ANL/NSE-22/45

Progress on Demonstration of a MOOSE-Based Coupled Capability for Hot Channel Factors in Fast Reactors

Nuclear Science and Engineering Division

ANL/NSE-22/45

Progress on Demonstration of a MOOSE-Based Coupled Capability for Hot Channel Factors in Fast Reactors

prepared by
Hansol Park, Yiqi Yu, Emily Shemon, and April Novak
Argonne National Laboratory

September 15, 2022

EXECUTIVE ABSTRACT

Hot channel factors (HCFs) are computed values that account for the impact on predicted peak fuel, cladding, and coolant temperatures due to uncertainties in the as-built reactor's material properties and geometry as well as uncertainties due to modeling approximations. Reduction in computed HCF values via reduction or elimination of modeling approximations may translate to significant economic savings if the reactor power can be raised due to the extra temperature margin gained. While limited historical datasets exist for sodium-cooled fast reactors (SFRs), there are no available HCF data for lead-cooled fast reactors (LFRs) outside of work generated previously within NEAMS. The computation of HCFs involves insights from reactor physics, thermal fluids and heat conduction calculations to determine how the peak temperatures respond to various uncertainties in the design.

Due to the significant advantages for multi-physics coupling offered by the MOOSE framework, Griffin (MOOSE-based reactor physics code), MOOSE Heat Conduction Module, and Cardinal (MOOSE-wrapped multi-physics application which includes the NekRS thermal fluids code) are being coupled together using the MOOSE MultiApp System to develop a high-fidelity multi-physics modeling capability for HCF simulations. This high-fidelity coupling workflow may also be beneficial for other fast reactor applications in the future. In previous work, Griffin and NekRS were individually assessed to ensure the necessary capabilities were in place. This work describes initial efforts to couple the codes (including folding in the MOOSE Heat Conduction Module) and determining the workflow for the perturbed calculations which will leverage the Stochastic Tools Module (STM). To our knowledge, this is the first coupling of Griffin and NekRS as well as the first exploratory use of Stochastic Tools Module for Cardinal.

In this report, the neutronics code Griffin, the heat conduction solver in MOOSE, and the MOOSE-wrapped application containing NekRS (Cardinal) are linked together to demonstrate the coupled capability. Griffin and Cardinal are linked dynamically by specifying shared libraries. Different coupling hierarchies are tested for selecting the most appropriate coupling strategy. A coupling scheme is selected based on the efficiency of calculation and ease of data communication. Multiple tests are performed to choose suitable mesh structure, model configurations, scheme setup and boundary conditions to avoid loss of energy due to data interpolation between different modules or weak imposition of fluxes in finite element codes. Computational experiments are performed to study the tolerance control of each type of iteration to avoid false convergence.

The coupled capability is demonstrated in both single pin and 7-pin models based on LFR materials and geometry. The study finds that the use of too large a time step size in the heat conduction module can lead to temperature oscillation even though the heat conduction equation does not have a time-derivative kernel, but only the time-dependent boundary condition. A 7-pin model without duct region achieved good convergence in the coupled calculation while a 7 pin model with duct region experienced data communication issues which need to be resolved.

ACKNOWLEDGEMENTS

This work was supported by the Department of Energy – Nuclear Energy Advanced Modeling and Simulation Program (NEAMS) under the Multiphysics Applications Technical Area. Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Nuclear Energy, under contract DE-AC02-06CH11357.

We gratefully acknowledge the computing resources provided on Bebop, a high-performance computing cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

Guidance and support from Cardinal developer and co-author April Novak was greatly appreciated, in addition to other Cardinal and NekRS developers (Elia Merzari, Dillon Shaver, Ananias Tomboulides, Paul Fisher), Griffin developers (Changho Lee, Javier Ortensi, Yaqi Wang, Zachary Prince, Vincent Laboure and Yeon Sang Jung), and the MOOSE framework team (Cody Permann).

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 Introduction

The U.S. Department of Energy, Office of Nuclear Energy Advanced Modeling and Simulation (NEAMS) [1] Campaign aims to develop, demonstrate, and deploy predictive computer methods for the analysis and design of nuclear reactor phenomena. The Multiphysics Applications Technical Area within NEAMS is tasked with assessing the readiness of physics tools for specific reactor applications. This work assembles the reactor physics code Griffin [2], thermal fluids code NekRS [3] via Cardinal [4], and MOOSE Heat Conduction Module [5] into a coupled simulation to compute high fidelity temperature distributions for lead cooled fast reactor (LFR) designs. Additionally, this work will explore the software requirements needed to leverage the MOOSE Stochastic Tools Module [6] in Cardinal for HCF simulations.

