ORNL/TM-2022/2601

Modeling, Performance Assessment, and Nodal Data Analysis of TRISO-Fueled Systems with Shift

Tara Pandya
Friederike Bostelmann
Matthew Jessee
Tarek Ghaddar
Philip Britt
Seth Johnson

October 14, 2022

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT–BATTELLE, LLC, FOR THE U.S. DEPARTMENT OF ENERGY

ORNL/TM-2022/2601

Nuclear Energy and Fuel Cycle Division

MODELING, PERFORMANCE ASSESSMENT, AND NODAL DATA ANALYSIS OF TRISO-FUELED SYSTEMS WITH SHIFT

Tara Pandya
Friederike Bostelmann
Matthew Jessee
Tarek Ghaddar
Philip Britt
Seth Johnson

Date Published: October 14, 2022

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

ABBREVIATIONS . . . . . **vii**
ABSTRACT **1**
1. INTRODUCTION **1**
2. MODELING TRISO-BASED SYSTEMS WITH SHIFT **2**
   2.1 SHIFT MODELING OPTIONS 2
   2.2 TRISO PARTICLE MODELING OPTIONS 6
   2.3 HTR-10 FULL CORE 6
   2.4 HTR-10 FUEL PEBBLE 9
   2.5 PBMR-400 FULL CORE 12
   2.6 TRISO HPR FULL CORE 13
   2.7 TRISO HPR UNIT CELL 13
3. SHIFT PERFORMANCE COMPARISONS **15**
   3.1 HTR-10 FUEL PEBBLE 15
   3.2 HTR-10 FULL CORE 17
   3.3 PBMR-400 FULL CORE 20
   3.4 TRISO HPR UNIT CELL 21
   3.5 TRISO HPR FULL CORE 22
   3.6 PERFORMANCE CONCLUSIONS 23
   3.7 FUTURE WORK 23
4. GENERAL SHIFT ENHANCEMENTS **26**
   4.1 VERIFICATION OF AUTOMATED CROSS SECTION GENERATION 26
   4.2 TITAN: AN ADVANCED REACTOR MULTIPHYSICS INTERFACE 28
   4.3 PATH-LENGTH SCATTERING MATRIX CALCULATION 40
5. TRISO HPR ANALYSIS **44**
   5.1 TRISO HPR UNIT CELL TWO-STEP RESULTS 44
   5.2 TRISO HPR FULL CORE TWO-STEP RESULTS 48
   5.3 FUTURE WORK 49
6. CONCLUSIONS AND FUTURE WORK **51**
   6.1 FUTURE WORK 52
7. ACKNOWLEDGMENTS **53**
REFERENCES **53**
A. TITAN OUTPUT **A–1**

LIST OF FIGURES

1 TRISO particle model. 2
2 Various methods used to model TRISO particles. 7
3 SCALE HTR-10 full-core model. 8
4 SCALE HTR-10 fuel pebble model. 9
5 SCALE PBMR-400 full-core model. 12
6 TRISO-fueled HPR full-core SCALE model. 13
7 SCALE model of a TRISO-fueled HPR unit cell. 14

8 Empire 2D assembly fuel cross section comparisons. 28
9 Empire 2D assembly fuel fission spectrum comparisons. 29
10 Empire 2D assembly moderator cross section comparisons. 30
11 Empire 2D assembly heat pipe cross section comparison. 31
12 Empire 2D assembly monolith cross section comparisons. 32
13 Empire 2D assembly fuel P_0 scattering cross section comparisons. 32
14 Empire 2D assembly moderator P_0 scattering cross section comparisons. 33
15 Empire 2D assembly heat pipe P_0 scattering cross section comparison. 33
16 Empire 2D assembly monolith P_0 scattering cross section comparisons. 34
17 The Titan pin cell model geometry. 38
18 The Titan HTR-10 pebble model geometry. 39

LIST OF TABLES

2 Overview of double-heterogeneous models for Shift performance assessment 15
3 Performance comparison for the HTR-10 fuel pebble – homogeneous 18
4 Performance comparison for the HTR-10 fuel pebble – TRISO particle lattice (no clipping) . 18
5 Performance comparison for the HTR-10 fuel pebble – TRISO particle random distribution . 19
6 Performance comparison for the HTR-10 full core 20
7 Performance comparison for the PBMR-400 full core 21
8 Performance comparison for the HPR unit cell 22
9 Performance comparison for the HPR full core 23
10 Overview of performance improvement with latest Shift development 24

11 Empire 2D assembly eigenvalue and pin power (PP) comparison 27
12 Empire 2D assembly SPH correction factor comparison 31

13 252 intermediate-group energy bounds for reaction rate tallies with Shift 45
14 252 intermediate-group energy bounds for reaction rate tallies with Shift (continued) . . . 46
15 11 coarse-group energy bounds for reaction rate tallies with Shift 47
16 Shift eigenvalue calculation parameters for TRISO HPR unit cell 47
17 Griffin calculation parameters for TRISO HPR unit cell (default) 47
18 TRISO HPR unit cell eigenvalue and pin power comparison 47
19 TRISO HPR unit cell SPH correction factor comparison 48
20 Shift eigenvalue calculation parameters for TRISO HPR full core 49
21 Griffin calculation parameters for all models (default) 49
22 TRISO HPR assembly and core models eigenvalue comparison 49

LIST OF LISTINGS

1 ORANGE *replica* geometry input for the HTR-10 fuel pebble (simplified for display). . . . 10
2 CSAS-Shift *randommix* input for the HTR-10 fuel pebble (simplified for display). 11
3 Example Shift tally input to generate multigroup cross sections. 26
4 Example Shift cross section output in HDF5 file. 27
5 Example geometry section in the Titan input file. 35
6 Example materials section in the Titan input file. 35
7 Example Shift options in the Titan input file. 36
8 Unit "W" geometry Titan output 37
9 Material Titan output 37
10 Shift Titan output 38
11 HTR-10 Titan input file. 39
A.1 Full Titan output A–2

ABBREVIATIONS

ANL	Argonne National Laboratory
API	application programming interface
CAD	computer aided design
CE	continuous energy
CPU	central processing unit
CSG	constructive solid geometry
DOE	Department of Energy
FHR	fluoride salt-cooled high temperature reactor
FY	fiscal year
HDF5	Hierarchical Data Format 5
HPC	high performance computing
HPR	heat pipe reactor
HTGR	high-temperature gas-cooled reactor
INL	Idaho National Laboratory
IRPhEP	International Reactor Physics Experiments Project
LWR	light-water reactor
M&S	modeling and simulation
MC	Monte Carlo
MG	multigroup
MPI	message passing interface
NEAMS	Nuclear Energy Advanced Modeling and Simulation
ORANGE	Oak Ridge Adaptable Nested Geometry Engine
ORNL	Oak Ridge National Laboratory
PBR	pebble-bed reactor
PWR	pressurized water reactor
RTK	reactor toolkit
SPH	super homogenization
TRISO	tristructural isotropic
UCO	uranium oxycarbide

ABSTRACT

This technical report documents several enhancements to the Shift Monte Carlo (MC) code under the US Department of Energy (DOE) Nuclear Energy Advanced Modeling and Simulation (NEAMS) program in fiscal year (FY) 2022. Performance enhancements were added to Shift specifically for tristructural isotropic (TRISO)–fueled reactor systems and guided based on performance analysis in FY 2021. For the pebble performance model developed in previous studies, the runtime improved by ~ 91× compared to the original model and ~ 2× compared to the user-optimized model. Compared to Serpent, Shift is ~ 3× slower if Serpent delta-tracking is enabled but ~ 2× faster when delta-tracking is disabled. The multigroup cross section generation was improved through simplifying tally input definitions, porting several post-processing tally operations from Python scripts into the Shift code base, and accounting for production reactions in the scattering multiplicity. Progress was also made on two emerging capabilities: (1) the development of Titan (a Shift reactor physics user interface) and (2) initial investigation into path-length tallies for computing multigroup scattering matrices.

In addition to these code enhancements, two-step neutronics calculations were performed using Shift and Griffin for a TRISO-fueled microreactor design. The results focused on the pin-homogenized approach, where Shift cross sections were used with a Griffin diffusion solver and the super homogenization (SPH) correction procedure. Results show good agreement between Shift reference solutions and Griffin, as well as Serpent solutions provided by Argonne National Laboratory (ANL).

1. INTRODUCTION

The US DOE Office of Nuclear Energy NEAMS program develops advanced modeling and simulation (M&S) tools to accelerate the deployment of advanced nuclear energy technologies. The reactor physics technical area focuses on the development of M&S tools for modeling reactor physics phenomena (i.e., neutral particle transport and isotopic depletion/decay). The three primary codes supported in the reactor physics technical area are Griffin for non–light-water reactor (LWR) reactor physics, MPACT for LWR reactor physics, and the Shift MC code used for reference solutions, ex-core dose assessment, cross section generation, and sensitivity and uncertainty assessment.

This report presents the performance enhancements, multigroup cross section production automation, and other advanced features added to Shift during FY22. This work builds upon several technical reports and the content of a published journal paper [1, 2]. The geometric tracking performance enhancements added to Shift in this work target more efficient computations of TRISO-fueled advanced reactors.

This report is organized as follows. Section 2 presents the various approaches to modeling TRISO systems in Shift, including an overview of the different options for execution, geometry input, and tracking methods. Section 2 also summarizes the different models used for performance testing, such as HTR-10 and the provided microreactor model from ANL [3, 4]. Section 3 presents detailed memory usage and runtime performance comparisons for the various TRISO models. Section 4 presents additional enhancements made to Shift this FY—including tally automation, new input interface, and scattering matrix tally improvements. Section 5 presents pin-homogenized two-step neutronics analysis with Shift and Griffin for the TRISO heat pipe reactor (HPR) system, both 3D unit cell and 3D core, along with comparisons to Serpent. Section 6 provides conclusions and recommendations for future work.

2. MODELING TRISO-BASED SYSTEMS WITH SHIFT

Recently, there has been renewed interest in advanced reactor designs using TRISO fuel particles such as pebble bed and prismatic high-temperature gas-cooled reactors (HTGRs), as well as fluoride salt-cooled high temperature reactors (FHRs). A TRISO fuel particle is roughly 1 mm in diameter and consists of a micro fuel kernel composed of uranium in oxide, carbide, or nitride form. The particle is enclosed by four concentric coatings: a porous graphite buffer, an inner pyrolytic carbon (PyC) layer, a ceramic silicon carbide (SiC) layer, and an outer pyrolytic carbon layer (Figure 1). Many thousands of these TRISO particles are distributed randomly in a graphite matrix to form a fuel component such as a pebble or a cylindrical compact. The computational modeling of so-called *double-heterogeneous systems* poses challenges for both MC and multigroup (MG) deterministic transport calculations.

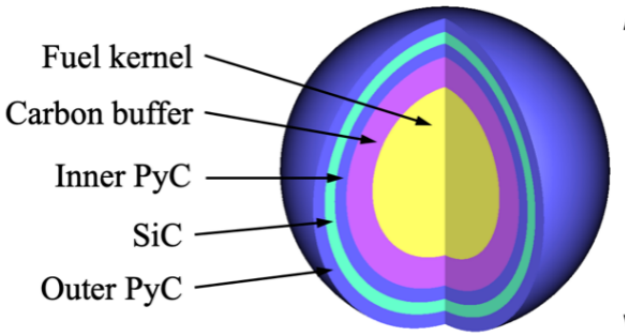


Figure 1. TRISO particle model.

2.1 SHIFT MODELING OPTIONS

There are a wide range of options for invoking `Shift` for MC analysis, based on the executable and input geometry. Moreover, there are many user and implementation options. These options are summarized below.

2.1.1 EXECUTABLE OPTIONS

`Shift` can be run in a variety of ways, most notably through the following four executables:

1. **`Omnibus`:** `Omnibus` is available through `SCALE` and is designed to launch `Shift` calculations in high performance computing (HPC) environments [5]. Runtime options such as scheduler type, number of nodes, walltime, processors per node, and message passing interface (MPI) command-line arguments are specified in the `Omnibus` input file. The `Omnibus` input file also provides control over the MC calculation, such as the number of active and inactive cycles, particles per cycle, tally definition, cross section options, etc. The model geometry and compositions can be defined directly in the `Omnibus` input syntax or provided in an existing model input file for `MCNP` or `SCALE`, as examples. Execution command: `/scale-install-path/bin/omnibus-run inputfile`
2. **`SCALE`:** `Shift` is also supported through several analysis sequences available through a `SCALE` installation. These sequences include CSAS (for criticality), TRITON (for depletion), TSUNAMI (for sensitivity and uncertainty), and MAVRIC (for adjoint-biased shielding). The number of MPI processes are specified on the command line. Additional MPI customization can be exported to the MPIARGS environment variable, such as `"-bind-to hwthread"` and through the scheduler script. Execution command: `/scale-install-path/bin/scalerte -N n inputfile`

3. **Titan:** Titan is a new reactor physics executable for Shift and will be available through upcoming releases of SCALE. Titan is described in more detail in Section 4.2.
   Execution command: `mpirun <options> /scale-install-path/bin/titan inputfile`
4. **VERA:** Shift can also be run as part of a VERA calculation with the VERAShift package. The Shift model is automatically generated based on the VERA input definition. Shift runtime options are available for customization in the Shift input block of the VERA input file.
   Execution command: `/vera-install-path/bin/verarun inputfile`

In this work, our analysis is based primarily on Omnibus and SCALE execution. Once Titan has matured, it will be adopted for future analysis with Shift.

2.1.2 GEOMETRY INPUT OPTIONS

In addition to the variety of executable options to run Shift, the Shift geometry can also be expressed through several input options. Most options rely on constructive solid geometry (CSG) principles for defining reactor geometry hierarchically through a series of units (or universes), where the unit media (or material cells) are defined through intersections or unions of input shapes (or surfaces). Each shape has a natural definition of inside or outside for media construction or unit boundary construction. Units are organized hierarchically through single-unit placement (hole) or multi-unit placement (array) of child units inside a parent unit. The input geometry options for Shift are as follows:

1. **ORANGE:** Oak Ridge Adaptable Nested Geometry Engine (ORANGE) is the underlying MC geometry engine for most Shift calculations. Most Shift calculations involve a step of converting the user input geometry into an underlying ORANGE geometry representation used for ray-tracing and MC particle tracking. In this context, the ORANGE input is an option for defining the geometry through CSG input options in the Omnibus input syntax. CSG constructs include universe, shape, cell, hole, and array. The Omnibus executable can accept an ORANGE geometry input file.
2. **KENO-VI:** KENO-VI input has been supported in SCALE since the 1990s. CSG constructs include units, shapes, media, hole, and array. The Omnibus executable can accept a SCALE input file based on KENO-VI geometry input.
3. **MCNP:** MCNP input geometry is similar in nature to KENO-VI, where universes are hierarchically constructed through surface and cell definitions. CSG constructs include universe, surface, cell, hole, and array. The Omnibus executable can accept an MCNP geometry input file.
4. **KENO-V.a:** KENO-V.a input has been supported in SCALE since the 1980s. CSG constructs include units, hole, and array. Shapes and media (or cells) are constructed implicitly in an inside-out order, where the inside shape does not intersect and is fully enclosed in the outside shape. Although the input options are less flexible compared to KENO-VI, the nested ordering of the geometry allows for optimizations in particle tracking. The Omnibus executable can accept a SCALE input file based on KENO-V.a geometry input.
5. **Titan**: Titan input geometry is described in more detail in Section 4.2.
6. **VERA**: VERA input geometry is LWR-specific and is described in the VERA input manual [6]. The VERA executable creates the Shift geometry model automatically. The Omnibus executable can also accept a VERA geometry input file (in XML format).
7. **Other:** Omnibus supports several other geometry options for computer aided design (CAD)–based modeling (DAGMC or SWORD), reactor modeling (reactor toolkit (RTK), which is pressurized water reactor (PWR)–specific), and high-energy physics modeling (GEANT4).

In this work, our analysis is based primarily on the ORANGE and KENO-VI geometry. Again, once Titan has matured, it will be used for geometry input.

2.1.3 USER MODELING STRATEGIES

In addition to the executable and geometry input options, the user can use several modeling strategies for defining the reactor geometry. These modeling choices, along with the underlying code implementation described in subsequent sections, can influence the memory and runtime performance. The basic modeling strategies include:

1. **Single unit vs. Unit tree:** Some models are simple enough to model as a single geometry unit. In theory, all models could be modeled as a single geometry unit, where repeatable geometry components (fuel rods, fuel assemblies, control drums, etc.) have a unique set of shapes and media definitions in the global unit. In complicated reactor geometries, a unit tree (i.e., hierarchy) can simplify the input requirements.
2. **Array vs. Hole:** In a unit tree, child units can be placed into parent units individually (hole) or as an arranged collection (array). All arrays can be modeled as holes, but the converse is not true. Holes are necessary in some cases.
3. **Unique material/tally regions:** In most cases, the reactor geometry needs repeatable geometry elements (e.g., fuel rods) to have unique IDs for assigning material isotopic densities and temperatures or for defining tallies of interest. These requirements can influence the user's modeling strategy.

In addition to these general modeling strategies, additional user modeling strategies should be considered that are specifically for TRISO modeling:

4. **Homogeneous vs. Heterogeneous:** In the former, TRISO particles are volume-homogenized and modeled as a single media (material cell) in the pebble or rod unit (universe). Homogeneous modeling can cause a large error in eigenvalue because of neglected spatial self-shielding of the individual TRISO particles (spatially distributed fuel causes overestimation of absorption in ^{238}U, which leads to a decrease in eigenvalue). However, the homogeneous modeling strategy can be useful for quick scoping calculations. The remaining strategies refer to heterogeneous modeling.
5. **Lattice vs. Random:** In the former, TRISO particles are placed in an array (square or hexagonal or dodecahedral). In the latter, TRISO particles are placed in randomized locations using holes such that there are no TRISO particle intersections with the parent unit boundary, sibling particles, or parent unit media (such as the FHR pebble inner graphite region). This modeling strategy can also apply to pebbles in a reactor core parent unit. The choice of using randomized locations is closer to the physical reality of most TRISO reactors/concepts.
6. **Lattice: Allow (vs. Prohibit) clipping:** The use of arrays can lead to TRISO particles intersecting with the parent unit boundary, and only the portion inside the boundary is kept in the model. In the former, the lattice pitch is adjusted to preserve the overall TRISO particle (or pebble) packing fraction. In the latter, the lattice pitch is set to yield the target TRISO particle (or pebble) packing fraction. The choice to prohibit clipping should be preferred over allowing clipping because (1) clipping can cause an unknown fuel mass to be included in the parent unit and (2) clipping can have an impact on moderation since a larger number of *partial* fuel kernels is directly surrounded by graphite moderator (this effect becomes especially pronounced in systems with thin layers of TRISO particles [7]).
7. **Random: User-input locations vs. Automated selection:** In the former, TRISO particles (or pebbles) are placed at user-specified locations. In the latter, `Shift` determines the locations based on randomization algorithms.

2.1.4 IMPLEMENTATION STRATEGIES

In addition to executable, geometry input, and user options, MC codes such as `Shift` have multiple implementation options for improved performance. These implementation options can be thought of as different algorithm strategies, some of which can be enabled or disabled as a user control option. In the options that follow, three MC operations are important to consider: `initialize-particle`, `distance-to-next-surface`, and `move-across-surface`. Given the particle position, the `initialize-particle` operation is the process of traversing the unit tree to determine the particle's geometry state (i.e., unit path and media). Given the particle's location, direction, and geometry state, the `distance-to-next-surface` operation determines the next surface crossing in the unit tree. The `move-across-surface` operation is the process of updating the particle's position and geometry state based on a surface crossing. The next surface may reside in the current unit, parent unit, or root unit boundary. Revisiting the user options above:

1. **Single unit vs. Unit tree:** In the latter, the MC operations require unit tree traversal, which requires the particle's position and direction to undergo coordinate system transformations (parent-to-child or child-to-parent). These transformations can influence runtime for deep unit tree models.
2. **Array vs Hole:** Array arrangement of child units allows for all of the MC operations to be optimized. In `move-across-surface`, the particle state can be updated based on the known adjacent unit.

The implementation strategies include the following:

3. **Partitioned vs. Unordered processing:** The former utilizes a space partitioning data structure (kd-tree, bvh, bih, uniform grid, etc.) to order the processing of a unit's media (material cells) and child units. The latter processes each child one at a time. Whereas the former is more efficient for many-children units (such as TRISO particles or pebbles), the partitioning data structures require more memory and may not lead to runtime gain for few-children units. (Note that the user has some flexibility in organizing child units spatially through the hierarchical input definition.)
4. **Allow (vs. Prohibit) child unit clipping:** Each child unit is placed in a parent unit with a closed boundary. Higher units (i.e., grandparent unit, up to root) may have boundaries that clip lower unit boundaries. Most implementations allow for boundary clipping or require clipping by design. In `distance-to-next-surface`, unit clipping requires distance checking at each level of the current particle's state because the particle may exit a parent unit before crossing the next surface in the current unit. However, some codes, such as `KENO-V.a`, prohibit unit clipping, which means distance checking is only done at the lowest unit level. In `KENO-VI`, arrays are allowed to be clipped, but holes are prohibited from clipping. This prohibition of hole clipping is a modeling limitation but provides for a runtime optimization.
5. **Surface tracking vs. delta tracking:** The former uses all of the MC operations above. The latter needs only `initialize-particle`, which is less expensive. Surface tracking can utilize path length or collision tallies, whereas delta tracking can use only collision tallies. (Path length tallies have lower variance than collision tallies for the same number of particle histories but require more runtime.) For delta tracking to be efficient, cross section access must be efficient, which generally requires more memory to store cross section data on a unified energy grid. Additionally, delta tracking efficiency is limited in models with strong absorbing materials.

