ANL/NSE-23/73
INL/RPT-23-74897

Improved Fast Reactor Capability of Griffin in FY23

September 2023

Changho Lee[1], Yeon Sang Jung[1], Shikhar Kumar[1], Namjae Choi[2], Joshua Hanophy[2], Yaqi Wang[2]

[1]*Nuclear Science and Engineering Division, Argonne National Laboratory*
[2]*Nuclear Science and Technology Division, Idaho National Laboratory*

Idaho National Laboratory

Argonne NATIONAL LABORATORY

ANL/NSE-23/73
INL/RPT-23-74897

Improved Fast Reactor Capability of Griffin in FY23

Changho Lee[1], Yeon Sang Jung[1], Shikhar Kumar[1], Namjae Choi[2], Joshua Hanophy[2], Yaqi Wang[2]

[1]**Nuclear Science and Engineering Division, Argonne National Laboratory**
[2]**Nuclear Science and Technology Division, Idaho National Laboratory**

September 2023

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 DOE-NE NEAMS program. In FY23, we enhanced capabilities required for fast reactor analysis. This effort included primarily updating the cross-section generation workflow using MC^2-3 for various reactor configurations, such as homogeneous, duct-heterogeneous, ring-heterogeneous, and fully-heterogeneous geometries. In addition, we initiated the implementation of a multi-cycle depletion and shuffling capability. To support fast reactor simulation capabilities, we significantly improved the performance of the DFEM-SN-based R-Z transport solver to efficiently solve ultrafine group (over 1000 groups) transport problems. Additionally, the performance of HFEM-PN was improved by introducing red-black iteration, the CMFD acceleration technique, and various optimizations. We also completed the pin power reconstruction capability to support multiphysics simulations while identifying and addressing issues associated with SPH equivalence parameter approach. These enhanced capabilities for fast reactor core simulations, specially HFEM-PN and pin power reconstruction features, were applied to benchmark problems involving ABTR and ABR-1000. These applications showcased excellent agreement with Monte Carlo and other code solutions in terms of eigenvalue, control rod worth, and assembly and pin powers.

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

FIGURES

TABLES

ACRONYMS

1D	one dimensional
2D	two dimensional
3D	three dimensional
ABR	Advanced Burner Reactor
ABTR	Advanced Burner Test Reactor
ANL	Argonne National Laboratory
BG	broad group
CMFD	coarse-mesh finite difference
CPM	collision probability method
CR	control rod
DFEM	discontinuous finite element method
DoF	degrees of freedom
HFEM	hybrid finite element method
HFG	hyperfine group
INL	Idaho National Laboratory
MOC	method of characteristics
MOOSE	Multiphysics Object-Oriented Simulation Environment
NEAMS	Nuclear Energy Advanced Modeling and Simulation
NR	narrow resonance
OOM	out of memory
PN	spherical harmonics expansion
RB	red-black
RGMB	Reactor Geometry Mesh Builder
RMS	root mean square
SFR	sodium-cooled fast reactor
SN	discrete ordinates
UAM	uncertainty analysis in modeling

UFG ultrafine group

1. Introduction

Under the DOE Nuclear Energy Advanced Modeling and Simulation (NEAMS) program, Griffin [1] is a reactor multiphysics analysis application jointly developed by Idaho National Laboratory (INL) and Argonne National Laboratory (ANL) using the Multiphysics Object-Oriented Simulation Environment (MOOSE) framework [2]. The Griffin's primary mission is to support the analysis of various advanced reactor designs, outside the scope of traditional commercial light-water reactor designs.

The objective of the FY23 milestone is to enhance Griffin's capabilities for fast reactor modeling, simulation, analysis, and design. In the previous year, we implemented a cross-section generation workflow that produces XML-format cross-section data based on MC^2-3 [3] inputs and temperature conditions provided by the user. This year, our goals are primarily twofold: first, to enhance user-friendliness by generating MC^2-3 inputs from reactor core geometry (mesh) and compositions; second, to introduce a fuel cycle capability that allows users to perform both non-equilibrium and equilibrium cycle calculations, with a primary focus on non-equilibrium cycles this year. This capability can also be applied to other reactor types.

In addition, to support capabilities for fast reactor modeling and simulation, we have made several improvements to HFEM-PN [4], pin power reconstruction, and an R-Z solver with DFEM-SN [5]. First, we optimized the DFEM-SN-based R-Z solver to enhance its performance for ultrafine group (UFG) transport calculations, which are required for cross-section generation with MC^2-3. Second, we improved the performance of HFEM-PN by implementing the red-black (RB) iteration, the coarse-mesh finite difference (CMFD) acceleration technique [5], and additional optimizations. As a result, the HFEM-PN solver now outperforms than DFEM-SN, being able to serve as the primary solver for fast reactor calculations, especially in cases involving cores with homogenized assemblies. Third, after solving for a core with homogenized assemblies, accurate pin power reconstruction within each assembly is essential. Initially implemented in the previous fiscal year, this capability has been further updated to support multiphysics simulation in this fiscal year.

While working on these tasks, we encountered challenges that were more complex than expected. For instance, in the fast reactor cross-section generation workflow, significant efforts were required to update the existing Reactor Geometry Mesh Builder (RGMB) mesh [6] in MOOSE, transfer data between the R-Z transport solver and the MC^2-3 module, and map between composition and geometry based on the meta data information generated from RGMB. Improving the performance of HFEM-PN was not trivial, as its

initial performance was not great as expected from the performance observed with DIF3D-VARIANT [7] (hereafter DIF3D). While the pin power reconstruction worked well for DFEM-SN and HFEM-PN solvers, we noticed an accuracy issue with a diffusion solver with SPH equivalence parameters because the SPH methodology does not guarantee the preservation of assembly surface quantities

The enhanced capabilities for fast reactor core simulations underwent individual testing and were applied to benchmark problems involving sodium-cooled fast reactors (SFRs), specifically Advanced Burner Test Reactor (ABTR) [8] and Advanced Burner Reactor (ABR)-1000 [9]. Verification tests utilized HFEM-PN and pin power reconstruction features, whose results were compared to DIF3D along with the pin power reconstruction tool in the ARC suite [10], and Serpent2 [11] Monte Carlo solutions.

Section 2 discusses the multigroup cross-section generation workflow tailored for fast reactors, while Section 3 focuses on the multi-cycle depletion and shuffling capability. In Section 4, we present enhanced capabilities crucial for the design and analysis of fast reactors. Our test results, showcasing the effectiveness of these improvements, are discussed in Section 5. Finally, Section 6 presents conclusions from our implementations and verification tests, and outlines future work.

2. Fast Reactor Cross Section Generation Workflow

In this section, we describe the cross-section generation workflow for fast reactors. In the previous fiscal year, the workflow began with user-provided MC^2-3 inputs. In the updated process, users need only supply core geometry and composition based on fully heterogeneous geometry. Griffin will then generate cross sections and inputs for cores with four different configurations: fully homogeneous, duct heterogeneous, ring heterogeneous, and fully heterogeneous. Further details will be discussed in the following sections, in terms of updates in mesh generation and cross-section generation workflow.

2.1 Cross-section Generation Process

The MC^2-3 code [3] is a multigroup cross-section generation code for fast reactor applications under the DOE NEAMS program. Griffin includes MC^2-3 as a submodule to generate cross sections for fast reactor systems.

The code generates region-dependent broad group (BG) neutron and gamma cross sections by solving one- or multi-dimensional transport equations in UFG or hyperfine group (HFG). The MC^2-3 code solves the consistent P_1 multigroup transport equation in the UFG or the HFG level. The calculation starts with preparing pointwise isotopic cross-sections by analytical Doppler-broadening the reconstructed cross-sections at specified isotopic temperatures in the resolved resonance energy range. The pointwise cross-sections are directly used in the HFG calculation, whereas self-shielded cross-sections are prepared for the UFG calculation by the numerical integration of pointwise cross-sections based upon the narrow resonance (NR) approximation. For both the HFG and UFG calculations, unresolved resonances are self-shielded using the analytic resonance integral method.

MC^2-3 generates self-shielded BG cross-sections in a two-step approach [3] to account for local and global effects separately. In the first step, the local spectrum and heterogeneity effects are considered in an assembly level. Self-shielded UFG macroscopic cross-sections are generated for each assembly type by solving a slowing-down problem for homogeneous mixture, a one dimensional (1D) pin or assembly cell problem with the collision probability method (CPM), or an assembly problem with two dimensional (2D) or three dimensional (3D) method of characteristics (MOC). Since the local heterogeneity effects are relatively small in fast spectrum reactors, homogeneous mixture calculations are often used for scoping calculations. However, the local heterogeneity effects are generally not negligible even in fast spectrum reactors, and thus

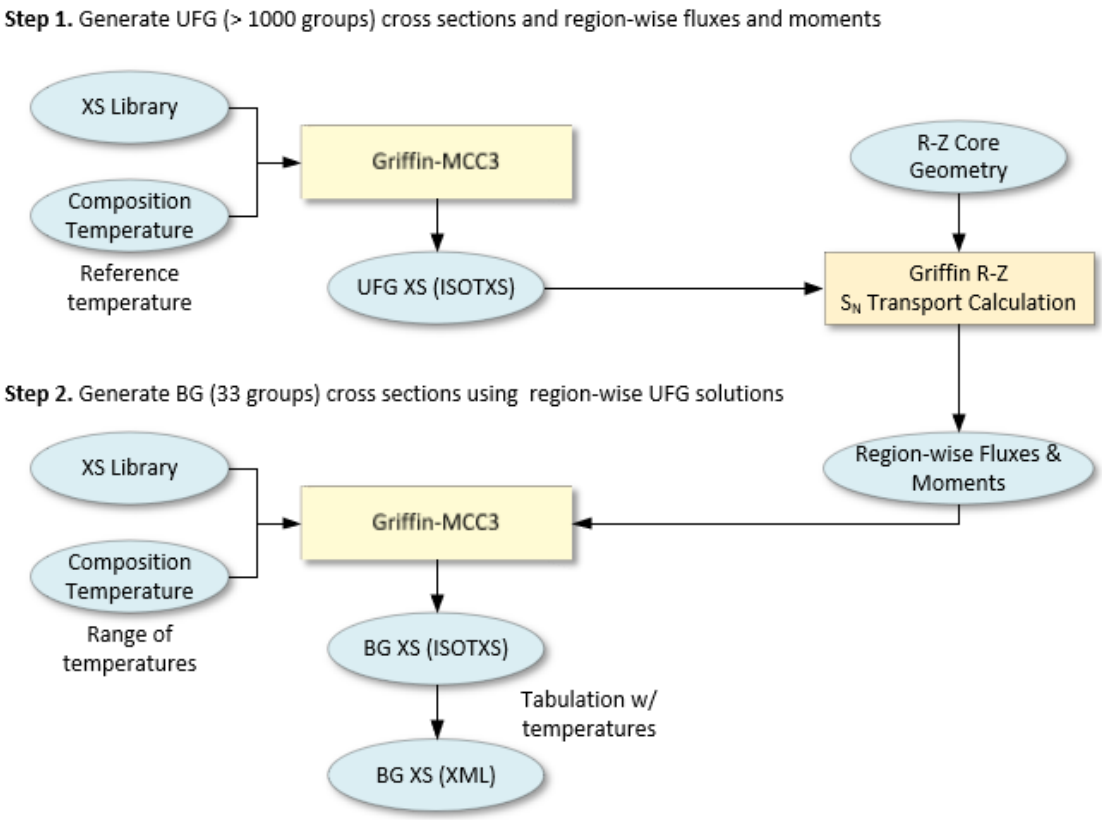


Figure 1: Two-step Cross-section Generation Procedure.

proper spatial self-shielding calculations need to be performed.

In the second step, using the homogenized UFG cross-sections of the first step, a whole-core transport calculation is performed in cylindrical-Z (R-Z) geometry to account for the region-to-region spectral transition effects (i.e., typically between inner core, outer core, blanket, reflector, and shield regions). After an R-Z core calculation is performed with high-order SN and anisotropic scattering options by preserving the regions volumes in the target core, region-dependent BG cross-sections are generated using group condensation with the R-Z flux and moment solutions.

2.2 Mesh Generation

Given the numerous types of homogenized, partially homogenized, and equivalent core configurations that might be used as part of the cross-section generation and full-core analysis workflow, a new type of mesh generator, called *EquivalentCoreMeshGenerator*, was created to assist users with generation of intermediate core meshes for the Griffin R-Z core UFG calculation given a fully heterogeneous input mesh with

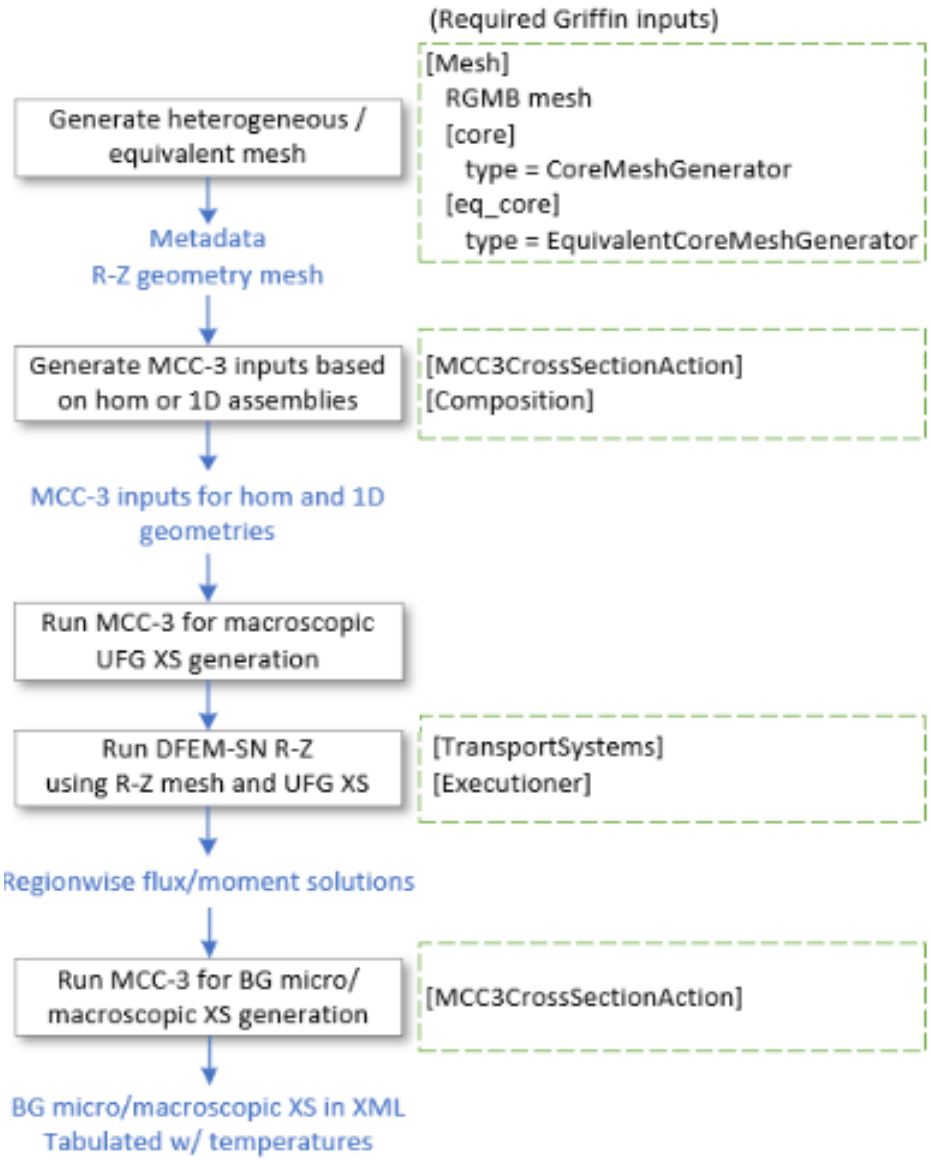


Figure 2: Fast reactor cross-section generation workflow using MC^2-3.

heterogeneous material regions defined directly on the mesh. This section describes the implementation details of *EquivalentCoreMeshGenerator*, what type of target geometries that it can be used to create, as well as how these meshes are used in the cross-section generation workflow as a whole. Broadly speaking, this mesh generator operates with minimal user specifications, and automates the task of defining the target geometry and region assignment specifications of the output mesh needed in order to generate equivalent core configurations. Currently, *EquivalentCoreMeshGenerator* supports four target geometries that can be generated from a fully heterogeneous input mesh: 2D R-Z, fully homogeneous, duct heterogeneous, and ring heterogeneous geometries.

2.2.1 Input Mesh Generation Through RGMB

In order to ensure that the input mesh to *EquivalentCoreMeshGenerator* follows a predictable structure in terms of geometrical features and reporting IDs used to tag like mesh elements together, the mesh generators found within the RGMB toolset, defined through the MOOSE *Reactor* module [12], are required to define the heterogeneous mesh that will serve as the input mesh to *EquivalentCoreMeshGenerator*. RGMB mesh generators are designed to streamline the geometry definitions associated with typical Cartesian and hexagonal reactor geometries, and can represent a 2D or 3D extruded mesh [13]. Specifically, these reactor geometries refer to typical Cartesian and hexagonal pin structures, lattices of pins defined as assembly structures, and lattices of assemblies defined as a core structure. Moreover, RGMB also automatically tags the output mesh elements with a number of region IDs allowing mesh groupings by pin location, pin type, assembly location, assembly type, and axial plane level. Finally, the direct definition of the ”region ID” reporting ID allows for users to directly define cross-section material regions directly onto the mesh. This ”region ID” reporting ID serves as a proxy for the ”material ID” used by Griffin to distinguish which mesh elements correspond to a specific cross-section material zone. In FY23, the MOOSE meshing team also added the capability of representing reactor meshes generated through RGMB as queryable metadata [6], which consolidates how the fully heterogeneous mesh is represented in order for *EquivalentCoreMeshGenerator* to determine how to define its equivalent target geometries. In addition, since the RGMB mesh generators provides a framework for generating the target geometries that *EquivalentCoreMeshGenerator* is responsible for, through a series of mesh subgenerator calls, RGMB can also be leverage to define the geometry and associated region IDs for a fully homogeneous, duct heterogeneous, and ring heterogeneous core, as will be described in Section 2.2.3.

2.2.2 *EquivalentCoreMeshGenerator* Pre-processing and Metadata Generation

Before *EquivalentCoreMeshGenerator* defines the equivalent core based on the user-provided *target_geometry* input parameter, a number of pre-processing steps are conducted in order to check the validity of the input RGMB mesh and to define additional metadata that could be used through the mesh generation process or by *MCC3CrossSectionAction*, *MCC3CrossSectionGeneration*, or *MCC3CrossSectionUserObject*. These include:

- *Checking validity of input heterogeneous mesh*: *EquivalentCoreMeshGenerator* checks to see if the input mesh is indeed a core-level mesh generated by RGMB, and will error out if the expected core-level metadata is not defined on the input mesh. There is also an additional check to ensure that the input mesh has a hexagonal lattice pattern, and that none of the input assemblies within the core are homogenized. There is no constraint on the dimensionality of the input mesh, and *EquivalentCoreMeshGenerator* can work with both 2D and 3D meshes created by RGMB

- *Representing pin and assembly lattices by ring location*: By default, metadata defined by RGMB defines the hexagonal pin and assembly lattice using a row and column based representation. For simplifying many of the downstream geometry conversions, the ability to represent the lattices by grouping pins and assemblies that belong to the same ring is preferred, where the central pin/assembly in the lattice belongs to the first ring, the six adjacent surrounding pins/assemblies belong to the second ring, the next twelve adjacent surrounding pins/assemblies belong to the third ring, and so on. This is done in *EquivalentCoreMeshGenerator* by representing each lattice of pins and assemblies within the core as a 2D vector, with the outer index corresponding to the ring number, and the inner index corresponding to the pin/assembly type ID in each azimuthal location for a given ring. This data is stored as metadata for the core and each assembly that has a constituent pin lattice.