1.1 Motivation: Hot Channel Factor Prediction through High Fidelity Simulation

Numerous uncertainties are involved in the predictions of reactor design parameters, including theoretical and experimental analysis uncertainties, instrumentation uncertainties, manufacturing tolerances, correlation uncertainties, and method and simulation uncertainties. These uncertainties impact the computed and actual peak cladding, fuel, and coolant temperatures in the system. The peak temperatures in the as-built system must nevertheless be maintained at safe margins away from maximum temperatures that could compromise the integrity and performance of the materials. The impact of uncertainties on the temperature predictions is typically accounted for through the assessment of HCFs, which consider the change in reactor temperatures due to specific uncertainties. The goal of this work is to reduce over-conservatism in HCFs using high-fidelity multi-physics simulations that better account for the uncertainties associated with HCFs. If HCF values can be reduced through advanced modeling and simulation, the nominal peak cladding temperatures can be raised, resulting in higher power and economic gains.

1.2 Summary of Prior Work

In previous work [7-11], the NEAMS campaign employed PROTEUS [12] and Nek5000 [13] for high-fidelity sodium-cooled fast reactor (SFR) and LFR HCF calculations using an offline, one-way coupled workflow. This resulted in the first known HCF dataset for LFRs. The high-fidelity simulations reduced and/or eliminated geometrical approximations, and the computational scalability of these codes permitted advanced physics models including pin-by-pin heterogeneous transport, large eddy simulations (LES) and Reynolds Averaged Navier Stokes (RANS) for turbulence modeling, and conjugate heat transfer for computing heat transfer in the fuel and cladding zones. When compared to legacy HCF data from the EBRII SFR [14], important conclusions were drawn. First, advanced modeling and simulation can reduce HCF in many cases. Second, HCF simulations should be performed for the design of interest rather than using a generic dataset based on reactor type. This is because geometric and material differences across designs may impact the resulting HCFs.

Building upon the success of PROTEUS and Nek5000, work began in FY21 to develop a Multiphysics Object Oriented Simulation Environment (MOOSE)-based [15] capability for more robust and automated coupling. The MOOSE-based Griffin reactor physics code and MOOSE-wrapped NekRS thermal fluids codes were individually assessed on a prototypic LFR assembly [16] since they can be coupled via MOOSE's MultiApp System [17] and both can perform highly heterogeneous simulations with few modeling approximations. The LFR design was selected due

to the absence of LFR HCFs in the literature, which creates an opportunity for advanced modeling and simulation to fill a critical need. Furthermore, the LFR design features an interesting combination of features different from conventional SFRs including annular fuel. The presence of the ducted assembly geometry in the LFR facilitates simple single-assembly modeling since fluid flow is isolated from the full-core flow within each assembly. The list of priority HCFs for LFRs is shown in Table 1.1 for context, although calculation of these specific values is a longer term goal planned in the next year.

While the work here is targeted at HCF simulations, the assessment has broader applicability as users seek to demonstrate and advance both Griffin and NekRS capabilities on various high fidelity (heterogeneous pin-by-pin) fast reactor assembly geometries.

Table 1.1 List of priority LFR HCFs

LFR Hot Channel Factor	**Description**
Cladding thickness (subchannel flow area)	Assess impact of variances in cladding thickness due to manufacturing tolerance *Assumption: +/- 0.05 mm tolerance; previous PROTEUS-Nek model changed uniformly by maximum value in all pins due to meshing complexity*
Fissile fuel maldistribution	Assess impact of uncertainties in fissile content due to manufacturing tolerance *Assumption: +/-5% tolerance on Pu-239 enrichment; sample stochastically in all pins*
Coolant specific heat	Assess impact of uncertainties in lead coolant specific heat *Assumption: +/-5% uncertainty in lead specific heat*
Coolant density	Assess impact of uncertainties in lead coolant density *Assumption: +/- 0.8% uncertainty in lead coolant density*
Coolant isotopics	Assess impact of uncertainties in lead coolant isotopics *Assumption: Use "low" quality and "high" quality lead with varying Pb-208 contents to simulate lead sourced from different mines.*
Cladding conductivity	Assess impact of uncertainties in cladding conductivity *Assumption: +/- 10% uncertainty in cladding conductivity*
Fuel conductivity	Assess impact of uncertainties in fuel conductivity *Assumption:+/- 21.3% uncertainty in fuel conductivity*