As mentioned above, the `Shift` calculations herein are run through `Omnibus` or `SCALE` executables, through either the ORANGE `Omnibus` input or `KENO-VI` input. In all cases, the MC calculation uses the ORANGE geometry engine for the random walk. The ORANGE implementation for `KENO-VI` in `SCALE` 6.3 employs surface tracking, allowing unit clipping for both holes and arrays, and it uses kd-tree

partitioning for `initialize-particle`. ORANGE uses a layered approach in which particles are transported through its universe hierarchy simultaneously. Therefore, modeling approaches using holes (i.e., a universe within a universe) are more efficient compared to very complex single universes. In this work, uniform-mesh partitioning has been added for `distance-to-next-surface` acceleration in user-identified units that contain TRISO particles. This option will be available in `SCALE` 7.0.

2.2 TRISO PARTICLE MODELING OPTIONS

The following modeling options were used for the TRISO performance calculations described in Section 3.

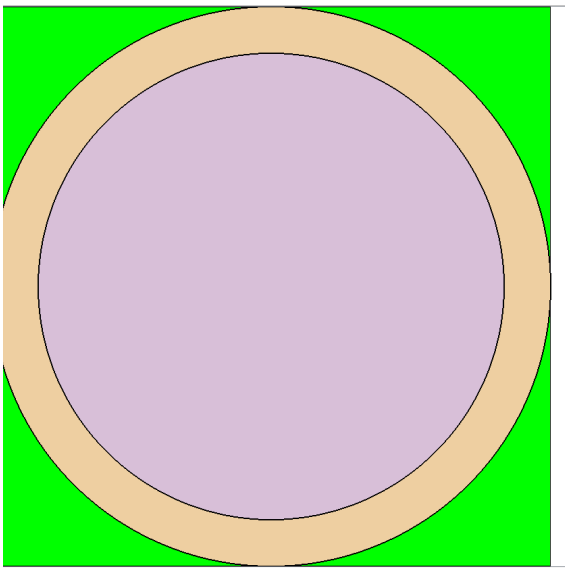
1. *homogeneous*: The TRISO particles are volume homogenized to demonstrate performance differences between a single-heterogeneous vs. a double-heterogeneous system.
2. *lattice*: The TRISO particles are placed in a square lattice, and the particles are removed at array locations that intersect the parent boundary.
3. *lattice+clipping*: The TRISO particles are placed in a square lattice and allowed to be clipped by the parent boundary.
4. *automated*: The TRISO particles are automatically determined and modeled.
5. *automated+accel*: The TRISO particles are automatically determined, and uniform grid acceleration is used for surface tracking.
6. *single-unit*: The TRISO particles are defined directly in the parent unit.
7. *single-unit+user-array*: The TRISO particles are directly defined in the parent unit, which is spatially partitioned into a user input array.
8. *multi-unit*: The TRISO particles are defined as child units inside the parent unit.
9. *multi-unit+user-array*: The TRISO particle units are spatially partitioned into a user input array.
10. *multi-unit+accel*: The TRISO particles are defined as child units inside the parent unit, and uniform grid acceleration is used for surface tracking.

Figure 2 displays some of these modeling options for a single HTGR pebble. Note that the pebble model shown in Figure 2d is the closest to reality.

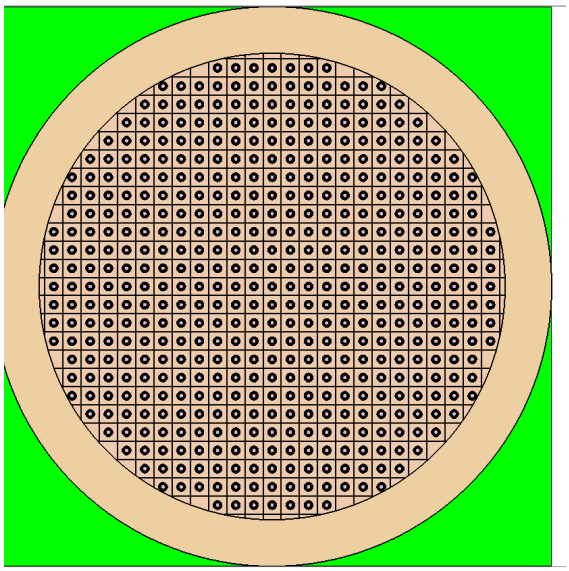
2.3 HTR-10 FULL CORE

The HTR-10 is a small 10 MWth prototype pebble-bed reactor that was operated at Tsinghua University in China. With a core diameter of 1.8 m and a mean height of 1.97 m, it contains almost 10,000 fuel pebbles surrounded by graphite reflector structures. This reactor was designed to help in the development of pebble-bed technology in China and to test fuel, safety features, operational behavior, and other factors. Construction began in 1995, first criticality was achieved in December 2000, and the reactor operated at full-power condition through January 2003 [8].

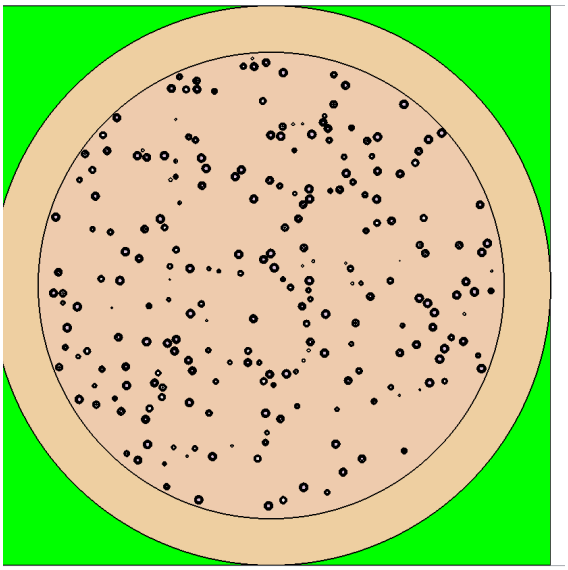
The initial critical configuration of HTR-10 was modeled based on the high-fidelity specifications provided in the International Reactor Physics Experiments Project (IRPhEP) handbook [8]. For this configuration, the conus, and discharge tube were filled with pure graphite “dummy” pebbles. The cylindrical core consisted of a mixture of 9,627 fuel pebbles and 7,263 dummy pebbles at a packing fraction of 61%. Criticality was achieved at room temperature while all control rods were withdrawn, and void spaces were filled with ambient air (Figure 3).



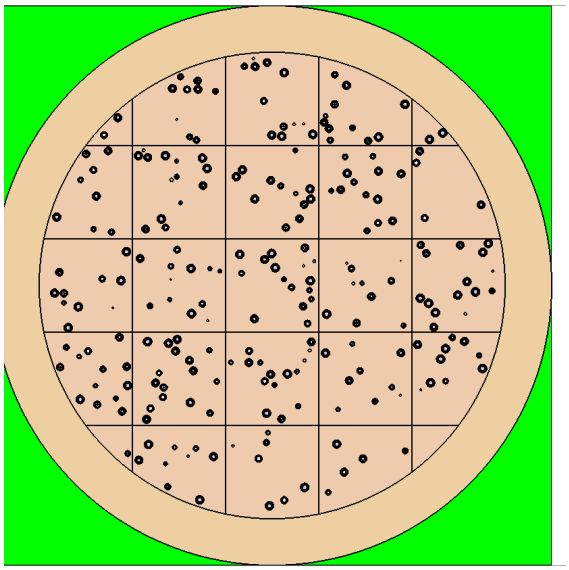
(a) Homogeneous



(b) Lattice



(c) Automated or single-unit or multi-unit



(d) Single-unit + user-array or multi-unit + user-array

Figure 2. Various methods used to model TRISO particles.

The HTR-10 fuel pebble consists of a fuel zone that is 5 cm in diameter and contains 8,385 TRISO fuel particles distributed randomly in a graphite matrix; this matrix is surrounded by a 5 mm graphite layer, which results in a pebble with a 6 cm outer diameter (Figure 4). The TRISO fuel particle is 0.91 mm in diameter and includes a microfuel kernel composed of UO_2 that is 0.5 mm in diameter. The fuel kernel is enclosed by four concentric coatings: a porous graphite buffer, an inner pyrolytic carbon layer, a ceramic silicon carbide layer, and an outer pyrolytic carbon layer. The ^{235}U enrichment of the fuel is 17 wt% in this HTR-10 configuration.

SCALE models prepared for earlier studies [9–11] were used for the present work. The pebbles were modeled in an hexagonal lattice that ensures that all pebbles are contained in the core, preventing pebble clipping by outer surfaces. The TRISO particle modeling options considered for the HTR-10 full core are *lattice*, *multi-unit*, and *multi-unit+user-array*.

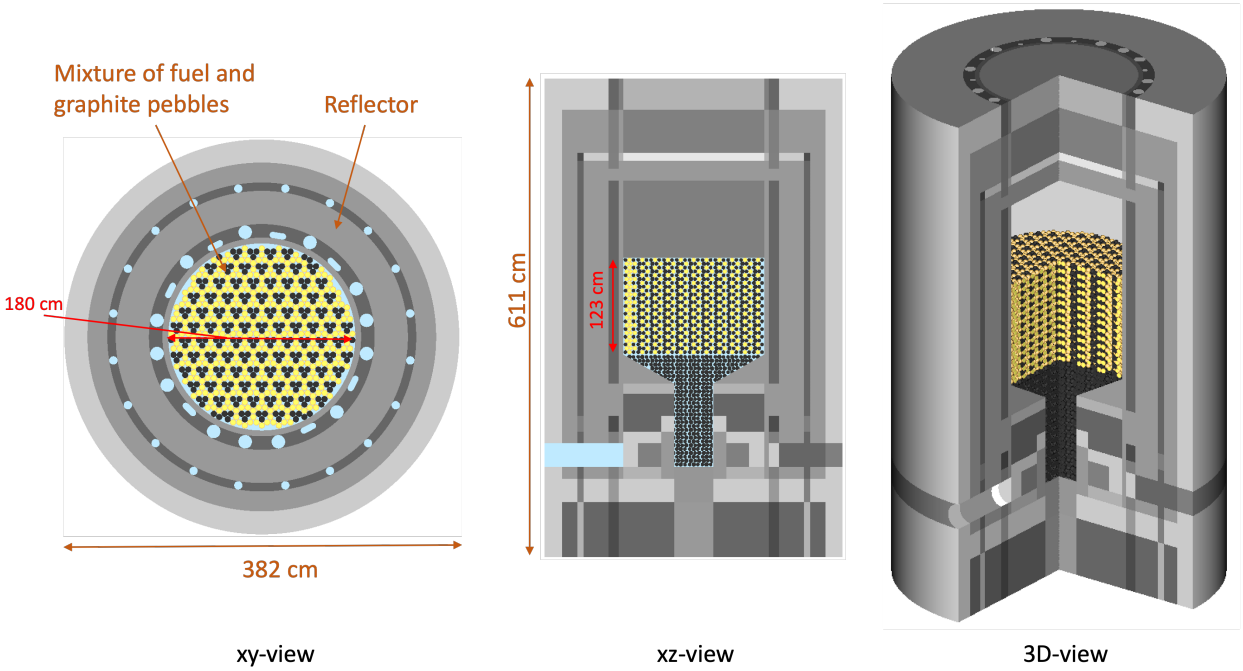


Figure 3. SCALE HTR-10 full-core model.

2.4 HTR-10 FUEL PEBBLE

Modeling and performance of a single HTR-10 fuel pebble model (Figure 4) based on the corresponding HTR-10 full-core model were investigated in detail. Reflective boundary conditions were specified on the pebble model's bounding cuboid to model an infinite, simple-cubic lattice of fuel pebbles surrounded by helium. `SCALE/KENO-VI` and `SCALE/Shift` models prepared for earlier studies [11] were used for the present work. The TRISO particle modeling options considered for the HTR-10 pebble are as follows:

- *homogeneous*,
- *lattice*,
- *automated*,
- *automated+accel*,
- *single-unit*,
- *single-unit+user-array*,
- *multi-unit*,
- *multi-unit+user-array*, and
- *multi-unit+accel*.

Examples of these modeling approaches are shown in Figure 2. Listing 1 shows a simplified version of an ORANGE geometry input for the HTR-10 pebble for the *multi-unit+accel* modeling option, which uses the *replica* keyword to activate the surface-tracking acceleration. Listing 2 shows a `SCALE` input for the *automated+accel* modeling option, which uses the *randommix* input block for instructing `Shift` to fill the pebble with TRISO particles.

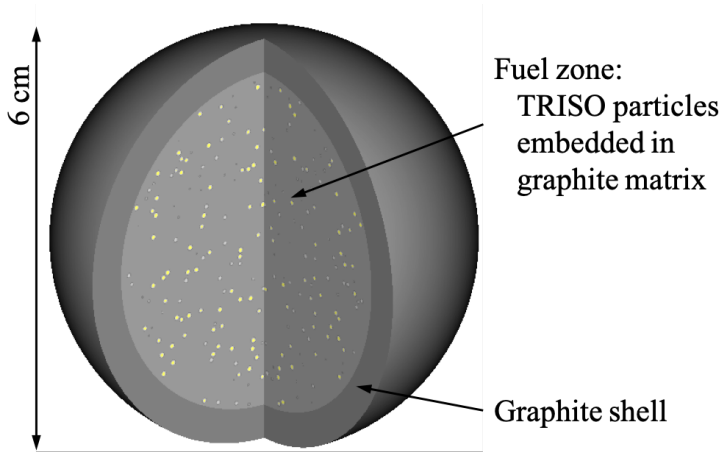


Figure 4. `SCALE` HTR-10 fuel pebble model.

Listing 1. ORANGE *replica* geometry input for the HTR-10 fuel pebble (simplified for display).

```
[GEOMETRY]
global "fuel_pebble"

! --- TRISO fuel particle ---
[UNIVERSE=general triso]
interior "or_opyc_triso"
[.][SHAPE=sphere or_opyc_triso]
radius 4.550000e-02
[..][SHAPE=sphere or_sic_triso]
radius 4.150000e-02
...
[..][SHAPE=sphere or_fuel_triso]
radius 2.500000e-02
[..][CELL opyc_triso]
comp opyc
shapes or_opyc_triso ~or_sic_triso
[..][CELL sic_triso]
comp sic
shapes or_sic_triso ~or_ipyc_triso
...
[..][CELL fuel_triso]
comp fuel_kernel
shapes or_fuel_triso

! --- TRISO fuel particles in sphere ---
[UNIVERSE=replica pebble_with_trisos]
background graphite_matrix
interior pebble_int
[.][SHAPE=sphere pebble_int]
radius 2.5
[..][HOLE triso]
x    :    y    :    z
     -6.635751446e-01  -1.608044047e+00  -1.672705785e+00
     -7.280667832e-01  -1.602319851e+00  -1.597842243e+00
     ...

! --- Fuel pebble ---
[UNIVERSE=general pebble_cell]
interior "box_pebble"
[.][SHAPE=cuboid box_pebble]
faces -3.0 3.0 -3.0 3.0 -3.0 3.0
reflect *
[..][SHAPE=sphere or_pebble]
radius 3.0
[..][HOLE pebble_hole]
fill pebble_with_trisos
[..][CELL pebble_coating]
comp graphite_shell
shapes or_pebble ~pebble_hole
[..][CELL coolant]
comp coolant
shapes box_pebble ~or_pebble
```

Listing 2. CSAS-Shift *randommix* input for the HTR-10 fuel pebble (simplified for display).

```
=csas6-shift
HTR-10 fuel pebble
ce_v7.1_endf
read composition
' fuel kernel: UO2
    u-235       100 0 3.99198e-03 300   end
    u-238       100 0 1.92441e-02 300   end
    o-16        100 0 4.64720e-02 300   end
' coatings: buffer/101, iPyC/102, SiC/103, oPyC/105
' graphite pebble shell/106
' saturated air/300
end composition

read parameter
 gen=250 npg=10000 nsk=50
end parameter

read geometry
unit 1
 com='TRISO particle'
 sphere 1    2.50e-02
 sphere 2    3.40e-02
 sphere 3    3.80e-02
 sphere 4    4.15e-02
 sphere 5    4.55e-02
 media  100 1    1
 media  101 1    2  -1
 media  102 1    3  -2
 media  103 1    4  -3
 media  104 1    5  -4
 boundary 5

global unit 10
 com='pebble'
 sphere 1    2.5
 sphere 2    3.0
 cuboid 3    6p3.0
 media  105 1    1  randommix='trisos'
 media  106 1    2  -1
 media  300 1    3  -2
 boundary 3
end geometry

read randomgeom
 randommix = 'trisos'
  type=random
  units=1 end
  pfs=0.05054954 end
  clip=no
  seed=1111
 end randommix
end randomgeom

read bounds all=refl end bounds
end data
end
```

2.5 PBMR-400 FULL CORE

The PBMR-400 design was originally considered for development by the South African utility ESKOM as part of a larger industrial consortium, PBMR Ltd. Although a reactor was never constructed, sufficient design details of the PBMR exist to provide a useful verification test case, and these details have resulted in specifications and evaluations of an international benchmark [12, 13].

The PBMR-400 design is based on a scaled-down version of prior HTGRs, designed to capture advantages and safety features of prior pebble-bed reactor (PBR) designs. The goal of the PBMR-400 design was to combine the excellent fission product retention capabilities demonstrated in TRISO-based spherical pebble fuel with the high-temperature outlet and high-efficiency Brayton cycle employed in HTGR designs. Designed for 400 MWth power output, the core consists of approximately 452,000 graphite-coated fuel pebbles, each containing approximately 15,000 UO_2 TRISO fuel particles. The core consists of an annular region between two cylindrical graphite blocks that serve as neutron moderators (an inner and outer reflector with radii 100 cm and 185 cm, respectively) with an active core height of approximately 10.117 m. Primary control through the reactor control system is provided by 24 boron carbide rods oriented equidistantly around the core in the outer reflector region at a pitch circle diameter of 3.974 m [12, 13].

A SCALE model prepared for earlier studies [14] was used as a basis for the present work (Figure 5). The pebbles were modeled in a dodecahedral lattice that permitted the clipping of pebbles by the surrounding surfaces. The TRISO particle modeling options considered for the PBMR-400 pebble are *lattice+clipping* and *multi-unit+user-array*.

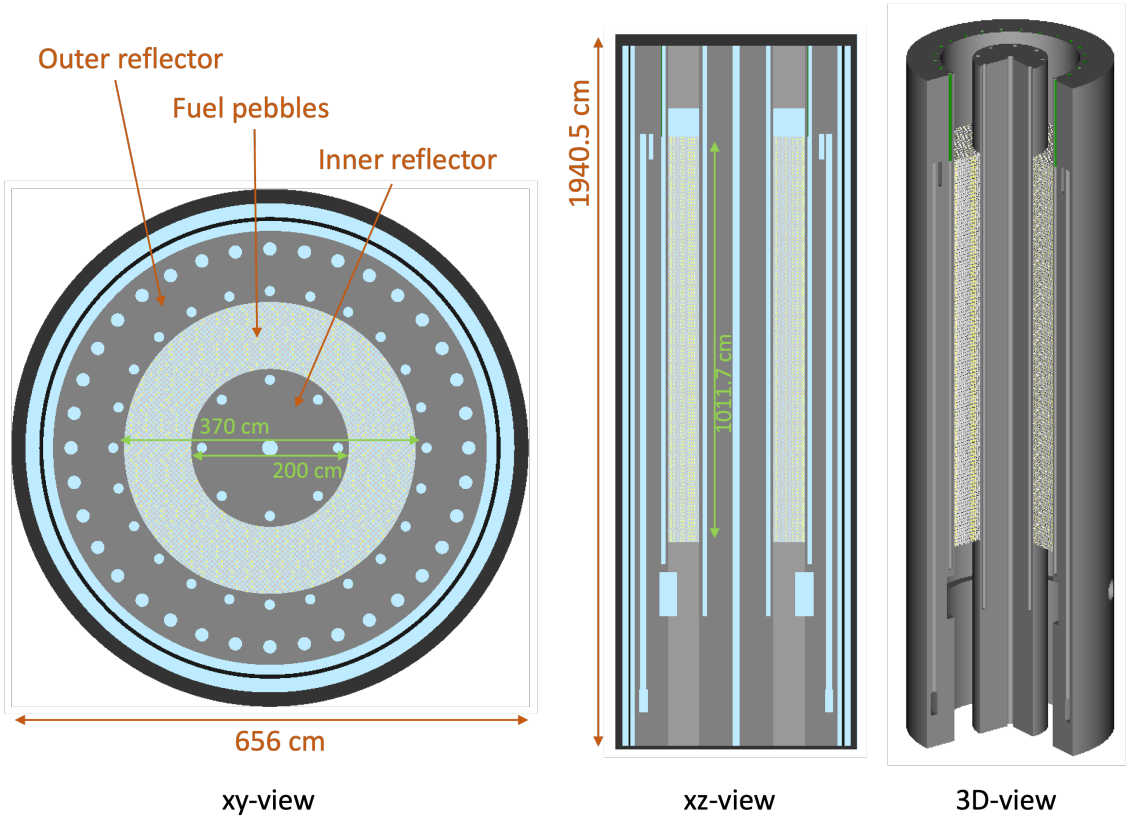


Figure 5. SCALE PBMR-400 full-core model.

2.6 TRISO HPR FULL CORE

ANL developed a microreactor concept as a modeling exercise that combines the use of heat pipe technologies to remove the nuclear heat, the use of TRISO fuel particles to enable operations at very high temperatures, and the use of rotating control drums in the radial reflectors [4]. This 2 MWth HPR concept comprises 30 hexagonal fuel assemblies surrounded by one ring of beryllium reflectors, 12 of which contain control drums. The active core length is 160 cm, axially surrounded by lower and upper beryllium reflectors of 20 cm thickness (Figure 6).

