



Improved Fuel Cycle Capability of Griffin for Fast Reactor Applications

September 2024

Changho Lee¹, Yeon Sang Jung¹, Shikhar Kumar¹, and Namjae Choi²

¹*Nuclear Science and Engineering Division, Argonne National Laboratory*

²*Nuclear Science and Technology Division, Idaho National Laboratory*

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

Improved Fuel Cycle Capability of Griffin for Fast Reactor Applications

Changho Lee¹, Yeon Sang Jung¹, Shikhar Kumar¹, and Namjae Choi²

¹**Nuclear Science and Engineering Division, Argonne National Laboratory**

²**Nuclear Science and Technology Division, Idaho National Laboratory**

September 2024

**Argonne National Laboratory
Idaho National Laboratory**

**Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under UChicago Argonne, LLC
Contract DE-AC02-06CH11357
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517**

Page intentionally left blank

ABSTRACT

Griffin is a MOOSE based reactor multiphysics analysis application jointly developed by Idaho National Laboratory and Argonne National Laboratory under the Department of Energy Office of Nuclear Energy Nuclear Energy Advanced Modeling and Simulation Program. This fiscal year, the fuel cycle capability has been significantly extended by improving the assembly shuffling option to allow flexible fuel reloading in the multi-cycle depletion calculation and incorporating decay between cycles. An equilibrium core calculation capability was also implemented to find an equilibrium core. Additionally, an enrichment search capability was added to determine the enrichment condition that allows a core to reach an equilibrium cycle with the end-of-cycle k -effective meeting a user-specified target value. The updated fuel management capability has been extensively tested using the three-dimensional ABTR problem with different batch schemes, exhibiting reasonable solutions in terms of manual shuffling, equilibrium cycle, and enrichment search calculations. The cross section generation workflow capability for fast reactors was further verified to produce microscopic cross sections as well as Griffin core inputs for an ring-heterogeneous configuration. Rigorous verification tests using the ABTR problem demonstrated that the RH core calculations, with cross sections and Griffin inputs generated from the cross section workflow of Griffin, produced accurate solution for fast reactor problems. In addition, an option to convert delay neutron parameter data generated from MC²-3 in the DLAYXS format into XML format was added to support transient calculations using MC²-3-generated data.

ACKNOWLEDGEMENTS

This work was supported by the U.S. Department of Energy, Office of Nuclear Energy, under contract DEAC02-06CH11357 for UChicago Argonne, LLC, and under contract DE-AC07-05ID14517 for DOE Idaho Operations Office and supported by the DOE Nuclear Energy Advanced Modeling and Simulation (NEAMS) program.

CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENT	iv
1 Introduction	1
2 Updates in Fuel Cycle Capabilities	2
2.1 Fuel Cycle Capabilities for Fast Reactor Analysis	2
2.2 Shuffling and Cycle Depletion	4
2.2.1 Implementation of Assembly Shuffling Capability	4
2.2.2 Implementation of Assembly Cooling During Core Reloading	4
2.2.3 Extended Verification Tests	5
2.3 Equilibrium Cycle Capability	9
2.3.1 Equilibrium Cycle Calculations	9
2.3.2 Implementation of Equilibrium Core Depletion Capability	10
2.3.3 Verification Tests	12
2.4 Search Capability	16
2.5 Demonstration Using the ABTR Core	17
3 Updates in Fast Reactor Cross Section Generation Workflow	21
3.1 Verification Tests	21
3.2 Delayed Neutron Data	24
4 Conclusions and Future Work	26
REFERENCES	26

FIGURES

Figure 1.	Overview of major fuel cycle models required for fast reactors.	3
Figure 2.	Example of user input for shuffling scheme	5
Figure 3.	Calculation flow of depletion calculation in Griffin	6
Figure 4.	ABTR core overview.....	7
Figure 5.	(a) A [FuelManagement] input example of fuel assembly shuffling and (b) illustration of the corresponding shuffling scheme of outer and inner cores	8
Figure 6.	Change of k -effective by outer and inner core fuel assembly shuffling.....	8
Figure 7.	Illustration of averaged core concept utilized in equilibrium core depletion	11
Figure 8.	Illustration of equilibrium core depletion scheme using a 3 batch cycle scheme	12
Figure 9.	Equilibrium cycle calculation workflow	13
Figure 10.	Fuel management scheme of equilibrium core depletion test problem (green region: 3-batch reloading with fuel type 1, blue region: 4-batch reloading with fuel type 2).....	14
Figure 11.	Eigenvalue result of equilibrium core depletion test problem.....	14
Figure 12.	Explicit modeling of batch reloading scheme for equilibrium core depletion test problem (green region: 3-batch with fuel type 1, blue region: 4-batch with fuel type 2, Number: batch reloading order).....	15
Figure 13.	Eigenvalue comparison between equilibrium core depletion and explicit core depletion for the 3-ring test problem	15
Figure 14.	A [FuelManagement] input example for equilibrium cycle calculation	18
Figure 15.	Comparison of equilibrium cycle and manual shuffling calculations.....	19
Figure 16.	Enrichment search results for the 3D ABTR problem.....	19
Figure 17.	Comparison of equilibrium conditions before and after enrichment search for the 3D ABTR problem	20
Figure 18.	Ring-wise fission (left) and absorption (right) reaction rate errors (%) of RH and SRH compared to FH for the 2D ABTR middle fuel assembly.	23
Figure 19.	Ring-wise fission (top) and absorption (bottom) reaction rate errors (%) of RH and SRH compared to FH for the 2D ABTR core.	23
Figure 20.	An input example for fast reactor cross section generation workflow using MC ² -3 for RH configuration.....	25

TABLES

Table 1. Comparison of FH, RH, and SRH meshes on a 2D ABTR fuel assembly.	21
--	----

ACRONYMS

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
ABTR	Advanced Burner Test Reactor
ANL	Argonne National Laboratory
BOC	beginning-of-cycle
EOC	end-of-cycle
FH	fully-heterogeneous
FY	fiscal year
INL	Idaho National Laboratory
MOOSE	Multiphysics Object-Oriented Simulation Environment
MOX	mixed oxide fuel
NEAMS	Nuclear Energy Advanced Modeling and Simulation
RGMB	Reactor Geometry Mesh Builder
RH	ring-heterogeneous
RMS	root mean square
SFR	sodium-cooled fast reactor
SRH	simple ring-heterogeneous

1. Introduction

Under the DOE-NE Nuclear Energy Advanced Modeling and Simulation (NEAMS) program, the main objective of Griffin [1], a reactor multiphysics analysis application developed collaboratively by Idaho National Laboratory (INL) and Argonne National Laboratory (ANL) utilizing the Multiphysics Object-Oriented Simulation Environment (MOOSE) framework [2], is to support the analysis of diverse advanced reactor designs, particularly for non-light water reactor configurations. In fast reactor applications, the primary objective of the FY24 work scope is to extend the cross section generation workflow and fuel cycle analysis capabilities to fulfill the required capabilities for fast reactor design and analysis.

As an extension of fuel cycle capabilities [3], we completed the fuel management system by adding fuel shuffling and assembly cooling options. This enables relocation of burned assemblies in the reloading process that occurs between reactor operation cycles. Users can easily specify the fuel shuffling pattern through user-friendly input formats implemented in the fuel management system. The radioactive decay of burned assemblies between cycles can also be considered as part of the core reloading options. This was achieved by extending the existing depletion functionality in Griffin. By leveraging these new capabilities in the fuel management scheme, we can simulate the realistic fuel cycles of both conventional and advanced reactors, including sodium-cooled fast reactors (SFRs). Additionally, a new capability to search for an equilibrium state for reactors operating under a fixed batch reloading scheme without shuffling was implemented. In this condition, the reactor approaches equilibrium core conditions through multiple depletion cycles. We implemented this equilibrium core depletion capability using an effective calculation scheme, which introduces an averaged core concept in the equilibrium status with implicit fuel reloading [4]. This allows users to quantify the overall performance and characteristics of a core over entire cycles.

This report is organized as follows: Section 2 describes the extension of fuel cycle capabilities in Griffin including the newly implemented equilibrium core search. Section 3 discusses the updates to cross section generation workflow. Finally, Section 4 presents the conclusions from our implementations and verification tests, and outlines future work.

2. Updates in Fuel Cycle Capabilities

In the previous year [3], the core functionality for shuffling and cycle depletion was implemented in Griffin. This fiscal year, it was enhanced by incorporating assembly cooling during fuel reloading. Comprehensive tests with realistic core sizes have been conducted to ensure this feature is suitable for practical design and analysis.

Additionally, a new capability for finding an equilibrium core has been introduced. This feature has been further extended to allow users to search for the conditions necessary to achieve equilibrium in the reactor core. However, it should be noted that the current implementation does not account for external cycles.

2.1 Fuel Cycle Capabilities for Fast Reactor Analysis

Fuel cycle management in fast reactors involves a series of activities to achieve performance objectives, such as fissile breeding, transuranic element burning, electricity generation, in an optimal manner throughout the reactor's operational lifetime. This includes managing fuel distribution, controlling power distribution, and managing the reactor's reactivity and refueling operations. In fast reactors, fuel cycle management is particularly important due to their use of high burnup fuel, which leads to complex isotope depletion and buildup patterns. Therefore, fuel management strategies need to be determined carefully in order to optimize reactor performance, improve fuel utilization, reduce waste production, and ultimately enhance safety margins.

The fuel management capabilities for fast reactors primarily requires two key types of analysis problems: 1) the explicit cycle-by-cycle (non-equilibrium) operation of a reactor under a specified periodic or non-periodic fuel management scheme and 2) the infinite-time or equilibrium cycle conditions of a reactor operating under a periodic fuel management scheme.