1.3 FY22 Scope

Last year's work focused on testing of standalone Griffin and NekRS simulations to ensure their computational readiness for high fidelity, heterogeneous fine-mesh assembly calculations. Between code development performed by the Griffin team (performance improvements) and the Application Drivers team (supporting microscopic cross section definition), Griffin is now computationally ready to perform these calculations using the DFEM-SN solver with CMFD acceleration. Standalone NekRS was also found to be ready by the end of FY21, but its wrapping within Cardinal and subsequent compatibility with Griffin had not been tested.

This year's work focuses on developing and assessing coupling schemes for Griffin, NekRS (via Cardinal) and heat conduction equation solver (provided by the MOOSE Heat Conduction Module). NekRS is at a lower maturity level than its predecessor Nek5000 and additionally, its MOOSE-wrapping implementation within Cardinal updates NekRS infrequently because the NekRS development team only enforces their test suite when a new master branch is to be released (which has historically occurred every 3-9 months). Due to the very high computational requirements of NekRS, a simplified single pin cell problem with LFR properties was examined for the majority of the work this year. This allowed the computational time to be kept somewhat reasonable while debugging and testing different strategies for the nominal condition case.

This report includes an examination of coupling strategies using the MultiApp System considering the specific Griffin solver required for this simulation which has a different implementation of sub-application calls than other executioners. Sensitivity of results to different tolerance criteria is considered as well the sensitivity to mesh / angular discretizations. An energy conservation study is performed to verify coupling accuracy. The future workflow of using the Stochastic Tools Module to perturb input parameters and propagate them throughout the entire toolsuite is considered in order to identify gaps needed to conduct stochastic analysis with NekRS via MOOSE. Consequently, some code development updates in Cardinal are required before this workflow can be executed. In the final chapter, we discuss preliminary results for a more realistic 7-pin model although most larger geometry work is deferred to FY23.

2 Description of Physics Solvers

2.1 Griffin

Griffin is a reactor physics analysis application using the MOOSE framework. Griffin can be natively combined with other MOOSE applications for coupled multi-physics simulations of neutronics and different physics phenomena. Griffin can run steady state and transient simulations in either assembly-homogenized core problems or fine-mesh high-fidelity heterogeneous assembly/core problems. Griffin solves the multigroup transport partial differential equations in weak forms using various discretization schemes with continuous or discontinuous finite element method (FEM) in space and S_N, P_N or diffusion in angle. By default, Griffin uses the PJFNK (Preconditioned Jacobian-Free Newton Krylov) method to solve a steady-state problem.

For a fine-mesh high-fidelity heterogeneous assembly problem like that being modeled in this work, the discontinuous FEM (DFEM)-SN solver needs to be used for local reaction rate conservation. The DFEM-SN solver was improved significantly with the implementation of the coarse mesh finite difference (CMFD) acceleration in FY21 [18]. The ability to perform multiphysics coupling was added to the fixed-point iteration of the CMFD call under the Richardson iteration [19]. Since this feature is of great importance to understand the coupling hierarchy, it is explained in detail here.

- Richardson iteration
  - Fixed point iteration
    - Sub app. @ timestep_begin
    - CMFD calculation (low order diffusion)
    - Prolongate CMFD solution to transport solution
    - Update power
    - Sub app. @ timestep_end
    - Update cross-section
    - Convergence check for fixed point iteration
  - Transport sweep to update transport solution
  - Convergence check for Richardson iteration

Figure 2.1. Pseudo code for multi-physics coupling in the DFEM-SN CMFD solver of Griffin.

Figure 2.1 shows the basic algorithm of the coupling scheme of the DFEM-SN CMFD solver of Griffin with its sub applications. At each Richardson iteration, fixed point iterations are performed, followed by transport sweep to update transport solution and Richardson iteration convergence check. In each fixed-point iteration, any sub applications declared to run on timestep_begin and timestep_end are called before and after the CMFD calculation, respectively, for multi-physics coupling. This scheme has an important implication for coupling of neutronics code with a CFD code as discussed in a later section.

The role of Griffin in the multiphysics workflow is to provide power distribution in the whole domain to the other solvers for use as a heat source. Since reactor power depends on temperature through fuel Doppler and fluid density feedback, iterations are needed between Griffin and the

thermal-fluid solvers, which are the MOOSE Heat Conduction Module and NekRS in this application.