The fuel assembly consists of a graphite monolith that contains heat pipes, cylindrical fuel components, and moderator rods. The fuel components consist of TRISO fuel particles that are dispersed in a graphite matrix at a packing fraction of 40%. The TRISO fuel particles contain 19.95 at.% enriched uranium oxycarbide (UCO). The core is cooled through a potassium heat transfer fluid in heat pipes that are encompassed by stainless-steel envelopes. In addition to the moderation through the graphite structure, yttrium hydride (YH_2) pins surrounded by stainless-steel envelopes provide additional moderation to allow a more compact core (Figures 6 and 7) [4].

The SCALE model development benefited greatly from the Serpent models shared by ANL to support this work [3, 4, 15]. The TRISO particle modeling options considered for the HPR full-core model are *lattice* and *multi-unit+user-array*. These modeling strategies were chosen based upon the limited amount of time for developing the full core models.

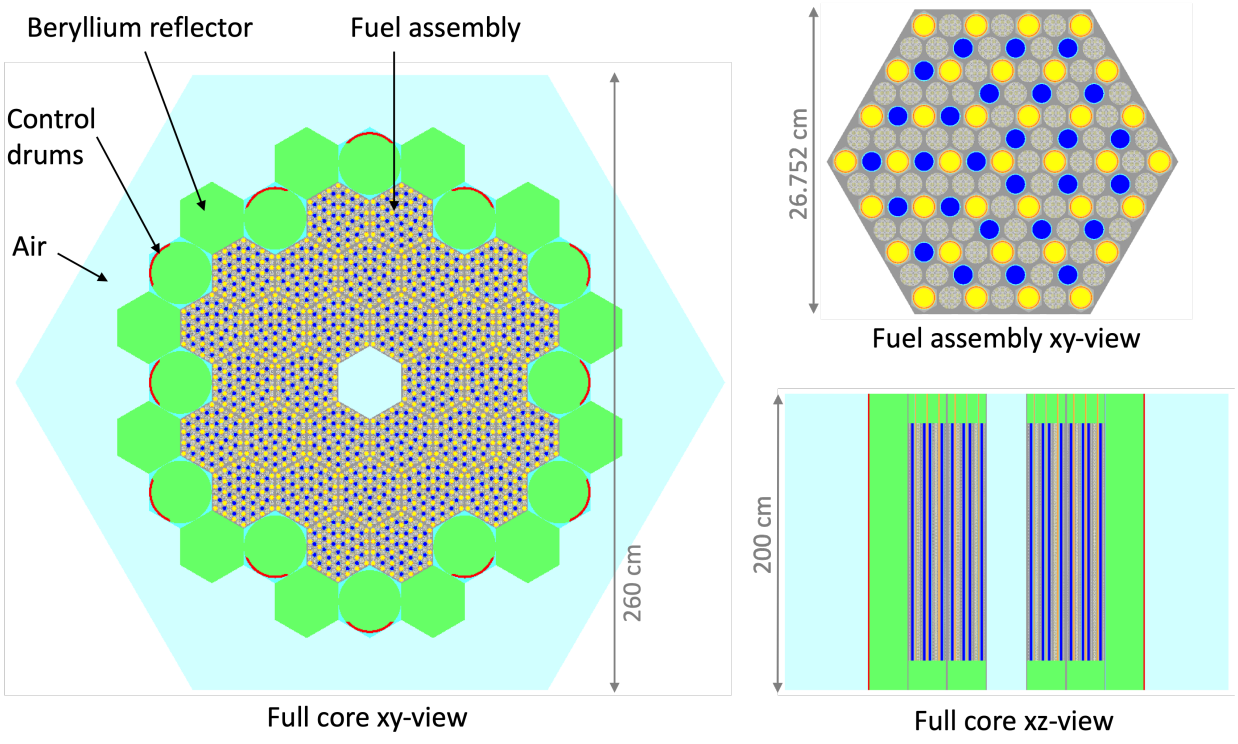


Figure 6. TRISO-fueled HPR full-core SCALE model.

2.7 TRISO HPR UNIT CELL

In addition to the TRISO-fueled HPR full core, ANL developed a 3D unit cell model based on the full core to allow for investigations of code performance and capabilities at reduced computational costs while

considering major modeling challenges, including explicit modeling of the TRISO fuel particles [4]. The general unit cell consists of a fuel rod surrounded by moderator rods and heat pipes in a hexagonal lattice. The active height and the lower and upper axial beryllium reflectors are consistent with the full core. While vacuum boundary conditions are applied in the axial direction, reflected radial boundary conditions create an infinite lattice of the fuel, heat pipe, and moderator rods (Figure 7).

The `Shift` model includes the fuel cylinder in the center of the unit cell, whereas ANL included the heat pipe in the center. This model change was chosen purely to reduce the modeler time of only needing to model one fuel cylinder. Due to the radially reflected boundary conditions, the ANL and `Shift` models are still consistent. Furthermore, the `SCALE` model development strictly followed ANL's provided `Serpent` models instead of details in the report [4], in which the unit cell includes simplifications of the moderator and heat pipe shells where the stainless-steel envelopes were homogenized with the surrounding gaps. However, such details have a negligible impact on performance assessment, which was the goal of this work.

The TRISO particle modeling options considered for the HPR unit cell are *lattice*, *multi-unit*, *multi-unit+user-array*, and *multi-unit+accel*.

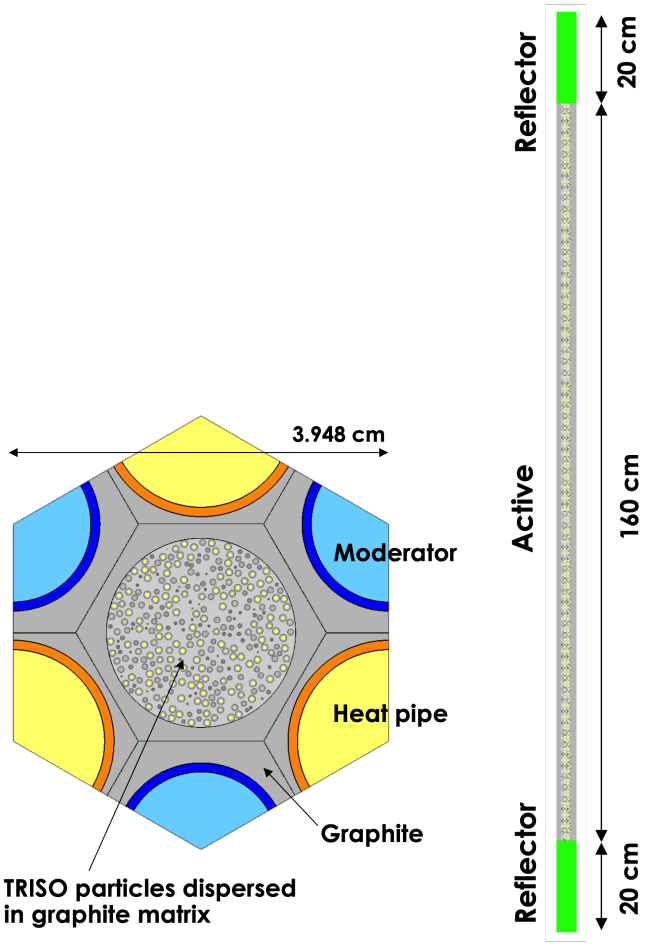


Figure 7. SCALE model of a TRISO-fueled HPR unit cell.

3. SHIFT PERFORMANCE COMPARISONS

Limited studies exist that use Shift to model HTGR fuel systems [1, 11]. In [11], Shift was applied to calculate the initial criticality of the HTR-10 benchmark and to run performance tests for a single fuel pebble model. Consistent results were obtained between Shift, SCALE's KENO-VI [16] MC code, and the Serpent [17] MC code. In [1], more detailed performance assessments were conducted for a single HTR-10 fuel pebble model, focusing on Shift within development versions of SCALE 6.3. It was found that the modeling choices described above can lead to largely different runtimes.

The study herein provides memory and runtime performance comparisons for the models listed in Table 2 and introduced in Section 2. The employed HTR-10 fuel pebble and full-core models are consistent with the models used for previous performance assessments in [1, 11]. Double-heterogeneous models provide challenges for the MC transport calculation, owing to the large number of surfaces and cells required to model the large number of TRISO fuel particles distributed in the fuel region. For the full-core models, the large number of fuel pebbles, each filled with a large number of particles, adds to this challenge.

All calculations presented here were run at Oak Ridge National Laboratory (ORNL) on a Linux cluster of Intel(R) Xeon(R) Gold 5118 central processing units (CPUs) running at 2.3 GHz. Each node consists of 48 CPUs with 192 GB of memory available per node. The full node was reserved when running these calculations, but only 1 out of 48 CPUs was utilized to mitigate the runtime complexities and performance implications of parallel calculations. The majority of the Shift calculations were performed using SCALE 6.3.0. To assess the latest implementations that enable additional input options through an ORANGE input and that enable various performance improvements, comparisons were added based on the latest development version of SCALE 7.0 (pre-beta 3). The ENDF/B-VII.1 nuclear data library was used for all calculations.

Table 2. Overview of double-heterogeneous models for Shift performance assessment

Model	# particles in pebble/rod	particle packing fraction	# pebbles/rods in model
HTR-10 pebble [8]	8,385	5.05%	1
HTR-10 full core [8]	8,385	5.05%	9,627
PBMR-400 full core [12, 13]	15,000	9.34%	452,000
TRISO-fueled HPR unit cell [4]	624,522	40.66%	1
TRISO-fueled HPR full core [4]	624,522	40.66%	1,890

3.1 HTR-10 FUEL PEBBLE

The HTR-10 fuel pebble was the first model analyzed for Shift performance assessment of computation time, memory, and reactivity prediction, as described in Section 2.4. A single-heterogeneous model with a smeared-fuel region and various double-heterogeneous models with explicit consideration of the fuel particles in the fuel region were developed.

Comparisons of Shift calculations were made between various model variations, SCALE 6.3.0, the latest development version of SCALE 7.0 (pre-beta 3), SCALE's MC code KENO-VI, Serpent, and MCNP. For Serpent, calculations were performed with delta tracking and with surface tracking using Serpent 2.1.30 [17]. The MCNP calculations were performed using MCNP 6.1 [18]. All MC calculations for this

model were run using 10,000 neutrons per cycle with 200 active cycles and 50 inactive cycles. These parameters were chosen only to show performance comparisons.

The performance metrics in Tables 3, 4, and 5 show that the single-heterogeneous, volume-homogenized (*homogeneous*) models are faster and use less memory than the double-heterogeneous models. All codes show similar computation times and memory footprints for the calculation of the volume-homogenized model. The additional surfaces in the double-heterogeneous models cause significant increases in computation time and memory use depending on the choice of code and modeling. The eigenvalue for the volume-homogenized model is consistently different from any of the double-heterogeneous models. This is a well known impact of TRISO fuel, owing to the spatial shielding effects of the fuel that cannot be captured by volume homogenizing directly. This case is only used to provide a baseline computation time for traditional single-heterogeneous fuel components.

When using the same `SCALE` version, the `Shift` calculations that use the `SCALE` frontend show the same performance compared to those that use the native `Omnibus` frontend as expected.

When running `Shift` through CSAS, the *automated+accel* model resulted in a runtime improvement compared to the explicit TRISO particle models (i.e., no mesh) because the applied geometry conditions in this block allowed for fewer geometry checks compared to an explicit model. Latest acceleration enhancements in `Shift` enable a dramatic reduction in runtime from approximately 58 hours to less than 2 hours. This acceleration makes this modeling option the most user-friendly modeling option for double-heterogeneous models while enabling a very reasonable runtime.

For the *single-unit* and *multi-unit* models, the subdivision of the fuel region into multiple zones using an overlaid mesh (as described in Section 2.2) caused a significant runtime decrease because the distance to boundary check is performed for TRISO particles within each mesh cell only, as opposed to checking the entire set of TRISO particles within the fuel region. Consequently, the *single-unit+user-array* calculations ran much faster than ones for the *single-unit* model, and the *multi-unit+user-array* models showed a significantly decreased runtime over the *multi-unit* models. Naturally, there is potential to further improve the runtime by optimizing the overlaid mesh size.

The *single-unit* models with particles modeled by explicitly specifying individual surfaces showed runtimes that were longer by a factor of approximately 11 compared to the *multi-unit* models that define one particle and place this particle into the fuel region multiple times via holes (as described in Section 2.2). The geometry engine ORANGE uses a layered approach in which particles are transported through its universe hierarchy simultaneously. Therefore, modeling approaches using holes (i.e., a universe within a universe) are more efficient compared to very complex single universes.

It is especially worth noting that the runtime and memory footprint for models with random particle distributions realized via the *multi-unit+user-array* approach, the *automated+accel*, or via the *multi-unit+accel* capability are similar or reduced with the latest `Shift` developments compared to the corresponding particle lattice models.

When comparing `Shift`'s performance between `SCALE` version 6.3.0 and the latest developments, the implementation of acceleration techniques caused runtime and memory reductions for all models. The runtime for the models besides the *automated+accel* model is reduced between 9 and 36%, and the memory is improved by 22 to 33%, depending on the modeling option. As mentioned above, the runtime for the *automated+accel* model was dramatically improved. The *multi-unit+accel* modeling option via an ORANGE geometry input yields an even more reduced runtime—the fastest runtime between all models with random particle distributions. The *automated+accel* and *multi-unit+accel* modeling options use the same geometry hierarchy and are constructed in an identical fashion; however, they use slightly different

algorithms for detecting unit boundary intersections. This difference leads to better runtimes for the *multi-unit+accel* model.

Comparing SCALE/Shift with SCALE/KENO-VI, similar runtimes were obtained for the *homogeneous* case and the lattice case, respectively. However, for the model with random particle locations, the *single-unit+user-array* model, KENO-VI required a substantially longer runtime compared to Shift. The KENO-VI calculations for the other random particle models took significantly more time—owing to the significant processing time required for the geometry—and were not completed in time for this report.

Looking at the performance of Shift vs. Serpent using the traditional surface tracking, which is also used in Shift, similar performance was obtained for the *homogeneous* models and the lattice models. However, Serpent clearly outperformed Shift when using models with random particle distributions within one fuel region (i.e., no subdivision with an overlaid mesh). The Shift-6.3.0 *automated+accel* and *multi-unit* models resulted in computation times 19× and 38× longer than the Serpent random calculation, respectively. The difference for the *multi-unit* model was slightly reduced to a factor of 23 when using Shift-7.0 beta. The subdivision of the fuel region into multiple zones using an overlaid mesh caused a significant runtime decrease because the distance to the boundary check is performed only for TRISO particles within each mesh cell, as opposed to checking the entire set of TRISO particles within the fuel region. Using holes to place TRISO particles into such a coarse mesh resulted in up to 32% shorter runtimes and up to a 27% smaller memory footprint of Shift compared with Serpent. When running Shift with the new *multi-unit+accel* capability, the fastest runtime was achieved, taking only 42% of Serpent's time with surface tracking. When running Shift with the *automated+accel* model, the performance is only slightly reduced.

By default, Serpent uses Woodcock delta tracking. Delta tracking provides significant advantages, especially for double-heterogeneous systems with thousands of fuel particles, because particles travel over many material boundaries [19]. For the TRISO particle lattice or the random TRISO particle distribution, the calculations using delta tracking took only 15%–20% of the time required by those using surface tracking. Although the runtimes for double-heterogeneous systems with Shift were previously multiple times slower than this fastest option for Serpent calculations, the latest acceleration techniques in Shift reduce this difference to Shift running only approximately 2.9 (*multi-unit+accel*) and 4.1 (*automated+accel*) times slower than the fastest Serpent calculation. Due to the general and wide range of Shift's application areas, as demonstrated in Section 2.1.4, it was decided not to implement Woodcock delta tracking in Shift. Implementing this type of tracking would limit the scope of applications and geometry models that could be run with Shift.

In addition to delta tracking, Serpent uses a unionized energy grid constructed by merging the energy grids of all nuclides in the problem [20]. This approach can significantly reduce computation time because it minimizes the number of grid search iterations. However, because the model investigated in this study has only a small number of materials and nuclides, the benefits of using a unionized grid are not significant in this particular case.

Despite using the same MC parameters, Shift shows larger standard deviations, σ, than Serpent and MCNP by a factor of 3. A true variance assessment based on 25 simulations using different random seeds for the Shift and Serpent *multi-unit+accel* models, respectively, confirmed the listed estimated k_{eff} standard deviations. The average result of the 25 Shift calculations is provided as reference in the first row of Table 5. Further investigations are needed to understand Shift's larger variances.

3.2 HTR-10 FULL CORE

In addition to a single HTR-10 fuel pebble, the HTR-10 full core was used for Shift performance assessments based on the model described in Section 2.3. Besides a model with a particle lattice, models

Table 3. Performance comparison for the HTR-10 fuel pebble – homogeneous

Code	k_{inf}	σ	memory [GB]	runtime [hh:mm:ss]
CSAS/Shift-6.3.0	1.56578	0.00124	0.439	0:36:02
Omnibus/Shift-6.3.0, KENO	1.56640	0.00124	0.426	0:33:59
Omnibus/Shift-7.0, KENO	1.56640	0.00124	**0.286**	**0:29:46**
CSAS/KENO-VI-6.3.0	1.56606	0.00047	0.458	0:23:40
Serpent-2.1.30 Delta tracking	1.56751	0.00039	0.310	0:18:32
Serpent-2.1.30 Surface tracking	1.56726	0.00038	0.310	0:24:28
MCNP-6.1	1.56739	0.00051	0.064	0:24:38

Table 4. Performance comparison for the HTR-10 fuel pebble – TRISO particle lattice (no clipping)

Code	k_{inf}	σ	memory [GB]	runtime [hh:mm:ss]
CSAS/Shift-6.3.0	1.67411	0.00136	0.457	1:54:03
Omnibus/Shift-6.3.0, KENO	1.67469	0.00120	0.436	1:52:43
Omnibus/Shift-7.0, KENO	1.67469	0.00120	**0.293**	**1:42:34**
CSAS/KENO-VI-6.3.0	1.67468	0.00043	0.467	1:26:20
Serpent-2.1.30 Delta tracking	1.67402	0.00040	0.402	0:23:34
Serpent-2.1.30 Surface tracking	1.67321	0.00037	0.402	2:03:12
MCNP-6.1	1.67443	0.00042	0.068	2:35:23

Table 5. Performance comparison for the HTR-10 fuel pebble – TRISO particle random distribution

Code	Model	k_{inf}	σ	memory [GB]	runtime [hh:mm:ss]
Reference Omnibus/Shift-7.0, ORANGE	multi-unit+accel	1.67575	0.00024	–	–
CSAS/Shift-6.3.0	multi-unit	1.67691	0.00132	0.500	112:12:46
CSAS/Shift-6.3.0	multi-unit+user-array	1.67484	0.00125	0.508	2:33:47
CSAS/Shift-6.3.0	single-unit	1.67655	0.00115	0.912	1176:25:59
CSAS/Shift-6.3.0	single-unit+user-array	1.67423	0.00122	0.893	29:10:14
CSAS/Shift-6.3.0	automated	1.67511	0.00131	**0.455**	**58:10:37**
Omnibus/Shift-6.3.0, KENO	multi-unit	1.67591	0.00122	0.440	109:58:39
Omnibus/Shift-6.3.0, KENO	multi-unit+user-array	1.67599	0.00119	0.441	2:31:02
Omnibus/Shift-6.3.0, KENO	single-unit*	1.67292	0.00132	0.485	1167:21:00
Omnibus/Shift-6.3.0, KENO	single-unit+user-array	1.67718	0.00116	0.490	28:52:13
Omnibus/Shift-7.0, KENO	multi-unit	1.67591	0.00122	0.296	70:25:14
Omnibus/Shift-7.0, KENO	multi-unit+user-array	1.67599	0.00119	**0.297**	**1:58:45**
Omnibus/Shift-7.0, KENO	single-unit	1.67230	0.00121	0.429	771:04:42
Omnibus/Shift-7.0, KENO	single-unit+user-array	1.67718	0.00116	0.383	24:16:12
CSAS/Shift-7.0	automated+accel	1.67280	0.00137	**0.451**	**1:42:28**
Omnibus/Shift-7.0, ORANGE	multi-unit+accel	1.67420	0.00117	**0.294**	**1:13:25**
CSAS/KENO-VI-6.3.0	single-unit+user-array	1.67571	0.00049	0.827	248:17:04
Serpent-2.1.30 Delta tracking	multi-unit	1.67638	0.00037	0.408	0:24:56
Serpent-2.1.30 Surface tracking	multi-unit	1.67538	0.00039	0.408	2:53:47
MCNP-6.1	single-unit	1.67632	0.00046	0.253	26:44:10

* Due to the very long runtime, this calculation was not completed in time for the release of this report. The memory footprint and runtime for this calculation were estimated based on about 70% completed neutron cycles.

with random particle locations via holes (*multi-unit*) and holes with an overlaid mesh (*multi-unit+user-array*) were used for this assessment. All MC calculations for this model were run using 50,000 neutrons per cycle, with 200 active cycles and 50 inactive cycles. Again these parameters were chosen only for performance comparisons and not for convergence.

The conclusions for these calculations are similar to those of the HTR-10 pebble. The performance is almost consistent between running `Shift` via CSAS or `Omnibus`. For the model with a particle lattice, the `KENO-VI` calculation is approximately 67% slower than the corresponding `Shift` calculation (Table 6). Due to different geometry restrictions in `KENO-VI` compared to `Shift`—in particular, the intersection of holes with outer boundaries of individual units—`KENO-VI` was not run for any other model variations.