For a basic non-equilibrium cycle problem, the depletion equations are solved for user-specified initial fuel compositions and fuel management scheme. Initial or charged fuel compositions can be determined from external feeds based on user-specified fabrication preference. Optionally, reprocessing may be included in the external fuel cycle specification, allowing discharged fuel to be recycled back into the reactor.

An equilibrium cycle implies a reactor condition that is invariant for successive operation cycles under a fixed fuel management scheme and specific operating requirements. Although a real equilibrium cycle is not possible, near-equilibrium cycles can be achieved after a few transient cycles, starting from the startup

configuration by employing special fuel management procedures. The equilibrium cycle option is practically valuable for assessing the core performance of new reactors, as it is a more valid basis than using arbitrary startup cycles or constructing a complex fuel cycle with a single assembly shuffling pattern. The equilibrium solution offers idealized estimates of burn cycle time, control requirements, fuel enrichment, and general system performance characteristics at a lower computational cost than the explicit calculation of a proposed fuel management scheme. After completing an equilibrium cycle calculation, the resulting parameters and partially burned fuel densities can serve as input to a detailed step-by-step non-equilibrium cycle calculation in order to obtain specific operating characteristics of the reactor in its post-startup equilibrium state.

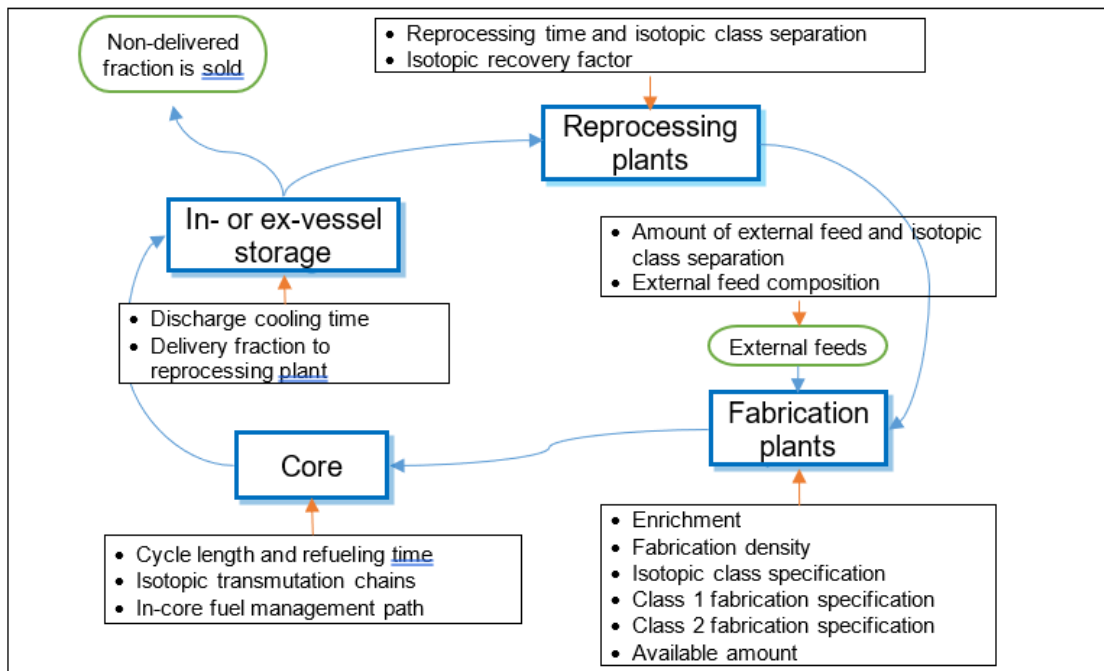


Figure 1: Overview of major fuel cycle models required for fast reactors.

The fuel cycle model can be subdivided into two parts: an in-reactor cycle and an ex-reactor (or external) cycle. Fig. 1 illustrates the models required for major fuel cycle calculations. The in-reactor fuel cycle calculation involves fuel loading, shuffling, and depletion. At the end of each cycle, some fuels may be discharged, and fresh fuels can be loaded into the core. In this process, the burned fuels remaining in the core can be shuffled based on a specified move sequence, referred to as fuel management scheme. The external cycle model represents the events that occur after fuel is discharged from the reactor. Hence, the ex-reactor model consists of the following successive steps: cooling, delivery to a reprocessing plant, reprocessing, re-fabrication using both reprocessed and external feed supplies, preloading storage, and reactor recharging.

In the previous fiscal year, we implemented the fuel management capability for fast reactors, focusing on multi-cycle depletion and shuffling functions. This fiscal year, we initiated implementing the equilibrium cycle capability without external cycle components, which will be discussed in the following sessions.

2.2 Shuffling and Cycle Depletion

2.2.1 Implementation of Assembly Shuffling Capability

The initial implementation of fuel cycle capabilities in FY23 [3] included support for a scattered reloading option. This option allows only for fresh fuel assemblies to be inserted into fixed locations, while the burned assemblies occupying those locations are discharged from the core. In FY24, the shuffling option was successfully added to the fuel management capabilities, which completes implementing the essential modeling features for realistic fuel cycle analysis. With this option, users can relocate burned assemblies to different core locations. An example of this updated shuffling option is shown in Fig. 2. The input parameter *assembly_id_pattern* specifies integer ID tags for fuel assembly locations. The *pattern* parameter in the *AssemblyShuffling* type designates assembly relocation using these integer ID tags. Negative IDs indicate fresh fuels specified in the core loading of the [Mesh] block. In the example, where 12 fuel assemblies are located in the second and third rings only, six fresh fuel assemblies are loaded into the third ring of the core, the six burned fuel assemblies in the third ring are relocated to the second ring, and the six burned fuel assemblies in the second ring are discharged from the core.

2.2.2 Implementation of Assembly Cooling During Core Reloading

The radioactive decay of isotopes loaded in the burned assembly during the time period spent for core reloading, commonly referred to as assembly cooling, should be considered in setting up the core compositions used at the beginning-of-cycle (BOC) in a multi-cycle depletion calculation. Short-lived isotopes decay out during this time period, which impacts the reactivity of the initial state of each cycle. The multi-cycle depletion capability implemented in FY23 did not consider this assembly cooling. To properly consider isotopic inventories of burned fuel assemblies in fuel cycle calculations, we implemented a functionality to incorporate assembly cooling in the fuel cycle calculation.

Assembly cooling is implemented by utilizing the existing depletion capabilities in Griffin. Specifically, the cooling calculation can be achieved by neglecting the neutron-induced reactions in the decay-

```

[FuelManagement]
  assembly_id_pattern =
    - - -;
    - 1 2 3 -;
    - 4 5 6 7 -;
    8 9 - 10 11;
    - 12 13 14 15 -;
    - 16 17 18 -;
    - - -;
[shuffle]
  type = AssemblyShuffling
  pattern =
    , 2 -2 7;
    -2 1 3 -2;
    4 8 - 11 15;
    -2 16 18 -2;
    12 -2 17'
[]
[]

```

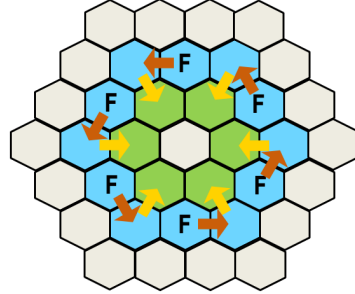


Figure 2: Example of user input for shuffling scheme

transmutation matrices used in the standard depletion calculation. We implemented a procedure to perform assembly cooling by reusing the existing functionalities in the depletion user objects. The neutron-related reaction can be removed from the depletion matrices by setting the flux level to zero in the standard depletion functions. After updating the core configurations based on the fuel management at the BOC, assembly cooling is performed. Users should additionally specify the time spent for core reloading for each fuel reloading scheme used in the fuel cycle calculations.

The calculation workflow for depletion calculations, including the newly added shuffling and assembly cooling, is shown in Fig. 3. As illustrated in the figure, the shuffling and assembly cooling treatments are performed at the BOC, which specifies the correct initial core inventories to be depleted during the cycle calculation.

2.2.3 Extended Verification Tests

We performed verification tests with more realistic cases, identifying and resolving additional issues during the process. Previously, tests were conducted with simple 2D three-assembly ring cores. This year, we used the three-dimensional (3D) Advanced Burner Test Reactor (ABTR) problem, applying practical

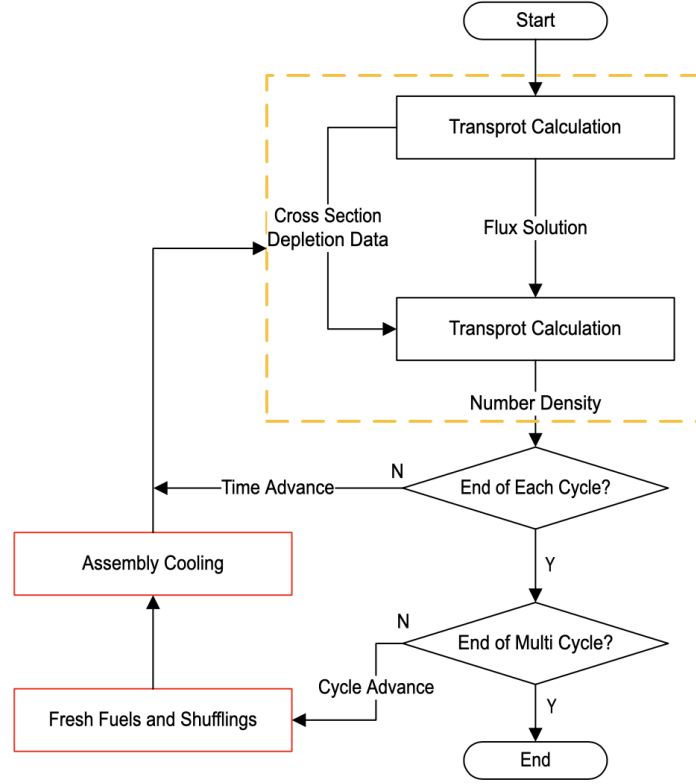


Figure 3: Calculation flow of depletion calculation in Griffin

shuffling and depletion schemes.