Griffin considers the temperature feedback effect by evaluating cross-sections at temperature evaluated at each quadrature point within an element. For this, cross sections need to be tabulated as a function of temperature in the isoxml cross section library. The density feedback effect is also considered quadrature-wise. A user needs to define an *aux kernel* to calculate density as a function of temperature and neutronics material in Griffin updates macroscopic cross sections by multiplying the ratio of newly-evaluated density to the reference density a user provides to cross sections evaluated at the reference density.

The cross section library generation procedure was described in detail in last year's report [16]. The MC^2-3/TWODANT [20] procedure was used to generate 9 energy group cross section sets for materials at different axial zones. For each material, two or three temperature points were selected for use with interpolation in Griffin. Since the target problem is a fast spectrum reactor problem, not many temperature points are needed. Temperatures of 600 and 800 K were used for axial regions below the active core region, 900 and 1300 K for upper regions, 700 and 1000 K for cladding and duct materials in the active core region, and 700, 1300 and 2400 K for the fuel material in the active core region.

2.2 MOOSE Heat Conduction Module

The MOOSE Heat Conduction Module models conduction, radiation between gray, diffuse surfaces, and contains provisions to couple temperature fields to fluid domains through boundary conditions. Boundaries in MOOSE Heat Conduction Module can be declared as a heat flux or fixed temperature. In this multi-physics coupled calculation, MOOSE Heat Conduction Module receives the heat source from Griffin and calculates the heat flux on the interface between solid and fluid domain based on that heat source. The MOOSE Heat Conduction Module is used to solve for energy conservation in the solid.

$$\rho_s C_{p,s} \frac{\partial T_s}{\partial t} - \frac{\partial}{\partial x_i}\left(k_s \frac{\partial T_s}{\partial x_i}\right) - q_s = 0$$

where T_s is the solid temperature, ρ_s is the solid density, $C_{p,s}$ is the solid specific heat capacity, k_s is the sold thermal conductivity, and q_s is volumetric heat source. The interface between solid and fluid domain are posed as a conjugate heat transfer problem, where NekRS computes the surface temperature (which is sent to the solid solver as a Dirichlet boundary condition) and MOOSE heat conduction computes a heat flux (which is sent to NekRS as a Neumann boundary condition). Since the target problem is a pseudo-transient problem, the time-derivative term is ignored in the solve. Instead, the solid-fluid boundary condition is time-dependent via coupling with NekRS.

2.3 NekRS / Cardinal

NekRS is a computational fluid dynamics (CFD) code developed at Argonne National Laboratory (ANL), University of Illinois at Urbana Champaign (UIUC), and Pennsylvania State University (PSU). NekRS aims to leverage the present trend in GPU-based high-performance computing (HPC) systems to perform CFD on GPU-accelerated systems. By using the OCCA library's unified API [21], NekRS can run on CPUs and on GPU-accelerated CPUs that support CUDA, HIP, or OpenCL. NekRS has been tested for 30 million elements (total 10.4 billion grid points) for high performance computing on Oak Ridge National Laboratory's leadership computing machine,

Summit. Moreover, by using GPUs, the computation speed of NekRS improves substantially when the coarse grid solver is based on the AMG solver. Because the Nek5000 code is somewhat of a predecessor to NekRS, some aspects of the current NekRS design were selected to enable faster translation of Nek5000 input files into NekRS input files, which make it flexible to convert the previous Nek5000 model into NekRS model. In this multi-physics coupled calculation, NekRS is responsible for predicting the flow velocity by solving the incompressible Navier Stokes equation and computing the temperature distribution using the heat flux from MOOSE heat conduction module. Typically, temperature and velocity inlet and pressure outlet boundary conditions are applied. In order to leverage NekRS in MOOSE-based multiphysics coupling, Cardinal is used.