When modeling the particles via *multi-unit* with `Shift`, an overlaid *mesh* (i.e., the *multi-unit+user-array* model) causes a significant runtime improvement, although with a factor of 6, it is less dramatic as that for the pebble, which showed a factor of 44. The application of the latest development version of `Shift` led to a memory decrease of approximately 33% and a slight runtime decrease that was most visible for the *multi-unit* model (35% less runtime) (Table 6).

Just as for the HTR-10 single pebble, the latest development efforts in `Shift` cause the runtime and memory footprint for models with random particle distributions modeled via the *multi-unit+user-array* approach to be similar or reduced when compared to the corresponding particle lattice models.

Table 6. Performance comparison for the HTR-10 full core

Code	Model	k_{inf}	σ	memory [GB]	runtime [hh:mm:ss]
CSAS/`Shift`-6.3.0	lattice	1.00208	0.00046	0.357	28:46:02
`Omnibus`/`Shift`-6.3.0, KENO	lattice	1.00179	0.00042	0.299	28:34:49
`Omnibus`/`Shift`-7.0, KENO	lattice	1.00195	0.00041	**0.199**	**27:01:07**
CSAS/`KENO-VI`-6.3.0	lattice	1.00343	0.00045	0.520	48:08:06
CSAS/`Shift`-6.3.0	multi-unit	1.00337	0.00043	0.401	186:04:40
CSAS/`Shift`-6.3.0	multi-unit+user-array	1.00250	0.00043	0.410	29:44:15
`Omnibus`/`Shift`-6.3.0, KENO	multi-unit	1.00261	0.00044	0.303	182:59:26
`Omnibus`/`Shift`-6.3.0, KENO	multi-unit+user-array	1.00228	0.00040	0.303	29:40:21
`Omnibus`/`Shift`-7.0, KENO	multi-unit	1.00315	0.00042	0.201	119:17:06
`Omnibus`/`Shift`-7.0, KENO	multi-unit+user-array	1.00243	0.00037	**0.202**	**27:27:45**

3.3 PBMR-400 FULL CORE

The PBMR-400 full-core model adds diversity to the performance assessments by incorporating a very large number of fuel pebbles in the model (452,000 as compared to less than 10,000 for the HTR-10 full core). Namely, this diversity is achieved by including 110 different fuel pebbles with fuel compositions according to 110 zones in the reactor core that include different levels of average burnup, as well as by including operating temperatures as compared to room temperature for HTR-10. Because of the complexity of the model, the particles were only modeled in a regular square lattice that follows the average packing fraction of the particles in the pebble while permitting particle clipping by the surrounding surface. All MC calculations for this model were run using 50,000 neutrons per cycle with 200 active cycles and 50 inactive cycles.

The KENO-VI calculation of the lattice model was 72% slower than the corresponding Shift calculation (Table 7). Shift's memory footprint and runtime were decreased by approximately 15% when using the latest development version. For the *multi-unit+user-array* model, the memory footprint did not change with the latest development version, but the runtime decreased by 23%.

Table 7. Performance comparison for the PBMR-400 full core

Code	Model	k_{inf}	σ	memory [GB]	runtime [hh:mm:ss]
CSAS/Shift-6.3.0	lattice	0.98373	0.00038	3.616	29:16:06
Omnibus/Shift-6.3.0, KENO	lattice	0.98385	0.00035	2.435	28:10:55
Omnibus/Shift-7.0, KENO	lattice	0.98351	0.00037	**2.104**	**24:02:18**
CSAS/KENO-VI-6.3.0	lattice	0.98316	0.00034	1.449	50:19:42
Omnibus/Shift-6.3.0, KENO	multi-unit+user-array	0.98700	0.00036	12.412	47:15:18
Omnibus/Shift-7.0, KENO	multi-unit+user-array	0.98609	0.00031	**12.253**	**36:37:18**

3.4 TRISO HPR UNIT CELL

The TRISO fueled HPR unit cell adds a computational challenge due to its very large number of TRISO particles (>600,000), which are densely packed in a long cylindrical fuel region. All MC calculations for this model were run using 10,000 neutrons per cycle with 200 active cycles and 50 inactive cycles.

The conclusions for the performance assessment are similar to those of the HTR-10 fuel pebble. When modeling the particles in a random distribution via holes, overlaying the model with a mesh reduces the runtime dramatically; the *multi-unit* models run approximately 43 times slower than the *multi-unit+user-array* models (Table 8). Using the latest development version of Shift reduces the runtime by 16% for the *multi-unit+user-array* model and 35% for the *multi-unit* model.

In contrast to the previous models, all models with random particle distributions showed longer runtimes compared to the lattice models (Table 8). This is most likely caused by the fact that less effort has been spent on optimizing the overlaid mesh compared to the pebble models. The 180 cm long cylindrical fuel region was divided into 32 axial zones because this is the discretization needed to obtain flux for SPH factors [21] in Section 5.1. However, no effort was taken to further improve the mesh axially, and it is anticipated that a better mesh could further reduce the runtime to be closer to the particle lattice model's runtime.

The *multi-unit+accel* capability through the ORANGE geometry input provides a convenient way to provide the individual particle locations without specifying an overlaid mesh in a holes input (which requires significant modeling effort) while providing an excellent runtime close to the *multi-unit+user-array* model's runtime, as well as a much reduced memory footprint.

In contrast to the previous models, the KENO-VI calculation of the lattice model ran approximately 25% faster than the corresponding Shift calculation. However, the runtime with Shift's latest development version already reduced this gap by 10%, and it is anticipated that future optimization implementations will further reduce the runtime.

The Shift *multi-unit+user-array* model shows similar runtime compared to Serpent when using surface tracking. Serpent's delta tracking calculation shows a runtime that is approximately 8 times shorter, thus outperforming all other calculations. The ratio of the runtime between the *multi-unit+accel* model and

Serpent's delta tracking calculation is about 12, whereas it was about 4 for the HTR-10 fuel pebble. This is caused by a larger packing fraction of 40% in this model compared to about 5% for the HTR-10 fuel pebble.

Table 8. Performance comparison for the HPR unit cell

Code	Model	k_{inf}	σ	memory [GB]	runtime [hh:mm:ss]
CSAS/Shift-6.3.0	lattice	0.97520	0.00071	1.022	1:46:57
Omnibus/Shift-6.3.0, KENO	lattice	0.97539	0.00054	0.849	1:46:43
Omnibus/Shift-7.0, KENO	lattice	0.97516	0.00054	**0.595**	**1:35:38**
CSAS/KENO-VI-6.3.0	lattice	0.97497	0.00052	0.594	1:11:23
CSAS/Shift-6.3.0	multi-unit	0.97637	0.00068	5.723	132:24:49
CSAS/Shift-6.3.0	multi-unit+user-array	0.97663	0.00058	9.796	3:04:23
Omnibus/Shift-6.3.0, KENO	multi-unit	0.97611	0.00058	3.415	131:09:35
Omnibus/Shift-6.3.0, KENO	multi-unit+user-array	0.97547	0.00059	6.068	3:13:18
Omnibus/Shift-7.0, KENO	multi-unit	0.97634	0.00056	3.419	85:14:45
Omnibus/Shift-7.0, KENO	multi-unit+user-array	0.97612	0.00055	**6.035**	**2:42:30**
Omnibus/Shift-7.0, ORANGE	multi-unit+accel	0.97509	0.00062	**1.100**	**3:22:40**
Serpent-2.1.30 Delta tracking	multi-unit	0.97801	0.00052	2.238	0:17:25
Serpent-2.1.30 Surface tracking	multi-unit	0.97815	0.00048	2.238	2:23:38

3.5 TRISO HPR FULL CORE

In addition to a single TRISO-fueled HPR unit cell, the corresponding full core was used for Shift performance assessments based on the model described in Section 2.6. Besides a model with a particle lattice, a model with random particle locations modeled via holes with an overlaid mesh (*multi-unit+user-array*) was used for this assessment. All MC calculations for this model were run using 50,000 neutrons per cycle with 200 active cycles and 50 inactive cycles.

The conclusions for these calculations are similar to results from the TRISO-fueled HPR unit cell. For the *multi-unit+user-array* model, the application of the latest development version of Shift did not lead to a noticeable memory decrease, and the runtime decreased only by approximately 7% (Table 9). However, for the lattice model, the memory footprint decreased by 33%, and the runtime decreased by approximately 61%. This runtime decrease is the largest observed for any lattice models.

Table 9. Performance comparison for the HPR full core

Code	Model	k_{inf}	σ	memory [GB]	runtime [hh:mm:ss]
Omnibus/Shift-6.3.0, KENO	lattice	1.06161	0.00035	0.873	24:05:44
Omnibus/Shift-7.0, KENO	lattice	1.06104	0.00033	**0.610**	**9:19:41**
Omnibus/Shift-6.3.0, KENO	multi-unit+user-array	1.06229	0.00035	6.072	31:51:22
Omnibus/Shift-7.0, KENO	multi-unit+user-array	1.06225	0.00034	**6.040**	**29:35:30**
Serpent-2.1.30 Delta tracking	multi-unit	1.06204	0.00023	2.771	7:51:48
Serpent-2.1.30 Surface tracking	multi-unit	1.06214	0.00024	2.771	49:35:32

3.6 PERFORMANCE CONCLUSIONS

The performance comparisons using models with varying complexity and with different modeling strategies to realize explicit TRISO particles in random distributions led to the following conclusions:

- The latest acceleration techniques implemented in Shift for SCALE 7.0 reduce the runtime by up to 61% and the memory by up to 34% compared to SCALE 6.3.0 for all modeling options besides the *automated+accel* modeling strategies. For the *automated+accel* simulations, a dramatic reduction in runtime by a factor of 34 was achieved.
- The *randommix* block that is available through SCALE's Shift interface is the most user-friendly modeling option for random particle distributions and provides one of the best runtime performance of all options (*automated+accel*).
- When running SCALE through the Omnibus frontend via an ORANGE geometry, the new *replica* capability allows for a simple definition of all particle coordinates (*multi-unit+accel*) and provides the best overall performance in terms of memory and runtime (Table 10).
- The most efficient modeling approach for random particle distributions that does not make use of frontend-specific implementations is to model the particles via holes and to overlay the fuel region with a mesh (*multi-unit+user-array*).
- The performance for the best random particle models is similar to the performance of models with regular particle lattice. Starting with Shift in SCALE 6.3.0, random particle realizations no longer have to cause a long runtime penalty.
- The performance of Shift is approximately identical when running Shift through SCALE's CSAS sequence or through the Omnibus frontend.
- Serpent's delta tracking is still outperforming all Shift calculations. However, Shift already shows similar or better performance than Serpent's surface tracking calculations, and further acceleration is expected.
- Despite using the same MC parameters, Shift shows larger standard deviations σ than Serpent and MCNP by a factor of 3.

3.7 FUTURE WORK

Based on the conclusions drawn from the performance assessment above, a high priority for future work is the automated generation of random TRISO particle positions for high packing fractions with Shift. The

Table 10. Overview of performance improvement with latest `Shift` development

Code	Model	SCALE-6.3.0		SCALE-7.0 beta		Relative difference	
		memory [GB]	runtime [hh:mm:ss]	memory [GB]	runtime [hh:mm:ss]	memory	runtime
HTR-10 pebble							
Omnibus/`Shift`, KENO	homogeneous	0.426	00:33:59	0.286	0:29:46	-33%	-12%
Omnibus/`Shift`, KENO	lattice	0.436	01:52:43	0.293	1:42:34	-33%	-9%
Omnibus/`Shift`, KENO	multi-unit	0.440	109:58:39	0.296	70:25:14	-33%	-36%
Omnibus/`Shift`, KENO	multi-unit+user-array	0.441	02:31:02	0.297	1:58:45	-33%	-21%
Omnibus/`Shift`, KENO	single-unit+user-array	0.490	28:52:13	0.383	24:16:12	-22%	-16%
CSAS/`Shift`, KENO	automated (accel for 7.0)	0.455	58:10:37	0.451	1:42:28		
Omnibus/`Shift`, ORANGE	multi-unit+accel	–	–	0.294	1:13:25	–	–
HTR-10 full core							
Omnibus/`Shift`, KENO	lattice	0.299	28:34:49	0.199	27:01:07	-33%	-5%
Omnibus/`Shift`, KENO	multi-unit	0.303	182:59:26	0.201	119:17:06	-34%	-35%
Omnibus/`Shift`, KENO	multi-unit+user-array	0.303	29:40:21	0.202	27:27:45	-33%	-7%
PBMR-400 full core							
Omnibus/`Shift`, KENO	inf-lattice	2.435	28:10:55	2.104	24:02:18	-14%	-15%
Omnibus/`Shift`, KENO	multi-unit+user-array	12.412	47:15:18	12.253	36:37:18	-1%	-23%
HPR unit cell							
Omnibus/`Shift`, KENO	lattice	0.849	1:46:43	0.595	1:35:38	-30%	-10%
Omnibus/`Shift`, KENO	multi-unit	3.415	131:09:35	3.419	85:14:45	0%	-35%
Omnibus/`Shift`, KENO	multi-unit+user-array	6.068	3:13:18	6.035	2:42:30	-1%	-16%
Omnibus/`Shift`, ORANGE	multi-unit+accel	–	–	1.100	3:22:40	–	–
HPR full core							
Omnibus/`Shift`, KENO	lattice	0.873	24:05:44	0.610	9:19:41	-30%	-61%
Omnibus/`Shift`, KENO	multi-unit+user-array	6.072	31:51:22	6.040	29:35:30	-1%	-7%

automatic generation of random coordinates is currently permitted for packing fractions of up to approximately 20%, which is sufficient for most HTGR designs. However, some systems such as FHRs and TRISO-fueled HPRs include particles at very high packing fractions of 40% or more, just as the TRISO-fueled HPR model studied here. The automatic TRISO particle placement should be supported for all advanced reactor designs that are currently under development.

A further study of the new *multi-unit+accel* implementation in `Shift` should be performed for additional higher packing fraction cases. This is needed to understand the runtime differences between this modeling option and the *multi-unit+user-array* modeling option, as well as to enable further optimizations to this acceleration algorithm.

Further investigations will focus on understanding the observed larger statistical standard deviations in `Shift` compared to `Serpent` and `MCNP` for identical MC parameters to identify if potential methods are needed to reduce `Shift`'s standard deviation.

Whereas the current performance assessment focused on the modeling of TRISO particles randomly distributed in the graphite matrix, the next level of modeling concerns fuel pebbles in PBRs. Instead of modeling fuel pebbles in regular lattices, performance assessments with pebbles included as holes will be performed, enabling two levels of modeling with holes. This testing will be conducted with *multi-unit* and *multi-unit+user-array* models just as for the TRISO particles, and also through the *multi-unit+accel* modeling option with ORANGE.

This performance assessment was focused on pure eigenvalue calculations; no tallies were enabled. Since the overarching goal is the generation of nodal data from these detailed models, the models will be extended to include all tallies necessary for a nodal data calculation. Another complexity will be added by performing depletion analyses which require additional tallies for cross sections and the neutron flux.

4. GENERAL SHIFT ENHANCEMENTS

This section discusses the enhancements made to Shift to support multigroup cross section generation and multiphysics calculations. Section 4.1 discusses verification of the automated cross section generation implemented in Shift using the Empire 2D assembly problem. Section 4.2 presents the new multiphysics interface to Shift, and section 4.3 presents details of a fully path-length based scattering matrix tally using a response function approach as implemented in Shift.

4.1 VERIFICATION OF AUTOMATED CROSS SECTION GENERATION

This section documents the verification of the automated cross section generation procedure implemented into Shift by comparing results to those of the Empire microreactor benchmark previously published [2].

Previously, the process to generate multigroup cross sections with Shift entailed processing the Shift raw tallies from an Hierarchical Data Format 5 (HDF5) output via a Python script to convert them to cross sections and then to the Griffin ISOXML format (see Section 2 in the associated journal paper [2] for full details). In FY22, Shift was updated to automatically calculate and output the multigroup cross sections directly to HDF5 along with the raw tallies. Along with this update, the scattering probability matrix tally was improved to now properly include the particle reaction multiplicity and to calculate and output the higher-order scattering probability moments.

An example of the new Shift input format needed to automatically generate the multigroup cross sections is given in Listing 3.

Listing 3. Example Shift tally input to generate multigroup cross sections.

```
[TALLY]
[TALLY][CELLNODAL myxs]
fine_neutron_bins    2.0e7 1.0e5 1.0e3 1.0 1.0e-5
coarse_neutron_bins  2.0e7 1.0e3 1.0e-5
cells_by_homog       "1":"2" "4":"5" "6"
cells_by_mat         "1":"2" "4":"5" "6"
```

The parameters in this simple example mean the following:

- *fine_neutron_bins*: The intermediate energy group structure to perform the Shift tallies (4 groups),
- *coarse_neutron_bins*: The coarse energy group structure to produce the multigroup cross sections (2 groups),
- *cells_by_homog*: The regions to calculate multigroup cross sections that are defined by the model cell names (3 regions),
- *cells_by_mat*: Regions to contribute to the scattering matrix cross section by material that are defined by the model cell names (3 regions). Currently, this variable should be the same as the homogenized regions. Details about using this variable is given in the last paragraph of this section.

The multigroup cross sections are located in a group called *mg_xs* in the Shift HDF5 output file. Listing 4 shows the output fields in the **mg_xs/myxs** group. Each of the groups shown in Listing 4 contains an HDF5 dataset that is an array of the cross section values.

Listing 4. Example Shift cross section output in HDF5 file.

```
coarse_flux                Group
fission                    Group
fission_spectrum           Group
kappa_fission              Group
nu_fission                 Group
reduced_absorption         Group
removal                    Group
scattering_matrix          Group
total                      Group
transport                  Group
```

Converting the cross section values in this HDF5 output to an ISOXML file is a simple procedure using the `Shift` Python post-processing tools.

To verify this new automated implementation, multigroup cross sections were generated with `Shift` for the 2D Empire assembly benchmark and compared to the previously generated cross sections from FY21 and published in Pandya [1, 2]. The same `Shift` model and input from FY21 were used with modification needed only to the tally input block. The `Shift` calculations were performed on an ORNL institutional computing cluster. The `Griffin` calculations were performed on the Idaho National Laboratory (INL) high-performance computing cluster Sawtooth [22].

Figures 8–16 show comparisons of the cross sections generated automatically with `Shift` and the previously manually generated cross section set using the same `Shift` tallies. The differences seen in these cross section comparisons can be mostly attributed to the statistical variation of the MC tallies. The relative error of the calculated coarse-group scalar flux in these homogenized regions between these two simulations varied within 9×10^{-3}, which is the main contributor to the differences in cross sections. The difference in the removal cross section is also due to the addition of multiplicity in the scattering probability matrix tally which is used to calculate the within-group scattering cross section.

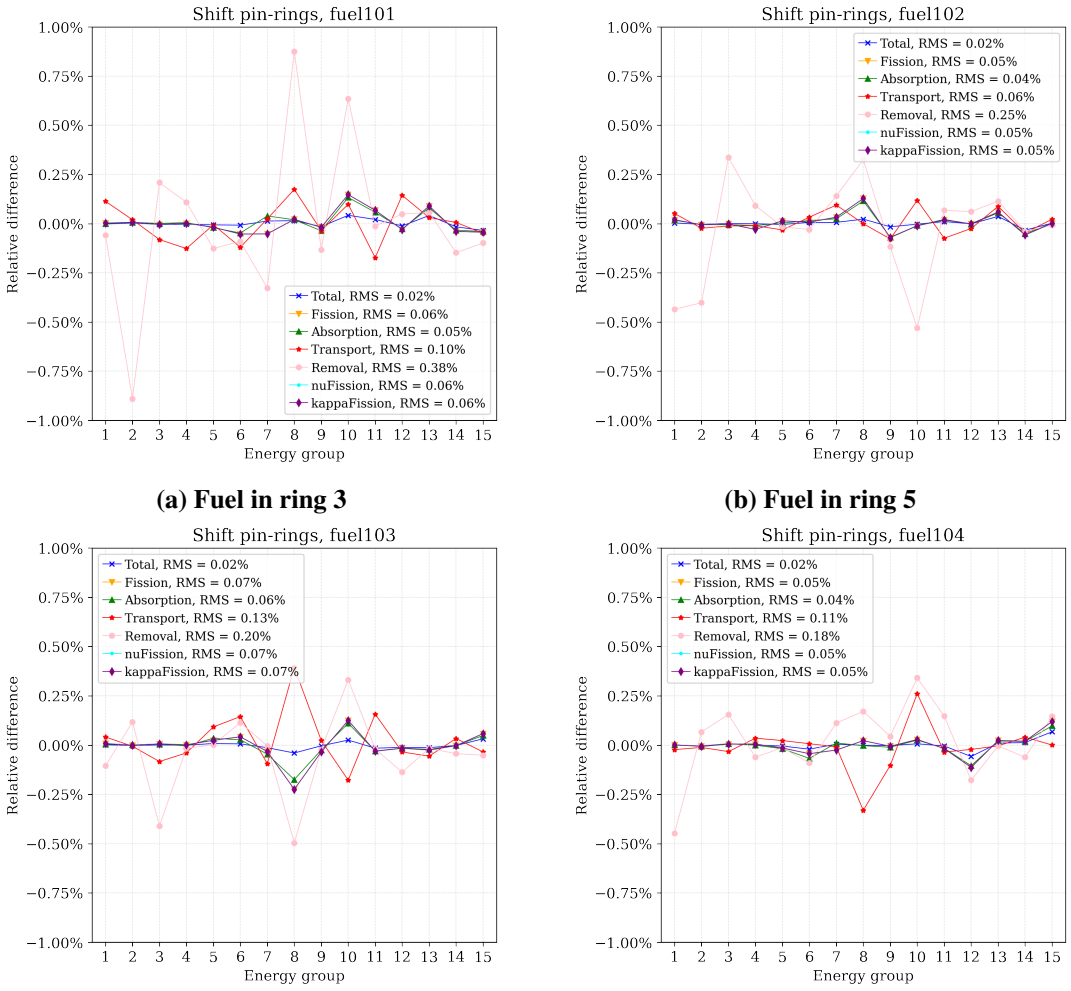
This new cross section set was also run through `Griffin` using the same inputs as those used for the results published in Pandya [2]. Table 11 shows the eigenvalue and pin power comparisons including the previously published results. Table 12 shows the minimum and maximum SPH factors used by the `Griffin` diffusion solver compared against the previously published results. These results provide verification of the automated cross section processing implemented in `Shift`.