In the ABTR problem shown in Fig. 4, three types of fuel assemblies are loaded in the active core: 24 inner core driver fuel assemblies, 30 outer core driver fuel assemblies, and 6 fuel test assemblies. Since the compositions of the test assemblies are not determined, the fuel test assembly locations are loaded with driver fuel assemblies, while the material test assembly locations are loaded with reflector assemblies. For simplicity, the wire-wrap separating fuel pins was smeared with cladding, and the gap between B4C pellet and cladding was neglected in the control and shield assemblies. It is also noted that all the control assemblies are fully withdrawn, leaving the active core section filled with sodium coolant. All assemblies have the same lower and upper structures.

For verification purposes, REBUS [5] calculations for the 3D ABTR problem are available [6], which include non-equilibrium cycle problems with assembly reloading and shuffling, and charge enrichment search in either equilibrium cycle analysis or non-equilibrium cycle analysis. In the future, these benchmark test cases will be used to verify Griffin's calculations.

For these verification tests, the multigroup cross sections with the typical 9- and 33-group structures were

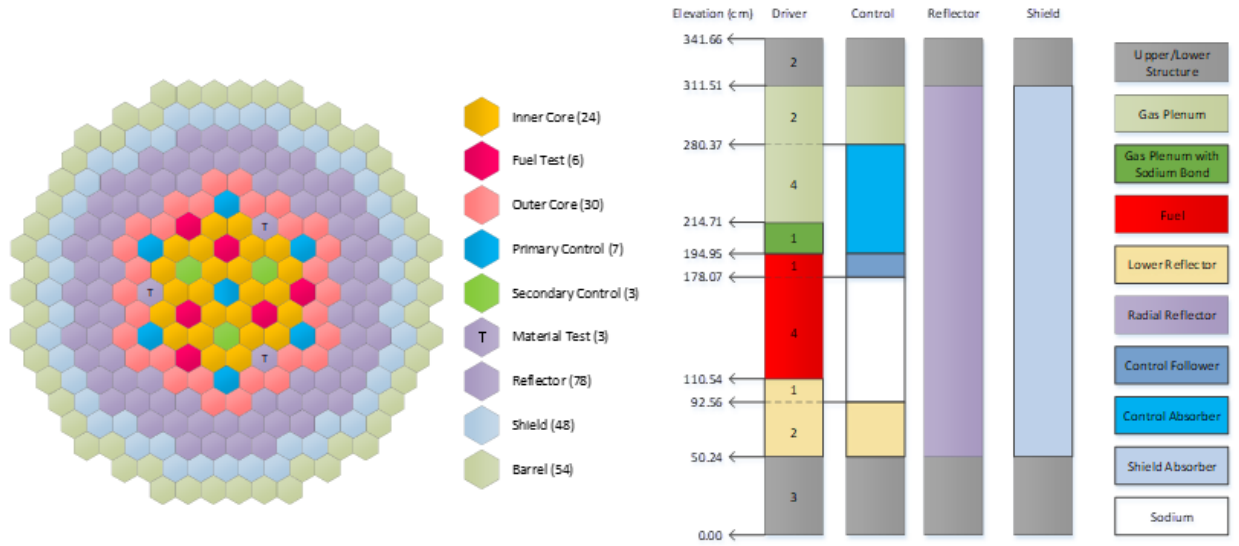
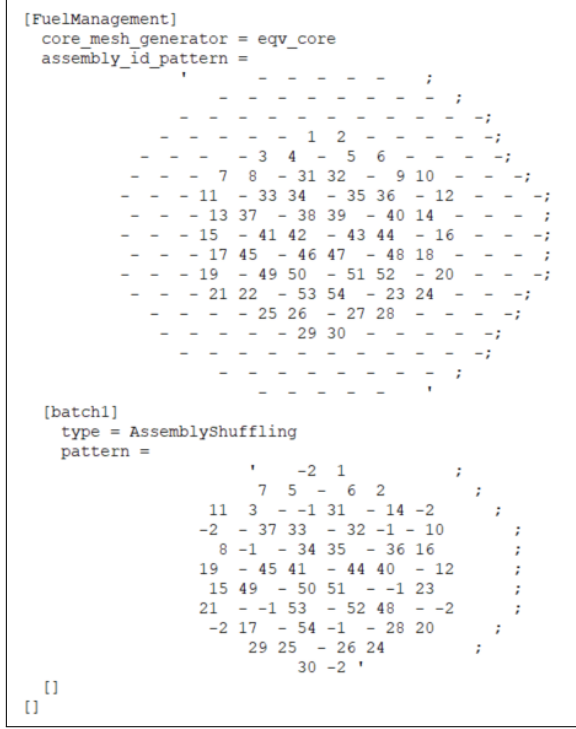


Figure 4: ABTR core overview

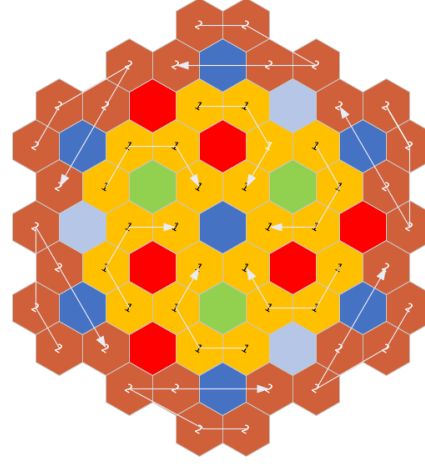
generated using the cross section generation workflow with MC²-3 in Griffin and the ENDF/B-VII.0 library. The Reactor Geometry Mesh Builder (RGMB) was used for mesh generation, producing an equivalent assembly-homogeneous geometry from a fully-heterogeneous geometry and material specification. For testing purposes, a diffusion solver with 9-group cross sections was used instead of a transport solver.

In the [FuelManagement] block of the Griffin input, *assembly_id_pattern* must be defined for all assemblies, where IDs are assigned only to fuel assemblies, as shown in Fig. 5 (a). A shuffling scheme is specified in the subblock with the *AssemblyShuffling* type. As discussed in the previous section, a negative ID indicates fresh fuel. These IDs are defined in the *CoreMeshGenerator* section of the [Mesh] block. Fuel assemblies should be assigned only with positive ID values (i.e., ID=0 cannot be used) in order to allow the insertion of fresh fuel assemblies using negative ID values. In the *pattern* parameter, positive IDs refer to the assembly locations of the batch of interest, based on IDs defined in *assembly_id_pattern*. Fuel assemblies with IDs not defined in the *pattern* are considered discharged.

In the example shown in Fig. 5 (b), 4 and 5 batches are performed for the inner and outer cores, respectively (the arrows in the figure indicate the sequence of fuel shuffling across cycles), where 6 fresh fuel assemblies are inserted to each of inner and outer cores and consequently 12 burned fuel assemblies are discharged. Repeating the shuffling scheme using *batch_cycles* in the [Executioner] block results in *k*-effective values over time, as illustrated in Fig. 6, which almost converged after seven cycles.



(a)



(b)

Figure 5: (a) A [FuelManagement] input example of fuel assembly shuffling and (b) illustration of the corresponding shuffling scheme of outer and inner cores

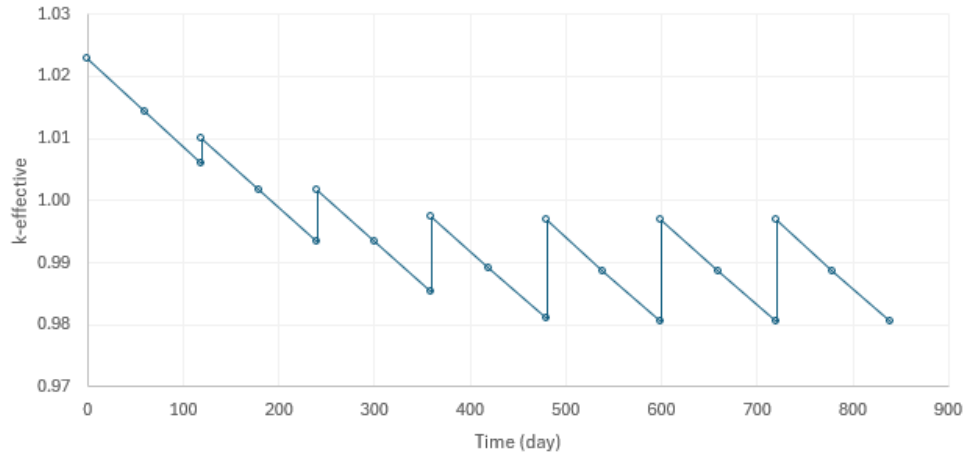


Figure 6: Change of k -effective by outer and inner core fuel assembly shuffling

2.3 Equilibrium Cycle Capability

2.3.1 Equilibrium Cycle Calculations

The equilibrium cycle problem can be categorized into four interrelated intermediate problems: cyclic mode, cyclic mode with a burnup constraint, unconstrained equilibrium cycle, and constrained equilibrium cycle. The cyclic mode equilibrium cycle is constrained only by the two-point boundary condition for a given charge density vector. It focuses solely on the in-reactor cycle and is independent of the external fuel cycle. Equilibrium is achieved by loading a fixed charge density vector into the reactor at the start of each burn cycle, operating the reactor for a fixed cycle length, and applying a consistent fuel management scheme at the end of each cycle. The fuel cycle is considered converged when the discharge density vector remains the same after each burn cycle.

