9074],   9080,    9081,  9082, 9082 ] ) # Nat U
cyl_cluster.add_material_id( [ [9101,9102], [9101,9102], 9160, [9171,9172], [9173,9174],   9080,    9081,  9082, 9082 ] ) # Enr U
cyl_cluster.add_material_id( [ [9003,9004], [9003,9004], 9061, [9075,9076], [9077,9078],   9083,    9084,  9085, 9085 ] ) # Nat U
```

```
cyl_cluster.add_material_id( [ [9840,9840], [9840,9840], 9840, [9841,9841], [9841,9841],   9842,    9843,  9844, 9844 ] )  # top cap to top of heater
cyl_cluster.add_material_id( [10017] ) # water and top crucible
cyl_cluster.add_material_id( [10019] ) # upper assembly w tank
cyl_cluster.add_material_id( [10019] ) # upper assembly w tank

cyl_cluster.add_material_id( [ [9919,9919], [9919,9919], 9919, [9919,9919], [9919,9919],   9919,    9919,  9919, 9920 ] ) # tang & botttom RPV
cyl_cluster.add_material_id( [ [9921,9921], [9921,9921], 9921, [9921,9921], [9921,9921],   9921,    9921,  9921, 9922 ] ) # cup water level to heater bottom
cyl_cluster.add_material_id( [ [9810,9810], [9810,9810], 9810, [9811,9811], [9811,9811],   9812,    9813,  9814, 9814 ] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_material_id( [ [9201,9202], [9201,9202], 9260, [9271,9272], [9273,9274],   9280,    9281,  9282, 9282 ] ) # Nat U
cyl_cluster.add_material_id( [ [9301,9302], [9301,9302], 9360, [9371,9372], [9373,9374],   9280,    9281,  9282, 9282 ] ) # Enr U
cyl_cluster.add_material_id( [ [9203,9204], [9203,9204], 9261, [9275,9276], [9277,9278],   9283,    9284,  9285, 9285 ] ) # Nat U
cyl_cluster.add_material_id( [ [9850,9850], [9850,9850], 9850, [9851,9851], [9851,9851],   9852,    9853,  9854, 9854 ] )  # top cap to top of heater
cyl_cluster.add_material_id( [10031] ) # water and top crucible
cyl_cluster.add_material_id( [10033] ) # upper assembly w tank

cyl_cluster.add_material_id( [ [9931,9931], [9931,9931], 9931, [9931,9931], [9931,9931],   9931,    9931,  9931, 9932 ] ) # tang & botttom RPV
cyl_cluster.add_material_id( [ [9933,9933], [9933,9933], 9933, [9933,9933], [9933,9933],   9933,    9933,  9933, 9934 ] ) # cup water level to heater bottom
cyl_cluster.add_material_id( [ [9933,9933], [9933,9933], 9933, [9933,9933], [9933,9933],   9933,    9933,  9933, 9934 ] ) # cup water level to heater bottom
cyl_cluster.add_material_id( [ [9820,9820], [9820,9820], 9820, [9821,9821], [9821,9821],   9822,    9823,  9824, 9824 ] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_material_id( [ [9401,9402], [9401,9402], 9460, [9471,9472], [9473,9474],   9480,    9481,  9482, 9482 ] ) # Nat U
cyl_cluster.add_material_id( [ [9501,9502], [9501,9502], 9560, [9571,9572], [9573,9574],   9480,    9481,  9482, 9482 ] ) # Enr U
cyl_cluster.add_material_id( [ [9403,9404], [9403,9404], 9461, [9475,9476], [9477,9478],   9483,    9484,  9485, 9485 ] ) # Nat U
cyl_cluster.add_material_id( [ [9860,9860], [9860,9860], 9860, [9861,9861], [9861,9861],   9862,    9863,  9864, 9864 ] )  # top cap to top of heater
cyl_cluster.add_material_id( [10045] ) # water and top crucible
cyl_cluster.add_material_id( [10047] ) # upper assembly w tank
cyl_cluster.add_material_id( [10047] ) # upper assembly w tank
cyl_cluster.add_material_id( [10047] ) # upper assembly w tank

cyl_cluster.add_material_id( [ [9943,9943], [9943,9943], 9943, [9943,9943], [9943,9943],   9943,    9943,  9943, 9944 ] ) # tang & botttom RPV
cyl_cluster.add_material_id( [ [9945,9945], [9945,9945], 9945, [9945,9945], [9945,9945],   9945,    9945,  9945, 9946 ] ) # cup water level to heater bottom
cyl_cluster.add_material_id( [ [9830,9830], [9830,9830], 9830, [9831,9831], [9831,9831],   9832,    9833,  9834, 9834 ] ) # heater bottom to bottom of fuel stack
cyl_cluster.add_material_id( [ [9601,9602], [9601,9602], 9660, [9671,9672], [9673,9674],   9680,    9681,  9682, 9682 ] ) # Nat U
cyl_cluster.add_material_id( [ [9701,9702], [9701,9702], 9760, [9771,9772], [9773,9774],   9680,    9681,  9682, 9682 ] ) # Enr U
cyl_cluster.add_material_id( [ [9603,9604], [9603,9604], 9661, [9675,9676], [9677,9678],   9683,    9684,  9685, 9685 ] ) # Nat U
cyl_cluster.add_material_id( [ [9870,9870], [9870,9870], 9870, [9871,9871], [9871,9871],   9872,    9873,  9874, 9874 ] )  # top cap to top of heater
cyl_cluster.add_material_id( [10059] ) # water and top crucible
cyl_cluster.add_material_id( [10061] ) # upper assembly w tank
cyl_cluster.add_material_id( [10061] ) # upper assembly w tank
cyl_cluster.add_material_id( [10061] ) # upper assembly w tank

cyl_cluster.add_material_id( [10063] ) # hanger rod to bottom of block
cyl_cluster.add_material_id( [10063] ) # hanger rod to bottom of block
cyl_cluster.add_material_id( [10065] ) # block
cyl_cluster.add_material_id( [10067] ) # hanger rod to plate
cyl_cluster.add_material_id( [10067] ) # hanger rod to plate
cyl_cluster.add_material_id( [10067] ) # hanger rod to plate

cyl_cluster.set_element_type( ["WEDGE","HEX","HEX","HEX","HEX","HEX","HEX","HEX","HEX"] )
vehicle.add_cyl_cluster(cyl_cluster)
core.add_assembly(vehicle)
```