Cardinal is a wrapping of the GPU-oriented spectral element Computational Fluid Dynamics (CFD) code NekRS and the Monte Carlo particle transport code OpenMC [22] within the MOOSE framework [4]. Cardinal provides high-resolution thermal-hydraulics and/or nuclear heating feedback to MOOSE multi-physics simulations. Multi-physics feedback is implemented in a geometry-agnostic manner with virtually no requirements on node/element/cell alignment, eliminating the need for rigid one-to-one mappings. A generic data transfer implementation also allows NekRS and OpenMC to be coupled to any MOOSE application, such as Griffin, enabling a broad set of multi-physics capabilities. Cardinal can also leverage combinations of MPI, OpenMP, and GPU resources to achieve in-memory coupling for delivering high-resolution simulations. By taking advantage of MOOSE user object and post-processing systems, Cardinal can gain improved insight into NekRS and OpenMC solutions and provide multiscale closures to other MOOSE applications, which significantly facilitate the information transfer from high fidelity model to low fidelity model.

Cardinal has its own MOOSE problem, *NekRSProblem*, which entails a MOOSE mesh, *NekRSMesh,* and a MOOSE timestepper, *NekTimeStepper,* for wrapping NekRS. In *NekRSProblem*, the "casename" parameter of native NekRS inputs listed in Table 2.1 must be specified as well as parameters for conversion of non-dimensional form of NekRS solutions into dimensional form (either for transferring field data of other MOOSE physics tools to/from NekRS or for postprocessing of NekRS solutions). *NekRSMesh* builds a mirror mesh of a NekRS mesh in a MOOSE-compatible format and takes care of data interpolation between NekRS's GLL points and the mirror mesh. *NekTimeStepper* allows NekRS to choose its time step via adaptive time-stepping (subject to its parent application's synchronization time points).

Table 2.1. List of NekRS native input files

File name	Function
{casename}.par	High-level settings for NekRS solver, boundary condition mappings to sidesets, and the equations to solve
{casename}.re2	NekRS mesh
{casename}.udf	User-defined C++ functions for on-line post-processing and model setup
{casename}.oudf	User-defined Open Concurrent Compute Abstraction (OCCA) kernels for boundary conditions and source terms
{casename}.f00001	Initial condition for fluid resolution

2.4 MOOSE Stochastic Tools and Reactor Modules

The MOOSE Stochastic Tools Module (STM) [6] samples parameters according to user-defined distributions which can then be propagated down to other applications in a MultiApp hierarchy to automate perturbation studies. STM has the ability to collect statistics on the series of calculations it spawns with the sampled parameters. For HCF calculations, parameters can be perturbed with STM automatically and propagated to sub applications at any level. However, present limitations in Cardinal prohibit templated parameter propagation down to NekRS. This is discussed in Section 3.2.

Additionally, the recently developed MOOSE Reactor Module [23] provides widely expanded native MOOSE meshing capability, therefore exposing meshing parameters and geometry definition to be perturbed with the STM where it is used. Use of Reactor Module objects (and other MOOSE mesh generators) within the [Mesh] block of an application therefore permits a geometry-based parameter study to be executed by MOOSE Stochastic Tools. The caveat for this work is that NekRS requires a postprocessed mesh in a special format (non-MOOSE based) on disk, requiring the Cardinal simulation be restarted rather than continuing the workflow directly. The "serial" operation mode of the STM does facilitate this NekRS limitation, though at an efficiency penalty by requiring the simulation to repeat many initialization tasks (such as initializing equation systems, compiling OCCA kernels, etc.) that do not need to be repeated for stochastic simulations. Improvements in NekRS/Cardinal are recommended to be able to leverage MOOSE mesh generators directly, which could be achieved by replacing NekRS's custom mesh format with Exodus or another mesh format also supported by MOOSE.

3 Multiphysics Model Description

There are multiple benefits for moving towards a MOOSE-based coupling approach for computing HCFs. First, integrated multi-physics calculations are necessary to achieve the most accurate temperature distribution because they account for all feedback effects simultaneously. The MOOSE MultiApp approach allows automated, in-memory coupling of any number of physics components with little to no code development work. Coupling all three physics together via MultiApps (with Picard iteration) allows tight, 2-way feedback between each communicating set of physics, which is difficult to achieve in offline computing schemes. Second, the MOOSE Stochastic Module can be used to sample input parameters, propagate these through the MultiApp hierarchy, and collect information/statistics after the runs conclude.

In the present LFR application, the normalized pin power distribution is typically not strongly influenced by small temperature changes in the coolant or fuel of a fast spectrum reactor. Therefore, a one-way, once-through coupling is typically appropriate if a reasonable temperature guess is assumed in neutronics. This could be achieved with a once-through scheme if desired rather than Picard iterations. However, the simulations in this work were set up as Picard iterations for full flexibility.