- *Checking all pins within each ring of the pin lattice have the same type*: In order to simplify the 1D and ring heterogeneous representations of the subassembly regions within the input heterogeneous mesh, an additional constraint is imposed onto the input mesh that requires that all pins that belong to the same hexagonal ring within a pin lattice must have the same type, where the pin type ID is specified by the RGMB mesh generators to distinguish the different pins that constitute the pin lattice. This constraint allows homogenization procedures to occur by hexagonal rings in the pin lattice, instead of

separately homogenizing two different pins that could otherwise belong to the same hexagonal ring within the pin lattice.

- *Determining unique subassembly regions in core*: In order to determine how to define equivalent core geometries, *EquivalentCoreMeshGenerator* iterates through all subassembly regions to determine uniqueness. The subassembly is represented by the associated assembly mesh generator name and axial plane ID, where a default plane ID of 0 is used for 2D meshes. For each subassembly region, *EquivalentCoreMeshGenerator* iterates through the geometrical specifications and region ID assignments of all subpin regions, background regions, and duct regions of the subassembly to see if they match with any other unique subassemblies that have been determined. If not, then a new "subassembly ID" is assigned to this subassembly region (identified by the combination of assembly name and axial plane ID). In this manner, each subassembly region within the core will be tagged with a subassembly ID denoting which subassembly regions share the same geometrical and region ID properties.

- *Determining 1D ring representation of each subassembly region*: For each unique subassembly region determined in the previous step, the equivalent 1D representation of this subassembly is determined. This 1D representation is based on a series of concentric rings. This 1D ring representation can be used to set the specifications for the MC^2-3 1D heterogeneous cross-section calculations within the fast reactor cross-section workflow [3], as well as for defining the regions and geometries of the equivalent ring heterogeneous geometry. The 1D ring representation for a representative fuel subassembly region is shown in Fig. 3

- *Compute homogenized volume fractions of each unique subassembly region*: In order to define the MC^2-3 input files used to generate fast reactor cross sections, the equivalent volume fractions of each unique subassembly region need to be determined. In order to accomplish this, the volumes of all heterogeneous regions within the subassembly are calculated and grouped by the heterogeneous region IDs specified by RGMB, and the volume fraction of each heterogeneous region ID within the subassembly is stored as metadata. MC^2-3 uses this homogenized volume fraction information to define the volume fraction of each region ID within the homogenized MC^2-3 input files, where each region ID is associated with a particular composition of isotopes and isotopic densities.

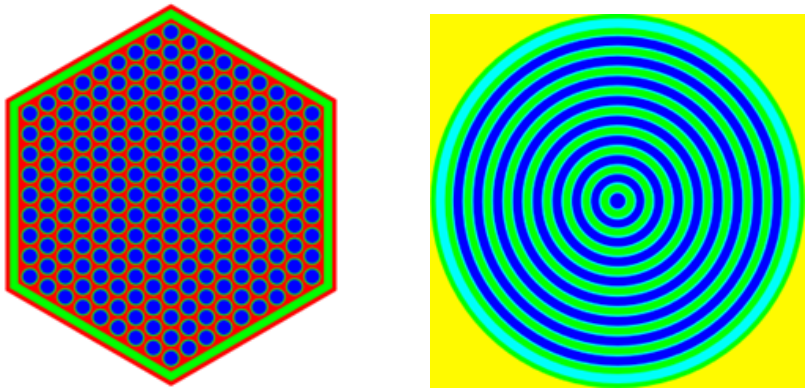


Figure 3: 1D ring representation (right) of a heterogeneous subassembly region (left). Ring boundaries are determined by preserving the volume of each ring of pins, followed by the assembly background and duct regions.

Through these pre-processing steps, a summary of metadata defined by *EquivalentCoreMeshGenerator* is provided in Table 1.

2.2.3 Generation of Equivalent Cores

With the additional metadata defined by *EquivalentCoreMeshGenerator* described in Section 2.2.2, the prerequisite data needed to define the equivalent target cores has all been computed. A brief description of the equivalent core mesh generation algorithms for each of the target geometry types is provided below. For each of the target geometries, a conversion from a simple 4-ring heterogeneous input mesh defined with RGMB. This heterogeneous mesh is a 3D mesh with specifications based on the ABTR core [14], with a central control assembly surrounded by two rings of fuel assemblies and a ring of reflector assemblies. Each of these three assembly types have different pin lattices, assembly duct specifications, and assembly-level heterogeneities to illustrate the wide range of assembly types that are supported through *EquivalentCoreMeshGenerator*.

2D R-Z Target Geometry A 2D R-Z mesh is required to run intermediate flux calculations in the cross-section generation workflow, so *EquivalentCoreMeshGenerator* has the added capability of generating an equivalent 2D R-Z mesh from a fully heterogeneous input mesh as well. A simple algorithm is used to group assemblies within the same hexagonal ring together to determine the radial boundaries of the R-Z mesh, where the radii are made to preserve the total volume of grouped hexagonal assemblies within the assembly lattice. However, the presence of multiple assembly types within the same ring of the assembly

Table 1: Name, datatype, and description of all additional metadata defined by *EquivalentCoreMeshGenerator* for use with equivalent core mesh generation and MC^2-3 cross-section workflow.

Metadata Name	Datatype	Description
`het_mesh_name`	std::string	Name of RGMB input core mesh generator, for retrieving properties of the heterogeneous input mesh
⟨**NAME**⟩`_assembly_lattice_by_ring`	std::vector⟨std::vector⟨int⟩⟩	2D vector of ring representation of assembly lattice for the core, where ⟨**NAME**⟩ is the name of the core mesh generator from RGMB. Outer indexing is the ring number, and the inner indexing is assembly type ID of each assembly within a particular ring of the assembly lattice
⟨**NAME**⟩`_pin_lattice_by_ring`	std::vector⟨std::vector⟨int⟩⟩	2D vector of ring representation of pin lattice for a particular assembly, where ⟨**NAME**⟩ is the name of the assembly mesh generator from RGMB. Outer indexing is the ring number, and the inner indexing is pin type ID of each pin within a particular ring of the pin lattice. This is defined for each assembly that is part of the core lattice
`region_subassembly_map`	std::map⟨unsigned int, std::vector⟨ std::pair⟨std::string, unsigned int⟩⟩⟩	Mapping of unique subassembly ID to vector of subassembly regions associated with unique subassembly ID, where each subassembly region is denoted by its assembly name and axial plane ID
`subassembly_region_map`	std::map⟨ std::pair⟨std::string, unsigned int⟩, unsigned int⟩	Mapping of subassembly region to its unique subassembly ID, where each subassembly region is denoted by its assembly name and axial plane ID
`subassembly_1d_boundaries`	std::map⟨unsigned int, std::vector⟨Real⟩⟩	Mapping of unique subassembly ID to vector of radii corresponding to equivalent 1D ring representation
`subassembly_1d_regions`	std::map⟨unsigned int, std::vector⟨unsigned int⟩⟩	Mapping of unique subassembly ID to vector of region IDs corresponding to equivalent 1D ring representation
`homogenized_volume_fracs`	std::map⟨unsigned int, std::map⟨unsigned int, Real⟩⟩	Mapping of unique subassembly ID to map of heterogeneous region IDs within subassembly and associated homogenized volume fraction

lattice creates ambiguity about how to define the radial boundaries of the equivalent R-Z mesh. In such cases, *EquivalentCoreMeshGenerator* also accepts a user-defined vector of radial boundaries and assembly type IDs to define custom boundary and region ID specifications of the R-Z mesh. A conversion from a fully heterogeneous RGMB mesh to a 2D R-Z mesh using the default R-Z mesh generation algorithm is shown in Fig. 5. The left and right boundaries represents the center and outer radial boundary of the R-Z mesh, respectively, and axial boundaries and regions are made to match those of the unique subassembly regions of the input heterogeneous mesh.

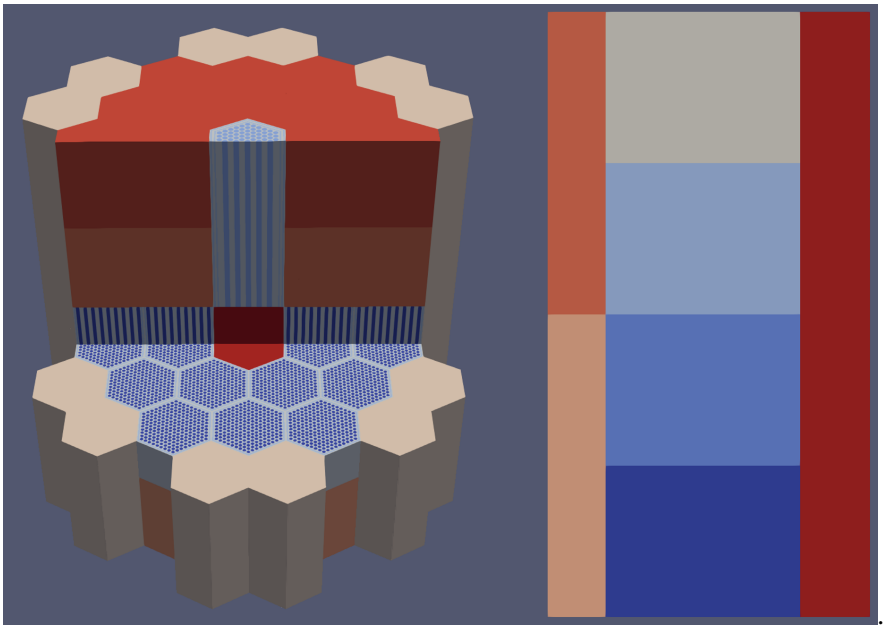


Figure 4: Conversion of fully heterogeneous RGMB mesh to 2D R-Z mesh through *EquivalentCoreMeshGenerator*. The left boundary of the R-Z mesh represents the radial centerline of the R-Z mesh, while the right boundary represents the outer radial boundary. Meshes are colored by region ID.

Fully Homogeneous Target Geometry The fully homogeneous target geometry is generated by taking the iterating through each combination of assembly name and axial plane ID and determining the associated subassembly ID defined by the `region_subassembly_map` metadata field. This subassembly ID is set as the region ID of the output mesh, and this subassembly ID is also used to generate the homogenized cross-sections associated with each unique subassembly region. This 1:1 mapping ensures that cross-section IDs used to calculate homogenized cross sections through MC^2-3 and *EquivalentCoreMeshGenerator* are consistent with the IDs assigned to each subassembly on the fully homogeneous output mesh, when running the full-core fast reactor analysis problem after cross-section generation has taken place. A conversion from

a fully heterogeneous RGMB mesh to a duct heterogeneous mesh through *EquivalentCoreMeshGenerator* is shown in Fig. 5.

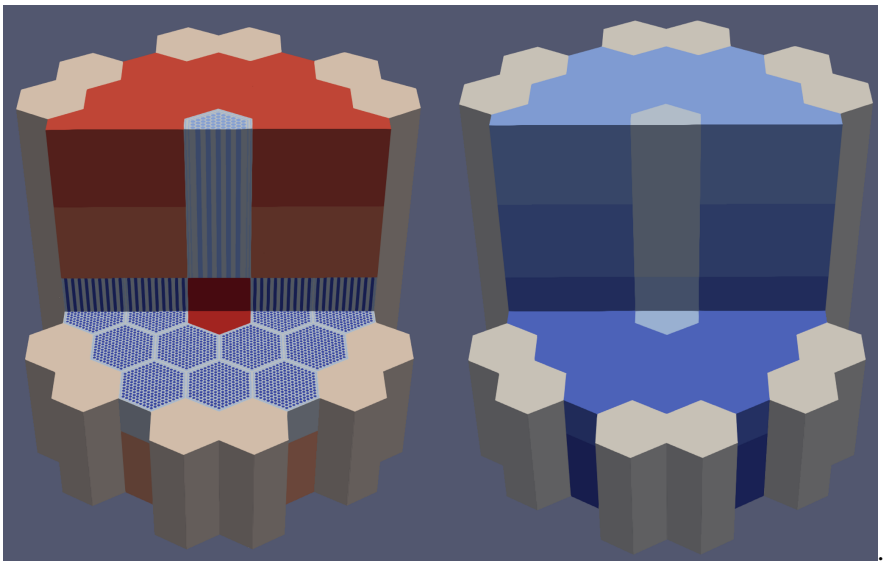


Figure 5: Conversion of fully heterogeneous RGMB mesh to fully homogeneous mesh through *EquivalentCoreMeshGenerator*. Meshes are colored by region ID.

Duct Heterogeneous Target Geometry The duct heterogeneous target geometry is generated by preserving the background and duct regions and boundaries of each subassembly region, while homogenizing the constituent pin lattice region into a single hexagonal region that preserves the volume of all pins within the assembly. A conversion from a fully heterogeneous RGMB mesh to a duct heterogeneous mesh through *EquivalentCoreMeshGenerator* is shown in Fig. 6.

Ring Heterogeneous Target Geometry The region IDs and geometry of the target ring heterogeneous core are determined by first figuring out the unique subassembly ID associated with each subassembly region, which is given by the `subassembly_region_map` metadata defined in Section 2.2.2. By iterating through each assembly name and axial level of the input core mesh, the unique subassembly ID associated with each subassembly region can then be used to figure out the associated 1D ring representation of ring boundaries and ring regions through the `subassembly_1d_boundaries` and `subassembly_1d_regions` metadata fields. Since this data gives the circular ring representation of the subassembly, the associated radii need to be converted to volume-preserved hexagonal duct half-pitches, which can be inputted to the mesh subgenerator calls to the RGMB mesh generators to define the ring heterogeneous mesh. A conversion from

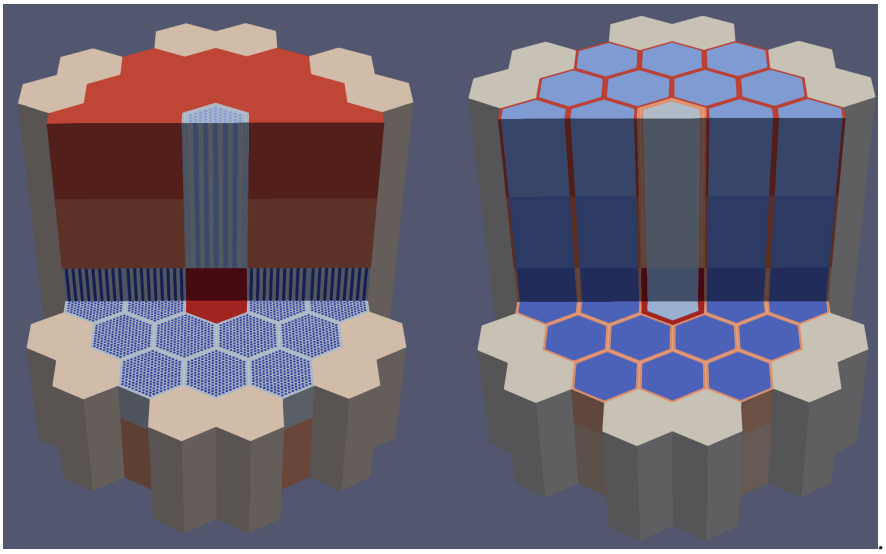


Figure 6: Conversion of fully heterogeneous RGMB mesh to duct heterogeneous mesh through *EquivalentCoreMeshGenerator*. Background and duct regions and boundaries and preserved between the input and output mesh, and a single homogenized hexagonal region is used to represent the pin lattice. Meshes are colored by region ID.

a fully heterogeneous RGMB mesh to a ring heterogeneous mesh through *EquivalentCoreMeshGenerator* is shown in Fig. 7.

Fully Heterogeneous Target Geometry Basically the process for this geometry is the same as that for the ring heterogeneous geometry, but there is no geometry simplification from a RGMB mesh. This still requires an R-Z mesh for running an R-Z core calculation with UFG cross-sections to obtain regionwise spectra for group condensation to BG cross-sections.

2.2.4 Chaining of *EquivalentCoreMeshGenerator*

An additional implementation detail of *EquivalentCoreMeshGenerator* involves the chaining of successive *EquivalentCoreMeshGenerator* calls to efficiently generate the target equivalent core mesh. Since all equivalent cores can be generated through the metadata shown in Table 1 and the metadata defined through RGMB, the metadata generation process and data pre-processing outlined in Section 2.2.2 does not need to be repeated in successive *EquivalentCoreMeshGenerator* calls. Instead, for the second *EquivalentCoreMeshGenerator* call, it will check to see if metadata defined by *EquivalentCoreMeshGenerator* already exists within the MOOSE mesh system, and if so, proceed straight to generating the equivalent core of interest. Such code design allows for one type of mesh to be generated for the cross-section generation

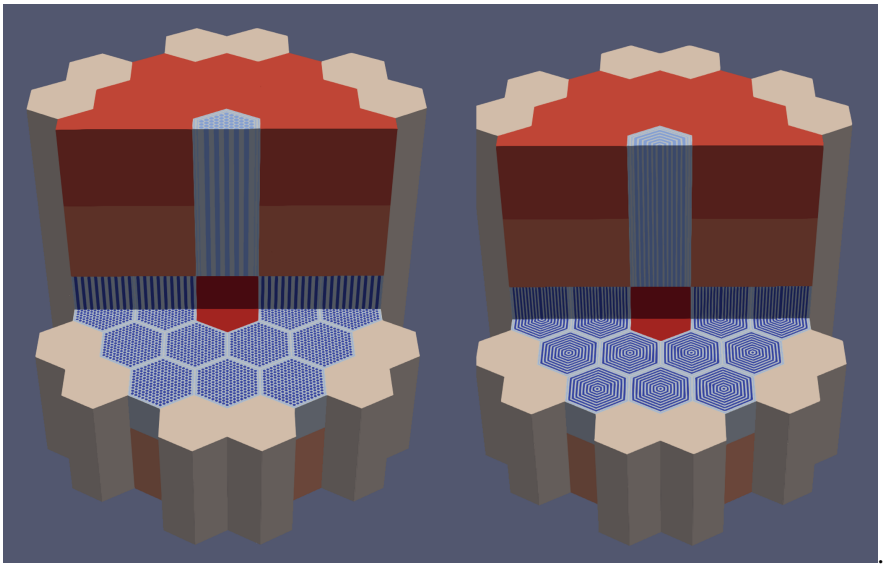


Figure 7: Conversion of fully heterogeneous RGMB mesh to ring heterogeneous mesh through *EquivalentCoreMeshGenerator*. Meshes are colored by region ID.

workflow, and then the checkpoint file generated through this process can be imported directly and read by the MOOSE mesh system to generate another type of equivalent core used for the full-core fast reactor flux calculation. Since this checkpoint file stores the metadata that was defined by *EquivalentCoreMeshGenerator* in the first mesh generator call, this metadata generation step and heterogeneous mesh generation step can be skipped entirely by the second *EquivalentCoreMeshGenerator* call.

2.2.5 Use of *EquivalentCoreMeshGenerator* in the Cross-section Generation Workflow using MC^2-3

Given the numerous ways in which *EquivalentCoreMeshGenerator* is used to generate all intermediate meshes and any metadata used for downstream steps in the overall fast reactor analysis workflow, this section summarizes the interplay between *EquivalentCoreMeshGenerator* and each step of this workflow, where it is assumed that a two-step cross-section generation procedure is used to generate homogenized cross sections for the ultimate full-core homogenized core calculation.