Table 11. Empire 2D assembly eigenvalue and pin power (PP) comparison

Code	k_{eff}	Δk [pcm]	Abs. PP RMS
`Shift` FY21 Reference[a]	$1.21343 \pm 1.0 \times 10^{-7}$	–	–
`Shift/Griffin` FY21[a]	1.21326	-17	0.09%
`Shift` FY22 Reference	$1.21337 \pm 1.0 \times 10^{-7}$	–	–
`Shift/Griffin`	1.21304	-33	0.08%

[a] From [2]

The authors are currently investigating a potentially more accurate calculation of the scattering matrix cross section. This analysis will be included in a future publication.



(a) Fuel in ring 3

(b) Fuel in ring 5

(c) Fuel in ring 7

(d) Fuel in ring 9

Figure 8. Empire 2D assembly fuel cross section comparisons.

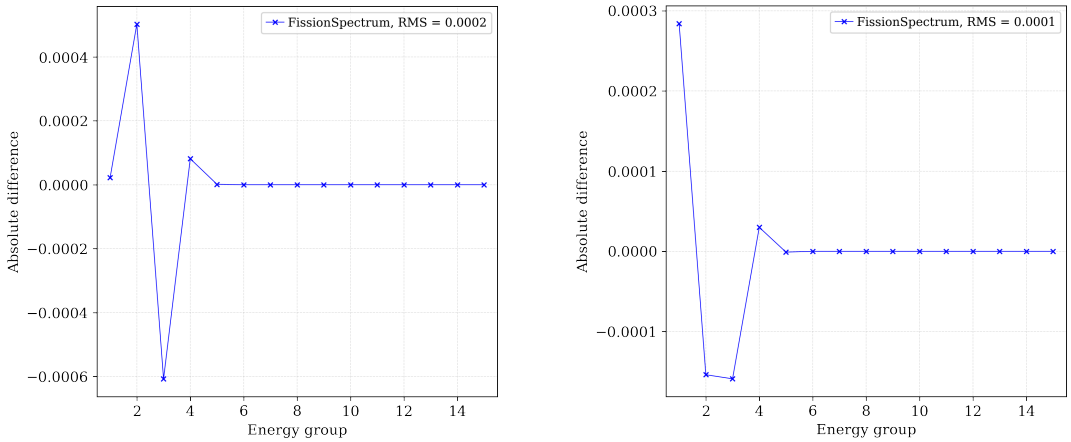
4.2 TITAN: AN ADVANCED REACTOR MULTIPHYSICS INTERFACE

`Titan` will be introduced in `SCALE` 7.0.0 with a beta release later in calendar year 2022 and provides an easy-to-use interface for `Shift`, as well as an application programming interface (API) to build ORANGE geometries for uses such as ray tracing. It can run `Shift` by reading in an input file or by populating a model via API. `Titan` uses a JSON-formatted input split into three main sections: geometry, materials, and `Shift` options. The input is validated prior to running `Shift`.

4.2.1 GEOMETRY INPUT SECTION

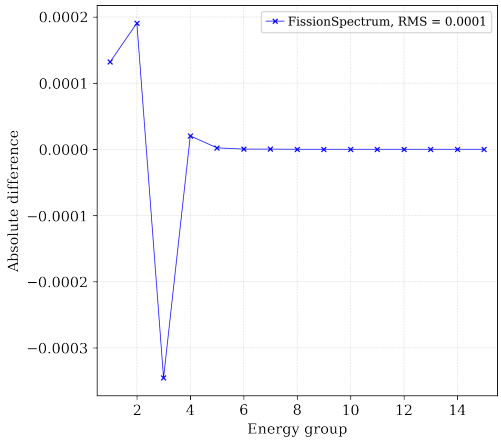
`Titan` does not define geometry with surfaces and cells but rather with concrete engineering–style unit types. In the example input shown in Listing 5, there are two units defined: "F" and "W". "F" is a unit of type "PIN", which is defined as a vector of cylindrical zones. Each cylindrical zone is defined by a material ("FUEL" below) and the outer radius of the zone (0.4096 below).

"W" is a unit of type "POLYGON_DOMAIN", which is defined by a material ("WATER"), a `Titan` Polygon, and an optional vector of inserted units ("inserts"). A `Titan` Polygon is defined by the number of sides (4) and the apothem of the polygon (0.63). There are optional parameters in the polygon for corner

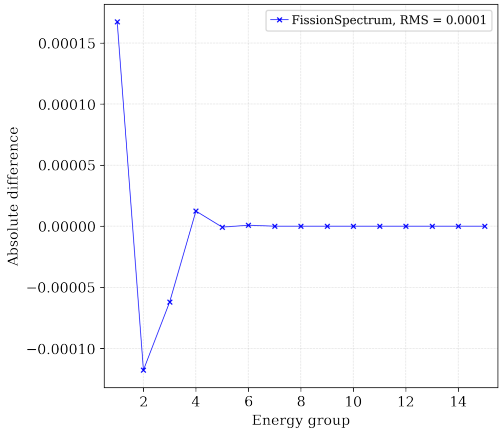


(a) Fuel in ring 3

(b) Fuel in ring 5



(c) Fuel in ring 7



(d) Fuel in ring 9

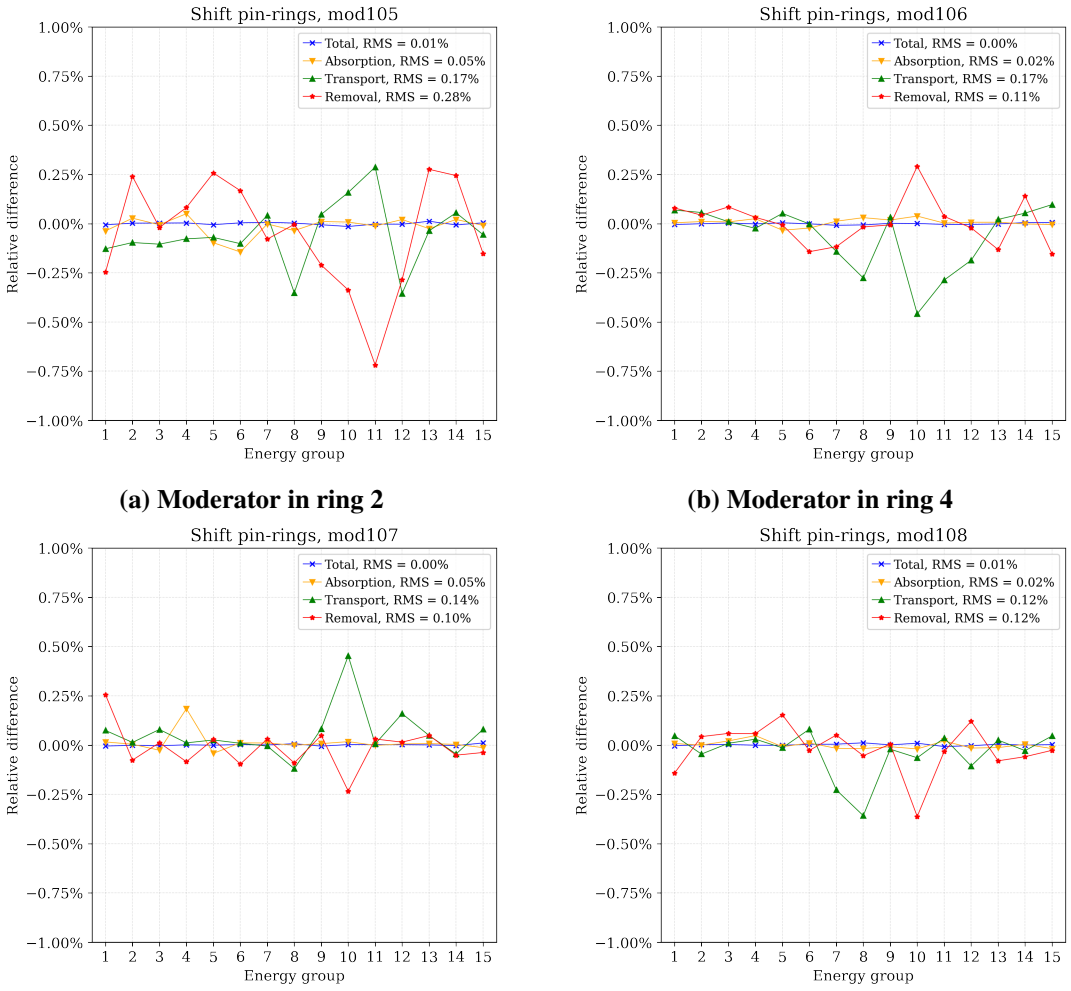
Figure 9. Empire 2D assembly fuel fission spectrum comparisons.

radius and side indentation. The inserts are a list of other units in the unit map that are in "W". This inserts "F" into "W" at the local center of "W". Inserts have an optional member that can transform the inserted unit within the parent unit.

Once units are defined, the geometry definition is completed with the geometry dimension, height, global unit, and boundary conditions. Dimension is either "2D" or "3D", where "2D" is a 2D extruded geometry. Height is used only in the case of a "2D" geometry and is the extruded height of the geometry. Global is the global unit ("W"), and boundary conditions are defined based on the shape of the global unit. In the example above, because "W" is an extruded square, the minimum and maximum z-planes are set to reflecting boundaries, as are the four planes defining the square. The default boundary condition is a vacuum boundary condition, and as of now, only reflecting boundary conditions are supported.

4.2.2 MATERIALS INPUT SECTION

Listing 6 shows the materials section of the `Titan` input, which is a map of each material to a `Titan` material definition. Each `Titan` material is defined by a temperature (in Kelvin) and a material concentration. A concentration is a vector of ZAIDs and corresponding number densities in atoms/b-cm.



(a) Moderator in ring 2

(b) Moderator in ring 4

(c) Moderator in ring 6

(d) Moderator in ring 8

Figure 10. Empire 2D assembly moderator cross section comparisons.

4.2.3 SHIFT INPUT SECTION

Listing 7 shows an example of the `Shift` section of the `Titan` input, which offers options for the number of particles per cycle, the total number of cycles, the number of inactive cycles, whether Doppler broadening is used, the random number seed, and the initial k_{eff} guess.

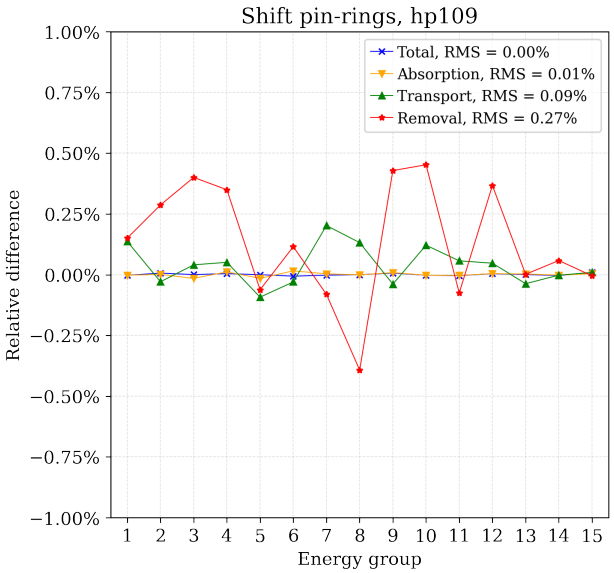
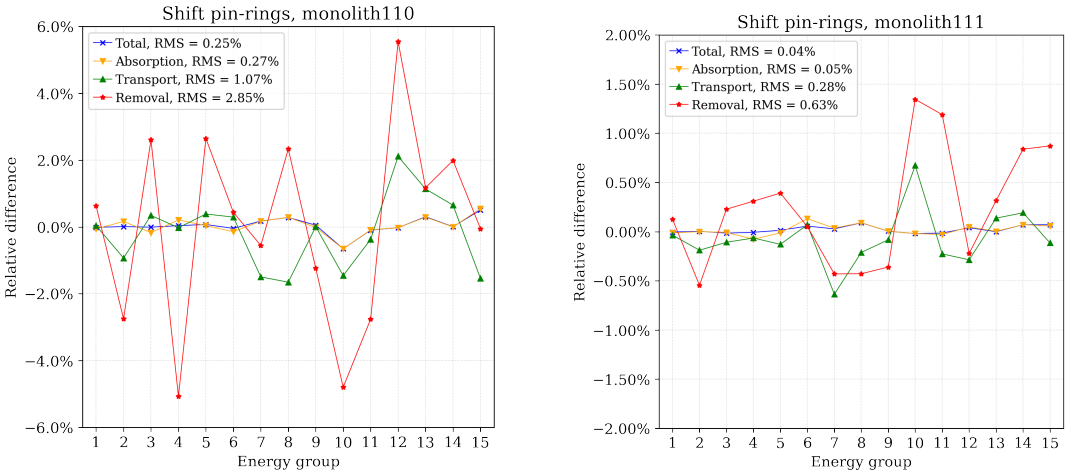


Figure 11. Empire 2D assembly heat pipe cross section comparison.

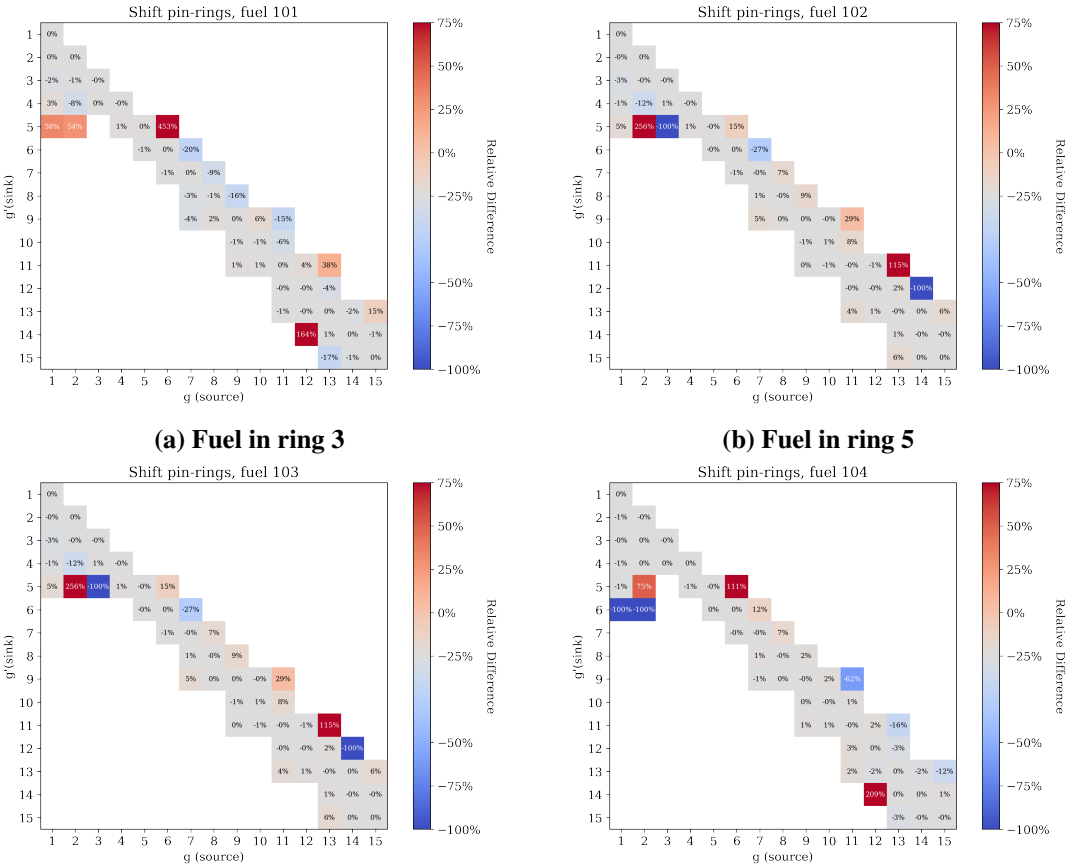
Table 12. Empire 2D assembly SPH correction factor comparison

Code	Nonlinear Iterations	Rel. Tol.	Min	Max
`Shift/Griffin` FY21	4	1.0×10^{-8}	0.47	1.31
`Shift/Griffin` FY22	4	1.0×10^{-8}	0.43	1.52



(a) Hex corner **(b) Hex side**

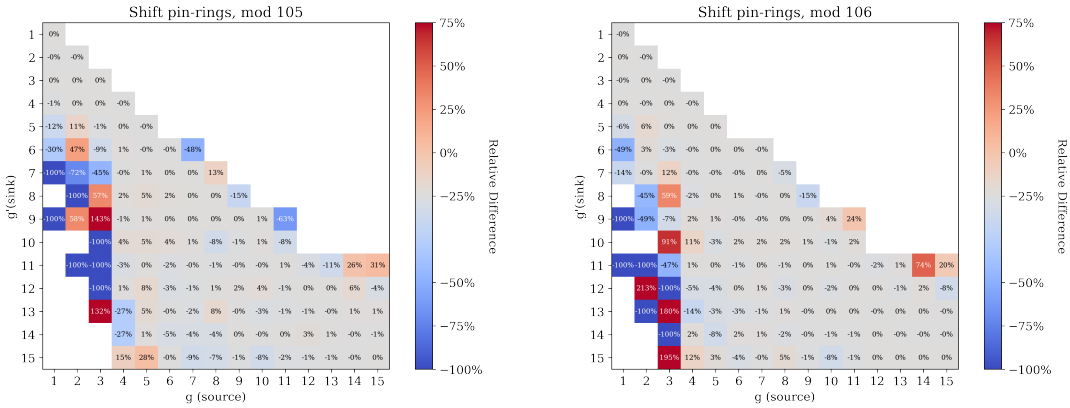
Figure 12. Empire 2D assembly monolith cross section comparisons.



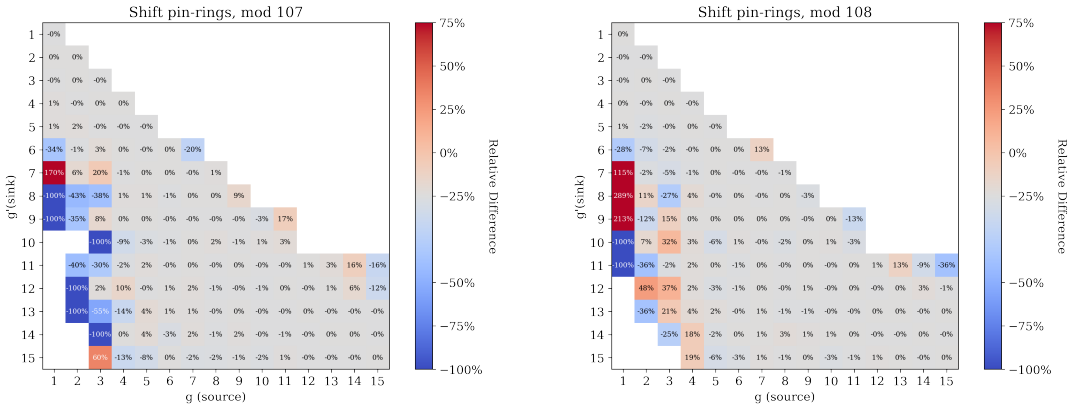
(a) Fuel in ring 3 **(b) Fuel in ring 5**

(c) Fuel in ring 7 **(d) Fuel in ring 9**

Figure 13. Empire 2D assembly fuel P_0 scattering cross section comparisons.



(a) Moderator in ring 2 **(b) Moderator in ring 4**



(c) Moderator in ring 6 **(d) Moderator in ring 8**

Figure 14. Empire 2D assembly moderator P_0 scattering cross section comparisons.

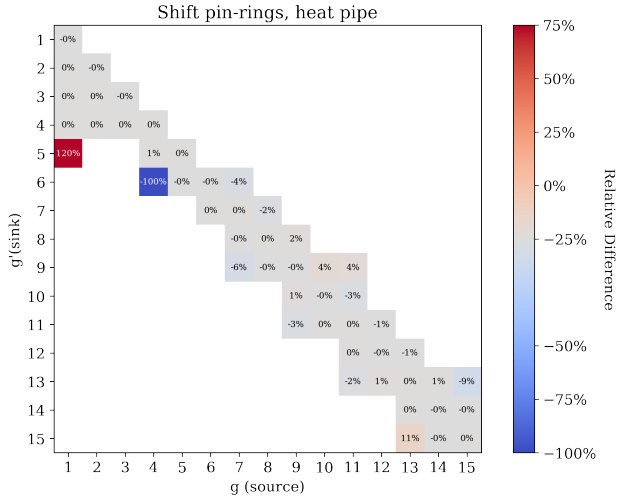
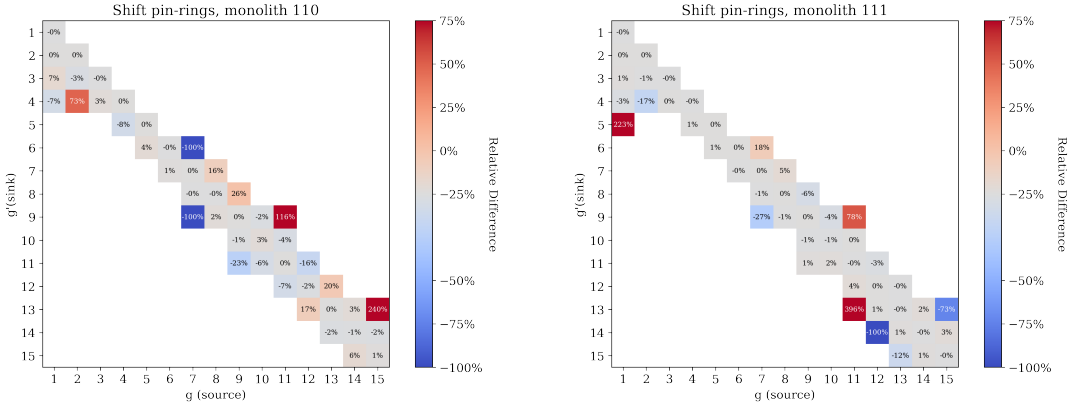


Figure 15. Empire 2D assembly heat pipe P_0 scattering cross section comparison.