In the cyclic mode equilibrium cycle with a burnup constraint, the discharge burnup is typically limited to reduce the risk of fuel failure. For a given charge density vector, cycle length, and fuel management scheme, the equilibrium solution is defined by the discharge burnup of each material, which is uniquely determined by the charge density vector and cycle length. In fast reactors, the burnup constraint is usually set by the maximum permissible discharge burnup in specific reactor regions. Under these conditions, burnup increases monotonically over time for a constant enrichment. As the cycle length increases, the discharge burnup from the core also increases due to the longer fuel residence time.

The unconstrained equilibrium mode is determined for a given enrichment vector by calculating the operating time and the charge density vector. When the external cycle is involved, the charge density vector is determined by both the discharged fuel composition and the external feed composition. Starting with an assumed charge density vector, the discharge density vector can be calculated by solving the cyclic mode equilibrium problem with a burnup constraint. A new charge density vector is then obtained through the reprocessing and re-fabrication processes. This updated charge density vector is used to solve the cyclic mode equilibrium problem again, and the charge density vector is further refined. This iterative process continues, with the charge density vector being updated after each step, until two successive charge density vectors converge within a specified tolerance.

In the constrained equilibrium cycle, the charge fuel composition is adjusted to meet an additional design constraint, such as maintaining criticality over a specified cycle length and power setting. For instance, the charge fuel must contain enough fissile material to sustain the reactor's criticality. The charged fuel

composition is modified to achieve an un-poisoned eigenvalue at a particular point in time, ensuring the reactor remains critical. This additional constraint introduces an equation that typically determines one of the design parameters, often related to the charged fuel composition.

In this FY, we focused on implementing the unconstrained equilibrium cycle capability along with the enrichment search functionality. The constrained equilibrium cycle mode will be implemented in the next FY. For a given initial reactor configuration, the end-of-cycle (EOC) state can be determined by solving the transmutation equation. The following algorithm is used for a cyclic mode equilibrium cycle calculation:

1. Assume all stage densities at the BOC are equal to the charge density (fresh fuel) of each material type,
2. Calculate the region-averaged nuclide densities at the BOC for all regions in the core,
3. Solve a depletion equation,
4. Check for convergence of stage densities. If not converged, return to step 2 with the updated stage densities and continue performing steps 3 and 4 until convergence is achieved.

2.3.2 Implementation of Equilibrium Core Depletion Capability

A typical fast reactor operates under a fixed fuel management scheme. After many cycles of operations, the core status reaches an equilibrium condition. Thus, the capability to achieve an equilibrium core condition is essential for fast reactor analysis. The purpose of the equilibrium core calculation is to quantify the overall performance and characteristics of the core over entire cycles. It is not intended to obtain detailed core power distributions and compositions for specific cycles and burnup levels.

This capability is designed to efficiently yield an equilibrium core condition operated under a scattered reloading scheme. In this scheme, assemblies of the same type but at different burnup stages are distributed across each core batch region. Given the characteristics of fast neutrons, we assume that core configurations, where the average compositions of neighboring assemblies at different burnup stages are assigned to individual assembly locations, can represent the overall core status, as illustrated in Fig. 7.

Based on this observation, fuel assemblies are implicitly reloaded using a defined batch reloading scheme in the equilibrium core depletion calculation. This approach allows us to perform efficient and

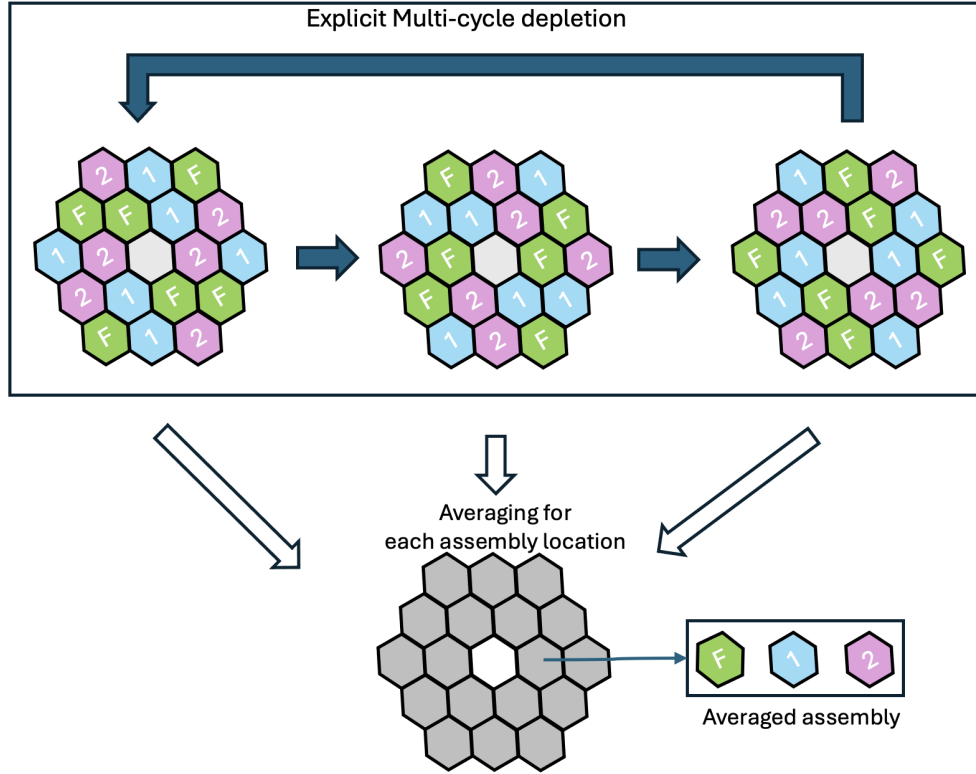


Figure 7: Illustration of averaged core concept utilized in equilibrium core depletion

practical calculations and simplify the modeling process. Here are the key steps in the equilibrium core depletion process:

- Multiple composition data (stage data) representing individual batch stages are defined for each fuel assembly (depletion zone). These stage data represent the isotopic composition of the fuel at different points in the batch cycle.
- During the depletion process, the stage data of each assembly are depleted over the cycle, using fluxes and transmutation matrices computed based on averaged number densities of the stage data.
- At the EOC, fresh fuel is reloaded for individual assemblies by replacing the stage data that have reached the batch cycle length.
- This process is repeated for multiple cycles (iterations) until the core reaches an equilibrium status.

The equilibrium core depletion process for a 3-batch cycle scheme is illustrated in Fig. 8. This approximate approach effectively yields an equilibrium core configuration in fast reactor applications. An assembly located in a region operating under the 3-batch cycle scheme has three sets of homogenized compositions

that represent the fresh, once-burned, and twice-burned stages. The averaged composition of these three sets is used in the transport calculation to determine the eigenvalue and flux distribution at a certain burnup time step. The resulting flux distribution is then utilized to deplete the three sets of compositions during the cycle operation. At the BOC of the next cycle, fresh fuel is loaded, and third-burned fuel is discharged from each assembly, as shown on the left side of Fig. 8. It should be noted that the shuffling of fuel assemblies is not supported in the equilibrium core calculation.

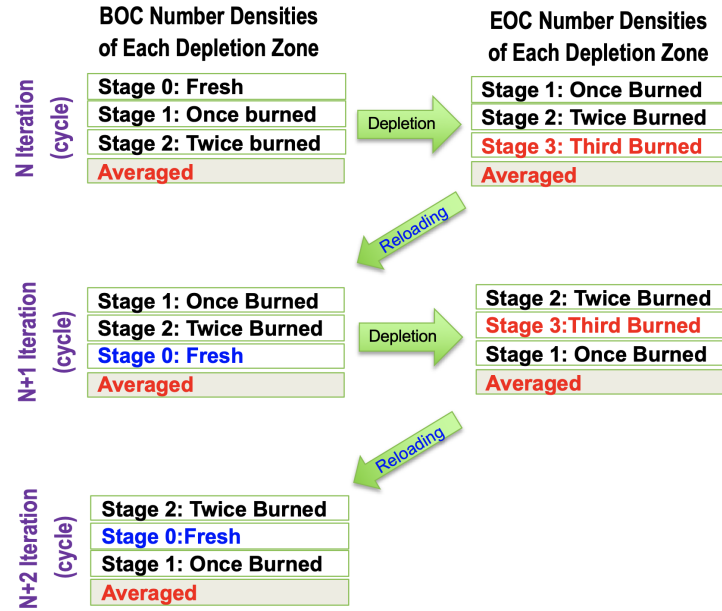


Figure 8: Illustration of equilibrium core depletion scheme using a 3 batch cycle scheme

We successfully implemented the equilibrium core depletion capability described above by extending the existing depletion functionality of Griffin. First, the depletion user object of Griffin was updated to handle the depletion of multiple stage compositions. Then, we refactored the depletion executioner system to properly support multiple depletion modes: single cycle depletion, multi-cycle depletion, and the newly added equilibrium core depletion. Additionally, a dedicated input block for defining the batch reloading scheme for equilibrium core depletion was incorporated into the fuel management system input of Griffin.