- *EquivalentCoreMeshGenerator* is invoked to define the geometry and region IDs associated with an equivalent mesh to set up any intermediate Griffin calculation. In this case, the 2D full-core R-Z flux calculation required as part of the cross-section generation workflow is handled by Griffin, so the ”R-Z” option is chosen as the target geometry for *EquivalentCoreMeshGenerator*, which automati-

cally generates the equivalent R-Z mesh and material/region ID assignment for use with this Griffin calculation

- Before the Griffin R-Z calculation is initiated, `MCC3CrossSectionAction` takes the subassembly IDs defined through the `region_subassembly_map` metadata field defined by *EquivalentCoreMeshGenerator*, and creates and runs the input files for the first step of the MC^2-3 cross-section generation workflow. Each unique subassembly ID found within `region_subassembly_map` will will have its own cross-section dataset generated by MC^2-3, and the homogenized volume fractions associated with each subassembly ID are used to populate the MC^2-3 input files for this step of the cross-section generation workflow, which can be found in the metadata field `homogenized_volume_fracs`.

- Now, the 2D R-Z flux calculation is conducted through Griffin, and the consistent 1:1 mapping of IDs used by MC^2-3 to generate cross-section datasets and the region IDs defined on the equivalent R-Z mesh ensures that the cross sections defined by the first step of MC^2-3 are correctly mapped onto the R-Z core without the user having to keep track of what these subassembly IDs actually are.

- `MCC3CrossSectionUserObject` passes the fluxes from the R-Z calculation to the second step of the cross-section generation workflow through MC^2-3. This step can involve the calculation of 1D heterogeneous cross-sections through CPM, and the 1D ring model dimensions and region IDs can be queried by `MCC3CrossSectionUserObject` through the `subassembly 1d boundaries` and `subassembly 1d regions` metadata fields that were defined by *EquivalentCoreMeshGenerator*

- With the homogenized cross sections for each unique subassembly region generated, *EquivalentCoreMeshGenerator* is called one more time to initiate the full core flux calculation, and a "homogeneous hexagonal" target geometry is specified in the input mesh block for this step. Keeping the heterogeneous mesh and region ID specifications identical between the cross-section generation workflow and the full-core homogeneous flux calculation step ensures that region/material IDs and cross-section library IDs generated by MC^2-3 and ISOXML are defined consistently, once again removing the need for the user to manually specify the material IDs on the homogeneous core mesh.

2.2.6 Future Improvements to *EquivalentCoreMeshGenerator*

While the use of *EquivalentCoreMeshGenerator* is an integral and necessary part of the cross-section and fast reactor analysis workflow, certain limitations are present which should be addressed as future functionality improvements, including:

- RGMB does not support the definition of wire wraps, and as such, the presence of wire wraps cannot be currently modeled with *EquivalentCoreMeshGenerator*.
- RGMB requires that all assembly duct and background regions on the input heterogeneous mesh be defined as concentric hexagonal regions, and cannot be used to define circular duct and background regions, which may be needed for certain types of fast reactor control assemblies.
- The ability to skip mesh generation of the input heterogeneous mesh is not possible though the MOOSE meshing system. Since *EquivalentCoreMeshGenerator* operates entirely by reading the metadata defined on the heterogeneous input mesh and simply ignores the heterogeneous finite element mesh that is generated, the ability to skip generation of the heterogeneous input mesh would provide significant computational time savings, especially if a very fine axial, radial, or azimuthal mesh discretization is needed to represent the heterogeneous mesh.

2.3 Updates of MC^2-3 Cross-section Workflow

In the generation of cross-sections using MC^2-3, it is a user's responsibility to create an appropriate input model for cross-section generations derived from a given fully heterogeneous core model. Depending on the target accuracy and the core design, the input for MC^2-3 can range from a simplified homogenized mixture model to a 1D cylindrical model with R-Z spectrum weighting. This task of input preparation and the execution of MC^2-3 is crucial in the fast reactor analysis, and it may pose challenges for users. The MC^2-3 cross-section workflow in Griffin aims to streamline this process by automating both the input preparation and execution steps.

The workflow offers a capability to automatically generate broad-group cross-section set for individual homogenized regions. For using the workflow, a full-core model should be provided in the mesh block using RGMB in the mesh block. Heterogeneous compositions and corresponding mapping onto the fuel core geometry model are specified in the composition and the cross-section action blocks. Then, the homoge-

nized regions and their compositions are automatically determined. Subsequently, the MC^2-3 are internally executed to generate the broad group cross-section sets. Optionally, the workflow performs the transport calculation for the simplified R-Z model yielded from the full-core model to apply the spectrum weighting to account for region-dependency of cross-sections. The cross-section generation for the SFR core can be done by utilizing the provided capabilities of workflow with the minimal user input preparation and interventions. The following sections will provide details information on the workflow's capabilities.

2.3.1 Overall Structures

The basis code structures of cross-section generation workflow was originally designed and implemented in FY22 to demonstrate a simple workflow of SFR cross-section generations by coupling the Griffin and MC^2-3 codes. In order to support the sophisticated capabilities for cross-section generations, the code structure of MC^2-3 workflow was completely refactored into the following three key components:

- `MCC3CrossSectionAction`: MOOSE action system to configure the overall calculation flow for cross-section generation accordingly based on the user setting.
- `MCC3CrossGeneration`: Driver object to coordinate the executions of MC^2-3 by setting up required input models and calculation options, and to post-process generated cross-section sets into the ISOXML format with tabulations.
- `MCC3CrossSetionUserObject`: Interface user-object to incorporate the R-Z spectrum obtained from the DFEM-SN solver into the MC^2-3.

By incorporating these components properly as schematically illustrated in Fig. 8, the MC^2-3 cross-section workflow can generate the cross-section sets for core analysis in the framework of Griffin.

`MCC3CrossSectionAction` also serves as the user interfaces for gathering the requisite input for cross-section generation. It also configures `MCC3CrossGeneration` and `MCC3CrossSetionUserObject` to properly coordinate the overall calculation flow. The key requisite information for cross-section generation is to define a configuration of heterogeneous composition on the full-core geometry. In a typical Griffin execution with the pre-generated cross-section data, the compositions and their mapping onto the geometry are defined in the composition and material blocks. In case of using this workflow, the heterogeneous compositions are defined in the *[Compositions]* block and their mappings onto the individual components consisting full core

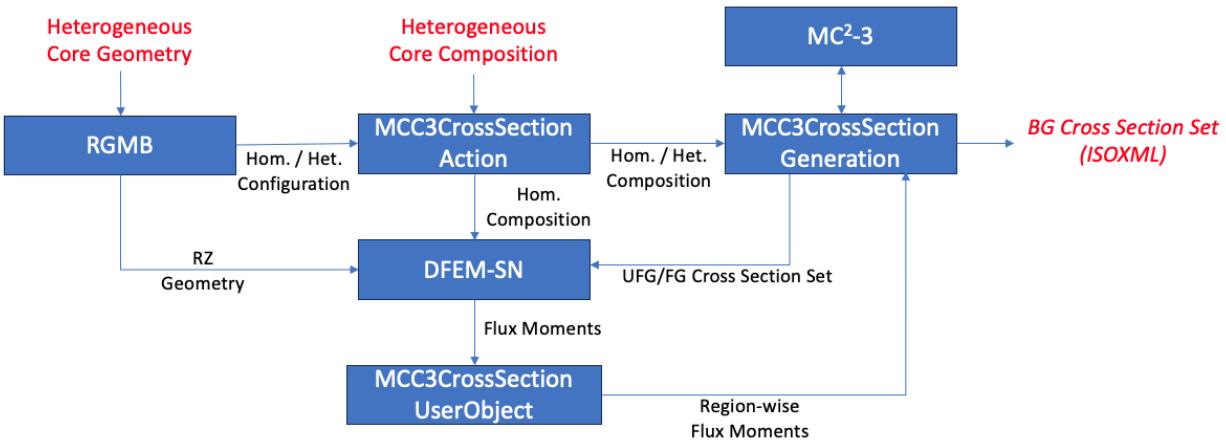


Figure 8: Overall code structure of cross-section generation workflow.

model are alternatively defined in the workflow input. The temperature information for the heterogeneous configuration should be provided, which is required for cross-section generations with temperature tabulations. The *[Materials]* block are internally configured by properly processing the mapping information if necessary, especially performing the R-Z spectrum calculation. Along with the composition information, the necessary settings for cross-section generations such as the group structure and the anisotropic scattering order should be specified.

2.3.2 Homogenized Cross-section Generation Preparation

The cross-section generation in the workflow starts from the preparation of homogenized compositions. When constructing the homogenized core model in RGMB as aforementioned in Section 2.2.5, the volume fractions of heterogeneous components, such as fuel, clad, and coolant, are computed and provided to `MCC3CrossSectionAction`. By combining the heterogeneous compositions and the mapping given in the input parameters of the workflow, the homogenized composition of each compositions is readily determined. By taking the resulting homogenized compositions, then, `MCC3CrossSectionGeneration` prepares the input homogenized mixture models for MC^2-3. Before executing the MC^2-3, the point-wise cross-section data are generated for the isotopes and temperatures used in the input homogenized mixtures. The executions of MC^2-3 are subsequently performed for the individual homogenized mixtures, which generates the BG cross sections for separate *ISOTXS* formatted files for each mixture and temperature grid point. Here, a user can select whether macro or micro cross-sections are prepared in the MC^2-3 executions. Finally, `MCC3CrossSectionGeneration` gathers and merges the output cross-section files into a single ISOXML

file with temperature tabulations, which will be utilized in the following full-core calculations.

2.3.3 Incorporation of R-Z Spectrum

In the standard procedure of MC^2-3 cross-section generation, the core-wide spectrum with the detailed energy group structure, typically 2082- or 1041-group, is obtained by performing the transport calculation on the R-Z geometry which can represent core characteristics. The core spectrum weighing is applied to the homogenized cross-section sets in order to account for the region-dependency of cross sections in the fast reactors. The workflow supports this generation procedure involving the spectrum weighting by utilizing the DFEM-SN solver of Griffin and *EquivalentCoreMeshGenerator* of RGMB.

Cross-section Preparation for R-Z Spectrum Calculations The core model for spectrum calculation is automatically constructed in a manner that accurately represents the key characteristics of core spectrum in R-Z geometry. Thus, the prerequisites of performing the spectrum calculation is preparing the homogenized cross-section sets. The homogenized cross-section set can be readily generated by the existing infrastructure of workflow. `MCC3CrossSectionGeneration` invokes the series of MC^2-3 executions to generate the homogenized cross-section sets for individual homogenized regions with the group energy structure for the spectrum calculation specified by users. It is noted that macro-scopic cross sections are generated in this stage by default to reduce the memory footprint of transport calculations. Here, users needs to specify the reference temperatures for individual heterogeneous compositions as the input parameters. The core spectrum calculation is performed based on the user-defined reference temperature distributions. The tabulation of cross sections is not used in the cross-section generation and the spectrum calculation. Once the cross-section sets are prepared, `MCC3CrossSectonAction` internally configures *[Compositions]* and *[Materials]* blocks to assign the prepared cross sections onto the R-Z geometry meshes, which can minimize the user's interventions for streamlining the workflow.

DFEM-SN Calculations The transport calculations are performed for the prepared R-Z core model and cross-section data to obtain the core-wide spectrum including the high order flux moments. Here, the higher flux moments are utilized as the weighting spectrum for the corresponding order of anisotropic scattering matrices. The DFEM-SN solver was significantly updated to support the R-Z core calculations with the fine energy group structures, which can be found in Section 4.1. The input blocks of *[TransportSystems]* and

[Executioner] for the DFEM-SN solver should be prepared by users. It is also recommended to use the CMFD acceleration option to accelerate the convergence of DFEM-SN calculation.

Cross-section Generation with Spectrum Weighting Once the DFEM-SN calculation is completed, `MCC3CrossSectionUserObject` computes the region-wise spectra of flux moments by taking average of obtained DFEM-SN solutions. Then, the obtained spectra are passed into `MCC3CrossSectionGeneration`. When conducting the MC^2-3 calculations for homogenized regions, the corresponding region-wise spectrum are additionally supplied. The BG cross-sections are generated by applying the weighting spectrum that are determined by super-imposing the region-wise spectrum and the UFG slowing down spectrum. Originally, the weighting spectrum could be supplied to MC^2-3 using the *RZMFLX* format file. currently in the new process, in-memory data transfer of weighting spectrum from Griffin is supported by MC^2-3, which replaces the usage of *RZMFLX* format file. In cases where temperature grid point input is provided for cross-section tabulations, MC^2-3 calculations are performed for individual temperature points and regions while the identical core spectrum are applied regardless of temperature grids.

2.3.4 1D CPM Modeling in Cross-section Generation

For fuel assemblies, 1D modeling is needed to account for heterogeneity effects in cross-section generation. Users need to indicate which assemblies require 1D modeling. Once identified, a 2D assembly is converted into a 1D cylindrical geometry with equivalent composition regions. For instance, an assembly with 9-ring fuel pins (each pin composed of fuel, cladding, and coolant) and duct is translated into a total of 37 regions (3 regions for a center fuel pin, 4 regions for each ring in the rest 8 fuel pins, and 2 regions for duct and coolant) where each fuel ring contains cladding-fuel-cladding-coolant after previous ring, as illustrated in the right one in Fig. 3.

For fully homogeneous assembly calculations, user-specified isotope names in the composition inputs of MC^2-3 should be the same for each isotope to use a single set of cross-sections and a single ID for an ISOXML file. In contrast, for heterogeneous assembly calculations, including duct-heterogeneous, ring-heterogeneous, and fully heterogeneous geometries, user-specified isotope names need to differ for fuel regions. This is based on the assumption that self-shielded cross-sections vary only for fuel regions or rings, and any differences in cross-sections for non-fuel regions (such as cladding, duct, and coolant) are insignificant in terms of their locations within an assembly.

3. Initial Implementation of Fuel Cycle Capability

3.1 Fuel Cycle Capabilities Required for Fast Reactor Analysis

Fuel cycle management in fast reactors involves a series of activities to achieve the targeted performance objective (e.g., fissile breeding, transuranic element burning, electricity generation, etc.) in an optimal way throughout the reactor's operational lifetime. This includes managing the fuel distribution, controlling the power distribution, and managing the reactor's reactivity and refueling operations. In fast reactors, fuel cycle management is particularly important because these reactors typically use high burnup fuel, which can result in complex isotope depletion and buildup patterns. Therefore, fuel management strategies need to be determined carefully in order to optimize the reactor's performance and safety, which can lead to improved reactor performance, increased fuel utilization, reduced waste production, and eventually enhanced safety margins.

The fuel management capabilities for fast reactors primarily requires two basic types of analysis problems: 1) the explicit cycle-by-cycle or non-equilibrium cycle operation of a reactor under a specified periodic or non-periodic fuel management scheme, 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 (i.e., coupled neutronics and isotopic transmutation equations) are solved for user-specified initial fuel compositions and fuel management scheme. The initial or charged fuel compositions can be determined from external feeds according to the user-specified fabrication preference. Optionally, reprocessing may be included in the specification of the external fuel cycle and discharged fuel may 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 has an important practical value when assessing the core performance of new reactors as it is a more valid basis than using an arbitrary sequence of startup cycles or constructing a complex fuel cycle that only considers one assembly shuffling pattern. The equilibrium solution can provide idealized estimates of the burn cycle time, control requirement, fuel enrichment, and general system performance characteristics at a lower overall

computational cost than the explicit calculation of a proposed fuel management scheme. Having completed an equilibrium cycle calculation, the resulting parameters and partially burned fuel densities can be used 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.

The fuel cycle model could be subdivided into an in-reactor cycle and an ex-reactor or external cycle. Fig. 9 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 by carrying out a specified move sequence, which is specified as fuel management scheme. The external cycle model is intended to represent the actual course of events following the discharge of fuel from the reactor. Hence this ex-reactor model consists of the following successive steps: cooling, delivery to a re-processing plant, reprocessing, re-fabrication using both reprocessed and external feed supplies, preloading storage, and reactor charge.

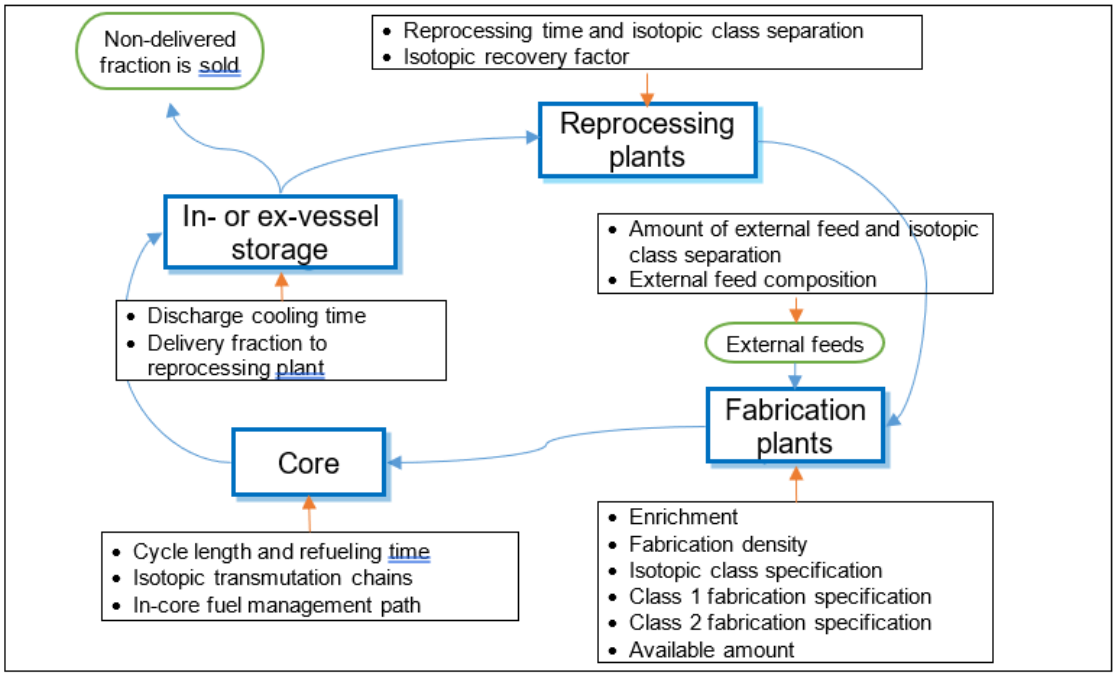


Figure 9: Overview of major fuel cycle models required for fast reactors.

In this fiscal year, we initiated implementing the fuel management capabilities for fast reactors, focusing on multi-cycle depletion and shuffling functions, which will be elaborated in the following section.

3.2 Multi-cycle Depletion and Shuffling

In the fuel management, 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 by carrying out a specified move sequence. Two possibilities arise in the case that fuel management procedures change after a burn cycle.

The first one is a non-repetitive fuel management scheme where a particular move sequence is to be carried out only once. Therefore, each fuel bundle in the reactor must be accounted for separately at the end of each burn cycle. Clearly no equilibrium condition can be defined for this case. This non-repetitive fuel management scheme permits completely general in-reactor shuffling of fuel bundles including temporary out-of-core storage, loading of fresh fuel, and subsequent retrieval and reloading of the fuel.