(a) Hex corner **(b) Hex side**

Figure 16. Empire 2D assembly monolith P_0 scattering cross section comparisons.

Listing 5. Example geometry section in the Titan input file.

```
"geometry" : {
  "units" : {
      "F" : [ "PIN",
              [ ["FUEL", 0.4096],
                ["GAP", 0.418],
                ["CLAD",0.475] ]
      ],
      "W" : [ "POLYGON_DOMAIN",
         ["WATER", [4, 0.63],
         {
           "inserts" : [ ["F"] ]
         }]
      ]
  },
  "dimension" : "2D",
  "height" : 1.0,
  "global" : "W",
  "bcs" : {
    "mz" : "REFLECT",
    "pz" : "REFLECT",
    "p0" : "REFLECT",
    "p1" : "REFLECT",
    "p2" : "REFLECT",
    "p3" : "REFLECT"
  }
}
```

Listing 6. Example materials section in the Titan input file.

```
"materials" : {
  "FUEL" : [ 300.0,
      [ [922350, 6.76350E-03],
        [922380, 2.74821E-02] ]
  ],
  "GAP" : [ 300.0,
      [ [70140, 1.0E-10] ]
  ],
  "CLAD" : [ 300.0,
      [ [40090, 2.217855E-02] ]
  ],
  "WATER" : [400.0,
       [ [ 80160, 2.202036E-02 ],
         [ 10010, 4.414292E-02 ] ]
  ]
}
```

Listing 7. Example Shift options in the Titan input file.

```
"shift" : {
   "num_particles" : 100,
   "num_cycles" : 15,
   "num_inactive_cycles" : 5,
   "dbrc" : false,
   "seed" : 1234,
   "keff_init" : 1.3
}
```

4.2.4 TITAN OUTPUT

Titan outputs a geometry description, materials description, and a summary of Shift results. An example of a full output is shown in Listing A.1. The geometry output for the unit "W" in Section 4.2.1 shows material, surface, and cell information as given in Listing 8. The material output shows the temperature and concentration information for each material defined in the geometry as shown in Listing 9.

The Shift output from Titan shows the solution and timing info for each cycle as given in Listing 10. Titan also outputs an image file of the geometry, shown in Fig. 17.

4.2.5 ONGOING AND FUTURE TITAN DEVELOPMENT

Titan development is currently focused on increasing the number of concrete unit types supported, such as cartesian and hexagonal assemblies, pebbles with TRISO particles, and control structures. In the near future, development will focus on time-dependent and depletion calculations, as Titan already supports automatic deduplication of materials to support higher fidelity materials analysis.

Listing 8. Unit "W" geometry Titan output

```
*********************************************
"unit[1](POLYGON_DOMAIN)" (Titan/csg/ORANGE_CSG.hh:66)
*********************************************
:Type:         masked unit
:# Cells:      3 (offset = 0)
:# Surfaces:   7 (offset = 0)
:Bounding box: ``{-0.63 -0.63 0 to 0.63 0.63 1}``
======= = ============================== ============================================================
Cell    Z Name                           Surface logic
======= = ============================== ============================================================
0       X unit[1](POLYGON_DOMAIN)_boundary 0 ~ 1 & 2 & 3 ~ & 4 & 5 ~ & ~
                                         (from ``Titan/csg/ORANGE_CSG.cc:89``)
1       H unit[1](PIN)                   4 5 ~ & 6 ~ &
                                         (from ``Titan/csg/ORANGE_CSG.cc:112``)
2       M [WATER]cell[4]                 0 ~ 1 & 2 & 3 ~ & 4 & 5 ~ &
                                         (from ``Titan/csg/ORANGE_CSG.cc:58``)
======= = ============================== ============================================================
Cells with reentrant surface tracking: "unit[1](POLYGON_DOMAIN)_boundary"
======= ============================ ================================================================
Surface Name                         Description
======= ============================ ================================================================
0                                    Plane: y=0.63
        [WATER]cell[4]shape[poly].p0 (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].p0 (from ``Titan/shapes/Shape.cc:59``)
1                                    Plane: x=-0.63
        [WATER]cell[4]shape[poly].p1 (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].p1 (from ``Titan/shapes/Shape.cc:59``)
2                                    Plane: y=-0.63
        [WATER]cell[4]shape[poly].p2 (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].p2 (from ``Titan/shapes/Shape.cc:59``)
3                                    Plane: x=0.63
        [WATER]cell[4]shape[poly].p3 (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].p3 (from ``Titan/shapes/Shape.cc:59``)
4                                    Plane: z=0
        [WATER]cell[4]shape[poly].mz (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].mz (from ``Titan/shapes/Shape.cc:59``)
        unit[1](PIN).mz              (from ``Titan/csg/ORANGE_CSG.cc:112``)
5                                    Plane: z=1
        [WATER]cell[4]shape[poly].pz (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].pz (from ``Titan/shapes/Shape.cc:59``)
        unit[1](PIN).pz              (from ``Titan/csg/ORANGE_CSG.cc:112``)
6                                    Cyl z: r=0.475
        unit[1](PIN).coz             (from ``Titan/csg/ORANGE_CSG.cc:112``)
======= ============================ ================================================================
```

Listing 9. Material Titan output

```
============================================================
:# Composition  ID= 1  (matid 1)  name= "FUEL"
    density=  13.503262 (g/cc)   temperature=   300.00 (K)
    fissionable= true   depletable= true
    ZAID     Number Density
 ----------  ---------------
     92235    6.763500e-03
     92238    2.748210e-02
============================================================
```

Listing 10. Shift Titan output

```
::: Beginning inactive cycles
 Cycle   k_cycle    Time (s)          Np
   0    1.658389   6.639334e-01        100
   1    1.730859   8.304776e-01        119
   2    1.475721   7.207497e-01        104
   3    1.666580   6.368681e-01         95
   4    1.636224   7.555143e-01        111
>>> Completed 5 inactive cycles
::: Beginning active cycles
 Cycle   k_cycle     k_avg     k_stdev        Time (s)       Np
   0    1.808716   1.808716   1.808716e+00   6.322631e-01         87
   1    1.570670   1.689693   1.190230e-01   6.775513e-01        101
   2    1.431242   1.603542   1.102001e-01   6.812073e-01        100
   3    1.345641   1.539067   1.011389e-01   6.022441e-01         92
   4    1.547398   1.540733   7.835960e-02   6.437300e-01         99
   5    1.587574   1.548540   6.445488e-02   7.588835e-01        115
   6    1.577632   1.552696   5.463262e-02   5.492430e-01         82
   7    1.494820   1.545462   4.786314e-02   6.323530e-01        100
   8    1.418737   1.531381   4.449782e-02   5.786129e-01         85
   9    1.561618   1.534405   3.991476e-02   7.594728e-01        111
>>> Completed 10 active cycles
```

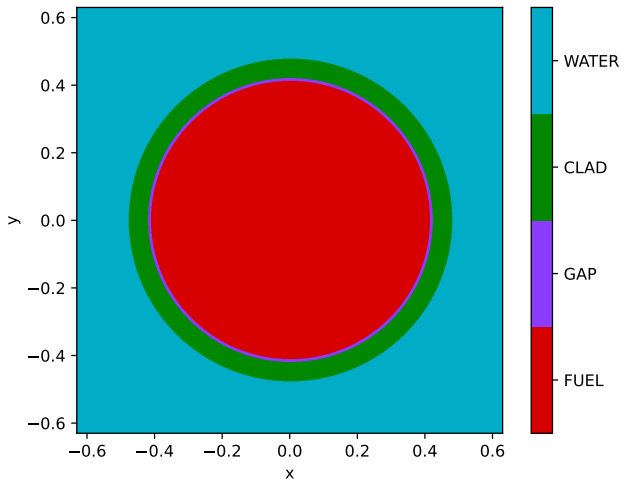


Figure 17. The Titan pin cell model geometry.

Listing 11. HTR-10 Titan input file.

```
{
  "title" : "HTR PEBBLE",
  "materials" : {
    "FUEL" : [ 300.0,
        [ [ 922350, 3.99198E-03],
          [ 922380, 1.92441E-02],
          [ 80160,  4.64720E-02],
          [ 50100,  4.06384e-07],
          [ 50110,  1.63575e-06]]
    ],
    "BUFFER" : [ 300.0,
        [ [ 30060000, 5.51511E-02],
          [ 50100,  1.58513e-08],
          [ 50110,  6.38035e-08] ]
    ],
    "IPC"  :[ 300.0,
        [ [ 30060000, 9.52610e-02],
          [ 50100, 2.73795e-08],
          [ 50110, 1.10206e-07] ]
    ],
    "SiC" : [ 300.0,
        [ [60000, 4.77597e-02],
          [140280,4.77597e-02]]
    ],
    "OPC" : [ 300.0,
        [ [ 30060000, 8.67377e-02],
          [ 50100, 2.49298e-08],
          [ 50110, 1.00345e-07] ]
    ],
    "MATRIX" : [ 300.0,
        [ [ 30060000, 8.67377e-02],
          [ 50100, 2.49298e-08],
          [ 50110, 1.00345e-07] ]
    ],
    "SHELL" : [300.0,
        [ [ 30060000, 8.67377e-02],
          [ 50100, 2.49298e-08],
          [ 50110, 1.00345e-07] ]
    ],
    "COOLANT" : [300.0,
        [ [70140, 3.95901e-05],
          [80160, 1.10499e-05],
          [180400, 2.36778e-07],
          [10010, 8.58028e-07] ]
    ]
  },
  "geometry" : {
    "units" : {
        "F" : [ "PEBBLE",
          {
            "pebble" : [
                ["MATRIX", 2.5],
                ["SHELL",3.0]
             ],
             "particle" : [
                 ["FUEL", 0.0250],
                 ["BUFFER", 0.0340],
                 ["IPC", 0.0380],
                 ["SiC", 0.0415],
                 ["OPC", 0.0455]
             ],
             "num_particles" : 8385
          }
        ],
        "C" : [ "POLYGON_DOMAIN", [
            "COOLANT",
            [ 4, 3.0, {"height" : 6.0
                 } ],
            { "inserts": [ ["F"] ] }
          ]
        ]
    },
    "dimension" : "3D",
    "global" : "C",
    "bcs" : {* : "REFLECT" }
  },
  "shift" : {
     "num_particles" : 10000,
     "num_cycles" : 250,
     "num_inactive_cycles" : 50,
     "dbrc" : false
  }
}
```

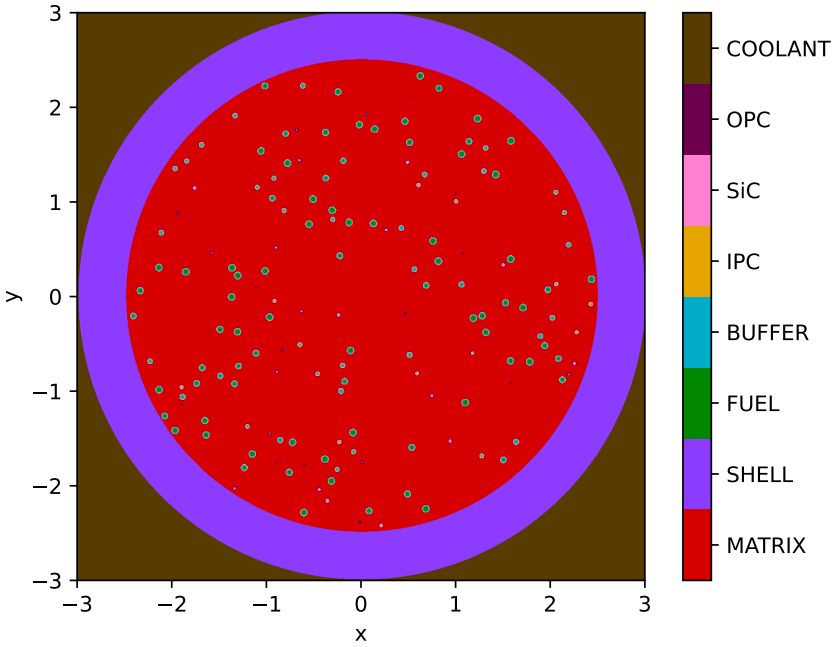


Figure 18. The Titan HTR-10 pebble model geometry.

4.3 PATH-LENGTH SCATTERING MATRIX CALCULATION

This section presents the initial investigation and implementation of a path-length–based scattering matrix tally into `Shift` to produce more accurate scattering matrix cross sections.

Part of the nodal tally capabilities of `Shift` includes the ability to calculate a scattering matrix—a matrix that contains the flux-averaged scattering cross sections from one user-defined energy group g to another for a given Legendre polynomial order l. This scattering matrix is a necessary component of the two-step neutronics method where cross section information is calculated for use in a nodal/deterministic method (such as diffusion). Mathematically, a given column of the scattering matrix can be expressed as follows:

$$\vec{\Sigma}_s^{l,g} = \vec{s}^{\,l,g} \oslash \vec{p}, \tag{1}$$

where $\oslash$ denotes Hadamard division and $\vec{s}^{\,l,g'}$ is as follows:

$$\vec{s}^{\,l,g} = \begin{Bmatrix} \int_{\Delta E_g} \int_{\Delta E_0} \Sigma_s^l(E' \rightarrow E)\phi(E')dE'dE \\ \vdots \\ \int_{\Delta E_g} \int_{\Delta E_{g'}} \Sigma_s^l(E' \rightarrow E)\phi(E')dE'dE \\ \vdots \\ \int_{\Delta E_g} \int_{\Delta E_{G'-1}} \Sigma_s^l(E' \rightarrow E)\phi(E')dE'dE \end{Bmatrix}, \tag{2}$$

noting that Σ_s^l is defined as follows:

$$\Sigma_s^l(E' \rightarrow E) = \int_{-1}^{1} \Sigma_s(E' \rightarrow E, \mu) P^{\,l}(\mu) d\mu, \tag{3}$$

where $P^{\,l}$ is simply the lth-order Legendre polynomial, and $\vec{p}$ is simply group-integrated flux:

$$\vec{p} = \begin{Bmatrix} \int_{\Delta E_0} \phi(E')dE' \\ \vdots \\ \int_{\Delta E_{g'}} \phi(E')dE' \\ \vdots \\ \int_{\Delta E_{G'-1}} \phi(E')dE' \end{Bmatrix}, \tag{4}$$

and so the matrix $\mathbf{\Sigma}_s^l$ is expressed as follows:

$$\mathbf{\Sigma}_s^l = \left\{ \vec{\Sigma}_s^{l,0} \cdots \vec{\Sigma}_s^{l,g} \cdots \vec{\Sigma}_s^{l,G-1} \right\}. \tag{5}$$

This scattering matrix includes non-fission particle production reactions; therefore it can be assumed that $\Sigma_s^l(E' \rightarrow E)$ includes the multiplicity of those reactions implicitly since the scattering cross section can be decomposed into the summation of all possible scattering reactions as follows:

$$\Sigma_s^l(E' \rightarrow E) = \sum_{r=0}^{R} \gamma^r \Sigma_s^{l,r}(E' \rightarrow E), \tag{6}$$

where γ^r would be the multiplicity of reaction r.

The "typical" ways to calculate this scattering matrix with the MC method include using a collision tally that generates a score when a particle scatters from one group to another. If there are G user-defined energy

groups for a given problem, then this means that there are roughly $\sim \frac{G(G+1)}{2}$ matrix elements to populate (allowing some flexibility for upscattering at lower energies for neutrons), each with varying probabilities of occurring in a given material. Some of these probabilities may be very low, and thus accurate statistics might be difficult to ascertain for the entirety of the scattering regime of a problem. This work sought to ameliorate this issue by introducing a vector response function for path length tallies. This response function is defined as follows:

$$\vec{r}^{\,l}(E') = \begin{Bmatrix} \int_{\Delta E_0} \Sigma_s^l(E' \rightarrow E) dE \\ \vdots \\ \int_{\Delta E_g} \Sigma_s^l(E' \rightarrow E) dE \\ \vdots \\ \int_{\Delta E_{G-1}} \Sigma_s^l(E' \rightarrow E) dE \end{Bmatrix}, \tag{7}$$

This response function allows for generating scores for an entire column of the scattering matrix $\mathbf{\Sigma}_s^l$ from a single path-length event instead of individual matrix elements from a single collision event, which will effectively reduce the dimensionality that the tally is directly responsible for scoring from $\sim \frac{G(G+1)}{2}$ to simply G. It is well known that the finer the tally energy bin structure, the slower a tally will converge—simply because each bin can generate a greater amount of scores if it has a wider acceptance criteria for particles to generate scores. The details of calculating this scattering matrix are presented in the next section.

4.3.1 RESPONSE FUNCTION CONSTRUCTION AND APPLICATION

4.3.1.1 Tabular Data

To calculate $\mathbf{\Sigma}_s^l$, $\vec{p}$ and $\vec{s}^{\,l,g}$ must be determined. The vector $\vec{p}$ is trivial to find using MC because it is the result of a path-length tally discretized over energy groups $[\Delta E_0, \ldots, \Delta E_g, \ldots \Delta E_{G-1}]$. The vector $\vec{s}^{\,l,g}$ poses some additional difficulty. In the abstract, it presents a simple-looking response function ($\vec{r}^{\,l}(E')$ in Eq. 7) for the same path-length tally mentioned earlier. However, there are some complications in calculating this response function.

First, it is important to decompose the scattering cross section into its respective parts. Σ_s as follows will now be denoted as a macroscopic scattering cross section for a given material (which can contain multiple nuclides). This decomposition can be done as follows:

$$\Sigma_s(E' \rightarrow E, \mu) = \sum_{i=0}^{I} n_i \sum_{r=0}^{R} \gamma^r \sigma_s^{i,r}(E') p^{i,r}(E' \rightarrow E, \mu), \tag{8}$$

where n is the number density for nuclide i in a given material, r is the reaction type, σ_s is the microscopic cross section, and p is the probability density function describing the kinematics of a reaction. The reason this decomposition is relevant is because these separate variables mirror the separate pieces of data that are stored in memory during a `Shift` simulation. The number density is typically provided by the user, but σ_s and p are stored separately as 1D cross section data (for energy E') and kinematics data.

The tabular microscopic cross section σ_s is stored point-wise on an energy grid. However, the cross section data for each individual reaction is stored on separate energy grids. To avoid an unnecessary memory burden, it is prudent to have a single response function describing the entirety of scattering mechanics for a given problem. To do this, the energy grids for the tabular data must be consolidated into a single energy grid such that the response function can be interpolated and evaluated on that single energy grid. Thus, the

method by Leppännen implemented in the *Union_Grid_Builder* class in SCALE is used to construct the energy grid and the cross section values for each reaction of each nuclide in a material, which are then interpolated onto that grid [20]. This method preserves local cross section minima and maxima, threshold energies, and $S(\alpha,\beta)$ energy points for each reaction and then consolidates the rest based on user input.

After forming this unionized energy grid, another complication arises. The kinematics probability density function p is still on its own incoming energy grid (separate from its respective original 1D cross section energy grid). The tabular kinematics data, after being processed from a cross section library, is stored on a multidimensional grid with dimension order $[E'][\mu][E]$. Due to this, for the tabular data, p is first integrated over the outgoing energy, integrated against the Legendre polynomial, and then interpolated onto the unionized energy grid. This integration is expressed as follows:

$$\int_{\Delta E_g} \Sigma_s^l(E' \rightarrow E)dE = \int_{-1}^{1}\int_{\Delta E_g} \Sigma_s(E' \rightarrow E,\mu)P^{\,l}(\mu)dEd\mu = \sum_{i=0}^{I} n_i \sum_{r=0}^{R} \gamma^r \sigma_s^{i,r}(E') \int_{-1}^{1}\int_{\Delta E_g} p^{i,r}(E' \rightarrow E,\mu)P^{\,l}(\mu)dEd\mu. \quad (9)$$

Currently, only P_0 scattering is supported for this implementation. Support for higher order Legendre polynomials will be added during future work. The above integration is fairly straightforward because the tabular kinematics data are treated entirely as a piece-wise linear function for the probability density functions in μ and E.