2.3.3 Verification Tests

For verifying the newly implemented capability, a test calculation was performed for a small SFR core problem composed of 2 fuel regions with 3- and 4-batch reloading schemes. As illustrated in Fig. 10, the core of the test problem consists of homogenized hexagonal assemblies arranged in 4 rings of hexagonal lat-

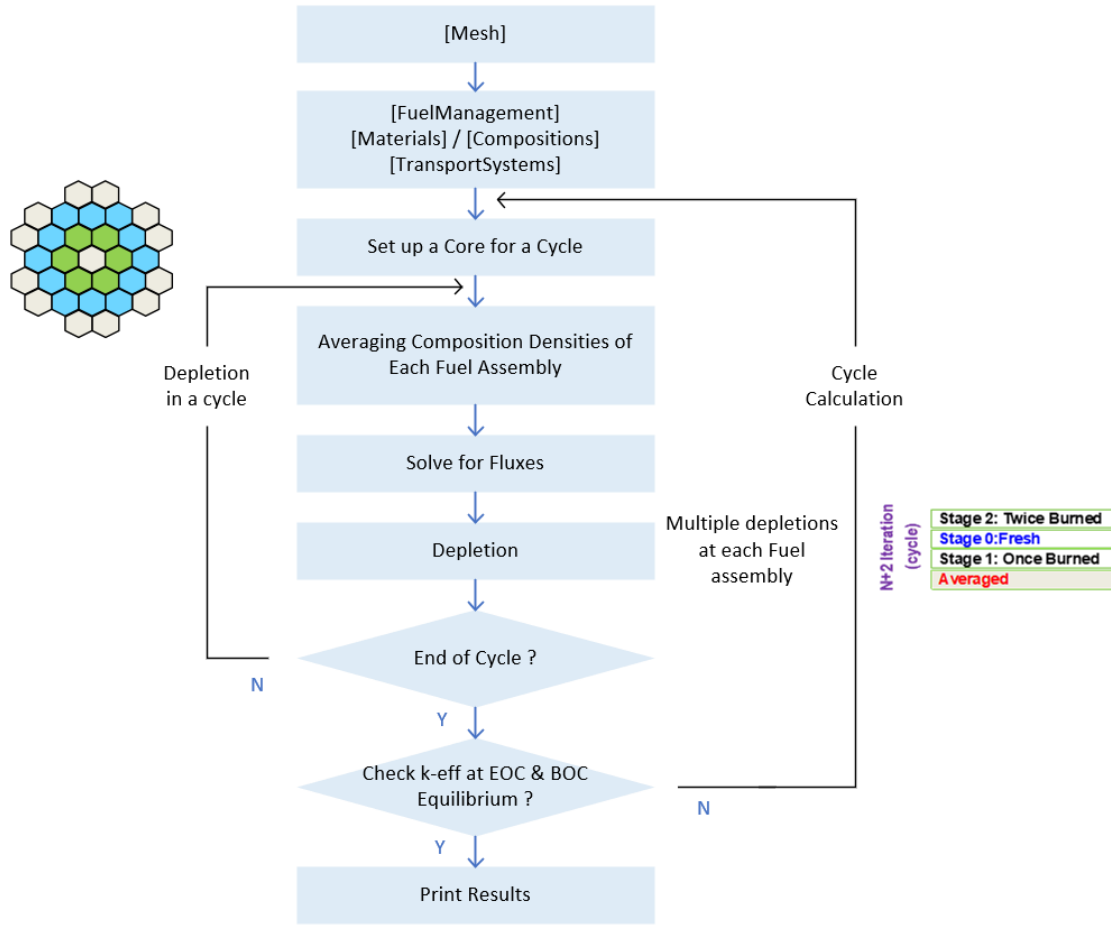


Figure 9: Equilibrium cycle calculation workflow

tice. The second and third rings are filled with fuel assemblies operated with 3-batch and 4-batch reloading schemes, respectively. A fuel management scheme for equilibrium core depletion can be specified using the *EquilibriumCoreBatchReloading* object in Griffin, as shown in Fig. 10. The *batch_id_pattern* parameter defines the batch reloading zones by assigning integer ID tags to fuel assembly locations, while the corresponding batch cycle lengths are provided in the *batch_reloading_cycles* parameter. The equilibrium core calculation for this test problem was performed using the cross section data set generated using the cross-section generation workflow. Depletion iterations to achieve equilibrium core conditions were conducted until the eigenvalues at the beginning and end of successive cycles converged. The convergence of eigenvalues over the depletion iterations is shown in Fig. 11, with the calculation completed with 8 depletion iterations. Prior to an equilibrium cycle calculation, a self-check can be performed by setting *batch_reloading_cycles* = '1 1' in the [FuelManagement] block to ensure that the depletion solutions of one

cycle are reproduced in the next, as the core loading is repeated with the same fresh fuel.

```
[FuelManagement]
[eq_core]
type = EquilibriumCoreBatchReloading
pattern =
    , -2 -2 -2;
    -2 -1 -1 -2;
    -2 -1 - -1 -2;
    -2 -1 -1 -2;
    -2 -2 -2'

batch_id_patten =
    , 2 2 2;
    2 1 1 2;
    2 1 - 1 2;
    2 1 1 2;
    2 2 2'
batch_reloading_cycles = '3 4'
[]
[]
```

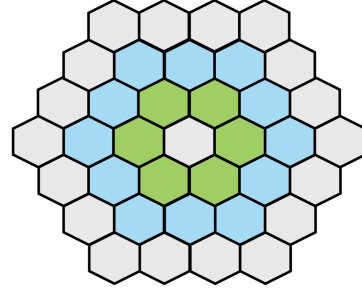


Figure 10: Fuel management scheme of equilibrium core depletion test problem (green region: 3-batch reloading with fuel type 1, blue region: 4-batch reloading with fuel type 2)

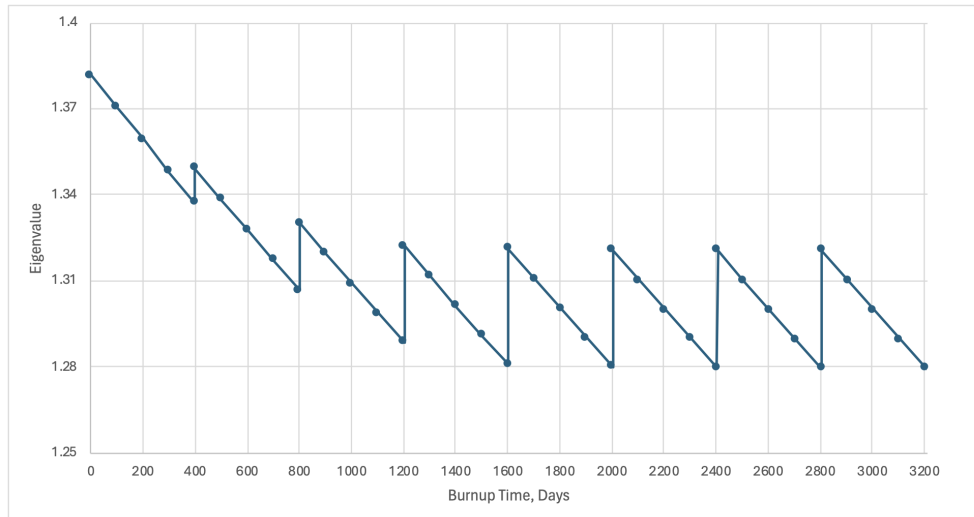


Figure 11: Eigenvalue result of equilibrium core depletion test problem

To confirm that the equilibrium core calculation properly yields the desired equilibrium condition, a reference calculation was additionally carried out by explicitly reloading fuel assemblies in the multi-cycle depletion scheme. This reference calculation uses the explicit batch reloading scheme shown in Fig. 12,

which requires the definition of 12 distinct loading patterns. To sufficiently converge the core to the equilibrium status, a total of 24 cycles of depletion were performed. For this problem, it was found that the eigenvalue result from the last 6 cycles of the reference calculation were less than 40 pcm of the converged eigenvalue from the equilibrium core calculation, as summarized in Fig. 13. This close agreement between the results of explicit cycle-by-cycle depletion and equilibrium core depletion suggests that the equilibrium core depletion capability was correctly implemented in Griffin.

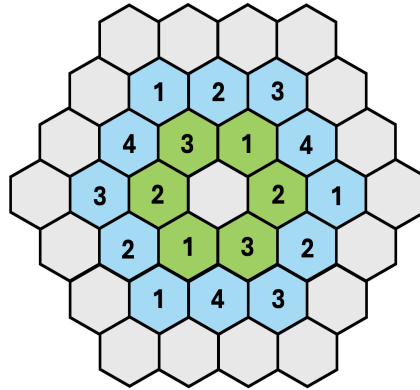


Figure 12: Explicit modeling of batch reloading scheme for equilibrium core depletion test problem (green region: 3-batch with fuel type 1, blue region: 4-batch with fuel type 2, Number: batch reloading order)

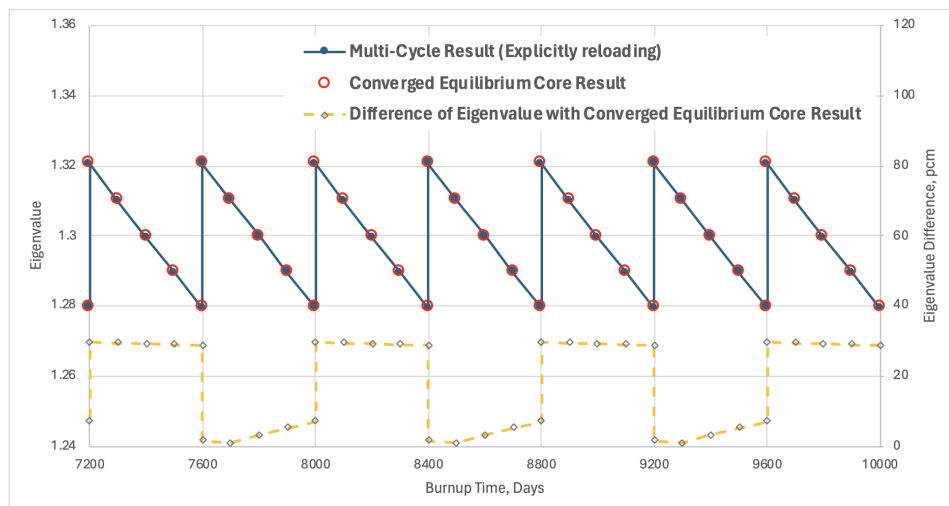


Figure 13: Eigenvalue comparison between equilibrium core depletion and explicit core depletion for the 3-ring test problem

2.4 Search Capability

With the equilibrium core calculation capability described in Section 2.3, the next step is to extend this work to support equilibrium search calculations. The goal of the equilibrium search is to find enrichments that allow the core to achieve a target k -effective at EOC in the equilibrium cycle. It is important to note that fuel enrichment is defined as the fraction of U-235 and fissile Pu isotopes in heavy metal. Instead of a single equilibrium core calculation with fixed initial core conditions, the user provides three additional input parameters in the [Executioner] block to trigger an equilibrium search calculation: *enrich_low*, *enrich_high*, and *target_keff*. The parameters *enrich_low* and *enrich_high* define the bounds of the equilibrium search routine, representing the lower and upper bounds of the enrichment multiplier applied to the fresh fuel compositions. In this case, the enrichment multiplier is expressed as a multiplicative factor of the original fuel enrichment, for example, a multiplier of 0.5 multiplies the initial number densities of U-235 and fissile Pu isotopes in the fresh fuels specified in all depletion zones by 0.5. The equilibrium search algorithm begins by incorporating an additional outer iteration to the existing equilibrium core calculation iterations, where two previous equilibrium calculations are conducted by applying the enrichment multiplier to fresh fuel compositions.