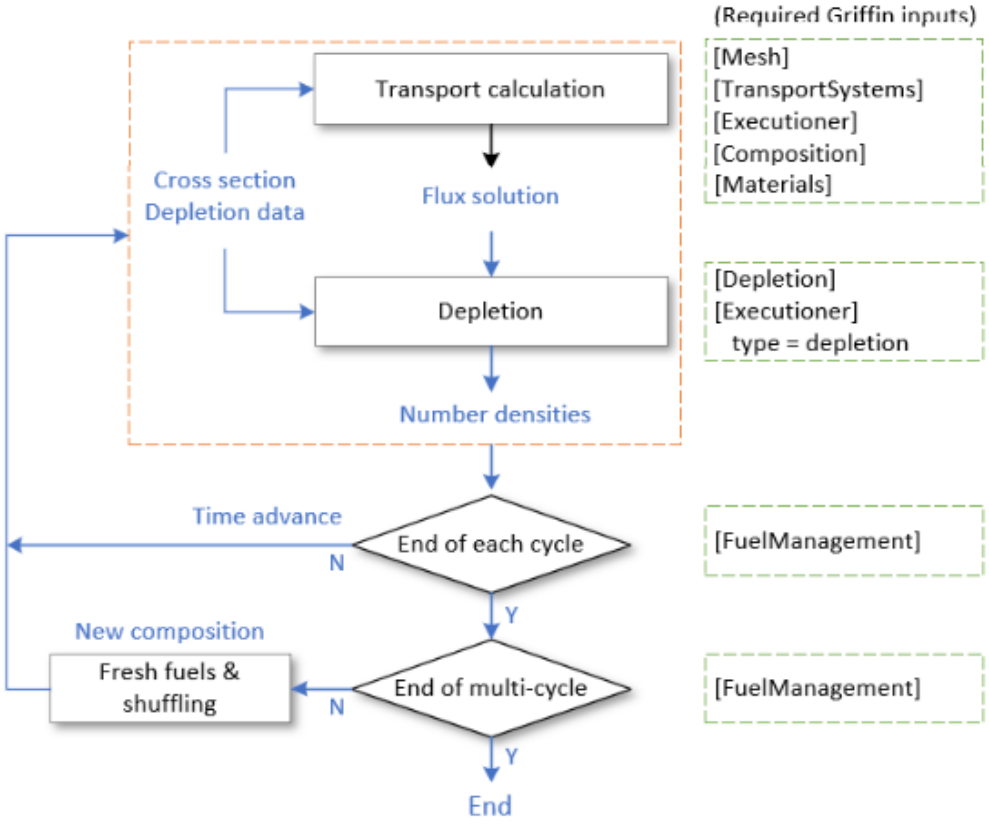


Figure 10: Multi-cycle depletion and fuel management.

The second one is the repetitive fuel management scheme. There exist two different types of repetitive fuel management schemes. One is a scattered reloading without shuffling and the other is a repetitive fuel

reloading with shuffling. In the scattered reloading scheme, often employed in liquid metal cooled fast reactors to reduce the risk of misplacing or dropping fuel assemblies in the opaque coolant, a number of burn cycles elapse without any fuel movement for a particular material type, then one discharge-charge move is made, and so on with a fixed repetition factor. This fuel management scheme is repetitive, leading to an equilibrium condition. This fiscal year, our focus is on multi-cycle depletion with reloading with shuffling for assembly-homogenized cores.

We implemented a flexible assembly-based fuel management scheme in Griffin. Griffin uses *element composition id* and *element depletion id* for an unstructured reactor mesh to manage the depletion calculation. Composition ID and depletion ID can be generated with a RGMB mesh generator, available in Griffin through MOOSE *Reactor* module. Elements with the same composition ID have the same material composition. Each composition is associated with a set of microscopic cross sections for SFR analysis. It is allowed to have multiple compositions share the same set of cross-section library. The composition ID is also used for setting up the initial atomic densities. Elements with the same depletion id, referred to as a depletion zone, are considered together during depletion calculation. Averaged fluxes, temperatures, etc. are used for evaluating the decay-transmutation matrix in size of the number of isotopes under tracking for updating the atomic densities at each depletion step for all depletion zones. Other quantities such as burnups in various units, the total time-integrated fission events, radiation damage, etc. are also tracked depletion-zone-wise. We refer these quantities and the atomic densities of depletion zones as the depletion data. Depletion data can be written to binary files and can be loaded later for calculations of in- or ex-core storage, reprocessing, and fabrication, where neutron flux solution is not needed. In Griffin, elements in one depletion zone must have the same composition id. While for a reactor model, the number of composition IDs is limited, the number of depletion zones can range from thousands to millions.

To perform fuel management, we utilize an element integer, *assembly id*, in Griffin. Elements having the same assembly ID are contained as one assembly and are considered together during fuel management. One assembly may contain multiple compositions and multiple depletion ids. The same composition can be used by multiple assemblies. However, it is not allowed to have the same depletion zone across multiple assemblies. The meshes for the current assembly and the assembly to be shuffled must be identical; in other words, fuel shuffling must not change the mesh. This constraint could be removed in future. Element depletion IDs will not be altered by assembly shuffling, which means that when shuffling, Griffin will figure out the mapping between the depletion IDs of the current assembly and the shuffling assembly and copy the

depletion data accordingly. If the shuffling assembly is a fresh assembly, atomic densities will be initialized based on the compositions of the assembly type. If the current assembly is discharged, depletion data of that assembly will be saved in a binary file for later considerations. After fuel shuffling, element composition ID could be changed if the current assembly and the shuffling assembly are in different types. It is noted that element assembly ID is not changed during assembly shuffling. This design automatically supports various types of assemblies, hexagonal or Cartesian.

Once users add the assembly ID in the mesh, the input syntax for fuel management is quite straightforward. The input parameters required for fuel shuffling are input in *[FuelManagement]* block as shown in Fig. 11, where 2D maps specify new locations of assemblies in each batch. The index in the 2D map indicates the location of an assembly, and the number of the 2D map is the ID of the assembly to be moved into this location. If a dash sign is provided, it means that this assembly is not moved during shuffling. If the number is negative, it means a fresh assembly is to be used and the absolute value of the number is the assembly type that users added in the mesh generators. Each cycle can have an assembly shuffling pattern associated, so that Griffin knows at end of the cycle which pattern is to be invoked. We remind that after each assembly shuffling, binary files for storing depletion data of discharged assemblies are generated. Griffin supports loading the depletion data in binary files and performing separate decay calculations for the loaded depletion data.

We successfully incorporated a scattered reloading capability into Griffin. In a scattered reloading scheme, fresh assemblies can be inserted to fixed locations, while replaced assemblies are discharged from a core without relocating to other locations. Consequently, the deletion data of those regions is reset to the fresh compositions when the burned fuel assemblies are discharged. This scattered reloading scheme is commonly used in typical SFR cores.

To test the implemented reloading capability, we prepared a simple 2D core case having 6 fuel assemblies with a 3-batch reloading scheme. In each cycle, one-third of the core is refueled with fresh assemblies. The core configuration and reloading scheme are illustrated in Fig. 12. A multi-cycle depletion calculations were performed with the 3-batch scheme, and the obtained eigenvalue results for each cycle are summarized in Fig. 13. The core reached equilibrium state after a few transition cycles, as expected, confirming the correct incorporation of the reloading scheme into the depletion framework of Griffin. More work is needed in the next fiscal year to complete the shuffling capability and perform extensive verification tests.

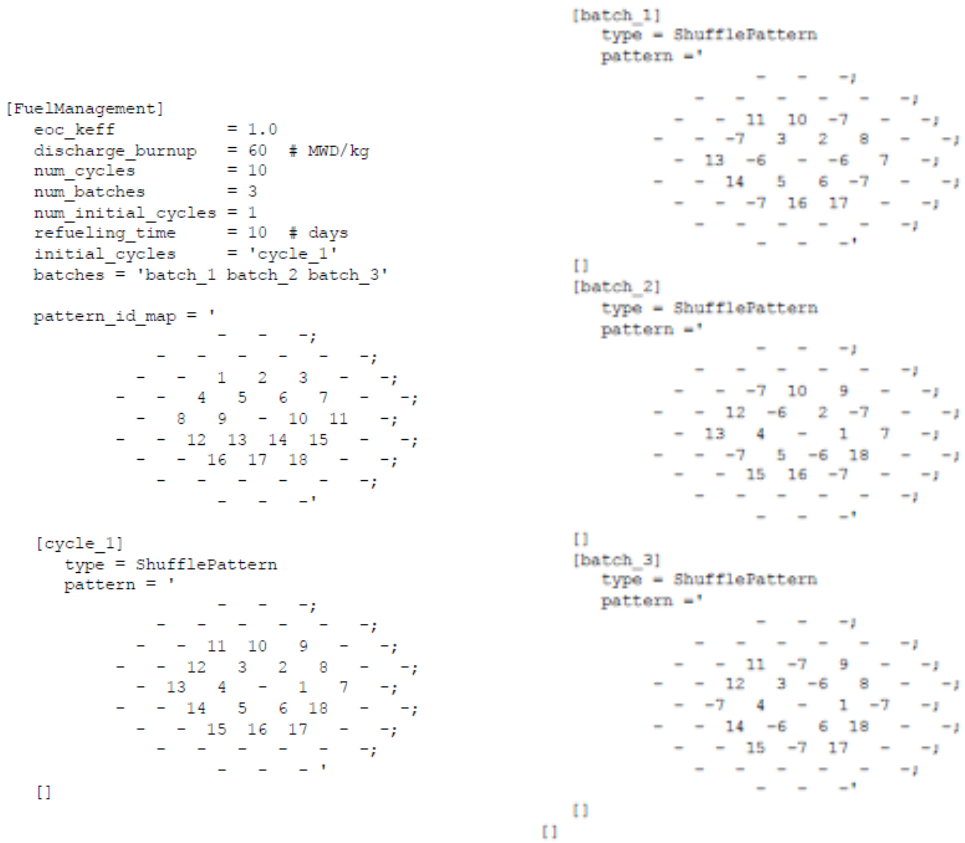


Figure 11: An example input of fuel management.

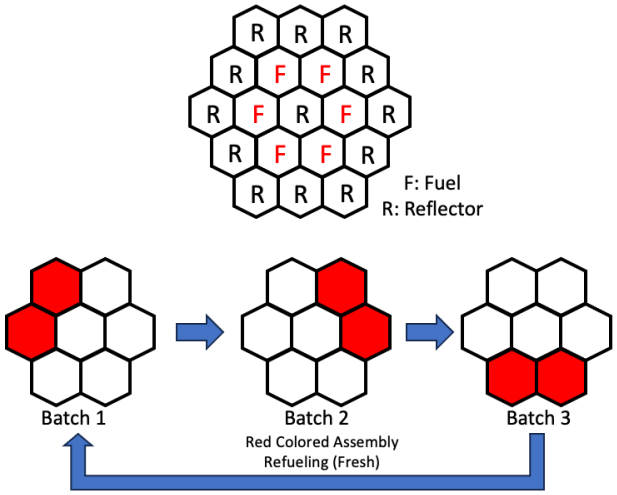


Figure 12: Test case for scattered reloading scheme (top: core configuration, bottom: 3-batch reloading scheme).

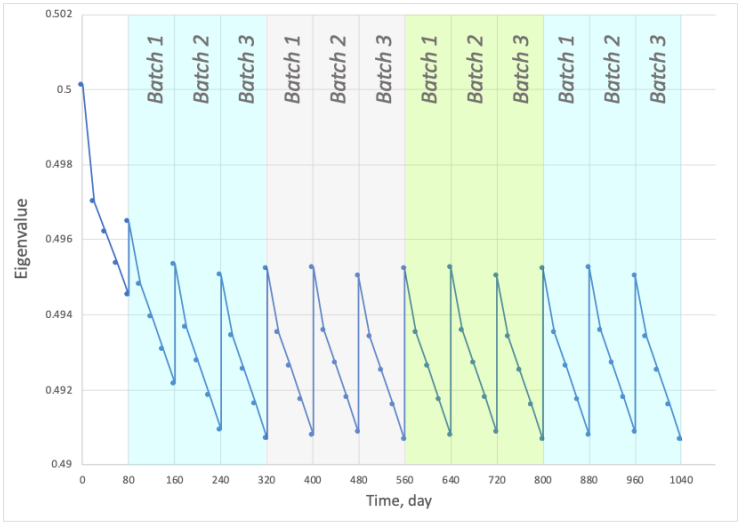


Figure 13: Multi-cycle depletion results for the test case (Fig. 12) of scattered reloading scheme.

4. Improved Capabilities Required for Fast Reactor Analysis

The capabilities required for fast reactor simulation have been improved during this fiscal year. In the previous year, HFEM-PN was functional, and the basic pin power reconstruction capability was implemented. In this fiscal year, both of these capabilities were significantly enhanced, and a new R-Z spectrum solver based on discontinuous finite element method (DFEM)-SN was introduced to efficiently solve R-Z problems with UFG cross sections. This solver replaces TWODANT [15] in the conventional process of the cross-section generation process with MC_2-3. This section details the R-Z spectrum solver, the pin power reconstruction module, and the HFEM-PN solver.

4.1 Improvement in R-Z Spectrum Solver

The SFR cross-section generation workflow involves solving an R-Z core with many energy groups (as many as 1,000 groups) using SN transport. There are a few ways to solve the SN transport problem in Griffin, but the DFEM-SN scheme paired with sweeper is considered the most efficient solution scheme. Therefore, the existing DFEM-SN sweeper was optimized for many-group calculations and the R-Z sweeper was newly implemented.

4.1.1 Optimization of Sweeper for Many-group Calculations

Griffin has not been tested for problems with an extreme number of energy groups, and initial examination with a test problem revealed a number of inefficiencies when solving a DFEM-SN system for many-group problems. These inefficiencies are not specific to the R-Z calculation but rather a generic problem. Therefore, these inefficiencies were tackled using Cartesian geometry first prior to implementing the R-Z discretization, which can benefit from these optimizations naturally.

Table 2 shows the Griffin calculation time using the DFEM-SN sweeper for a 425-group Cartesian core test problem. The Cartesian core test problem was obtained by simply changing the coordinate of an existing R-Z core problem to the Cartesian coordinate, so the computational cost is approximately the same with the R-Z core problem. The table presents a progressive improvement of runtime with each optimization, where each optimization is explained in the following:

- Scattering source calculation optimization: In many-group calculations, evaluating the scattering source becomes more costly than the sweeping itself, as the cost of scattering source calculation

increases quadratically with respect to the number of groups. The scattering source calculation algorithm in Griffin was not optimal, and therefore, it was optimized.

- Removal of `fission_source_integral` postprocessor: The `fission_source_integral` postprocessor is used to compute the eigenvalue. This postprocessor is executed every iteration and requires a re-evaluation of cross sections which is costly. However, when CMFD acceleration is employed, the eigenvalue is obtained from the result of CMFD acceleration, and the `fission_source_integral` postprocessor becomes meaningless due to an arbitrary re-scaling of flux by CMFD acceleration. Therefore, the `fission_source_integral` postprocessor was conditionally disabled when CMFD acceleration is enabled.

- Cross-section averaging and custom residual evaluation: Residual has been evaluated using MOOSE kernels, which compute quadrature point-wise cross-sections on-the-fly. In order to avoid the costly on-the-fly cross-section evaluation, an option to substitute MOOSE kernels with customized residual evaluation functions that utilize sweeper was implemented. The custom residual evaluation makes use of the element-averaged cross-sections that have been already stored in the sweeper. This option also allows users to control the frequency of cross-section update instead of calling it on every residual evaluation. If the cross-sections are known to be constant during the calculation, they can be computed only once in the initialization stage and stored using this option, which can significantly reduce the cross-section evaluation time. Residual evaluation itself also becomes faster with this option by utilizing pre-assembled elemental matrices (such as mass matrix) in the sweeper instead of doing numerical quadrature evaluation. CMFD acceleration also benefits from this option by using pre-stored element-averaged cross sections in coarse-mesh homogenization instead of evaluating quadrature point-wise cross sections on-the-fly with the MOOSE assembly system, which substantially reduces the CMFD projection time.

- Inner iteration scattering order truncation: Since Griffin uses the residual correction method, the inner iteration does not have to be exact. Therefore, the high-order scattering sources whose importance is marginal compared to the zeroth (isotropic) scattering source can be neglected during the inner iteration without increasing the number of iterations significantly. By skipping the evaluation of high-order scattering sources, the scattering source calculation cost can be substantially reduced.

- Direction-major angular flux ordering: The angular flux data was originally stored such that the direction variables are contiguous (group-major ordering). However, the sweeper solves all groups together which requires accessing the group variables contiguously, and due to the group-major storage of angular flux, accessing the angular flux data suffered from cache misses. Therefore, the angular flux ordering was changed such that the group variables are stored contiguously (direction-major ordering).

As the result, the runtime was improved by more than 17 times, reducing the base runtime of more than 3 hours to 11 minutes.

Table 2: Progressive improvement of runtime using sweeper with each optimization for a 425-group Cartesian core test problem. 24 processors were used on a single Sawtooth node.

Optimization	Runtime (min)	Relative Performance
Base	193.8	1.00
Scattering Source Calculation Optimization	82.2	2.36
Removal of `fission_source_integral` Postprocessor	51.4	3.77
Cross-section Averaging and Custom Residual Evaluation	16.0	12.11
Inner Iteration Scattering Order Truncation	12.0	16.15
Direction-major Angular Flux Ordering	11.2	17.30

Many-group calculations not only revealed issues in the computational performance but also in the memory usage. This prevented the test problem from being run with all the available processors (48 processors) on a single Sawtooth node. Since the target number of energy groups to be used for the cross-section generation goes over 1,000, it can be easily expected that Griffin will not be able to run the actual target problems with a reasonable amount of resources due to the memory constraint. Therefore, the excessive memory usage of Griffin for many-group calculations was tackled as well.

Table 3 shows the progressive reduction of peak memory usage by each optimization, depending on the number of processes used on a single Sawtooth node. OOM stands for out of memory, which means that the memory usage is over 186.9GB (per-node memory limit). The custom residual evaluation that was discussed in the performance optimization also reduced some of the memory usage by removing the memory added by MOOSE kernels. Removing redundant memory allocation in `MixedMatIDNeutronicsMaterial` was achieved by not allocating the memory for face and neighbor materials that are automatically created by MOOSE but not required in DFEM-SN calculations, and by selectively loading the cross-section libraries only for the material IDs that belong to the current partition of the domain. As the result, the scalability

of memory usage was improved, and the test problem could be run with all 48 processors with sufficient memory margin. The memory usage reduction from the base case is about 30% with 24 processes, and that of 48 processes is thought to be around 40%, although the precise value cannot be known as the base case ran out of memory.

Table 3: Progressive reduction of peak memory usage using sweeper with each optimization for the 425-group Cartesian core test problem. A single Sawtooth node was used.

Optimization	Memory Usage (GB)		
	1 Process	24 Processes	48 Processes
Base	49.0	144.7	OOM[1]
Custom Residual Evaluation	48.7	136.7	OOM[1]
Removal of Redundant Memory Allocation in `MixedMatIDNeutronicsMaterial`	47.8	98.9	133.4

[1] OOM: out of memory

However, the memory footprint is still high, and the fundamental cause of this high memory footprint has to do with the storage of angular flux and the inefficient memory usage of the libMesh `GHOSTED` vector type used by MOOSE to store solution and residual vectors. It has been observed that a libMesh `GHOSTED` vector uses more than three times of memory compared to its data size. Removal of angular flux storage or memory optimization of the libMesh `GHOSTED` vector type is a longer-term goal which should be achieved in FY24.

4.1.2 Implementation of R-Z Sweeper

As discussed in the previous sections, several optimizations were completed to dramatically increase the efficiency with which Griffin solve DFEM-SN problems with many energy groups. The optimizations were performed considering a 2D Cartesian problem to simplify the work. Griffin includes an efficient sweep solver for DFEM-SN transport in Cartesian geometry [5], the core sweeper itself did not need to be updated for many energy group problems. However, a new discretization was required for R-Z geometry. Griffin previously included the ability to solve DFEM-SN transport in R-Z geometry; however, the method previously implemented handled the angular derivative using a spherical harmonics expansion (PN) expansion [1]. No purpose built solver was developed for this democratization existed in Griffin and so a large matrix including the coupling between all directions had to be build and solved with a general linear solver. This was only practical for relatively small problems. A new DFEM-SN discretization for R-Z geometry was im-

plemented in Griffin where the angular derivative is handled by weighted diamond difference. The existing DFEM-SN sweeper was modified to be able to solve this new discretization.