It is of note that for neutrons certain isotopes contain "double-sided" data where the point-wise grid for μ is not in strict ascending order, but instead has a form of $[1,\ldots,\mu_i,\ldots,1]$. This is a workaround for certain kinematic situations in which a neutron has multiple possible exit energies for a single exit angle in elastic scattering. These and any other data that do not have a strict ascending order abscissa point-wise grid for scattering angle μ or exit energy E were excluded from the scattering response function. The ZAIDs that have this issue are as follows:

- 1001
- 12024
- 12025
- 12026
- 14030
- 16032
- 16033
- 16034
- 16036
- 19039
- 19041
- 69168
- 1001001
- 1014030
- 2001001
- 4001001
- 5001001
- 6001001
- 7001001
- 8001001
- 9001001

4.3.1.2 Elastic Scattering

The section above covers most reaction types, including how to handle tabular data for elastic scattering. However, elastic scattering can also be modeled using continuous functions that are directly sampled from including Target-at-Rest and Free-Gas. In addition to this, the tabular data are not a single set for the entire energy distribution: depending on the energy of an incoming neutron, it might be scattered using $S(\alpha,\beta)$ tables with or without Bragg Edges. All of these models represent the same scattering reaction type and

must thus be pieced together for the different energy ranges, user inputs, and data availability. Due to the complicated nature of handling this, currently only $MT = 2$ elastic scattering tabular data are supported in this response function generator. Eventually, according to the work by Adam Nelson, the entirety of elastic scattering will be supported in `Shift` [23].

4.3.1.3 Application

After the response function is formed, the response vectors are interpolated for each element onto the unionized grid. Once on the unionized grid, the response function becomes available in the same way as a 1D cross section—simply interpolate it between incoming energy points and multiply the scores by the response function for a path length tally:

$$\hat{\vec{s}}^{l,g} = \frac{1}{N} \sum_{k=0}^{K} \vec{r}^{\,l}(E'_k) w_k T_k, \tag{10}$$

where T and w are the path length and weight for transport event k and $\hat{\vec{s}}^{l,g}$ is a statistical estimation of $\vec{s}^{l,g}$. Note that the MC simulation itself takes care of the outer integral over the incoming energy bins. Finally, the scattering matrix can be evaluated simply as follows:

$$\hat{\vec{\Sigma}}_s^{l,g} = \hat{\vec{s}}^{l,g} \oslash \hat{\vec{p}}, \tag{11}$$

$$\hat{\mathbf{\Sigma}}_s^{l} = \left\{ \hat{\vec{\Sigma}}_s^{l,0} \cdots \hat{\vec{\Sigma}}_s^{l,g} \cdots \hat{\vec{\Sigma}}_s^{l,G} \right\}. \tag{12}$$

4.3.2 FUTURE WORK

There are two major pieces of future work that must be completed to fully implement this path-length–based scattering matrix tally. The first is to include higher-order Legendre polynomials in the response function—as of now, only $l = 0$ is supported. The $l = 0$ response function involves only integration over a piece-wise linear function since $P_0(\mu) = 1$ and the kinematics data are stored as a piece-wise linear function. Higher-order Legendre polynomials will require a slightly more clever integration routine.

The second important piece of future work that must be completed is the full inclusion of elastic scattering kinematics in the response function. As mentioned earlier, only $MT = 2$ data are used for the elastic scattering kinematics in the current implementation of the response function. To include the full elastic scattering physics in this response function, an algorithm that carefully accounts for the data that are available and the energy range of group g is required to effectively piece together thermal and fast neutron elastic data, including continuous function integration where applicable. There are also physics, such as Doppler broadening, which do not present a simple way to include in this response function and will require further investigation.

Other possible areas of future work would be the reduction of dimensionality of the response function due to its potential large memory footprint. This includes accounting for cross section data that do not have strong dependence on temperature when there is a material at multiple temperatures in a problem. A more clever storage scheme could be explored such that response function elements with a value of 0 may not be stored. Overall, there is quite a bit of future work that should be explored to make this feature more viable.

5. TRISO HPR ANALYSIS

This section presents the analysis of two TRISO HPR problems using Shift-generated cross sections with Griffin.

5.1 TRISO HPR UNIT CELL TWO-STEP RESULTS

This section presents the Shift-generated cross sections and the Griffin results from two-step neutronics calculations of the TRISO HPR unit cell model. For this two-step neutronics analysis, the infinite stacked lattice of the TRISO unit cell model described in Section 2.7 was used for the Shift calculations.

The same 252 intermediate energy group structure was used for the Shift reaction rate tallies as was used for the Empire assembly analysis and was previously published [1, 2]. This intermediate-group structure is given for ease of reference in Tables 13 and 14. An 11 coarse-energy group structure was used for the cross sections and for the Griffin calculations to coincide with the previous analyses by Stauff et al. [3, 4] and is shown in Table 15 for ease of reference.

Two different sets of homogenized cross sections were generated with Shift at a temperature of 800 K. Each set of cross sections includes the following homogenized regions:

- One region including the TRISO particles in the fuel pin and surrounding monolith hex,
- One region including the heat pipe wick, the surrounding air, and the surrounding monolith hex,
- One region include the moderator, the surrounding air, and the surrounding monolith hex.

The difference between the set of four cross sections and the set of five cross sections is that the former combines the axial reflector and heat pipe extension into the reflector region into one homogenized region, and the latter splits the reflector into a lower homogenized region and an upper homogenized region which includes the heat pipes above 180 cm.

The Shift calculations were performed on an ORNL institutional computing cluster. The Griffin calculations were performed on the INL high-performance computing cluster Sawtooth [22]. Tables 16 and 17 show some of the relevant Shift eigenvalue and Griffin diffusion solver parameters used in these calculations.

Table 18 shows the eigenvalue and pin power comparisons between the reference Shift calculation and various Griffin calculations using the Shift-generated cross sections. All of the Griffin results shown in this table used the CFEM-diffusion solver, with and without SPH correction. All of the SPH corrected results applied correction only to the fuel regions. Finally, Griffin was run with a uniform mesh refinement of zero and one. Overall conclusions are:

- Shift-generated cross section sets are consistent, and using SPH correction allows for using only four homogenized regions,
- Griffin CFEM-diffusion with SPH correction using Shift-generated cross sections produces consistent eigenvalue and axial pin powers to Shift, and
- No uniform mesh refinement is needed to achieve comparable results to the MC reference.

Table 19 shows the flux correction needed in the hex fuel pin axial zones for both sets of cross sections and both mesh refinements settings. Overall, all require the same amount of correction to the flux in the fuel pin.

Table 13. 252 intermediate-group energy bounds for reaction rate tallies with `Shift`

Grp	Upper Energy (eV)	Grp	Upper Energy (eV)	Grp	Upper Energy (eV)	Grp	Upper Energy (eV)
1	2.000×10^{7}	41	2.700×10^{5}	81	2.020×10^{2}	121	3.700×10^{1}
2	1.733×10^{7}	42	1.830×10^{5}	82	1.930×10^{2}	122	3.600×10^{1}
3	1.568×10^{7}	43	1.490×10^{5}	83	1.915×10^{2}	123	3.550×10^{1}
4	1.455×10^{7}	44	1.283×10^{5}	84	1.885×10^{2}	124	3.500×10^{1}
5	1.384×10^{7}	45	1.000×10^{5}	85	1.877×10^{2}	125	3.375×10^{1}
6	1.284×10^{7}	46	8.500×10^{4}	86	1.800×10^{2}	126	3.325×10^{1}
7	1.000×10^{7}	47	8.200×10^{4}	87	1.700×10^{2}	127	3.175×10^{1}
8	8.187×10^{6}	48	7.500×10^{4}	88	1.430×10^{2}	128	3.125×10^{1}
9	6.434×10^{6}	49	7.300×10^{4}	89	1.220×10^{2}	129	3.000×10^{1}
10	4.800×10^{6}	50	6.000×10^{4}	90	1.190×10^{2}	130	2.750×10^{1}
11	4.304×10^{6}	51	5.200×10^{4}	91	1.175×10^{2}	131	2.500×10^{1}
12	3.000×10^{6}	52	4.900×10^{4}	92	1.160×10^{2}	132	2.250×10^{1}
13	2.479×10^{6}	53	4.500×10^{4}	93	1.130×10^{2}	133	2.175×10^{1}
14	2.354×10^{6}	54	3.000×10^{4}	94	1.080×10^{2}	134	2.120×10^{1}
15	1.850×10^{6}	55	2.000×10^{4}	95	1.050×10^{2}	135	2.050×10^{1}
16	1.500×10^{6}	56	1.700×10^{4}	96	1.012×10^{2}	136	2.000×10^{1}
17	1.400×10^{6}	57	1.300×10^{4}	97	9.700×10^{1}	137	1.940×10^{1}
18	1.356×10^{6}	58	9.500×10^{3}	98	9.000×10^{1}	138	1.850×10^{1}
19	1.317×10^{6}	59	8.030×10^{3}	99	8.170×10^{1}	139	1.700×10^{1}
20	1.250×10^{6}	60	5.700×10^{3}	100	8.000×10^{1}	140	1.600×10^{1}
21	1.200×10^{6}	61	3.900×10^{3}	101	7.600×10^{1}	141	1.440×10^{1}
22	1.100×10^{6}	62	3.740×10^{3}	102	7.200×10^{1}	142	1.290×10^{1}
23	1.010×10^{6}	63	3.000×10^{3}	103	6.750×10^{1}	143	1.190×10^{1}
24	9.200×10^{5}	64	2.500×10^{3}	104	6.500×10^{1}	144	1.150×10^{1}
25	9.000×10^{5}	65	2.250×10^{3}	105	6.300×10^{1}	145	1.000×10^{1}
26	8.750×10^{5}	66	2.200×10^{3}	106	6.100×10^{1}	146	9.880×10^{0}
27	8.611×10^{5}	67	1.800×10^{3}	107	5.800×10^{1}	147	8.100×10^{0}
28	8.210×10^{5}	68	1.550×10^{3}	108	5.340×10^{1}	148	7.150×10^{0}
29	7.500×10^{5}	69	1.500×10^{3}	109	5.060×10^{1}	149	7.000×10^{0}
30	6.790×10^{5}	70	1.150×10^{3}	110	4.810×10^{1}	150	6.875×10^{0}
31	6.700×10^{5}	71	9.500×10^{2}	111	4.520×10^{1}	151	6.750×10^{0}
32	6.000×10^{5}	72	6.830×10^{2}	112	4.400×10^{1}	152	6.500×10^{0}
33	5.730×10^{5}	73	6.700×10^{2}	113	4.240×10^{1}	153	6.250×10^{0}
34	5.500×10^{5}	74	4.540×10^{2}	114	4.100×10^{1}	154	6.000×10^{0}
35	4.920×10^{5}	75	3.050×10^{2}	115	3.960×10^{1}	155	5.400×10^{0}
36	4.700×10^{5}	76	2.850×10^{2}	116	3.910×10^{1}	156	5.000×10^{0}
37	4.400×10^{5}	77	2.400×10^{2}	117	3.800×10^{1}	157	4.700×10^{0}
38	4.200×10^{5}	78	2.200×10^{2}	118	3.763×10^{1}	158	4.000×10^{0}
39	4.000×10^{5}	79	2.095×10^{2}	119	3.727×10^{1}	159	3.730×10^{0}
40	3.300×10^{5}	80	2.074×10^{2}	120	3.713×10^{1}	160	3.500×10^{0}

Table 14. 252 intermediate-group energy bounds for reaction rate tallies with `Shift` (continued)

Grp	Upper Energy (eV)	Grp	Upper Energy (eV)	Grp	Upper Energy (eV)
161	3.300×10^{0}	201	1.030×10^{0}	241	7.500×10^{-3}
162	3.100×10^{0}	202	1.020×10^{0}	242	5.000×10^{-3}
163	3.000×10^{0}	203	1.010×10^{0}	243	4.000×10^{-3}
164	2.970×10^{0}	204	1.000×10^{0}	244	3.000×10^{-3}
165	2.870×10^{0}	205	9.750×10^{-1}	245	2.500×10^{-3}
166	2.770×10^{0}	206	9.500×10^{-1}	246	2.000×10^{-3}
167	2.670×10^{0}	207	9.250×10^{-1}	247	1.500×10^{-3}
168	2.570×10^{0}	208	9.100×10^{-1}	248	1.200×10^{-3}
169	2.470×10^{0}	209	8.500×10^{-1}	249	1.000×10^{-3}
170	2.380×10^{0}	210	8.000×10^{-1}	250	7.500×10^{-4}
171	2.300×10^{0}	211	7.500×10^{-1}	251	5.000×10^{-4}
172	2.210×10^{0}	212	7.000×10^{-1}	252	1.000×10^{-4}
173	2.120×10^{0}	213	6.500×10^{-1}	253	1.000×10^{-5}
174	2.000×10^{0}	214	6.250×10^{-1}		
175	1.940×10^{0}	215	6.000×10^{-1}		
176	1.860×10^{0}	216	5.500×10^{-1}		
177	1.770×10^{0}	217	5.000×10^{-1}		
178	1.680×10^{0}	218	4.500×10^{-1}		
179	1.590×10^{0}	219	4.000×10^{-1}		
180	1.500×10^{0}	220	3.750×10^{-1}		
181	1.450×10^{0}	221	3.500×10^{-1}		
182	1.400×10^{0}	222	3.200×10^{-1}		
183	1.350×10^{0}	223	3.000×10^{-1}		
184	1.300×10^{0}	224	2.750×10^{-1}		
185	1.250×10^{0}	225	2.500×10^{-1}		
186	1.225×10^{0}	226	2.200×10^{-1}		
187	1.200×10^{0}	227	2.000×10^{-1}		
188	1.175×10^{0}	228	1.750×10^{-1}		
189	1.150×10^{0}	229	1.500×10^{-1}		
190	1.140×10^{0}	230	1.250×10^{-1}		
191	1.130×10^{0}	231	1.000×10^{-1}		
192	1.120×10^{0}	232	9.000×10^{-2}		
193	1.110×10^{0}	233	8.000×10^{-2}		
194	1.100×10^{0}	234	7.000×10^{-2}		
195	1.090×10^{0}	235	6.700×10^{-2}		
196	1.080×10^{0}	236	5.000×10^{-2}		
197	1.070×10^{0}	237	4.000×10^{-2}		
198	1.060×10^{0}	238	3.000×10^{-2}		
199	1.050×10^{0}	239	2.530×10^{-2}		
200	1.040×10^{0}	240	1.000×10^{-2}		

Table 15. 11 coarse-group energy bounds for reaction rate tallies with `Shift`

Grp	Upper Energy (eV)	Grp	Upper Energy (eV)
1	2.000×10^{7}	7	4.000×10^{0}
2	1.353×10^{6}	8	1.300×10^{0}
3	5.000×10^{5}	9	6.250×10^{-1}
4	1.830×10^{5}	10	1.800×10^{-1}
5	9.118×10^{3}	11	8.000×10^{-2}
6	1.487×10^{2}	12	1.000×10^{-5}

Table 16. `Shift` eigenvalue calculation parameters for TRISO HPR unit cell

kcode parameter	Value
total cycles	1030
inactive cycles	30
histories per cycle	1×10^{6}

Table 17. `Griffin` calculation parameters for TRISO HPR unit cell (default)

Method	Nonlinear Tol. Rel/Abs	Linear Tol. Rel
Diffusion	$1 \times 10^{-8}/1 \times 10^{-50}$	1×10^{-5}
SPH	$1 \times 10^{-8}/1 \times 10^{-50}$	1×10^{-5}
SPH-corrected Diffusion	$1 \times 10^{-08}/1 \times 10^{-50}$	1×10^{-5}

Table 18. TRISO HPR unit cell eigenvalue and pin power comparison

Code	Mesh Refine	SPH	XS Sets	k_{eff}	Δk	Abs. PP RMS
`Serpent-ANL`	–	–	–	0.97811 ± 4 pcm[a]	–	–
`Shift`	–	Reference	–	0.97694 ± 3 pcm		–
`Shift-Griffin`	0	No	4	0.83828	-13,866	10.60%
`Shift-Griffin`	1	No	4	0.83631	-14,063	10.55%
`Shift-Griffin`	0	Yes	4	0.97694	0	0.70%
`Shift-Griffin`	1	Yes	4	0.97694	0	0.70%
`Shift-Griffin`	0	No	5	0.83785	-13,909	1.99%
`Shift-Griffin`	1	No	5	0.83589	-14,105	1.86%
`Shift-Griffin`	0	Yes	5	0.97694	0	0.70%
`Shift-Griffin`	1	Yes	5	0.97694	0	0.70%

[a] From [4]

Table 19. TRISO HPR unit cell SPH correction factor comparison

Code	Mesh Refine	XS Sets	Min	Max
Shift-Griffin	0	4	0.91	1.83
Shift-Griffin	1	4	0.93	1.79
Shift-Griffin	0	5	1.20	1.84
Shift-Griffin	1	5	1.23	1.79

5.2 TRISO HPR FULL CORE TWO-STEP RESULTS

This section presents the Shift-generated cross sections and reference solution and the Griffin results from two-step neutronics calculations of the TRISO HPR full-core model. For this analysis, the full-core model using an infinite lattice described in Section 2.6 was used for the Shift calculations.

The same energy group structure as that used for the TRISO unit cell was used to generate the cross section set for the full core. The set of ten homogenized cross sections was generated with Shift at a temperature of 300 K and included the following homogenized regions:

- One region including the TRISO particles in the fuel pin and surrounding monolith hex over all fuel pins,
- One region including the heat pipe wick, the surrounding air, and the surrounding monolith hex over all heat pipe pins,
- One region including the moderator, the surrounding air, and the surrounding monolith hex over all moderator pins,
- One region for the beryllium reflector region below the fuel assemblies,
- One region for the beryllium reflector region above the fuel assemblies including the heat pipes extending in this region,
- One region for the pure monolith hexes,
- One region for the beryllium reflector hexes,
- One region for the center void that was modeled as air,
- One region for the beryllium and air in the control drums, and
- One region for the B_4C in the control drums.

The center void cross section values were replaced such that it had a diffusion coefficient of unity and all other cross sections were zero.

To ensure the cross section set generated with Shift was adequate, a 2D assembly, 3D assembly, 2D core, and 3D core were run with Griffin using the same cross sections. The Shift calculations were performed on an ORNL institutional computing cluster. The Griffin calculations were performed on the INL high-performance computing cluster Sawtooth [22]. Again, CFEM-diffusion was used for the Griffin solver. Tables 20 and 21 show some of the relevant Shift eigenvalue and Griffin diffusion solver parameters used in these calculations.

Table 22 shows the calculated eigenvalues from Shift, Serpent, and Griffin. First, both MC reference solutions match very well. Second, a higher eigenvalue for the 2D assembly, 3D assembly, and 2D core can be observed, as expected from using the same homogenized cross section set for all of these calculations, but approaching the 3D core eigenvalue as the geometry becomes more relevant. Third, the Griffin eigenvalue without SPH correction is 322 pcm within the Shift reference. Current work is focusing on generating 3D full core SPH factors and running the corrected diffusion calculation with Griffin.

Table 20. `Shift` eigenvalue calculation parameters for TRISO HPR full core

kcode parameter	Value
total cycles	2100
inactive cycles	100
histories per cycle	1×10^6

Table 21. `Griffin` calculation parameters for all models (default)

Method	Nonlinear Tol. Rel/Abs	Linear Tol. Rel
Diffusion	$1 \times 10^{-8}/1 \times 10^{-50}$	1×10^{-5}

5.3 FUTURE WORK

One area of ongoing work regarding cross section generation analysis is the investigation of a hybrid scattering matrix tally in `Shift` to produce a more accurate approximation of the multigroup scattering matrix. The scattering matrix cross section generated in the above calculations uses a collision-based tally in `Shift`. The authors propose to investigate the combination of typical reaction path-length–based and collision-based estimators to achieve better variance and tally scores on the off-diagonal components of the scattering matrix. This approach should provide a better estimate of the scattering cross section in regions of small volume with low collision scores. Note that this hybrid scattering matrix tally would not use the path-length scattering response function tally described in Section 4.3.2; it would use the traditional MT reaction path-length–based tallies in `Shift`.

The following example provides an illustration. Consider a pincell that consists of a fuel region, f, and a coolant region, c. The scattering reaction rate for this pincell can be written as:

$$\Sigma_{s,fc}^{g'\to g}\phi_{fc}^{g'}V_{fc} = \Sigma_{s,f}^{g'\to g}\phi_{f}^{g'}V_{f} + \Sigma_{s,c}^{g'\to g}\phi_{c}^{g'}V_{c}, \tag{13}$$

where

$$\Sigma_{s,x}^{g'\to g} = \Sigma_{s,x}^{g'}\rho_{x}^{g'\to g} \text{ for } x \in \{f, c\}, \tag{14}$$

Table 22. TRISO HPR assembly and core models eigenvalue comparison

Code	Model	$\mathbf{k}_{\text{eff}}$	$\Delta\mathbf{k}$
`Serpent`-ANL	Full Core	1.06609 ± 15 pcm[a]	–
`Serpent`	Full Core	1.06188 ± 3 pcm	–
`Shift`	Full Core	1.06188 ± 3 pcm	Reference
`Shift-Griffin`	2D Assembly	1.24952	
`Shift-Griffin`	3D Assembly	1.19809	
`Shift-Griffin`	2D Core	1.09998	
`Shift-Griffin`	Full Core no SPH	1.05866	322

[a] From [15]

and $\rho_x^{g' \to g}$ is the collision-based scattering probability matrix for a material region. Therefore, the multigroup scattering cross section for the pincell can be calculated as

$$\Sigma_{s,fc}^{g' \to g} = \frac{\Sigma_{s,f}^{g'} \rho_f^{g' \to g} \phi_f^{g'} V_f + \Sigma_{s,c}^{g'} \rho_c^{g' \to g} \phi_c^{g'} V_c}{\phi_{fc}^{g'} V_{fc}}. \tag{15}$$

The usual collision-based scattering matrix probability tally in `Shift` is used to calculate $\rho_x^{g' \to g}$ in Eq. 15, and the scattering reaction rates are determined by reaction rate path-length–based tallies in `Shift`.