Linear interpolation is then applied to make successive guesses in adjusting the enrichment multiplier until the equilibrium core eigenvalue at the EOC matches the target k -effective within a user-specified tolerance. The enrichment multiplier for the next search iteration is updated using the equation:

$$m_{i+1} = m_i + (k_{\text{target}} - k_i) \frac{m_i - m_{i-1}}{k_i - k_{i-1}}, \quad (1)$$

where m_{i+1} , m_i , and m_{i-1} are the enrichment multipliers for the next, current, and previous search iterations, respectively; k_{target} is the target equilibrium core k -effective for the search calculation; and k_i and k_{i-1} are the equilibrium core k -effectives at the EOC from the current and previous search iterations, respectively.

In the first search iteration, the enrichment multiplier is set to 1, meaning that an equilibrium core calculation is conducted using the original fresh fuel composition (i.e., no number density adjustment of fissile isotopes). In the second search iteration, if the first search iteration yields an equilibrium k -effective greater than the target k -effective, the enrichment multiplier is set to *enrich_low*. Conversely, if the equilibrium k -effective is smaller than the target k -effective, the enrichment multiplier is set to *enrich_high* in the second

search iteration. From the third search iteration onward, Eq. (1) is used to update the enrichment multiplier for the current search iteration and compute the next equilibrium core k -effective at the EOC.

2.5 Demonstration Using the ABTR Core

The equilibrium cycle and enrichment search calculations have been tested using the 3D ABTR problem described in Fig. 4, where the inner and outer cores are composed of 15.1 wt.% and 18.9 wt.% mixed oxide fuel (MOX) fuel, respectively. The equilibrium cycle calculation is initiated by setting the input parameter *eq_core_depletion_batch* to *eq_core* in the [Executioner] block. The results of the equilibrium cycle vary depending on the batch inputs. In Fig. 14, the core utilized 4 and 5 batches for inner and outer cores, respectively, with a cycle length of 120 days per cycle, achieving equilibrium after 7 cycles. As discussed in Section 2.3.2, each fuel assembly includes a mixture of all stage compositions. The convergence behavior can be verified by explicit cycle depletion calculations using the manual shuffling scheme (Fig. 5 and Fig. 6), as demonstrated in Fig. 15, indicating that the cycle depletion results are closely aligned.

For the enrichment search, two different batch schemes were tested: (a) 4 and 5 batches and (b) 12 and 15 batches for inner and outer cores, respectively. In both cases, the search condition was set for the EOC equilibrium k -effective to equal unity. The progression of the BOC and EOC k -effectives after each search iteration is shown in Fig. 16. In the first search iteration (with no enrichment multiplier adjustment), the 12-15 batch core reaches equilibrium after 21 cycles, while the 4-5 batch core achieves equilibrium after 7 cycles. As the equilibrium k -effective is lower than unity after the first iteration, a higher enrichment multiplier guess is applied in the next step, overshooting the target k -effective. From the third iteration onward, the next multiplier is estimated based on interpolation of the previous two steps. Finally, the enrichment multiplier converges after 4 search iterations for the 4-5 batch core and 6 search iterations for the 12-15 batch core, resulting in enrichment multipliers of 1.036 and 1.169, respectively. The final enrichments of fresh fuel in the inner and outer cores are changed almost by the amount of the enrichment multiplier, maintaining the initial enrichment gap between the inner and outer cores.

Using the calculated enrichment multipliers from the search, the densities of fissile isotopes in the fresh fuel were manually adjusted, and an equilibrium core calculation was then conducted to verify that the target k -effective could be reproduced. This is illustrated in Fig. 17 and serves as additional verification that the equilibrium search capability produces the accurate result. In the future, a burnup constraint will be accounted for as an input parameter, allowing the maximum permissible discharge burnup to be accounted

for in equilibrium cycle calculations by adjusting the cycle length.

```
[FuelManagement]
assembly_id_pattern =
    '
        - - - - -;
        - - - - - - -;
        - - - - - - - -;
        - - - - - 1 2 - - - -;
        - - - - 3 4 - 5 6 - - - -;
        - - - 7 8 - 9 10 - 11 12 - - -;
        - - - 13 - 14 15 - 16 17 - 18 - - -;
        - - - 19 20 - 21 22 - 23 24 - - -;
        - - - 25 - 26 27 - 28 29 - 30 - - -;
        - - - 31 32 - 33 34 - 35 36 - - -;
        - - - 37 - 38 39 - 40 41 - 42 - - -;
        - - - 43 44 - 45 46 - 47 48 - - -;
        - - - - 49 50 - 51 52 - - - -;
        - - - - - 53 54 - - - - -;
        - - - - - - - - -;
        - - - - - - - -;
    '

core_mesh_generator = eqv_core

[eq_core]
type = EquilibriumCoreBatchReloading
pattern = '
        -2 -2;
        -2 -2 - -2 -2;
        -2 -2 - -1 -1 - -2 -2;
        -2 - -1 -1 - -1 -1 - -2;
        -2 -1 - -1 -1 - -1 -2;
        -2 - -1 -1 - -1 -1 - -2;
        -2 -1 - -1 -1 - -1 -2;
        -2 - -1 -1 - -1 -1 - -2;
        -2 -2 - -1 -1 - -2 -2;
        -2 -2 - -2 -2;
        -2 -2'

batch_id_pattern = ' 2 2;
        2 2 - 2 2;
        2 2 - 1 1 - 2 2;
        2 - 1 1 - 1 1 - 2;
        2 1 - 1 1 - 1 2;
        2 - 1 1 - 1 1 - 2;
        2 1 - 1 1 - 1 2;
        2 - 1 1 - 1 1 - 2;
        2 2 - 1 1 - 2 2;
        2 2 - 2 2;
        2 2'

batch_reloading_cycles = '4 5'
[]
[]
```

Figure 14: A [FuelManagement] input example for equilibrium cycle calculation

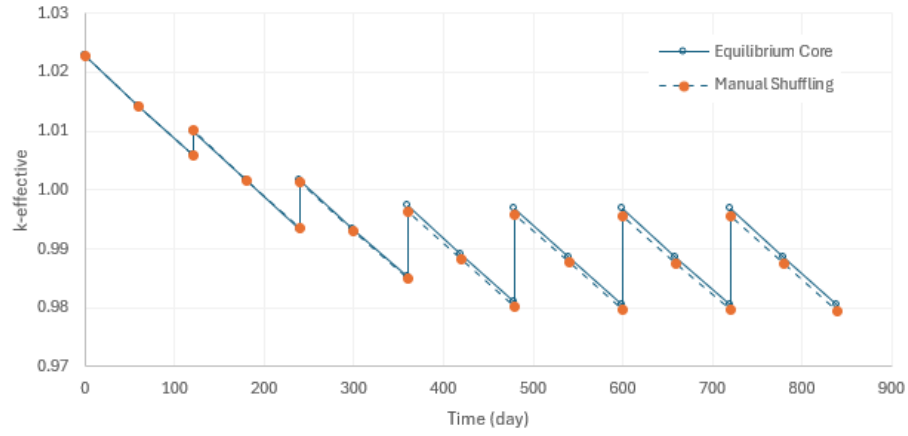
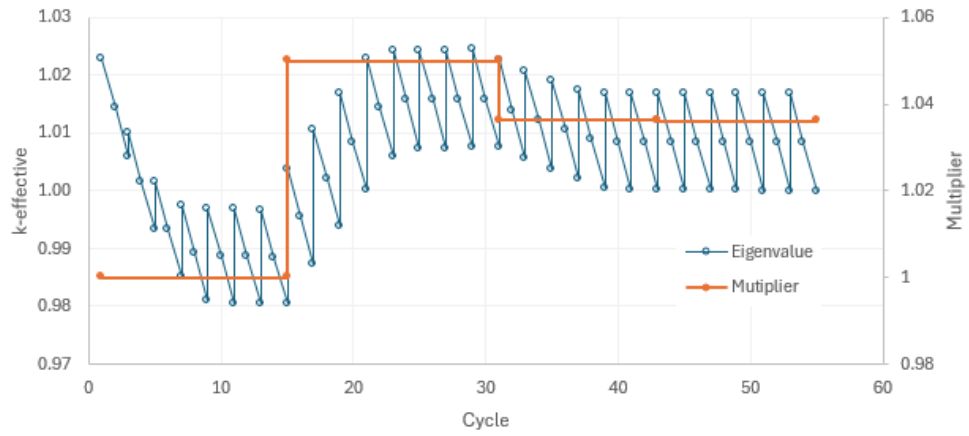
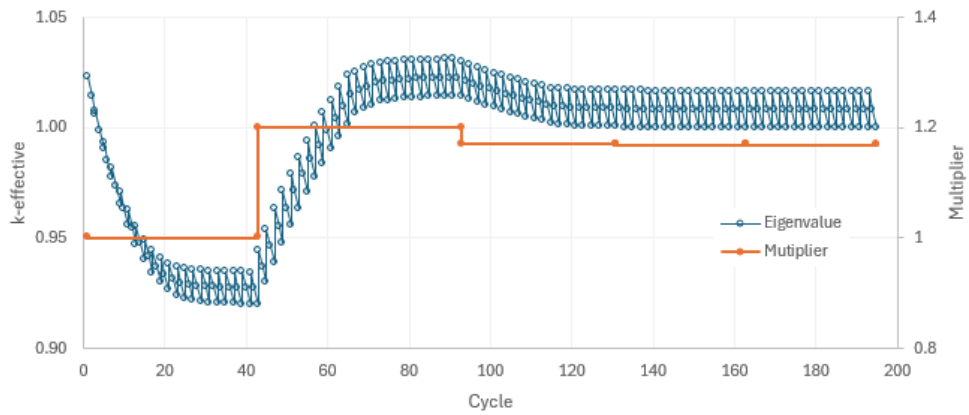