To verify the new DFEM-SN sweeper in R-Z geometry, it was first determined that the new solver produced the same answer as the existing method using a PN expansion for several problems. Since a primary purpose of the new sweeper is to solve problems with many energy groups, the efficiency of the solver was of principal concern. The expectation was for the modified sweeper to solve an R-Z problem with slightly worse efficiency than the existing sweeper in 2D Cartesian geometry. The efficiency is expected to be slightly worse because there are additional terms such as the angular derivative in R-Z geometry not present in Cartesian which take some additional computations. Furthermore, the parallel efficiency of the R-Z sweeper will be lower than that for Cartesian geometry since the angular derivative results in some coupling between directions not present in the Cartesian equations. Despite these additional complexities, in a test performed on 24 processors, it was found that for a test problem with 1,041 energy groups, an R-Z sweep was only 10 % slower than the 2D Cartesian geometry sweep.

4.2 Improvement in Pin Power Reconstruction

For problems that leverage spatial homogenization to compute the resultant flux distribution, pin power reconstruction is a necessary feature to evaluate pin-by-pin power distributions within fuel assemblies of interest. In FY22, `PinPowerReconstructUserObject` was implemented in Griffin to conduct pin power reconstruction on steady-state eigenvalue problems, and the methodology was tested on a variety of benchmark cases, including the Empire, ABTR, and C5 benchmark problems [16]. In FY23, further improvements to `PinPowerReconstructUserObject` were added to streamline the calculation of full-core form functions, integrate this user object with existing multiphysics solvers in MOOSE, and test multiphysics feedback problems with pin power reconstruction based on varying thermal and transient conditions. The majority of test cases for these applications were based on the thermal spectrum 2D Empire core [17], the methods outlined here can be easily generalized and applied to fast reactor problems as well.

4.2.1 Updates to Form Function Library Generation and Tabulation

Form functions are typically defined assuming single assembly or colorset conditions, and applied to the full-core problem. However, the approach employed in much of this year's analysis was to generate form functions starting from a full-core Serpent2 reference calculation [18]. Form functions in Griffin are defined

as [19, 20]:

$$f_{p,g} = \frac{\int_{r\in V_p} d^3r \int_{E_g}^{E_{g-1}} dE\, \kappa\Sigma_f(r,E)\phi(r,E)}{\kappa\Sigma_{f,m,g}\Phi_{p,g}}, \tag{1}$$

where p is the index of a pin within the reactor, g is the energy group index, and V_p refers to the volume of the volume p. The numerator represents the fission production reaction rate within pin p in the heterogeneous system, and these values are computed as Monte Carlo tallies from the Serpent2 calculation. On the other hand, the denominator represents the same integral reaction rate within the homogeneous system defined in Griffin. Since the homogeneous solution is typically computed in conjunction with equivalent methods, $\kappa\Sigma_{f,m,g}$ is the fission production cross section in group g defined over equivalence zone m on which the pin p is defined, while $\Phi_{p,g}$ is the homogeneous flux in energy group g integrated over the volume of pin p with equivalence correction applied.

Form functions are stored and read from the ISOXML library, and helper functions were created in FY23 to automate the process of form function loading and calculation, which is done in two steps. In the first step, ISOXML is called to populate the numerator of Eq. (1) by reading the output of the full-core Serpent2 simulation. This step assumes that heterogeneous fission production rates are computed in Serpent2 as a lattice detector tally for each pin and energy group over which form functions should be calculated for a given fuel assembly. In the second step, an equivalent homogeneous full-core problem is set up, and `PinPowerReconstructUserObject` is called to compute the denominator of Eq. (1) and update the existing data entries in ISOXML with the correct quotient for form functions. In order to simplify the workflow for running the entire workflow for full-core analysis with pin power reconstruction, this step can be done in the same input file that is responsible for computing assembly-level equivalence factors. For the remainder of the analysis that will follow, an SPH-based equivalence strategy is employed to preserve reaction rates between the heterogeneous and homogeneous systems.

In FY23, the ability to tabulate form functions according to an input tabulation grid has been added, similar to how equivalence data and cross-section data are tabulated. Through the ISOXML data processing capabilities, any tabulated form function data from Serpent2 or Griffin can be automatically populated in the form function library and used for statepoint calculations. This feature will be heavily used for the work in Section 4.2.2 to investigate how form function vary as a function of input state variables. In addition, linear interpolation of form functions data can also be carried out, and this feature is used in **??** in order

to run coupled feedback problems with pin power reconstruction, where form functions are not assumed constant as a function of input grid variables.

4.2.2 Investigation of Form Functions as a Function of Material Temperature and Control Drum Rotation Angle

By defining form functions with Eq. (1) and using SPH-based equivalence methods, it can shown that assembly-integrated and pin-by-pin power levels can be preserved between the homogeneous and heterogeneous systems. In practice however, storing form functions over all tabulation points can be a data intensive strategy, as form functions are computed as a function of pin location and energy group for all assemblies that contain fuel pins. Therefore, this section explores the dependence of form functions on the temperature and drum rotation angle tabulation variables, to see if a coarser tabulation grid for form functions can be used.

Form Function Dependence on Input Material Temperatures By computing form functions according to Eq. (1) and tabulating them as a function of material temperatures, the temperature-dependence of form functions for particular pin locations in the 2D Empire problem can be determined. Here, the temperature grid variables of interest are T_{fuel}, the average fuel pin temperature, $T_{mod;monolith}$, the lumped averaged moderator pin and monolith temperature, and $T_{refl;cd}$, the lumped average reflector and control drum temperature. For the purposes of this analysis, four representative fuel pin locations are chosen to compute groupwise form functions over, as shown in Figure 14. These pins are chosen from a fuel assembly located in the outer ring of fuel assemblies, and are located in the third, fifth, seventh, and ninth ring of pins in this assembly. These pin locations were chosen as they are in closest proximity to the reflector region where largest flux depressions - and potential changes to the form functions - are expected.

Figure 15(a) shows the value of the form function for each of these four pin locations and each of the six energy groups for the 2D Empire model, and these values are tabulated as a function of fuel pin temperature, while keeping the lumped moderator pin / monolith temperature and lumped reflector / control drum temperature fixed at 625K. Even with a large temperature variation ranging from 475K to 775K, the form functions for this tabulated variable exhibit relatively constant behavior. For each of these plots, multigroup cross sections and SPH factors are computed at each tabulation point independently, and the homogenized mesh shown in Figure 20 is used for all homogeneous Griffin calculations. Similar form function plots for

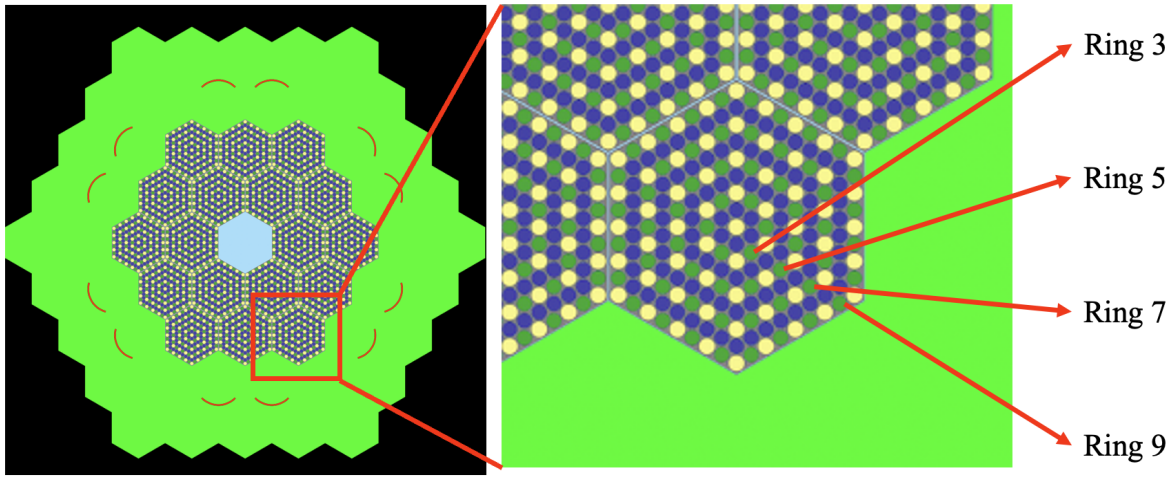
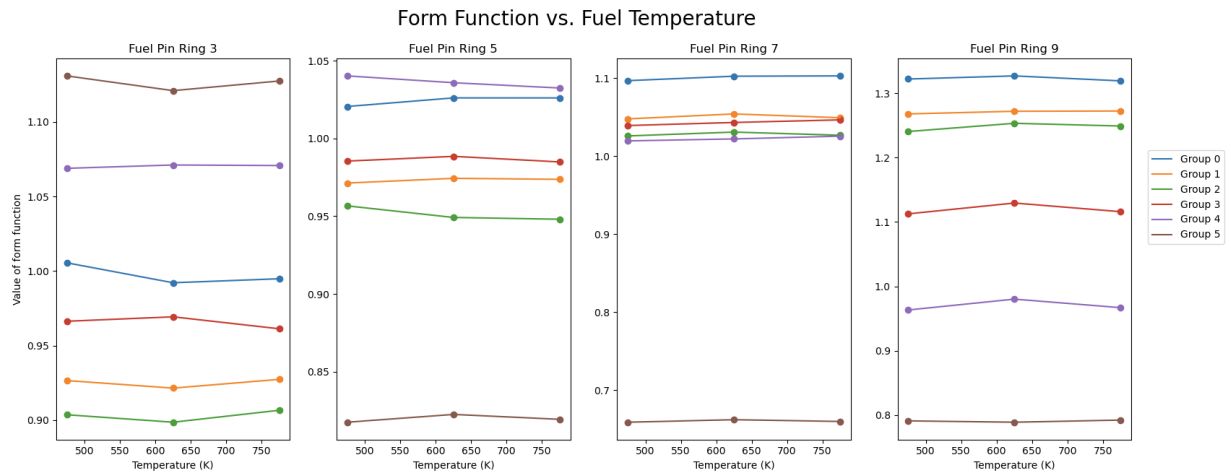


Figure 14: Diagram showing assembly location and pin locations where form functions are calculated for the analysis in Section 4.2.2. These four fuel pin locations (fuel pins are colored in green) will be referenced based on which ring of pins they belong to in the pin lattice of the fuel assembly.

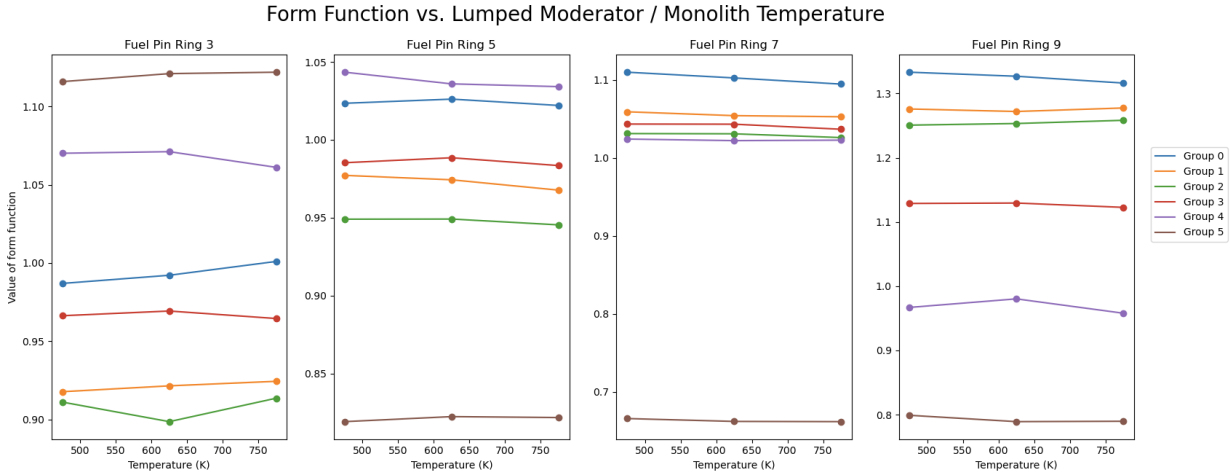
lumped moderator pin / monolith temperature and lumped reflector / control drum temperature can be seen in Figures 15(b) and 15(c), respectively, and neither of these temperature fields induce large changes to the computed form functions for the four representative pin location. While not shown here, all other fuel pin locations in the core exhibit similar trends with respect to the three temperature-dependent tabulation variables.

While Figure 15(c) shows that a non-constant reflector / control drum temperature dependence may be present, overall the plots in this section show a relatively constant behavior of form function with respect to input material temperatures, for a given pin location and energy group. In order to reduce the data requirements of storing form function data at all temperature-dependent tabulation points, it will be assumed that all temperature-dependent form functions can be determined from the form functions computed at a fixed tabulation grid with isothermal conditions of 625K throughout the entire reactor core. Later in this section, the errors in reconstructed pin powers will be shown from pursuing such a strategy of assuming that form functions from a fixed temperature grid point can be applied to all other temperature grid points.

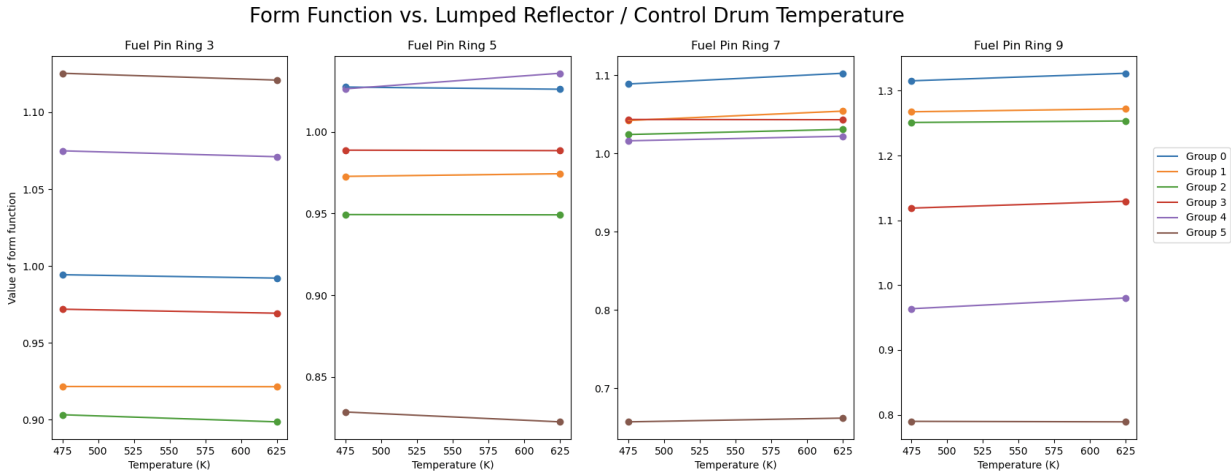
Form Function Dependence on Control Drum Rotation Angles Now, input material temperatures will be kept fixed at 625K, and the control drum rotation angle will be increased in 60 degree increments, where 0 degrees represents a completely outward orientation, while 180 degrees represents a completely inward orientation of control drums. Figure 16 shows what what the core layout looks like for each of these input



(a) Keeping lumped moderator pin / monolith temperature and lumped reflector / control drum temperatures at 625K



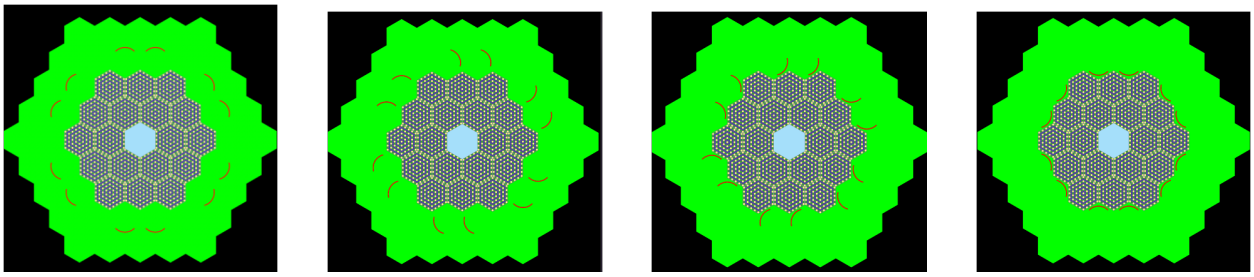
(b) Keeping fuel pin temperature and lumped reflector / control drum temperatures at 625K



(c) Keeping fuel pin temperature and lumped moderator pin / monolith temperatures at 625K

Figure 15: Variations of form functions as a function of temperatures for four pin locations shown in Figure 14.

angles, and the impact that this drum rotation has on the tabulated form function values is provided in Figure 17.



(a) (b) (c) (d)

Figure 16: 2D geometry of Empire core with a control drum rotation angle of (a) 0 degrees, (b) 60 degrees, (c) 120 degrees, and (d) 180 degrees.

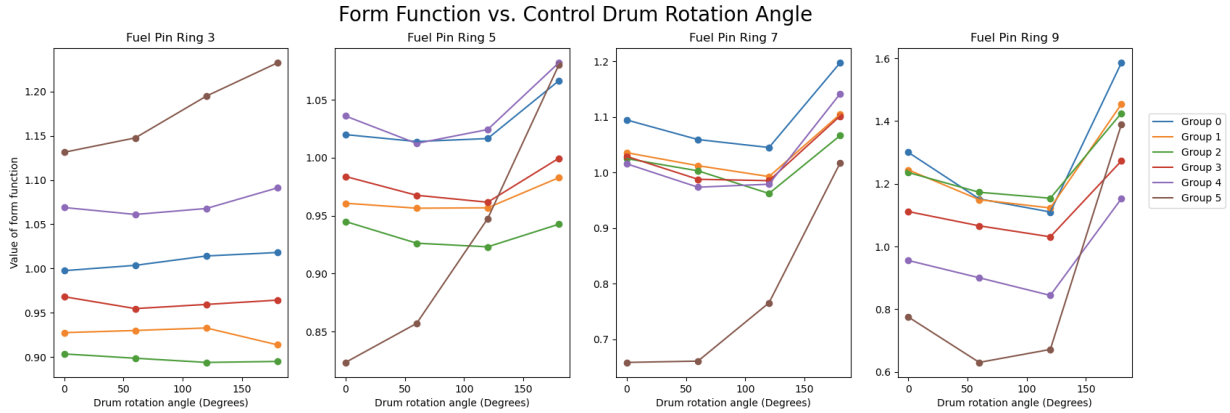


Figure 17: Value of form function as a function of tabulated control drum angle, for four pin locations shown in Figure 14. All material temperatures are kept constant at 625K.

It can be seen from Figure 17 that the control drum rotation angle has a much more pronounced effect on the form functions for the four pin locations shown in Figure 14, so form functions can no longer be assumed constant as a function of tabulated drum rotation angle. With this insight, the ability to apply a linear interpolation scheme for tabulated form functions was added to ISOXML in order to support the execution of transient simulations.

Steady-state Pin Power Reconstruction Results using a Fixed Temperature Form Function Tabulation Grid Now, we will look at the pin power reconstruction errors associated with applying form functions at a specific temperature-dependent grid point and applying them directly to all other grid points. In this analysis, form functions are calculated when the fuel pin temperatures, moderator pin / monolith temperatures, and

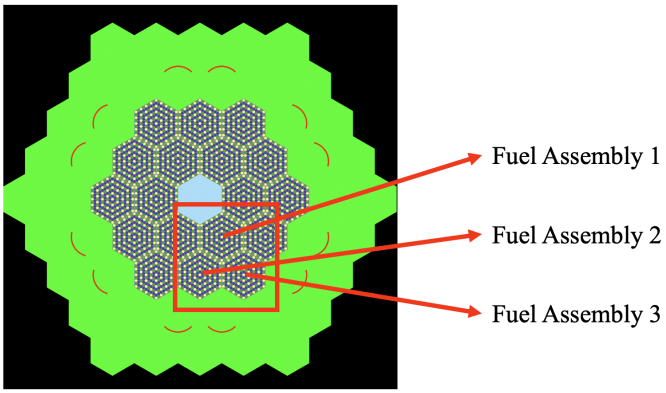


Figure 18: Diagram showing three fuel assemblies that are considered for error analysis shown in Table 4 and Figure 19. Pin powers are averaged over all sixth-core symmetric positions in Griffin and Serpent2 before computing RMS errors over these three assembly locations.

reflector / control drum temperatures are set to 625K. Three representative assembly locations, shown in Figure 18, are chosen to compute the pin power errors. These assemblies represent the three locations where all other fuel assemblies can be mapped to a sixth-core symmetric location, and all pin powers computed by Serpent2 and Griffin are averaged over their corresponding sixth-core symmetric positions. Table 4 shows the root-mean squared (RMS) error in pin powers computed for these three assemblies, as well as the entire core-averaged RMS error, where the pin powers from the heterogeneous Serpent2 calculation are used as the reference. These errors are shown for each of the temperature-dependent tabulation points of interest, and control drum rotation angles are kept fixed in an outward orientation. The variables T_{fuel} $T_{mod;monolith}$ and $T_{refl;cd}$ are used to represent the fuel pin temperature, lumped moderator pin / monolith temperature, and lumped reflector / control drum temperature respectively for a particular grid point.