6. CONCLUSIONS AND FUTURE WORK

This report documents the improvements made to Shift targeted at TRISO modeling, automated multigroup cross section generation, and reactor physics advanced reactor workflows.

The key takeaways from the TRISO modeling memory and performance comparisons are as follows:

- Performance comparison done with multiple models of varying complexity (single pebble up to complex full core) and with different modeling approaches to realize random particle locations.
- The *randommix* block that is available through SCALE's Shift interface is the most user-friendly modeling option for random particle distributions and provides among the best performance.
- The latest acceleration techniques implemented in Shift for SCALE 7.0 reduce the runtime by up to 36% and the memory by up to 34% compared to SCALE 6.3.0 for all modeling options besides the *randommix* model. For the *randommix* model, a dramatic reduction in runtime by a factor of 34 was achieved.
- When running SCALE through the Omnibus frontend via an ORANGE geometry, a new capability *replica* allows for a simple definition of all particle coordinates and provides the best overall performance in terms of memory and runtime.
- The performance for the best random particle models is similar to the performance of models with regular particle lattice. Starting with Shift in SCALE 6.3.0, random particle realizations no longer cause a long runtime penalty.
- Serpent's delta tracking is still outperforming all Shift calculations. However, Shift already shows similar or better performance than Serpent's surface tracking calculations, and further acceleration is expected.

The generation of multigroup cross sections with Shift is now automated to directly output these values to the Shift HDF5 output file. The tally input specification to generate these cross sections has also been simplified to allow for more automation and less user specification. Also, multiplicity and higher-order moments were added to the scattering matrix tally. This automation has been validated by reproducing previous two-step neutronics results for the Empire 2D assembly. Further validation was performed by running two-step neutronics calculations with Shift and Griffin for a TRISO HPR unit cell and full core. Comparison to previous Serpent and Griffin results shows that Shift-generated cross sections for these two models produce comparable eigenvalues and pin powers.

An initial implementation of a new reactor physics interface to Shift, Titan, will be available in the beta release of SCALE 7.0.0. This new interface allows for an engineering-style model definition with ray tracing capability and the ability to easily couple Shift to other codes for advanced reactor workflows under NEAMS. It currently supports basic geometric types, especially for modeling TRISO particles and pebbles, and performs automatic deduplication of materials.

Finally, an initial implementation of a path-length–based scattering matrix tally was incorporated into Shift. This tally uses a vector response function to form a path-length version of a scattering matrix tally (based on the work by Nelson [23]). Investigation into the effectiveness of reducing statistical variation of the scattering matrix and performance of this path-length tally is ongoing.

6.1 FUTURE WORK

Future analysis with and enhancements to `Shift` for advanced reactors are given in detail in Sections 3.7, 5.3, 4.2.5, and 4.3.2. A summary of the future work suggestions from these sections is given below.

- *MC tracking performance on TRISO models*
  - Enable the automatic generation of random coordinates for low and high packing fractions,
  - Investigate performance of pebbles and particles as holes,
  - Investigate performance with tallies enabled,
  - Investigate difference in standard deviation between the difference MC codes.
- *Multigroup cross section generation*
  - Investigate advantage of hybrid scattering matrix tally (path length and collision)
- *`Titan` reactor physics interface*
  - Add support for more concrete unit types,
  - Add ability to perform time-dependent and burnup calculations,
  - Integrate with deterministic codes.
- *Path-length–based scattering matrix*
  - Add support for higher-order scattering moments,
  - Include all elastic scattering kinematics,
  - Investigate performance and memory utilization and tradeoff with scattering matrix accuracy.

Beyond these considerations, there are also new features and enhancements needed for the tallies and multigroup cross sections generated with `Shift`. These include the following:

- `Shift` can be modified to compute energy-released and energy-deposited cross sections based on continuous energy (CE) library κ and heating values.
- Add nuclide-dependent ν-fission path-length tallies, which can also be used to compute forward-weighted homogenized kinetic parameters.
- Improve the `Shift` hexagonal mesh tally capability.
- Implement hydrogen-based transport correction ratios for diffusion coefficient calculations of thermal systems dominated by hydrogen scattering.

7. ACKNOWLEDGMENTS

The authors would like to thank the `Griffin` developers and users for their invaluable assistance especially Yan Cao, Nicholas Stauff, and Javier Ortensi. Also, thanks to Cihangir Celik for helping with generating the `SCALE` TRISO models.

This research made use of Idaho National Laboratory computing resources which are supported by the Office of Nuclear Energy of the US Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

REFERENCES

[1] Tara Pandya, Matthew A. Jessee, Tarek Ghaddar, and Friederike Bostelmann. Methods and Usability Enhancements in Shift for Non-LWR Applications. Technical Report ORNL/TM-2021/2280, Oak Ridge National Laboratory, Oak Ridge, TN, 2021.

[2] Tara Pandya, Friederike Bostelmann, Matthew A. Jessee, and Javier Ortensi. Two-step neutronics calculations with Shift and Griffin for advanced reactor systems. *Annals of Nuclear Energy*, 173(1), August 2022.

[3] Nicolas E Stauff, Kun Mo, Changho Lee, Zhi-Gang Mei, Yeon Sang Jung, Christopher Matthews, and Bo Feng. Multi-Physics Simulations of Heat-Pipe TRISO-Fueled Micro-Reactor. In *ANS Virtual Winter Meeting, November 16–19*, 2020.

[4] Nicolas Stauff, Yan Cao, Justin Thomas, Yinbin Miao, L. Zou, D. Nunez, Emily Shemon, Bo Feng, and K. Ni. Detailed analyses of a TRISO-fueled microreactor: Modeling of a Micro-Reactor System using NEAMS tools. Technical Report ANL/NEAMS-21/3, ANL, 2021.

[5] Seth Johnson, Thomas Evans, Gregory Davidson, Steven Hamilton, Tara Pandya, Katherine Royston, and Elliott Biondo. *Omnibus User Manual*. Technical Report ORNL/TM-2018/1073, ORNL, August 2020.

[6] S. Palmtag, A. T. Godfrey, M. Baird, and E. Walker. *VERAIn User's Manual*. Technical Report ORNL/SPR-2022/2509, ORNL, August 2022.

[7] Friederike Bostelmann, Cihangar Celik, Robert Kile, and William Wieselquist. SCALE Analysis of a Fluoride Salt Cooled High Temperature Reactor in Support of Severe Accident Analysis. Technical report, Oak Ridge National Laboratory, 2021.

[8] William K. Terry, Leland M. Montierth, Soon Sam Kim, Joshua J. Cogliati, and Abderrafi M. Ougouag. Evaluation of the Initial Critical Configuration of the HTR-10 Pebble-Bed Reactor. Technical Report HTR10-GCR-RESR-001, NEA/NSC/DOC(2006)1, Rev. 0, OECD/NEA, 2007.

[9] Eva E Sunny and Germina Ilas. SCALE 6 Analysis of HTR-10 Pebble-Bed Reactor for Initial Critical Configuration. In *PHYSOR 2010, Pittsburgh, Pennsylvania, USA, May 9–14*, 2010.

[10] Germina Ilas. On SCALE validation for PBR analysis. *International Conference on the Physics of Reactors 2010, PHYSOR 2010*, pages 1202–1211, 2010.

[11] Friederike Bostelmann, Cihangir Celik, Mark L. Williams, Ronald J. Ellis, Germina Ilas, and William A. Wieselquist. SCALE Capabilities for High Temperature Gas-Cooled Reactor Analysis. *Annals of Nuclear Energy*, 147:107673, 2020.

[12] NEA. PBMR Coupled Neutronics/Thermal-hydraulics Transient Benchmark, The PBMR-400 Core Design - Volume 1: The Benchmark Definition. Technical Report NEA/NSC/DOC(2013)10, OECD/NEA, 2013.

[13] IAEA. Evaluation of High Temperature Gas Cooled Reactor Performance: Benchmark Analysis Related to the PBMR-400, PBMM, GT-MHR, HTR-10 and the ASTRA Critical Facility. Technical Report IAEA-TECDOC-1694, IAEA, 2013.

[14] Steven E. Skutnik and William A. Wieselquist. Assessment of ORIGEN Reactor Library Development for Pebble-Bed Reactors Based on the PBMR-400 Benchmark. Technical Report ORNL/TM-2020/1886, Oak Ridge National Laboratory, Oak Ridge, TN, 2021.

[15] Nicolas E Stauff, Kun Mo, Yan Cao, Justin W Thomas, Yinbin Miao, Changho Lee, Christopher Matthews, and Los Alamos. Preliminary Applications of NEAMS Codes for Multiphysics Modeling of a Heat Pipe Microreactor. In *ANS Virtual Annual Meeting, June 14–16*, 2021.

[16] William A. Wieselquist, R. A. Lefebvre, and Matthew A. Jessee. SCALE Code System, Version 6.2.4. Technical Report ORNL/TM-2005/39, Oak Ridge National Laboratory, Oak Ridge, TN, 2020.

[17] Jaakko Leppänen, Maria Pusa, Tuomas Viitanen, Ville Valtavirta, and Toni Kaltiaisenaho. The Serpent Monte Carlo Code: Status, Development and Applications in 2013. *Annals of Nuclear Energy*, 82:142–150, 2015.

[18] T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. J. Cox, J. Durkee, J. Elson, M. Fensin, R. A. Forster, J. Hendricks, H. G. Hughes, R. Johns, B. Kiedrowski, R. Martz, S. Mashnik, G. McKinney, D. Pelowitz, R. Prael, J. Sweezy, L. Waters, T. Wilcox, and T. Zukaitis. Initial MCNP6 Release Overview. *Nuclear Technology*, 180:298–315, 2012.

[19] Jaakko Leppänen. Performance of Woodcock Delta-Tracking in Lattice Physics Applications Using the Serpent Monte Carlo Reactor Physics Burnup Calculation Code. *Annals of Nuclear Energy*, 37(5):715–722, 2010.

[20] Jaakko Leppänen. Two practical methods for unionized energy grid construction in continuous-energy monte carlo neutron transport calculation. *Annals of Nuclear Energy*, 36(7):878–885, 2009.

[21] A. Hébert and G. Mathonnière. Development of a third-generation superhomogénéisation method for the homogenization of a pressurized water reactor assembly. *Nuclear Science and Engineering*, 115(2):129–141, 1993.

[22] Idaho National Laboratory High Performance Computing. https://hpc.inl.gov/sitepages/home.aspx, May 2021.

[23] Adam Nelson. *Improved Convergence Rate of Multi-Group Scattering Moment Tallies for Monte Carlo Neutron Transport Codes*. PhD thesis, University of Michigan, 2014.

APPENDIX A. TITAN OUTPUT

APPENDIX A. TITAN OUTPUT

Listing A.1. Full Titan output

```
/ :
* geometry :
  * units :
    * [F] :
      * type = PIN
      * value :
        * [0] :
          * material = "FUEL"
          * radius = 4.0960000000000e-01
        * [1] :
          * material = "GAP"
          * radius = 4.1800000000000e-01
        * [2] :
          * material = "CLAD"
          * radius = 4.7500000000000e-01
    * [W] :
      * type = POLYGON_DOMAIN
      * value :
        * material = "WATER"
        * polygon :
          * num_sides = 4
          * apothem = 6.3000000000000e-01
          * corner_rad = 0.0000000000000e+00
          * rotation = 0.0000000000000e+00
          * indent_thickness = 0.0000000000000e+00
          * indent_length = 0.0000000000000e+00
          * indent_offset_length = 0.0000000000000e+00
        * inserts :
          * [0] :
            * unit = F
            * transform = Empty vector
  * global = W
  * dimension = "2D"
  * height = 1.0000000000000e+00
  * bcs :
    * [mz] = "REFLECT"
    * [p0] = "REFLECT"
    * [p1] = "REFLECT"
    * [p2] = "REFLECT"
    * [p3] = "REFLECT"
    * [pz] = "REFLECT"
* materials :
  * [CLAD] :
    * temp = 3.0000000000000e+02
    * concentrations :
      * [0] :
        * zzzaaam = 40090
        * density = 2.2178550000000e-02
  * [FUEL] :
    * temp = 3.0000000000000e+02
    * concentrations :
      * [0] :
        * zzzaaam = 922350
        * density = 6.7635000000000e-03
      * [1] :
```

```
        * zzzaaam = 922380
        * density = 2.7482100000000e-02
  * [GAP] :
    * temp = 3.0000000000000e+02
    * concentrations :
      * [0] :
        * zzzaaam = 70140
        * density = 1.0000000000000e-10
  * [WATER] :
    * temp = 4.0000000000000e+02
    * concentrations :
      * [0] :
        * zzzaaam = 80160
        * density = 2.2020360000000e-02
      * [1] :
        * zzzaaam = 10010
        * density = 4.4142920000000e-02
* title = "PINCELL MODEL"
* shift :
  * num_particles = 100
  * num_cycles = 15
  * num_inactive_cycles = 5
  * seed = 1234
  * dbrc = false
  * keff_init = 1.3000000000000e+00

***********************************************
"unit[1](POLYGON_DOMAIN)" (Titan/csg/ORANGE_CSG.hh:66)
***********************************************
:Type:         masked unit
:# Cells:      3 (offset = 0)
:# Surfaces:   7 (offset = 0)
:Bounding box: ``{-0.63 -0.63 0 to 0.63 0.63 1}``

======= = ================================ ============================================================
Cell    Z Name                             Surface logic
======= = ================================ ============================================================
0       X unit[1](POLYGON_DOMAIN)_boundary 0 ~ 1 & 2 & 3 ~ & 4 & 5 ~ & ~
                                           (from ``Titan/csg/ORANGE_CSG.cc:89``)
1       H unit[1](PIN)                     4 5 ~ & 6 ~ &
                                           (from ``Titan/csg/ORANGE_CSG.cc:112``)
2       M [WATER]cell[4]                   0 ~ 1 & 2 & 3 ~ & 4 & 5 ~ &
                                           (from ``Titan/csg/ORANGE_CSG.cc:58``)
======= = ================================ ============================================================

Cells with reentrant surface tracking: "unit[1](POLYGON_DOMAIN)_boundary"

======= ============================ ============================================================
Surface Name                         Description
======= ============================ ============================================================
0                                    Plane: y=0.63
        [WATER]cell[4]shape[poly].p0 (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].p0 (from ``Titan/shapes/Shape.cc:59``)
1                                    Plane: x=-0.63
        [WATER]cell[4]shape[poly].p1 (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].p1 (from ``Titan/shapes/Shape.cc:59``)
2                                    Plane: y=-0.63
        [WATER]cell[4]shape[poly].p2 (from ``Titan/shapes/Shape.cc:59``)
        [WATER]cell[4]shape[poly].p2 (from ``Titan/shapes/Shape.cc:59``)
```

```
3                                   Plane: x=0.63
        [WATER]cell[4]shape[poly].p3 (from ``Titan/shapes/Shape.cc:59'')
        [WATER]cell[4]shape[poly].p3 (from ``Titan/shapes/Shape.cc:59'')
4                                   Plane: z=0
        [WATER]cell[4]shape[poly].mz (from ``Titan/shapes/Shape.cc:59'')
        [WATER]cell[4]shape[poly].mz (from ``Titan/shapes/Shape.cc:59'')
        unit[1](PIN).mz             (from ``Titan/csg/ORANGE_CSG.cc:112'')
5                                   Plane: z=1
        [WATER]cell[4]shape[poly].pz (from ``Titan/shapes/Shape.cc:59'')
        [WATER]cell[4]shape[poly].pz (from ``Titan/shapes/Shape.cc:59'')
        unit[1](PIN).pz             (from ``Titan/csg/ORANGE_CSG.cc:112'')
6                                   Cyl z: r=0.475
        unit[1](PIN).coz            (from ``Titan/csg/ORANGE_CSG.cc:112'')
======= ========================== ============================================================

=============================== ================================================
Cell                            Fill
=============================== ================================================
unit[1](POLYGON_DOMAIN)_boundary ---
unit[1](PIN)                    unit[1](PIN) {Rotate 1 0 0 0 1 0 0 0 1}
[WATER]cell[4]                  Material 4
=============================== ================================================

Reflecting boundaries: [WATER]cell[4]shape[poly].p0, [WATER]cell[4]shape[poly].p1, [WATER]cell[4]shap

**********************************************
"unit[1](PIN)" (Titan/csg/ORANGE_CSG.hh:66)
**********************************************
:Type:         simple unit
:# Cells:      4 (offset = 3)
:# Surfaces:   4 (offset = 7)
:Bounding box: ``{-0.475 -0.475 0 to 0.475 0.475 1}''

======= ==================== ================================================================
Cell    Name                 Surface logic
======= ==================== ================================================================
0       unit[1](PIN)_boundary * ~
                             (from ``Titan/csg/ORANGE_CSG.cc:89'')
1       [FUEL]cell[1]        0 ~ * &
                             (from ``Titan/csg/ORANGE_CSG.cc:58'')
2       [GAP]cell[2]         0 * & 3 ~ &
                             (from ``Titan/csg/ORANGE_CSG.cc:58'')
3       [CLAD]cell[3]        * 3 & * &
                             (from ``Titan/csg/ORANGE_CSG.cc:58'')
======= ==================== ================================================================

Implicitly defined cells: "unit[1](PIN)_boundary"

======= ========================== ============================================================
Surface Name                       Description
======= ========================== ============================================================
0                                  Cyl z: r=0.4096
        [FUEL]cell[1]shape[cyl].coz (from ``Titan/shapes/Shape.cc:59'')
        [GAP]cell[2]shape[ring].ciz (from ``Titan/shapes/Shape.cc:59'')
1                                  Plane: z=0
        [CLAD]cell[3]shape[cyl].mz  (from ``Titan/shapes/Shape.cc:59'')
        [CLAD]cell[3]shape[ring].mz (from ``Titan/shapes/Shape.cc:59'')
        [FUEL]cell[1]shape[cyl].mz  (from ``Titan/shapes/Shape.cc:59'')
        [GAP]cell[2]shape[ring].mz  (from ``Titan/shapes/Shape.cc:59'')
```

```
2                                   Plane: z=1
        [CLAD]cell[3]shape[cyl].pz   (from ‘‘Titan/shapes/Shape.cc:59‘‘)
        [CLAD]cell[3]shape[ring].pz  (from ‘‘Titan/shapes/Shape.cc:59‘‘)
        [FUEL]cell[1]shape[cyl].pz   (from ‘‘Titan/shapes/Shape.cc:59‘‘)
        [GAP]cell[2]shape[ring].pz   (from ‘‘Titan/shapes/Shape.cc:59‘‘)
3                                   Cyl z: r=0.418
        [CLAD]cell[3]shape[ring].ciz (from ‘‘Titan/shapes/Shape.cc:59‘‘)
        [GAP]cell[2]shape[ring].coz  (from ‘‘Titan/shapes/Shape.cc:59‘‘)
======= ============================ ================================================================

==================== ============================================================
Cell                 Fill
==================== ============================================================
unit[1](PIN)_boundary ---
[FUEL]cell[1]        Material 1
[GAP]cell[2]         Material 2
[CLAD]cell[3]        Material 3
==================== ============================================================


>>> Loading SCALE Standard Composition Library from /scale/data/cache/v1.3.0/scale.rev40.sclib
****************************************
 Compositions Description
****************************************


============================================================
:# Composition  ID= 1  (matid 1)  name= "FUEL"
    density=  13.503262 (g/cc)   temperature=   300.00 (K)
    fissionable= true   depletable= true
    ZAID     Number Density
 ----------  ---------------
     92235    6.763500e-03
     92238    2.748210e-02

============================================================
:# Composition  ID= 2  (matid 2)  name= "GAP"
    density=   0.000000 (g/cc)   temperature=   300.00 (K)
    fissionable= false   depletable= false
    ZAID     Number Density
 ----------  ---------------
      7014    1.000000e-10

============================================================
:# Composition  ID= 3  (matid 3)  name= "CLAD"
    density=   0.331904 (g/cc)   temperature=   300.00 (K)
    fissionable= false   depletable= false
    ZAID     Number Density
 ----------  ---------------
      4009    2.217855e-02

============================================================
:# Composition  ID= 4  (matid 4)  name= "WATER"
    density=   0.658739 (g/cc)   temperature=   400.00 (K)
    fissionable= false   depletable= false
    ZAID     Number Density
 ----------  ---------------
      8016    2.202036e-02
   8001001    4.414292e-02
```

```
============================================================
::: Beginning inactive cycles
 Cycle  k_cycle   Time (s)         Np
   0   1.658389  6.639334e-01        100
   1   1.730859  8.304776e-01        119
   2   1.475721  7.207497e-01        104
   3   1.666580  6.368681e-01         95
   4   1.636224  7.555143e-01        111
>>> Completed 5 inactive cycles
::: Beginning active cycles
 Cycle  k_cycle    k_avg    k_stdev        Time (s)      Np
   0   1.808716  1.808716  1.808716e+00  6.322631e-01         87
   1   1.570670  1.689693  1.190230e-01  6.775513e-01        101
   2   1.431242  1.603542  1.102001e-01  6.812073e-01        100
   3   1.345641  1.539067  1.011389e-01  6.022441e-01         92
   4   1.547398  1.540733  7.835960e-02  6.437300e-01         99
   5   1.587574  1.548540  6.445488e-02  7.588835e-01        115
   6   1.577632  1.552696  5.463262e-02  5.492430e-01         82
   7   1.494820  1.545462  4.786314e-02  6.323530e-01        100
   8   1.418737  1.531381  4.449782e-02  5.786129e-01         85
   9   1.561618  1.534405  3.991476e-02  7.594728e-01        111
>>> Completed 10 active cycles
Titan lives and has a keff of: 1.534405e+00 +- 3.991476e-02
```