An issue that may arise is the disparate computational resource requirements for neutronics and thermal fluid codes, which tend to be dominated by the latter when using CFD. The option to run them separately on different resources may override the benefits of online coupling depending on whether the neutronics solution is frequently re-used. To reduce these concerns for this demonstration, small geometries were selected in which online iterations can be performed on small compute clusters.

3.1 MultiApp Hierarchy–- Nominal Condition

Figure 3.1 shows the relationship among the neutronics code Griffin, the heat conduction solver in MOOSE, and the CFD code NekRS. Technically, the MOOSE-wrapper of NekRS, Cardinal, is coupled with the other two codes as an interface. Griffin's role is to provide power to the heat conduction module, which solves the heat conduction equation using the volumetric heat source and the Dirichlet boundary condition of solid-fluid wall temperature provided by NekRS. Resulting solid temperature and heat flux at a solid-fluid interface are sent back to Griffin and NekRS, respectively. NekRS computes fluid temperature using the Neumann boundary condition of heat flux provided by the heat conduction module and resulting wall temperature and volumetric temperature are sent back to the heat conduction module and Griffin, respectively. Using solid and fluid temperatures, Griffin updates cross-sections for Doppler and fuel density feedback effect, respectively, and re-computes power to proceed iterations.

These individual physics solvers are assembled into a multiphysics workflow using the MultiApp System to achieve the coupling described above. There are many ways to construct a MultiApp simulation due to the flexibility of MOOSE, so different hierarchies were examined. Figure 3.2 shows possible hierarchies satisfying two conditions. First, any two codes cannot be run in a single app, because both Griffin and NekRS (via Cardinal) use custom MOOSE Problems/Executioners that do not fully support additional libMesh equation solves for additional equation sets. Second, NekRS should be located at the lowest level because it uses the smallest time step size, and all applications must at a minimum run for every main application time step.

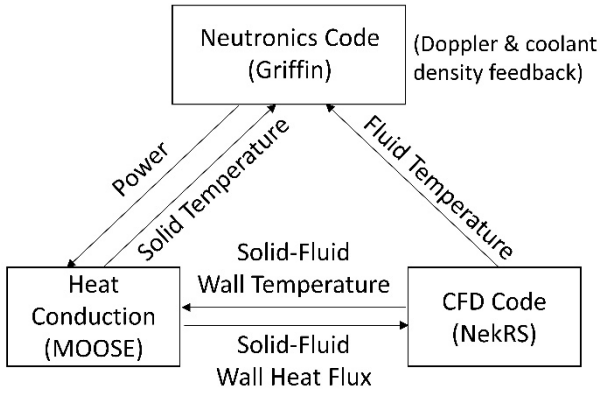

Figure 3.1. Relations among three physics codes.

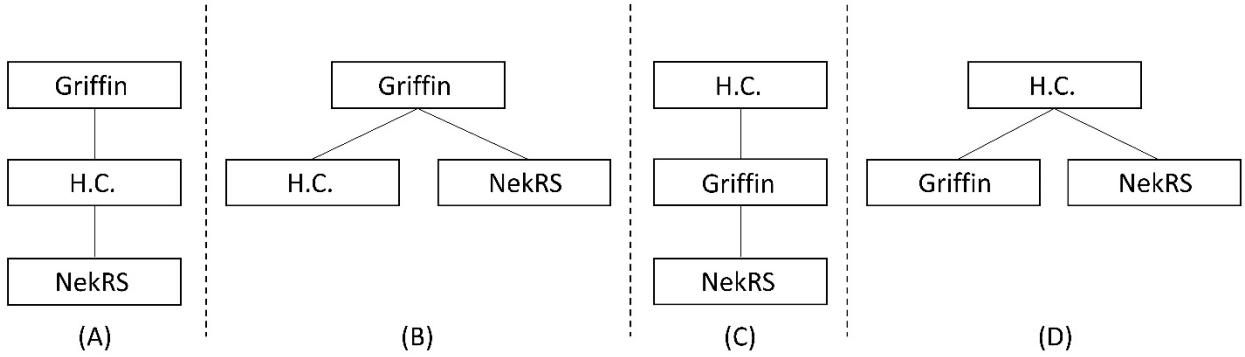

Figure 3.2. Possible coupling hierarchies satisfying that 1) any two codes cannot be run in a single app using strong coupling and 2) NekRS should be located at lowest level.