The RMS errors in pin powers for these three representative assemblies range from 0.3% to 0.8% over the entire temperature tabulation grid, and the maximum power error for any single pin within this grid is 2.3%. In addition, Figure 19 shows the spatial distribution of pin power errors for the three representative assemblies shown in Figure 18. Here, the tabulation point where $T_{fuel} = T_{mod;monolith} = 775K$ and $T_{refl;cd} = 475K$ is chosen, since this tabulation grid point exhibit largest overall errors.

While additional tabulation points can be added to the computation of form functions to reduce overall reconstruction errors, the data storage requirements with such a strategy would be quite onerous, since form functions are tabulated as a function of energy group, pin location, and fuel assembly location. Thus, the slight increase in pin power error is justified given that form functions are only tabulated once for the entire

Table 4: Pin power errors at all temperature-dependent grid points, assuming constant form functions computed at a single grid point. Assembly IDs in table are based on assembly locations shown in Figure 18

Temperature Tabulation Grid (K)			RMS Error in Pin Power (%)			
T_{fuel}	$T_{mod;monolith}$	$T_{refl;cd}$	Assembly 1	Assembly 2	Assembly 3	Core-wide
475	475	475	0.3	0.3	0.4	0.4
475	475	625	0.4	0.5	0.4	0.5
475	625	475	0.3	0.5	0.6	0.5
475	625	625	0.3	0.4	0.4	0.4
475	775	475	0.3	0.7	0.7	0.6
475	775	625	0.3	0.5	0.4	0.4
625	475	475	0.3	0.3	0.4	0.4
625	475	625	0.3	0.5	0.6	0.5
625	625	475	0.3	0.5	0.5	0.5
625[1]	625[1]	625[1]	-	-	-	-
625	775	475	0.4	0.6	0.8	0.6
625	775	625	0.3	0.4	0.4	0.4
775	475	475	0.3	0.3	0.4	0.4
775	475	625	0.4	0.5	0.5	0.5
775	625	475	0.3	0.6	0.6	0.5
775	625	625	0.3	0.4	0.3	0.3
775	775	475	0.3	0.7	0.8	0.6
775	775	625	0.3	0.4	0.5	0.4

[1] Since form functions are computed at this grid point, reconstructed pin powers are calculated so as to match reference pin powers exactly.

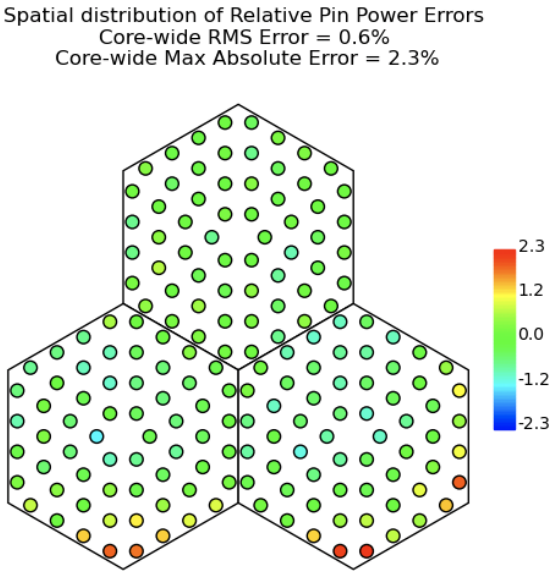


Figure 19: Spatial distribution of pin power errors for the three representative fuel assemblies shown in Figure 18. Results are shown for the tabulation grid point $T_{fuel} = 775K$, $T_{mod;monolith} = 775K$, and $T_{refl;cd} = 475K$. The core-wide RMS error in pin powers is 0.6% and the maximum absolute pin error for a single pin is 2.3%

temperature domain. It should be emphasized that the analysis in this section has been conducted for fixed tabulation grid points, where material temperatures and control drum angles were kept constant for the duration of the pin power reconstruction process. This was done primarily to assess the impact of assuming fixed temperature conditions for computing form functions on errors levels with pin power reconstruction over a larger domain of input material temperatures.

4.3 Implementing Two-way Transfer of Pin Powers with Multiphysics Solvers

In order to demonstrate a simple multiphysics coupling scheme using `PinPowerReconstructUserObject`, the MOOSE *Heat Conduction* module [21] will be used to set up a thermal conduction coupling problem. The MOOSE *Heat Conduction* module is responsible for solving the heat conduction equation as [21]:

$$\rho(t,\vec{x})c(t,\vec{x})\frac{dT(t,\vec{x})}{dt} = \nabla \cdot k(t,\vec{x})\nabla T(t,\vec{x}) + \dot{q}(t,\vec{x}), \tag{2}$$

where T represents the temperature, t is the time, and $\vec{x}$ represents the vector of spatial coordinates. Additionally, ρ is the density, c is the specific heat capacity, k is the thermal conductivity, and $\dot{q}$ is the heat source. In this case, the heat source is computed as the pin-by-pin power density from Griffin. By solving for the temperature over the spatial domain, assembly-averaged temperatures can be transferred back to Griffin to inform how to update cross sections, SPH factors, and form functions through the tabulated ISOXML data library. In addition, Griffin is the parent application that drives the multiphysics simulation and defines all forward and backward data transfers with the MOOSE *Heat Conduction* module, which is designated as the child application. For the purposes of this analysis, only the steady-state behavior of the thermal problem will be analyzed.

While pin power reconstruction is typically applied to a homogenized Griffin problem, the MOOSE *Heat Conduction* module will solve the heat conduction equation on a full-core pin-resolved mesh that takes all material heterogeneities into account. Figure 20 shows the full-core mesh used to solve the heat conduction equation, where the various material regions are explicitly meshed to solve for the temperature fields within each mesh element. The mesh discretization used for the problem can also be seen in Fig. 20, along with the homogeneous mesh used for the Griffin solve. The colorations shown in Fig. 20 represent subdomains that share the same IDs, and these IDs are used to assign the thermal properties for each region of the heterogeneous thermal problem and the material cross-section properties of the homogeneous neutronics

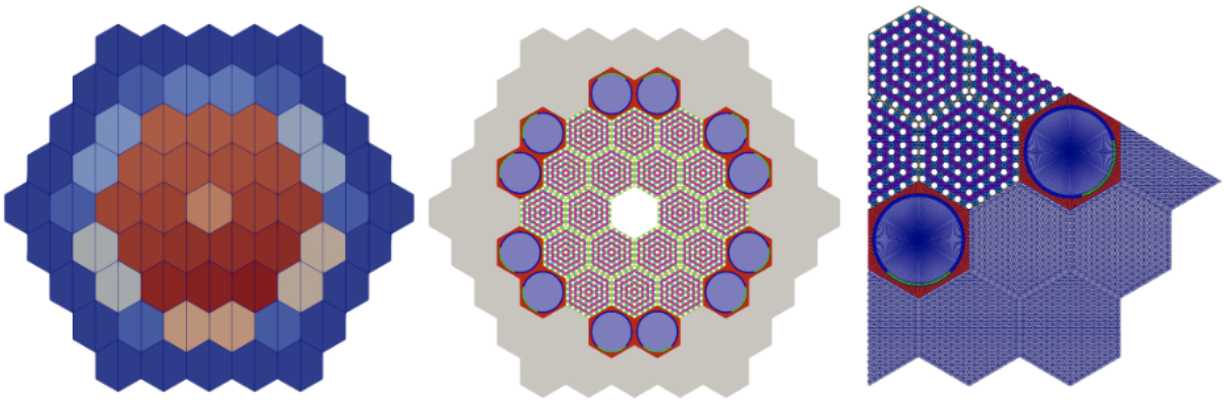


Figure 20: Full-core homogeneous 2D mesh of Empire core used to solve Griffin pin power reconstruction problem (left), full-core heterogeneous 2D mesh of Empire core used to solve heat conduction problem (middle), one-sixth core slice of heterogeneous mesh showing mesh discretization for the heat conduction problem (right).

problem.

The heat source in Eq. (2) is defined as the pin power density, which will come from the Griffin homogeneous solve, and `PinPowerReconstructUserObject` will inspect the heterogeneous mesh to automatically populate the pin power density values for each pin region, based on the specifications in the ISOXML form function library, which informs `PinPowerReconstructUserObject` where it should expect pin locations within each fuel assembly of the heterogeneous mesh. Therefore, the user must provide a heterogeneous mesh where the pin lattice specifications (pin radius, pin pitch, and number of rings in the pin lattice) are defined consistently with the pin mesh overlay that is used by `PinPowerReconstructUserObject` in the Griffin solve. Since the heterogeneous input mesh to the MOOSE *Heat Conduction* module and the homogeneous input mesh to Griffin both make use of the MOOSE *Reactor* module, however, the hierarchical nature of mesh definition reduces user burden with defining hexagonal assembly and pin lattices and reduces potential for user error, as many parameters and mesh generation steps are shared between the two input meshes.

Figure 21 shows the interaction between Griffin, ISOXML, the MOOSE *Reactor* module, and the MOOSE *Heat Conduction* module when running pin power reconstruction with thermal feedback. The general feedback mechanism here is to transfer pin-by-pin power densities computed using `PinPowerReconstructUserObject` from Griffin to the MOOSE *Heat Conduction* module. This pin power density is used to set the initial heat source in the heat conduction equation, which can then be used for solve for the tem-

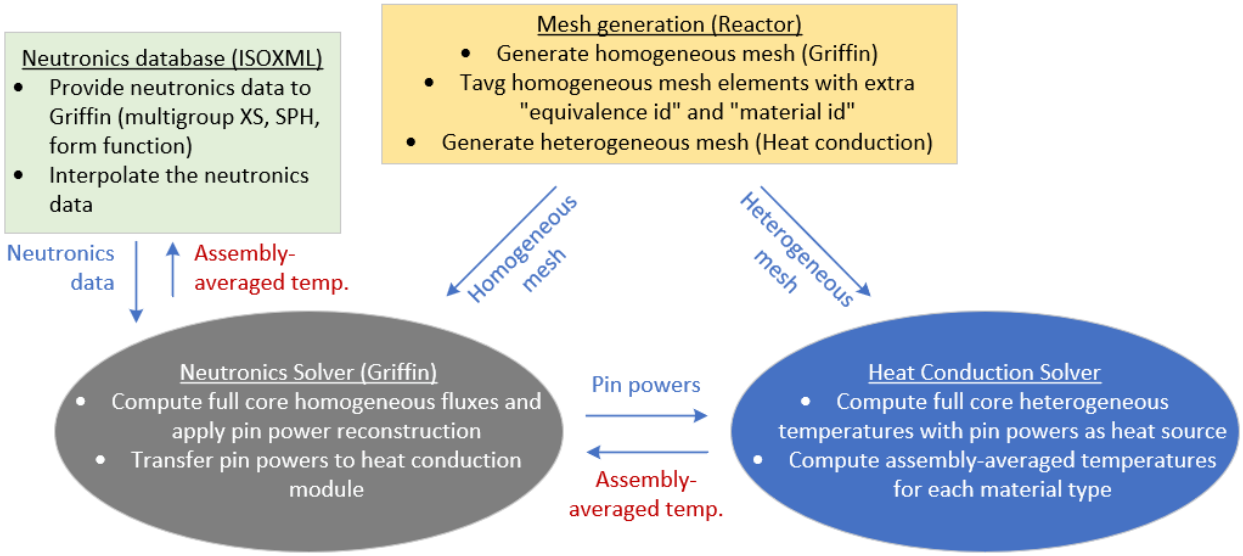
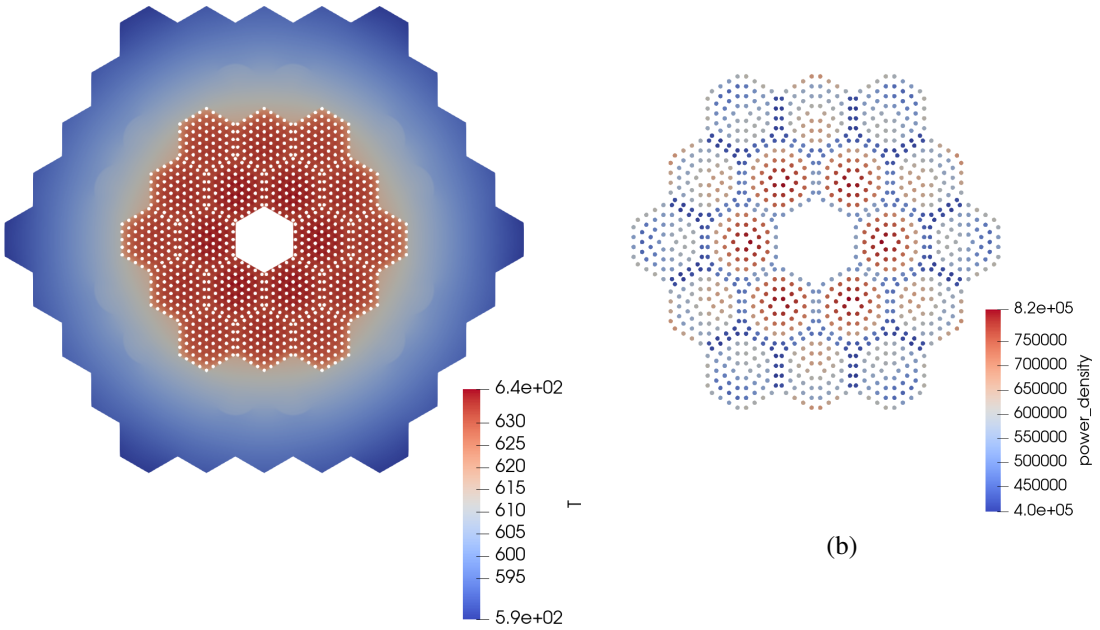


Figure 21: Workflow diagram showing coupling of Griffin with MOOSE Heat Conduction Module using `PinPowerReconstructUserObject`

perature distribution on the input heterogeneous mesh of the MOOSE *Heat Conduction* module. From this temperature distribution, assembly-averaged values for each material type are computed for each homogenized assembly regions, which are then transferred back to Griffin. For this problem, the temperature grid variables of interest are T_{fuel}, $T_{mod;monolith}$, and $T_{refl;cd}$. With the assembly averaged temperature fields transferred back to Griffin, cross-sections, SPH factors, and form functions are updated for each homogenized assembly, which in turn allows Griffin to solve for the new homogeneous flux solution and apply pin power reconstruction. Here, it is assumed that cross sections, SPH factors, and form functions are populated in ISOXML according to a grid of temperature fields, with linear interpolation applied at intermediate temperature points. This feedback loop between Griffin and the MOOSE *Heat Conduction* module is iterated on using a Picard iteration scheme until convergence is met.

On the Griffin side, `PinPowerReconstructUserObject` is responsible for computing the pin power densities by dividing the power in each pin region by the volume of the corresponding pin. Then, the heterogeneous mesh on the heat conduction side is inspected and any mesh element that fits within the volume of the pin mesh overlay is assigned the power density associated with the pin region. This transfer of data between Griffin and the MOOSE *Heat Conduction* module is done automatically by `PinPowerReconstructUserObject`, which sets the initial conditions necessary to solve the heat conduction equation.

Once the new temperature distribution is solved for, the MOOSE *Heat Conduction* module is responsible



(a) (b)

Figure 22: Converged map of (a) heterogeneous material temperatures and (b) pin power densities, assuming a pin power reconstruction workflow with thermal feedback through the MOOSE *Heat Conduction* module.

for computing assembly-averaged values of T_{fuel}, $T_{mod;monolith}$, and $T_{refl;cd}$ for each homogenized hexagonal assembly region. This is done by categorizing all mesh elements within an assembly as those that belong to the fuel pin regions, the moderator pin / monolith regions, or the reflector / control drum region. For each of these three mesh element groupings, the volume-averaged temperatures are computed and stored as T_{fuel}, $T_{mod;monolith}$, and $T_{refl;cd}$, respectively, and these are the temperature values that are transferred back to Griffin to inform how cross sections, SPH factors, and form functions are calculated for the subsequent Griffin solve. Here, the temperature of the moderator pins and monolith and the temperatures of the reflector and control drum were lumped into single temperature fields to reduce the number of tabulation variables needed to represent the temperature dependence of input neutronics data.

The final piece of the feedback mechanism that requires some further explanation is the manner in which input neutronics data is updated at the beginning of each Griffin solve. The MOOSE *Heat Conduction* module is responsible for computing T_{fuel}, $T_{mod;monolith}$, and $T_{refl;cd}$ for each homogenized assembly region m, and we assume that cross sections and SPH factors are dependent on these input temperature fields as $\Sigma_{m,g}(T_{fuel}, T_{mod;monolith}, T_{refl;cd})$, and $\mu_{m,g}(T_{fuel}, T_{mod;monolith}, T_{refl;cd})$, respectively, where m is the equivalence zone of interest. As was shown above, form functions do not have a strong dependence on input

material temperatures, so we continue to assume constant form functions for the coupled thermal problem. As cross-sections and SPH factors are tabulated on a grid of temperatures that depends on T_{fuel}, $T_{mod;monolith}$, and $T_{refl;cd}$, a simple linear interpolation is used to compute this data at intermediate temperatures. More specifically, Griffin passes the input temperature fields to ISOXML, and ISOXML is responsible for interpolating and providing the input neutronics data to Griffin for use in the subsequent solve.

Using the workflow and implementation details described in this section, the pin power density and core temperatures are iterated on until convergence is met using a Picard iteration scheme. The first solve occurs in Griffin, which assumes isothermal conditions of 625K everywhere in the core. The final converged temperature distribution and pin power density map is shown in Figs. 22a and 22b, respectively.

4.4 Improvements in HFEM Solver

In FY23, significant efforts were directed towards refining the hybrid finite element method (HFEM) solver option in Griffin, aiming to boost its performance and broaden its capacity for multi-physics coupling.

4.4.1 Performance Improvement

In the profiling of HFEM-PN solver performance, we observed the substantial overhead of residual evaluations in the Richardson iterations, which took about 50 % of total computing times. The most of the residual evaluations were consume by computing the Lagrange multiplier kernels and the odd parity flux moments. Therefore, the optimization of residual evaluation was focused on these two components of the HFEM solver. The Lagrange multiplier kernels evaluate the following components of the HFEM formulation:

$$(\Psi_o^*(\vec{\Omega}), \vec{\Omega}\cdot\vec{n}\,[\Psi_e(\vec{\Omega})])_{\Gamma\otimes\mathscr{S}} + ([\Psi_e^*(\Omega)], \vec{\Omega}\cdot\vec{n}\,\Psi_o(\vec{\Omega}))_{\Gamma}, \tag{3}$$

where the operators $(f,g)_\Gamma$ are the inner products of two generic variables f and g on internal surfaces, and boundary, respectively, and $[f]$ represents the jump of a variable f on an internal surface. Here, it can be noted that the Lagrange multiplier kernels do not depend on the material properties on elements. Thus, their residual contribution can be evaluated without using material evaluations. The residual evaluation for the Lagrange multiplier kernels action on the element sides was optimized by directly assembling the residual contributions, circumventing unnecessary material property evaluations within the residual evaluation

function of the MOOSE framework. Additionally, the element-wise matrices required for evaluating the Lagrange multipliers are pre-computed and stored. By making use of these pre-computed matrices, the quadrature point integrals can be bypassed in the residual evaluation.