Figure 15: Comparison of equilibrium cycle and manual shuffling calculations

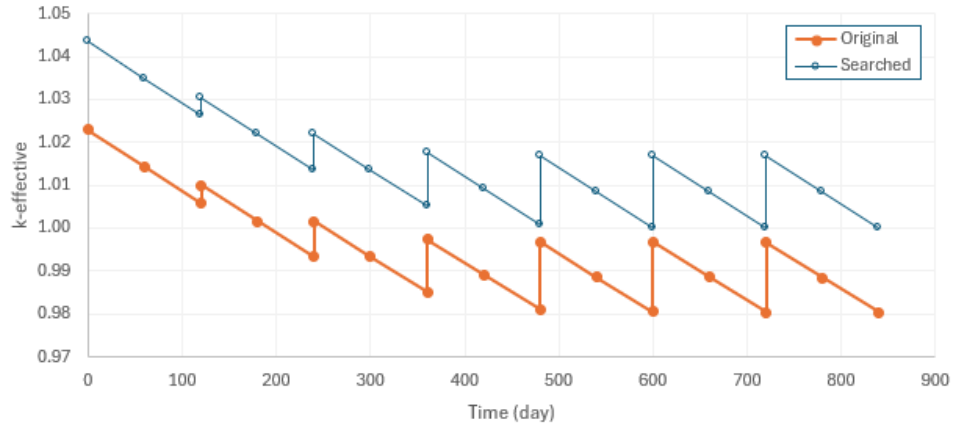


(a) 4 and 5 batches

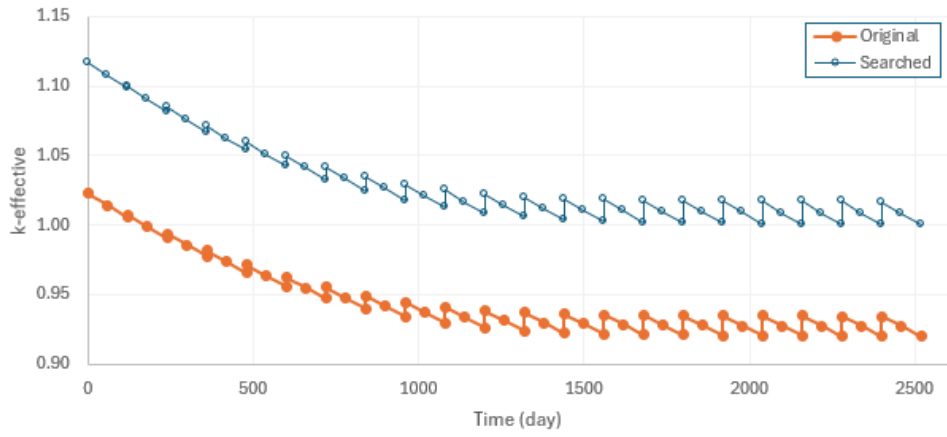


(b) 12 and 15 batches

Figure 16: Enrichment search results for the 3D ABTR problem



(a) 4 and 5 batches



(b) 12 and 15 batches

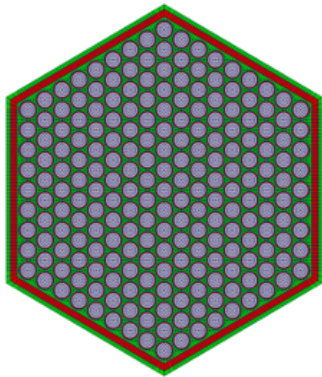
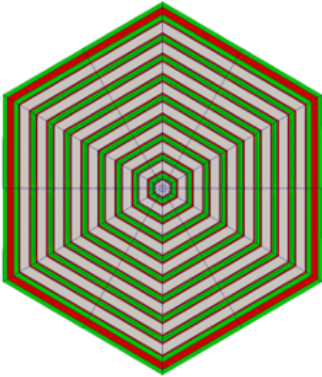
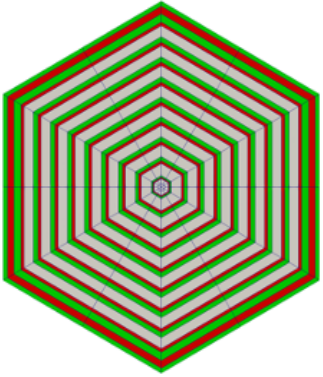
Figure 17: Comparison of equilibrium conditions before and after enrichment search for the 3D ABTR problem

3. Updates in Fast Reactor Cross Section Generation Workflow

3.1 Verification Tests

The ring-heterogeneous approximation was rigorously verified using the ABTR benchmark in a separate study [7]. Both the ordinary ring-heterogeneous (RH) approach (hereafter simply referred to as RH) and the simple ring-heterogeneous (SRH) approach, described in Table 1, were verified against the fully-heterogeneous (FH) mesh and Serpent calculations for various problems, ranging from a two-dimensional (2D) single assembly to a 3D full core. The RH approach preserves the original material arrangement (e.g., fuel surrounded by cladding and coolant) during the approximation, while the SRH approach makes a more aggressive approximation by representing each ring of pins with a single band per material (e.g., only one band for each of cladding and coolant per ring of fuel pins) to gain additional performance with the assumption that fast neutrons are not sensitive to such geometric details due to their long mean-free paths.

Table 1: Comparison of FH, RH, and SRH meshes on a 2D ABTR fuel assembly.

Mesh	FH	RH	SRH
Geometry			
# Elements	22,056	546	354

The detailed verification results can be found in the reference report [7], and important results and takeaways are summarized as follows:

- For single assembly problems, both the RH and SRH approaches showed at most a 0.1% error in the ring-wise fission reaction rate compared to the FH reference. Since fast neutrons have long mean-free paths and fast fission is dominant in this system, the fission reaction rate is not significantly affected by detailed geometry, validating the assumption of SRH. For absorption reaction rates, however, while RH maintains a comparable error level with the fission reaction rate, SRH shows higher errors of up

to 0.6%. In contrast to fission, resonance absorption is not negligible and is more sensitive to detailed geometry that alters the spatial self-shielding effect. The ring-wise fission and absorption reaction rate error distributions of RH and SRH in the 2D ABTR middle fuel assembly are presented in Figure 18. As the result, RH yields eigenvalue errors of only a few pcm, while those of SRH rise up to 60 pcm.

- For a 2D core problem with leakage, the difference between RH and SRH becomes more apparent, as illustrated in Figure 19. RH presents less than 0.1% error in both ring-wise fission and absorption reaction rates, and the root mean square (RMS) errors are only 0.02% and 0.01%, respectively, which are negligible. However, SRH shows a maximum fission reaction rate error of 0.34% and a RMS error of 0.16%. An in-out tilt in the error distribution is observed, with positive errors at the periphery, which is caused by smaller leakage at the boundary as the fuels are moved towards the assembly center by the SRH approximation. For the absorption reaction rate, the maximum error reaches 1%, with an RMS error of 0.3%, and the error distribution is more localized to each assembly, driven by the spatial self-shielding. As the result, the eigenvalue error of SRH is significantly larger than that of RH, which are 117 pcm and 19 pcm, respectively.

Consequently, the RH approach is considered more rigorous and desirable, and was selected over the SRH approach as the base ring-heterogeneous mesh generation option in *EquivalentCoreMeshGenerator*. While the ring-heterogeneous mesh option in *EquivalentCoreMeshGenerator* was implemented in FY23, this verification work conducted in FY24 confirms that the implemented approach is valid.