To understand Figure 3.2, first of all, one should be reminded that Griffin solves a steady-state problem (eigenvalue) while others solve transient problems. In a MultiApp setting, Griffin couples its sub applications with *FullSolveMultiApp* and others do with *Transient*. Thus, if Griffin calls its sub applications, transient calculations are performed until either the end time of sub applications is reached or a steady-state solution is detected (optional).

For (A) and (B) where temperature calculators are sub applications, they are performed inside the fixed-point iteration steps per Richardson iteration of neutron transport update in Figure 2.1. If the target quantity of interest is a neutronics quantity, not from sub applications, the fixed-point iteration may not need to be fully converged per Richardson iteration and let it just go through Richardson iterations without guarantee of its convergence. However, since our target quantity is temperature, its convergence needs to be guaranteed, which means that fixed point iterations need to be fully converged per Richardson iteration to avoid false convergence. In principle, this requires heat conduction + CFD calculations fully converged using the initial power distribution calculated by the CMFD calculation before the first transport calculation. Otherwise, it would be likely to have a false convergence if one is not careful about tolerance control of each type of iterations. One should note that CFD calculation requires a lot of time steps to converge and only the Richardson residual error is checked as the final convergence of the whole calculation by the DFEM-SN CMFD solver (*SweepUpdate* executioner). Between (A) and (B), (A) makes more sense unless (B) has any

technical advantages in coupling because solid and fluid temperatures are more tightly coupled via conjugate heat transfer than temperature and power are. This is especially true for fast reactor applications. One technical advantage of (B) in coupling is that the MOOSE Stochastic Tool Module would have easier access to all codes than in (A) for perturbation calculations in the future, as MOOSE STM would be the parent app above everything (although the STM can also be present at multiple levels in the MultiApp hierarchy, and does not preclude option (A)).

Figure 3.3 to Figure 3.5 illustrate the convergence history of fluid temperature and neutronics solution for a single pin-cell problem detailed in the next section with different fixed-point convergence criteria in Griffin. For this test, the velocity calculation in NekRS was turned off, such that NekRS is only solving the energy conservation equation for fluid temperature. The relative difference of L2-norm of fluid temperature over two fixed-point successive iterations was monitored to check the convergence. Figure 3.3 corresponds to the steady-state tolerance of 1E-5 (tolerance on L2-norm of NekRS fluid temperatures) for fixed-point iteration. It is clearly seen that the first Richardson iteration starts at the point when the fixed-point solution difference goes below 1E-5 and fluid temperatures are almost converged. Power solution obtained using just CMFD solution was used to calculate temperature until that point. The calculation ended when the Richardson iteration error went below 1E-3, which will be shown enough in the next section to get converged power solution in Griffin for a given set of temperatures. This scheme is computationally inefficient as CFD calculation needs to be forced to continue executing beyond its convergence.

Figure 3.4 and Figure 3.5 used the fixed-point iteration tolerance of 1E-4 and 1E-3, respectively. Richardson iteration started before temperature solution was converged. The 1E-4 case could luckily get temperature solution converged within the Richardson iteration tolerance, whereas the other case could not. These results indicate the importance of using tight fixed-point iteration convergence under the scheme (A). Or, it is suggested that any user-defined quantity is used as a convergence check together with the transport solution error in the Richardson iteration level to force Richardson iterations to continue until the user-specified tolerance is met.