The odd-parity moment evaluation is required to properly treat the anisotropic scattering sources in the HFEM formulation. Previously, odd-parity moments were computed for each quadrature points of the volumetric element since they were stored as material properties. The computational cost of their evaluation is especially expensive because odd-parity moments are coupled with the entire PN orders and energy groups. To alleviate the computational burden in computing odd-parity moments, these moments are currently stored using the *aux variables*, and they are directly assembled utilizing element-wise matrices and element-averaged cross sections. This update significantly reduces the required floating point operations by skipping the quadrature point sweeping, and the use of element-averaged cross sections does not degrade solution accuracy.

Along with the optimizations in the residual evaluations, the implementation of RB iterations was also elaborated. First, Efficient caching of degrees of freedom (DoF) maps was implemented to optimize the access of MOOSE solution data during RB iterations. The anisotropic scattering evaluation in the Richardson iterations was optimized as well. Previously, the anisotopric scattering sources were updated in both the outer Richardson iterations and the inner RB iterations. It was found that the update in inner iterations is not necessary because the anisotopric scattering can be effectively handled by Richardson iterations. The update of the anisotropic scattering source is now only performed in the Richardson iterations, which allows to skip evaluations of anisotropic scattering source terms during the RB iteration level. For demonstrating the performance improvement attained from these optimization, the 2D ABTR benchmark problems with 33-group cross sections were solved. As shown in Table 5, the computational times were compared with the results obtained with the FY22 version of HFEM solver without these optimizations. It was confirmed that more than a three-fold of performance improvement was obtained with the optimizations.

Table 5: Performance Improvements in HFEM-PN Solver with Optimizations

Angular Order		Spatial Order		# DoFs	Computing Times, sec	
PN	Scattering	Volume	Side		w/o Optimiation	w/ Optimization
3	3	3	1	0.98M	70.9	19.8
5	3	3	3	2.20M	191.3	34.1

The CMFD acceleration scheme was implemented in the HFEM solver options in order to accelerate

the convergence of HFEM solvers. Since the weak-form of the HFEM formulation provides conservative discretization, the solution of HFEM-PN solver exactly matches the neutron balance for each element. Consequently, the conventional CMFD acceleration scheme can be readily applied for the HFEM solvers for diffusion and P_N equations. Building upon the existing implementation for the DFEM-SN solver, CMFD functionalities have been extended to accommodate HFEM solvers. In order to configure the CMFD systems, coarse mesh structures and currents on the coarse mesh interfaces are required. The unstructured coarse mesh capability of DFEM-SN was adopted for the CMFD acceleration in the HFEM-solver. The current information can be directly obtained from the variable for Lagrange multiplier kernels, as the P_0 components of the variable correspond to the net current. The Richardson executioner of HFEM solvers was updated to incorporate the CMFD acceleration. Prior to performing each Richardson iteration, the CMFD calculation is performed, and the updated solutions are prolongated into the PN flux moment variables. The performance examinations were performed for the 2D and 3D ABTR benchmark problems with several energy group structures. In the 3D ABTR case, about 60% reduction in computational time was observed by effectively reducing the total Richardson iterations, as shown in Fig. 23.

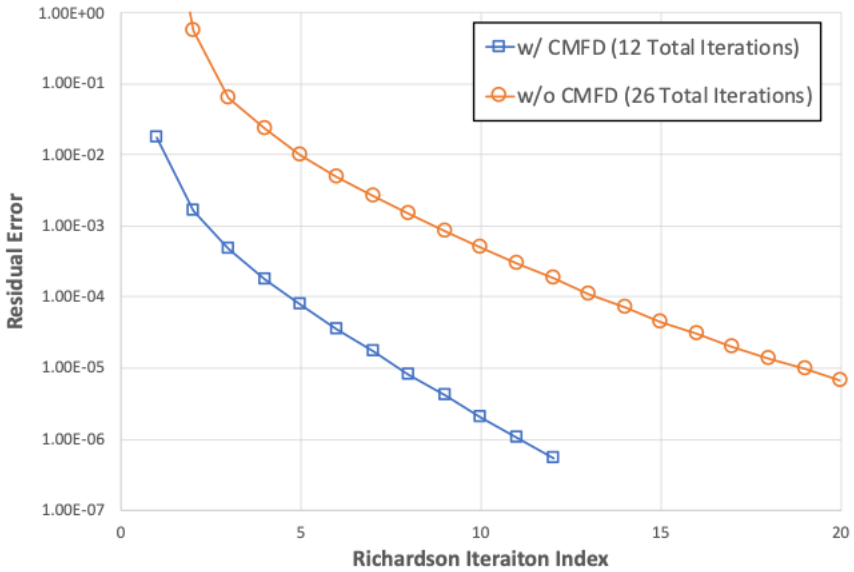


Figure 23: Residual Convergence of HFEM-PN solver with and without applying CMFD Acceleration for 3D ABTR Benchmark Problem.

The RB object was updated to construct response and element-wise matrices only for selected element types, referred to unique elements, possessing unique cross-section values and element shapes among the entire elements and energy groups. By introducing the unique element concept, redundant matrix storage is

eliminated for response matrix constructions and operations. Consequently, this optimization substantially decreases the memory footprint during RB iterations without sacrificing performance. As shown in Table 6, the memory optimizations significantly reduced the memory requirements for running the realistic core cases using the HFEM-PN solver. It was also confirmed that these optimizations worked properly without altering the convergence and solution.

Table 6: Memory Reuction of HFEM-PN for 3D ABTR Benchmark Problems (16 Processors)

PN Order	Spatial Order		DoFs	Memory, GB	
	Volume	Side		w/o Opt.	w/ Opt.
3	3	1	3.87 M	108.7	27.7
	4	1	5.76 M	159.1	42.0
	5	2	9.76 M	OOM[1]	81.3
5	3	1	9.68 M	OOM[1]	71.8
	5	2	24.4 M	OOM[1]	249.8

[1] OOM: out of memory

4.4.2 Performance Examinations

The overall performance of HFEM-PN solver was investigated by solving the 3D ABTR benchmark problems with the pre-generated 33G cross-section sets prepared by MC^2-3. In the ABTR model, the individual hexagonal assemblies were represented using two quadrilateral elements. Axially, a total 21 planes, roughly 20 cm height, were used to descretized the active fuel and other structure regions. The HFEM-PN calculations were conducted using various PN order and spatial basis function orders, incorporating P3 anisotropic scattering treatments. In addition, HFEM-Diffusion calculations were carried out with isotropic scattering treatment. The obtained eigenvalue and computing time results with the different calculation settings are summarized in Table 7. For comparing the accuracy and performance, the reference calculations were performed using the DFEM-SN solver with different spatial orders and mesh discretization options as summarized in Table 8. The both HFEM and DFEM-SN calculatiosn were performed using the 32 processors, and the CMFD acceleration scheme was applied. In the accuracy comparison purpose, the eigenvalue obtained with the most refined option, 1.05568, was used. From the comparisons of performance with various calculation setting and the DFEM-SN results, we can obtain the following observations:

- When using the diffusion equation, the eigenvalue discrepancy was observed about 1700 pcm. With the P3 equation, the discrepancy is reduced to about 200 pcm level, which suggests that the P_3 equation

can capture the most of transport effect. When increasing the PN order to 5, the eigenvalues converges to the reference values with about 50 pcm difference. Thus, users can utilize the P3 calculation option for the typical design calculation. The P5 calculation can be occasionally conducted for confirmatory calculation purpose.

- In the P3 and P5 case, the eigenvalue obtained with the third and the first order monomial for volume and side elements are essentially same with the the results with the fifth and second order monomials. For the ABTR benchmark calculation, this basis function setting is sufficient to yield the accurate eigenvalue results. Interestingly, the increasing the basis function order for volume only is worsen the eigenvalue accuracy, which suggests that we need to increase both order of basis functions for volume and side.

- The residual evaluation in the HFEM-PN calculation took significant portion of total computing time ranging 30% to 70%. To further improve the overall performance, we need to reduce the residual evaluation by introducing the optimization employing the average cross section which allows to by-pass the time consuming the quadrature integrals involving high order basis functions.

- The overall performance of HFEM-PN is generally faster than the DFEM-SN more than factor of two or three with the acceptable accuracy, especially with P3 option.

The radial power distributions of HFEM-PN solution was compared against the reference distributions from the DFEM-SN calculation. As shown in Fig. 24, the assembly-wise power distributions are matched well less than 0.3% of relative differences. The parallel efficiency of HFEM-PN solver was also examined by performing the 3D ABTR problems with the different number of processors. The obtained computing times for the HFEM-PN calculations are shown in Table 9. As using the more number of processors, the overall computing times were monotonically reduced as expected. Especially, the computing portion of residual evaluation shows good parallel scaling behavior. However, the parallel efficiency of the RB iteration is notably degraded as using more than 16 processors, which suggests that we need to further investigate its parallel performance.

Table 7: Performance Examinations of HFEM-PN Solver on 33G 3D ABTR Benchmark Problems (P_3 Anisotropic Scattering for P3 and P5 Cases, Two QUAD Elements for Hexagonal Assembly, 32 Processors)

PN Order	Spatial Order		# DoFs	Eigenvalue	Computing Time, Sec			
	Volume	Side			Residual	Red-Black	CMFD	Total
Diffusion	3	1	6.45M	1.03628	29.1 (42%)	10.9 (16%)	24.1 (35%)	68.5
	4	1	9.61M	1.03701	107.3 (64%)	12.3 (7%)	38.2 (23%)	167.1
3	3	1	38.7M	1.05290	77.9 (43%)	63.4 (35%)	29.8 (16%)	182.2
	4	1	57.6M	1.05450	202.9 (62%)	72.6 (22%)	40.8 (13%)	326.4
	5	1	84.1M	1.05521	420.0 (70%)	96.6 (16%)	72.8 (12%)	598.4
	5	2	97.6M	1.05307	465.1 (58%)	252.9 (32%)	75.7 (9%)	799.7
5	3	1	96.8M	1.05494	107.4 (32%)	182.0 (54%)	32.2 (10%)	334.1
	4	1	144M	1.05700	284.2 (50%)	221.6 (39%)	51.6 (9%)	564.9
	5	1	210M	1.05830	566.4 (61%)	264.2 (29%)	72.9 (8%)	921.0
	5	2	244M	1.05510	592.2 (45%)	631.2 (48%)	72.7 (6%)	1313.2

Table 8: Reference DFEM-SN Calculations on 33G 3D ABTR Benchmark Problems (P3 Anisotropic Scattering, 32 Processors)

Angular Quadrature	Mesh	Spatial Order	# DOFs	Eigenvalue	Computing Time, sec
S4 (24 Directions)	2 QUAD / Hex. Assembly	Lag 1	85.7 M	1.05330	516
		Lag 2	239 M	1.05565	2387
	6 TRI / Hex. Assembly	Lag 1	192 M	1.05484	1219
		Lag 2	578 M	1.05568	3363

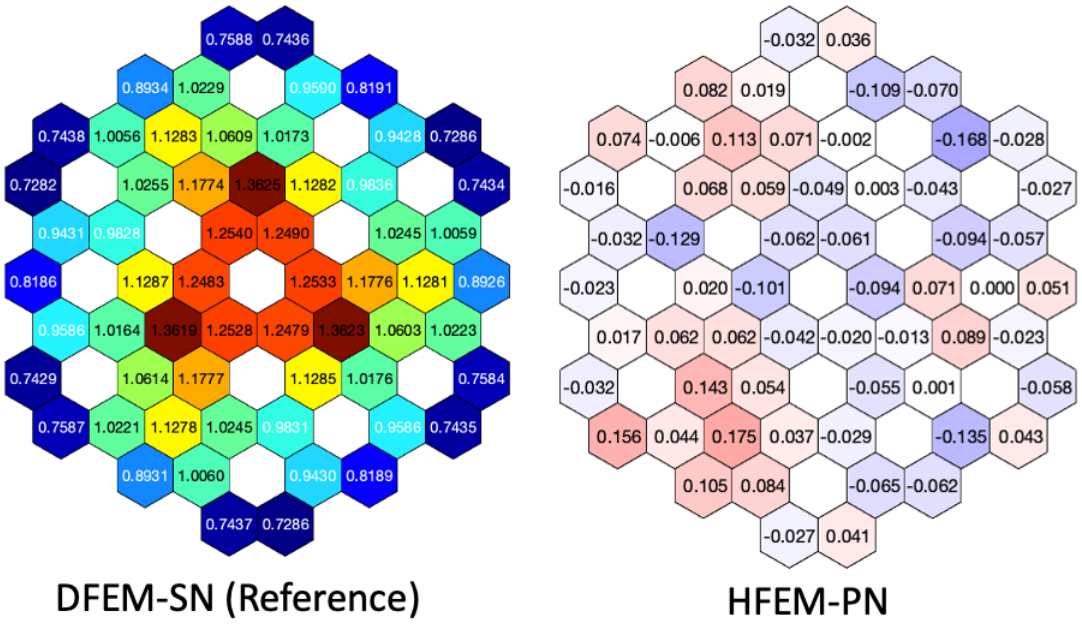


Figure 24: Comparisons of Power Distribution between DFEM-SN and HFEM-PN Solutions for 3D ABTR Benchmark Problem (Left: Reference DFEM-SN Power Distribution, Right: Relative Error of HFEM-PN Power Distribution with P3 options)

Table 9: Parallel Performance of HFEM-PN Solver on 33G 3D ABTR Benchmark Problems

# Processor	Computing Time, sec			
	Total	Residual	Red-Black	CMFD
4	1523.7	1154.4	186.2	158.5
8	795.4	597.7	106.3	88.0
16	505.3	318.6	75.6	53.2
32	326.4	202.9	72.6	40.8
64	225.9	112.5	67.0	39.4

4.4.3 Multi-physics Coupling

The multi-physics support of HFEM solver was implemented in the framework of Richardson iteration with CMFD acceleration. The HFEM Richardson executioner was refactored to properly support the Picard iterations among multi-physics apps.

The RB iteration object was also updated to re-evaluate the response matrices to account for cross-section changes from the multi-physics simulations. The multi-physics implementations was verified by comparing the test case solutions obtained by using the DFEM-SN and the HFEM-PN solvers. A heat conduction problem along with Griffin calculation was solved in a test case to model the thermal feedback in the fuel region. Comparing the obtained solutions, the temperature and power distributions were essential identical showing less than 1% differences.

5. SFR Benchmark Core Calculation Results

For the verification tests, two fast reactor benchmark problems, ABTR and ABR-1000, were selected, as illustrated in Fig. 25. The ABTR is a small-size fast reactor core generating 250 MWth of power. The inner and outer cores are composed of 24 and 30 driver fuel assemblies, respectively, and 6 fuel test assemblies (total 60 fuel assemblies). There are nine test assembly locations, among which three assemblies located in the fourth ring are designated for material test assemblies, and six test assemblies located in the third and fourth rings are designated for fuel test assemblies.

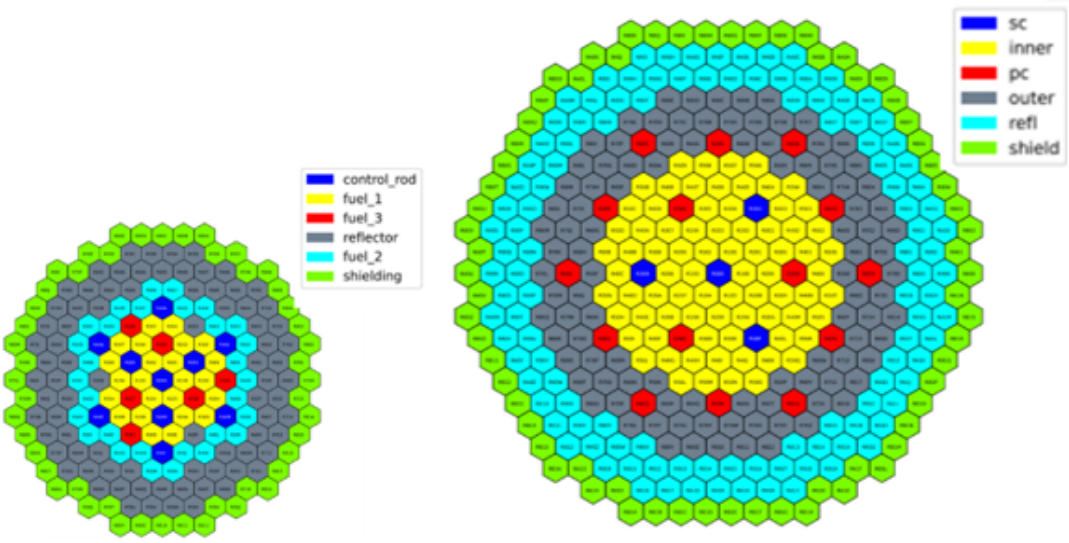


Figure 25: ABTR and ABR-1000 core benchmarks

The ABR core is a medium-size metallic-fueled SFR concept designed by ANL as part of the Global Nuclear Energy Partnership program. Its specifications was developed under the OECD/NEA subgroup on the Uncertainty Analysis in Modeling (UAM) of SFRs. The ABR core generates 1,000 MWth (400 MWe) composed of 180 fuel assemblies, 114 radial reflectors, 66 radial shields, and 19 control assemblies. It is known for its compact design with a conversion ratio of 0.7 that provides a one-year cycle length with a 90% capacity factor. Conventional core components, such as U-Pu-10Zr metallic fuel alloy with HT-9 cladding and HT-9 structural components, were used in the ABR design. Fig. 25 illustrates the 2D layouts of the ABTR and ABR cores.

As reference solution calculations, Serpent2 was run for the two benchmark cores with fully heterogeneous geometry configurations for all assemblies including reflector and shield regions. Eigenvalue, assem-

bly power and pin power solutions were tallied using a total of 300 million active particles such that the 1σ uncertainty of eigenvalue is less than 5 pcm.

In the deterministic simulation conducted with DIF3D and Griffin, each assembly was homogenized, whose heterogeneity effect was implicitly accounted for in the cross-section generation. In the DIF3D calculations, a single hexagonal element per assembly was used with high-order spatial approximations (i.e., 6th-order flux and source and 1st-order leakage approximations), while in Griffin the MOOSE meshing tool [10] was used to generate meshes with 6 triangles per assembly and either 1st-order or 3rd-order spatial approximations. Axially total 34 and 53 axial meshes were specified for ABTR (a height of 260 cm) and ABR (a height of 480.2 cm) cores, respectively, in both DIF3D and Griffin calculations.

BG (33 groups) cross sections were generated using MC^2-3 with the aforementioned two-step procedure where 1D calculations were performed for fuel and control assemblies only and homogeneous mixture calculations were conducted for all other assemblies including reflector and shield assemblies. It was noticed that the heterogeneity effect of reflector and shield assemblies is as small as 60 pcm or less for both cores. Note that the ENDF/B-VII.0-based cross sections were used for all calculations in this study.

In the DIF3D calculations, P3 transport and P3 anisotropic scattering approximations were used, whereas in Griffin DFEM-SN with 48 angles and P3 anisotropic approximation was employed. According to the mesh and angle refinement study, DIF3D and Griffin solutions were nearly converged in terms of mesh and transport order.

The DIF3D and Griffin solutions were compared with the corresponding Serpent2 solutions in terms of k-effective, control rod (CR) worth, and assembly power and pin power distributions. Gamma transport calculations were not performed and included in this study. Regarding the CR worth, all CR assemblies, including the primary and secondary control systems, were fully inserted into the core.