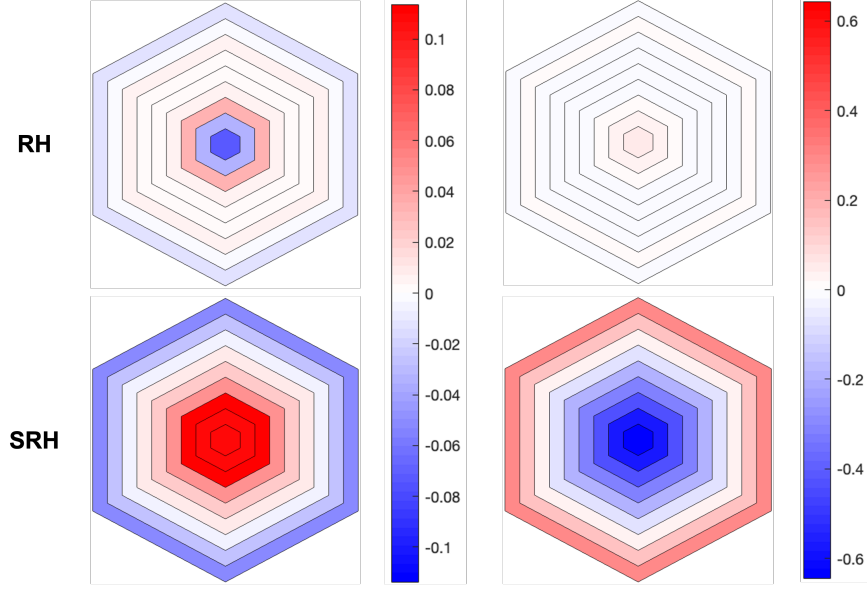


Figure 18: Ring-wise fission (left) and absorption (right) reaction rate errors (%) of RH and SRH compared to FH for the 2D ABTR middle fuel assembly.

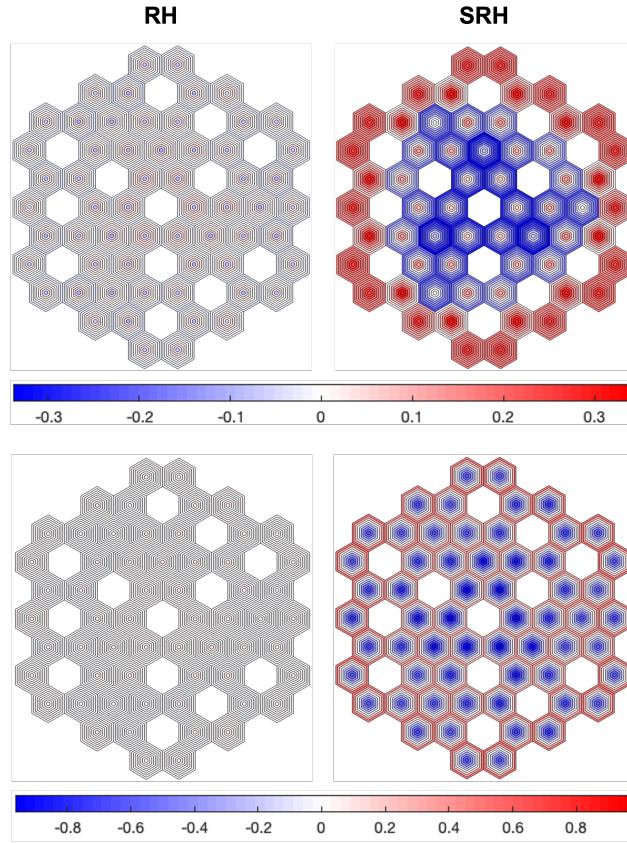


Figure 19: Ring-wise fission (top) and absorption (bottom) reaction rate errors (%) of RH and SRH compared to FH for the 2D ABTR core.

During this verification work, the cross section generation workflow for fast reactors was verified to produce microscopic cross sections and Griffin core inputs for an RH configuration, once a user provides the [MCC3CrossSection] input block along with an RGMB-based mesh input, as shown in Fig. 20. Essentially, most inputs remain the same as for an assembly-homogeneous core input, except for the *taget_core = ring_het* input card. This input triggers multiple MC²-3 jobs with homogeneous or one-dimensional (1D) configurations, as well as a simplified R-Z core calculation with ultrafine-group cross sections. Ultimately, it produces broad-group cross sections and a Griffin input for an RH core configuration, allowing the user to run an RH core problem with minor updates to the Griffin core input.

3.2 Delayed Neutron Data

The MC²-3 code [8] integrated into Griffin via the cross section generation workflow, has the capability to generate delayed neutron parameters in the DLAYXS data format [9]. To support kinetics calculations using MC²-3 generated cross section data, the delayed neutron parameters (delayed neutron decay constants, spectra, and precursor yields) in the DLAYXS format should be converted into the XML data format. Therefore, a functionality was added in the ISOXML module of Griffin to process and convert the DLAYXS format in Griffin, providing two conversion options:

- Convert a DLAYXS file together with the corresponding ISOTXS file.
- Append a DLAYXS file into an existing XML-format cross-section file.

These options allow a user to either add delayed neutron data from DALYXS into an existing cross-section file in XML format or convert ISOTXS and DLAYXS files into an XML-format cross-section file. It was confirmed that the converted delay parameters can be correctly utilized in Griffin kinetics calculations.

```

[MCC3CrossSection]
    remove_pwfiles = true
    remove_inputs  = true
    remove_outputs = true
    remove_xmlfiles = true
    griffin_data    = 'griffin/griffin_data'
    endfb_version   = 'ENDF/B-VII.0'
    library_pointwise = 'PW_DATA'

    xml_macro_cross_section = true
    xml_filename             = 'mcc3xs.xml'
    generate_core_input      = true

    rz_calculation          = true
    rz_spectrum_pn_order = 3
    group_structure         = 'ANL33'
    rz_group_structure      = 'ANL1041'
    map_het_grid_name = 'fuel1 Tfuel
                        fuel2 Tfuel
                        fuel3 Tfuel
                        control_b4c Tcool
                        ht9 Tcool
                        sodium Tcool
                        radial_reflector Tcool
                        radial_shielding Tcool
                        lower_reflector Tcool
                        upper_na_plenum Tcool
                        upper_gas_plenum Tcool
                        control_empty Tcool'
    map_het_grid_ref_value = 'Tfuel 900
                            Tcool 700'
    map_het_grid_values = 'Tfuel 900
                          Tcool 700'
    use_micro_cross_section = false
    rz_meshgenerator        = rz_core
    het_cross_sections       = 'fuel1 fuel2 fuel3 control_b4c'
    max_het_mesh_size       = 0.5
    target_core = ring_het
[]

```

Figure 20: An input example for fast reactor cross section generation workflow using MC²-3 for RH configuration.

4. Conclusions and Future Work

The preliminary fuel cycle capability, including shuffling and cycle calculations, implemented in Griffin last year, has been significantly extended this fiscal year to provide users with a comprehensive fuel cycle analysis tool for fast reactors. First, the assembly shuffling option was improved to allow flexible fuel reloading in multi-cycle depletion calculations and to incorporate decay between cycles. An equilibrium core calculation capability was implemented to determine an equilibrium core when users provide equilibrium cycle-related input parameters in the [FuelManagement] block. In the equilibrium calculation, fuels of the same type are treated in the same manner, including all stage fuels, which are depleted with the same region flux during a cycle. In fast reactors, the approach was verified by comparing it with explicit reloading and shuffling fuel assemblies, demonstrating that it provides accurate solutions.

Additionally, an enrichment searching capability was added to determine the enrichment condition that allows a core to reach an equilibrium cycle with the EOC k -effective matching a user-specified target value. The updated fuel management capability has been extensively tested using the 3D ABTR problem with different batch schemes, exhibiting reasonable solutions in terms of manual shuffling, equilibrium cycle, and enrichment search calculations.

The cross-section generation workflow capability for fast reactors was further verified to produce microscopic cross sections as well as Griffin core inputs for an RH configuration. Rigorous verification tests using the ABTR problem demonstrated that the RH approximation produced comparable solution with the FH mesh for fast reactor problems. In addition, an option was added to convert delay neutron parameter data generated from MC²-3 in the DLAYXS format into XML format or to add them to an existing XML-format cross section file, to support transient calculations using MC²-3-generated data.

In the future, the following tasks are required to complete the fuel management capability of Griffin: 1) Extend the equilibrium cycle capability to include external cycles, 2) Update the equilibrium cycle capability with constraints such as discharge burnup, 3) Enhance the fuel management scheme to include assembly rotations during shuffling, fuel management support for heterogeneous cores, and burned assembly insertion from ex-core storage, and 4) Compare solutions between Griffin and REBUS [5] for various equilibrium search benchmark problems [6].

REFERENCES

- [1] C. H. Lee and J. Ortensi, et al., “Griffin Software Development Plan,” Research Report INL/EXT-21-63185, ANL/NSE-21/23, Idaho National Laboratory, Argonne National Laboratory, June 2021.
- [2] A. D. Lindsay et al., “2.0 - MOOSE: Enabling massively parallel multiphysics simulation,” *SoftwareX*, vol. 20, p. 101202, 2022.
- [3] C. H. Lee, Y. S. Jung, S. Kumar, N. Choi, J. Hanophy, and Y. Wang, “Improved Fast Reactor Capability of Griffin in FY23,” Tech. Rep. ANL/NSE-23/73, INL/RPT-23-74897, Argonne National Laboratory and Idaho National Laboratory, 2023.
- [4] W. S. Yang and M. A. Smith, “Theory manual for the fuel cycle analysis code REBUS,” Tech. Rep. ANL/NE-19/21, Argonne National Laboratory, 2020.
- [5] M. A. Smith and A. G. Nelson, “Verification of the REBUS Software,” Tech. Rep. ANL-VTR-50, Argonne National Laboratory, 2021.
- [6] P. Deng and W. S. Yang, “ABTR fuel cycle analysis benchmark problems,” Tech. Rep. UM/NRDSL-23/01, University of Michigan, 2023.
- [7] S. Terlizzi et al., “Sodium-Cooled Fast Reactor Reference Plant Model,” Tech. Rep. INL/RPT-24-76937, Idaho National Laboratory, 2024.
- [8] C. H. Lee and W. S. Yang, “MC²-3: Multigroup cross section generation code for fast reactor analysis,” *Nucl Sci Eng*, vol. 187, pp. 268–290, 2017.
- [9] R. D. O’Dell, “Standard interface files and procedures for reactor physics codes, version IV,” Tech. Rep. LA-6941-MS, Los Alamos Scientific Laboratory, 1977.