Table 10: Eigenvalue and Control Rod Worth from Serpent2, DIF3D, and Griffin.

Reactor	Parameter	Serpent2 [1]	Difference [2]		
			DIF3D	Griffin DFEM-SN	Griffin HFEN-PN
ABTR	k-eff	1.08188	-32 pcm	-85 pcm	-135 pcm
	CR worth	13,054 pcm	-1.21%	0.31%	-0.69%
ABR-1000	k-eff	1.01707	211 pcm	-2 pcm	-11 pcm
	CR worth	9,811 pcm	-0.45%	1.35%	0.46%

[1] standard deviation of Serpent2 solutions less than 10 pcm
[2] difference from Serpent2

As shown in Table 10, the eigenvalue differences between DIF3D and Griffin are about 100 pcm for ABTR and about 200 pcm for ABR. The differences from Serpent2 solutions are -32 pcm and -85 pcm for DIF3D and Griffin (DFEM-SN), respectively, for ABTR, while they are 211 pcm and -2 pcm for DIF3D and Griffin (DFEM-SN), respectively, for ABR. The CR worths from the two deterministic codes are in very good agreement with the Serpent2 solutions within 1.3% for the two cores, as shown in Table 10. In principle, both DIF3D and Griffin should theoretically converge to similar solutions, given the same cross-sections and similar angle and spatial approximations. However, the relatively large k-effective difference between the codes for ABR requires further investigation.

Assembly and pin powers between DIF3D or Griffin and Serpent2 were compared as presented in Table 11. As aforementioned, DIF3D used the ARC tool [8] for pin power reconstruction, while Griffin employed its built-in pin power reconstruction capability [9] discussed in Section 4. For ABTR, RMS and maximum assembly power differences are 0.58% and 0.93% between DIF3D and Serpent2, and 0.37% and 0.72% between Griffin and Serpent2, respectively. RMS and maximum pin power differences are 0.93% and 4.72% between DIF3D and Serpent2, and 0.75% and 4.98% between Griffin and Serpent2, respectively. It is worth noting that the assembly and pin power differences of DIF3D are slightly larger than those of Griffin. It is also noticed that the assembly with maximum assembly power difference differs from the one with maximum pin power difference, but both of these assemblies are located adjacent to the reflector region, as illustrated in Fig. 26. For the ABTR core, it is noteworthy that the assembly and pin power difference distributions are similar between DIF3D and Griffin. We also observed that the axial power differences for ABTR support the similarity between the two code solutions as well.

Table 11: Assembly and pin powers from Serpent2, DIF3D, and Griffin.

Reactor	Power	% Difference [1]					
		DIF3D		Griffin DFEM-SN		Griffin HFEM-PN	
		RMS	Max	RMS	Max	RMS	Max
ABTR	Assembly	0.58	0.93	0.37	0.72	0.44	0.66
	Pin	0.93	4.72	0.75	4.98	0.73	5.01
ABR-1000	Assembly	1.38	2.19	0.85	1.65	0.90	1.86
	Pin	1.46	5.40	1.12	6.82	1.15	6.34

[1] % difference from Serpent2 solutions

For ABR, the differences in assembly and pin powers between DIF3D or Griffin and Serpent2 are larger than those observed for ABTR. RMS and maximum assembly power differences are 1.38% and 2.19% for

DIF3D and 0.85% and 1.65% for Griffin, respectively. RMS and maximum pin power differences are 1.46% and 5.40% for DIF3D and 1.12% and 6.82% for Griffin, respectively. An interesting point to note is that even though the RMS pin power differences of DIF3D are larger than those of Griffin, the maximum pin power differences of DIF3D are smaller. This may indicate that the pin power reconstruction of DIF3D seems to overall perform better than that of Griffin, but further detailed investigation is needed to support this conclusion. In contrast to ABTR where the assembly power peak occurs at the 2nd ring of the core, it is located at the 6th ring of the ABR core because its active core size is radially about 2 times larger than ABTR. Similar to ABTR, the assembly with maximum pin power error is located adjacent to the reflector region in ABR, as shown in Fig. 27.

Peak pin powers were also examined for the cores, which would be of interest in safety calculations. It was noticed that the differences of peak pin powers between DIF3D or Griffin and Serpent2 are less than 0.7% (relative power 1.40) and 0.4% (relative power 1.27) for ABTR and ABR, respectively. The ABR axial powers between Griffin and Serpent2 overall agree well with those from Serpent2. However, Griffin axial power differences are up to 2% near the top of the fuel region, while DIF3D ones are less than 0.6%, compared to the Serpent2 solutions. The axial powers are slightly bottom-skewed because of the high neutron leakage to the region above the fuel where the gas plenum is located.

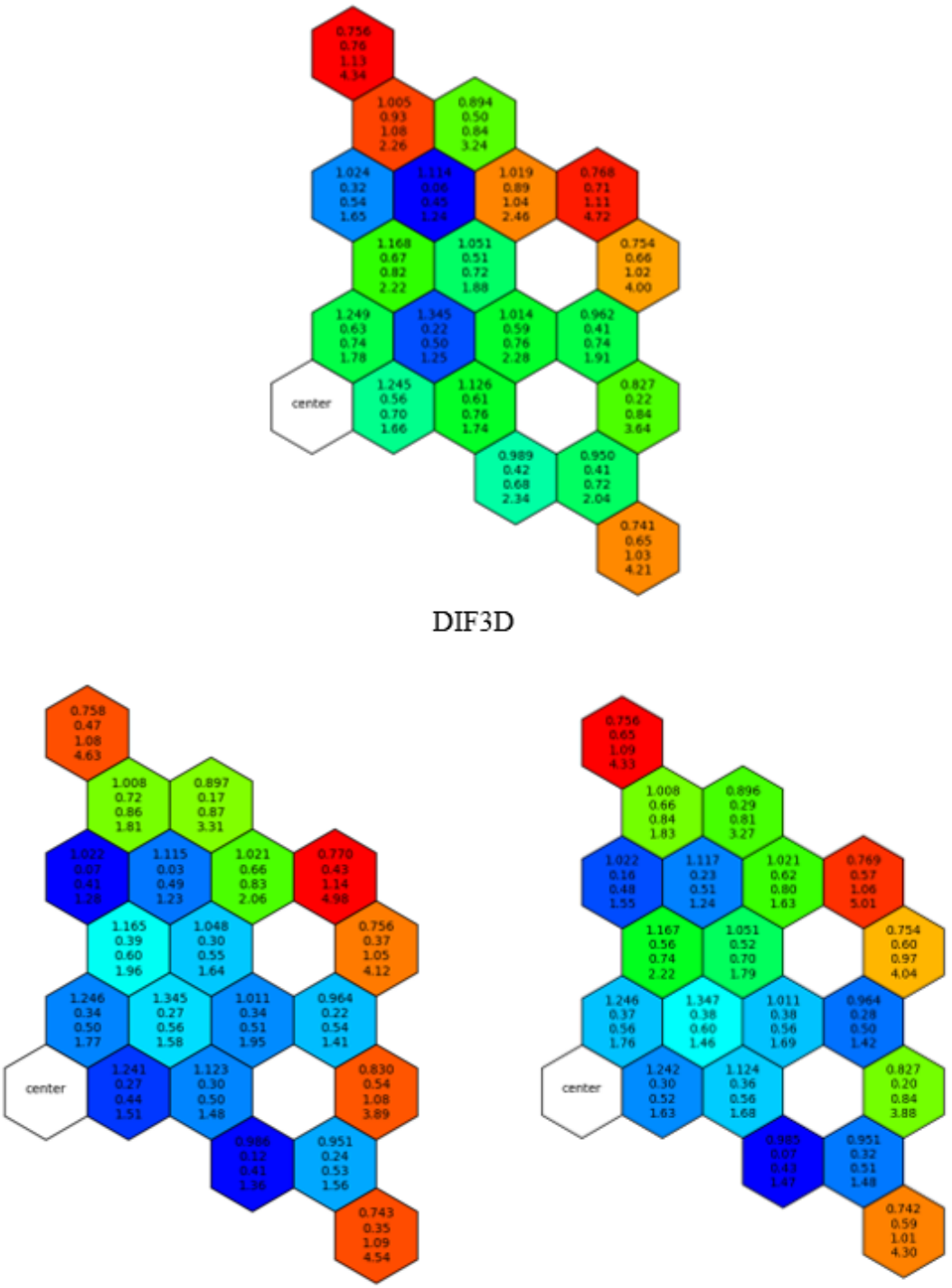


Figure 26: Assembly and pin powers from DIF3D, Griffin, and Serpent2 for ABTR

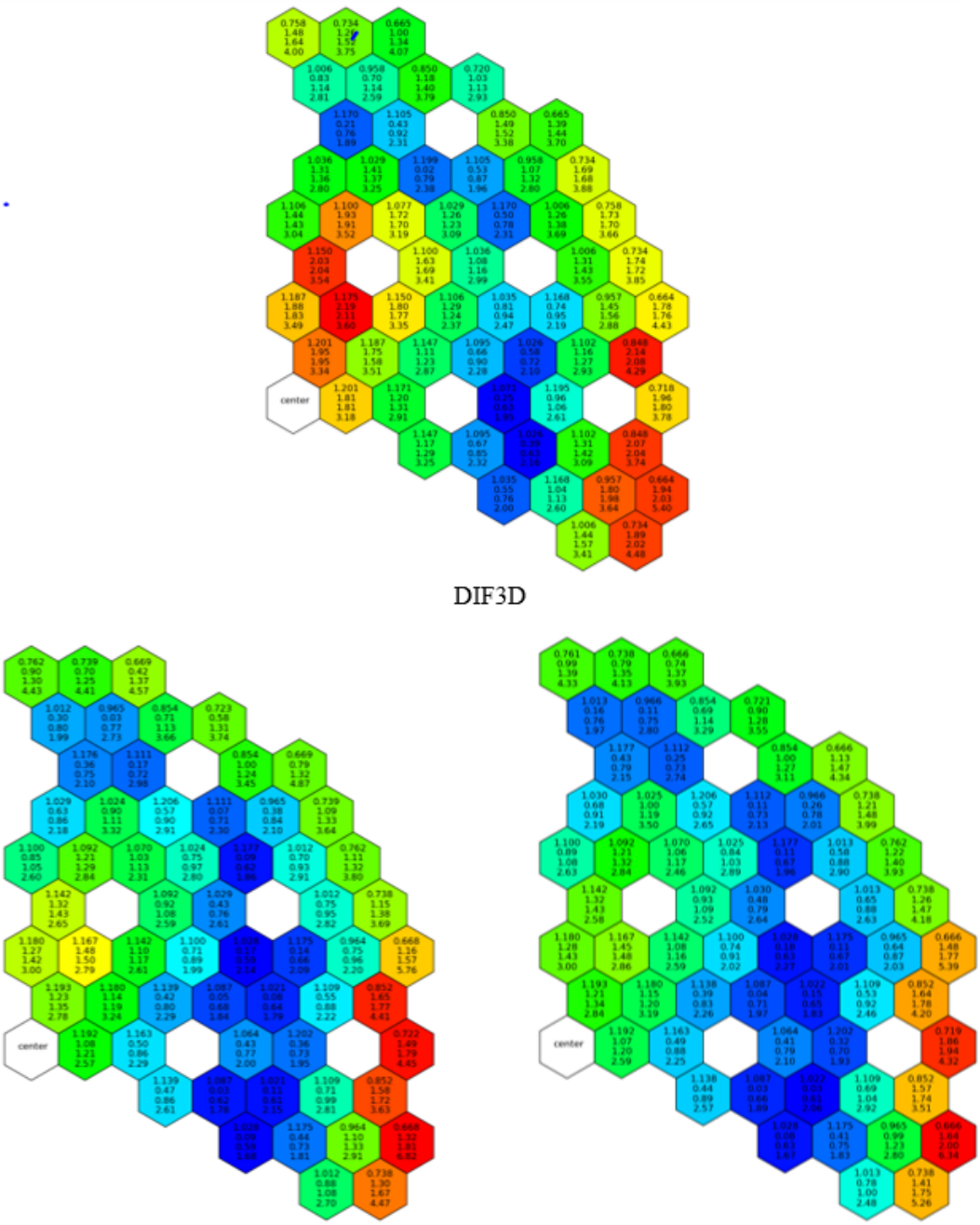


Figure 27: Assembly and pin powers from DIF3D, Griffin, and Serpent2 for ABR-1000

6. Conclusions and Future Work

In this fiscal year, we enhanced Griffin's capabilities for fast reactor analysis. This primarily involved improving the cross-section generation workflow using MC^2-3 for various reactor configurations, including homogeneous, duct-heterogeneous, ring-heterogeneous, and fully-heterogeneous geometries. Additionally, we began implementing a multi-cycle depletion and shuffling capability. While the main structure of this capability was designed this year, its full completion is scheduled for the next fiscal year.

Furthermore, to enhance our capabilities for fast reactor modeling and simulation, we've implemented several improvements in HFEM-PN, pin power reconstruction, and an R-Z solver with DFEM-SN. First, we optimized the R-Z solver to improve its performance, particularly for ultrafine group (over 1,000 groups) transport calculations, which are crucial for cross-section generation with MC^2-3. Second, we enhanced the performance of HFEM-PN by introducing red-black iteration, the coarse-mesh finite difference (CMFD) acceleration technique, and additional optimizations. Consequently, the HFEM-PN solver now surpasses DFEM-SN in performance, making it the primary choice for fast reactor calculations, especially when dealing with cores featuring homogenized assemblies. Third, accurate pin power reconstruction within each assembly is essential after solving for a core with homogenized assemblies. While this capability was initially implemented in the previous fiscal year, it has been further refined to support multiphysics simulations in this fiscal year.

The enhanced capabilities for fast reactor core simulations underwent individual testing and were applied to benchmark problems involving sodium-cooled fast reactors (SFRs), specifically ABTR and ABR-1000. Verification tests utilized HFEM-PN and the pin power reconstruction feature, whose results were compared to DIF3D along with the pin power reconstruction tool in the ARC suite, and Serpent2 Monte Carlo solutions.

As future work, we plan to conduct further tests using actual core problems that involve various heterogeneous geometry configurations in the fast reactor cross-section generation workflow. While we have streamlined the cross-section generation process, there are still a few steps and input options for users to select. Therefore, a further extension of a GUI tool Workbench/PyGriffin will be necessary to assist users in preparing inputs and providing visualization of inputs and outputs. As previously mentioned, the majority of the fast reactor fuel cycle capability will be completed in the next fiscal year. In the case of HFEM-PN, our remaining work involves implementing transient capability, which is scheduled to be completed next

year. Regarding pin power reconstruction, we will continue to investigate and resolve accuracy issues associated with the approach using SPH equivalence parameters. Additionally, cross-section generation using MC^2-3 was initially developed and verified for cores with homogenized assemblies. Therefore, improvements and verification are needed to adapt these cross-sections for use in configurations with heterogeneous geometries.

REFERENCES

[1] The INL/ANL joint Griffin development team, *https://griffin-dev.hpc.inl.gov/site/theory_manual.html*, 2023.

[2] A. D. Lindsay et al., "2.0 - MOOSE: Enabling massively parallel multiphysics simulation," *SoftwareX*, vol. 20, p. 101202, 2022.

[3] C. H. Lee and W. S. Yang, "MC^2-3: Multigroup cross section generation code for fast reactor analysis," *Nucl Sci Eng*, vol. 187, pp. 268–290, 2017.

[4] P. A. Raviart and J. M. Thomas, "Primal hybrid finite element methods for 2nd order elliptic equations," *Mathematics of Computation*, vol. 31, no. 138, pp. 391–413, 1977.

[5] Y. Wang, C. H. Lee, Y. Jung, Z. Prince, J. Hanophy, L. Harbour, H. Park, and J. Ortensi, "Performance improvements to the Griffin transport solvers," Tech. Rep. INL/LTD-21-64272, ANL/NSE-21/51, Idaho National Laboratory and Argonne National Laboratory, 2021.

[6] E. Shemon, Y. Miao, S. Kumar, K. Mo, Y. S. Jung, A. Oaks, S. Lee, C. Permann, G. Giudicelli, L. Harbour, R. Stogner, C. Icenhour, M. Li, R. Lefebvre, and B. Langley, "Meshing, language server protocol, and other user-oriented moose framework improvements to enhance reactor analysis capabilities and workflows," tech. rep., American Nuclear Society-ANS, La Grange Park, IL, 2023.

[7] M. A. Smith, E. E. Lewis, and E. R. Shemon, "DIF3D-VARIANT 11.0, A Decade of Updates," Tech. Rep. ANL/NE-14/1, Argonne National Laboratory, 2014.

[8] Y. I. Chang et al., "Advanced Burner Test Reactor Preconceptual Design Report," Tech. Rep. ANL-AFCI-173, Argonne National Laboratory, 2006.

[9] N. Stauff, T. Fei, and M. Smith, "ARDP Natrium Neutronic Methodology: Argonne Neutronic Assessment of ABR-1000," Tech. Rep. ANL/NSE-22/31, Argonne National Laboratory, 2022.

[10] *ARC 11.0: Code System for Analysis of Nuclear Reactors*, Argonne National Laboratory, 2014.

[11] J. Leppänen, "Serpent – a continuous-energy Monte Carlo reactor physics burnup calculation code," tech. rep., VTT Technical Research Centre of Finland, 2015.

[12] E. Shemon, K. Mo, Y. Miao, Y. S. Jung, S. Richards, A. Oaks, and S. Kumar, "MOOSE framework enhancements for meshing reactor geometries," tech. rep., American Nuclear Society-ANS, La Grange Park, IL, 2022.

[13] E. Shemon, Y. Miao, S. Kumar, K. Mo, Y. S. Jung, A. Oaks, S. Richards, G. Giudicelli, L. Harbour, and R. Stogner, "MOOSE Reactor Module: An Open-Source Capability for Meshing Nuclear Reactor Geometries," *Nuclear Science and Engineering*, vol. 197, 2023.

[14] Y. Chang, P. Finck, C. Grandy, J. Cahalan, L. Deitrich, F. Dunn, D. Fallin, M. Farmer, T. Fanning, T. Kim, *et al.*, "Advanced burner test reactor preconceptual design report.," tech. rep., Argonne National Laboratory, 2008.

[15] R. E. Alcouffe, F. W. Brinkley, D. R. Marr, and R. D. O'Dell, "User's Guide for TWODANT: A Code Package for Two-Dimensional, Diffusion-Accelerated, Neutral-Particle Transport," Tech. Rep. LA-10049-M, Los Alamos National Laboratory, 1990.

[16] Z. M. Prince, Y. Wang, J. T. Hanophy, V. M. Laboure, S. Kumar, Y. S. Jung, and C. Lee, "Improvements to the griffin transport solvers," tech. rep., Idaho National Laboratory, 2022.

[17] C. Matthews, V. Laboure, J. Ortensi, M. DeHart, Y. Wang, and R. Martineau, "Evaluation of the moose tool-set for analysis of thermo-mechanical-neutronics coupling in micro-reactors," *Los Alamos National Laboratory, LAUR-19-31443*, 2019.

[18] J. Leppänen, M. Pusa, T. Viitanen, V. Valtavirta, and T. Kaltiaisenaho, "The Serpent Monte Carlo code: Status, development and applications in 2013," *Annals of Nuclear Energy*, vol. 82, pp. 142–150, 2015.

[19] Y. Xu and T. Downar, "Code for Generating the PARCS Cross Section Interface File PMAXS," *Purdue University, West Lafayette, Indiana, USA*, 2006.

[20] J. Leppänen, M. Pusa, and E. Fridman, "Overview of methodology for spatial homogenization in the serpent 2 monte carlo code," *Annals of Nuclear Energy*, vol. 96, pp. 126–136, 2016.

[21] MOOSE Website, "MOOSE Website. Heat Conduction Module." `https://mooseframework.inl.gov/modules/heat_conduction/index.html`. Accessed: 2023-08-01.