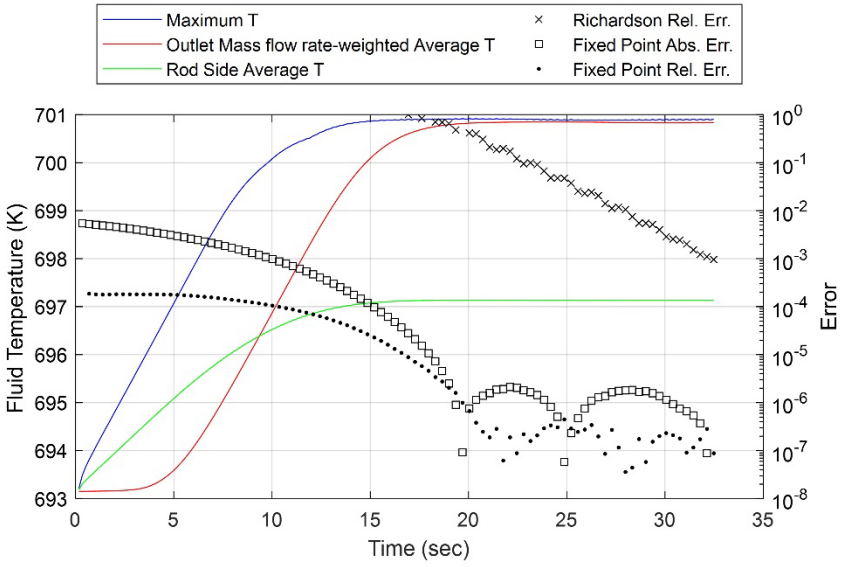

Figure 3.3. Temperature evolution and neutronics convergence with relative tolerance of fixed-point solution of 1E-5 for coupling scheme (A) ($\Delta t_{H.C.} = 20\Delta t_{nekRS}$).

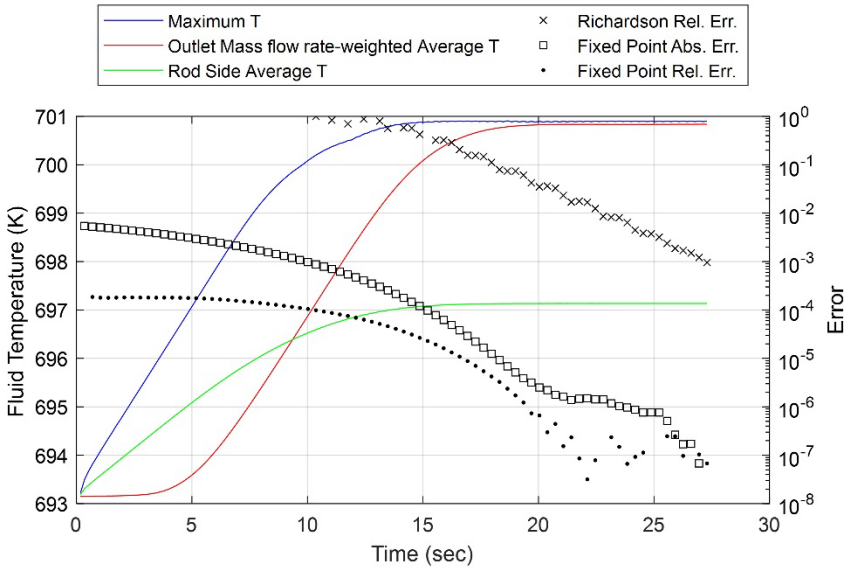

Figure 3.4. Temperature evolution and neutronics convergence with relative tolerance of fixed-point solution of 1E-4 for coupling scheme (A) ($\Delta t_{H.C.} = 20\Delta t_{nekRS}$).

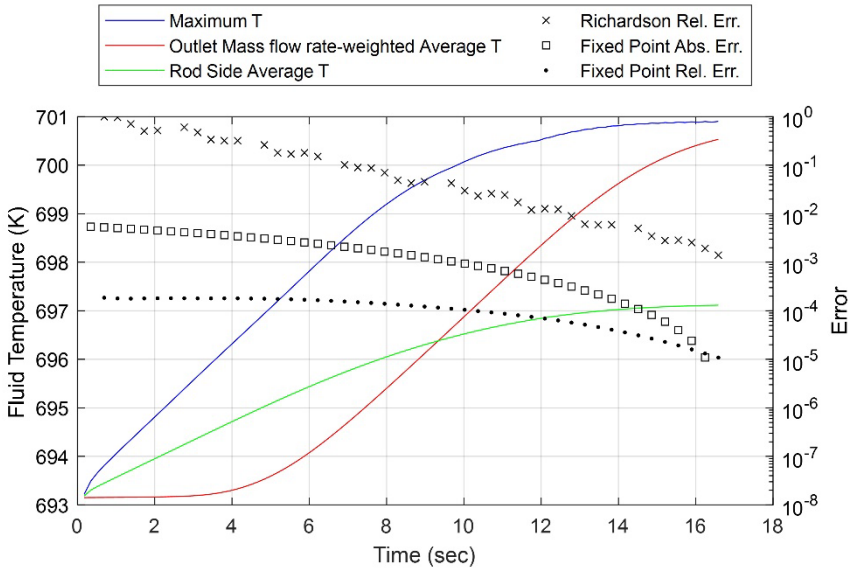

Figure 3.5. Temperature evolution and neutronics convergence with relative tolerance of fixed-point solution of 1E-3 for coupling scheme (A) ($\Delta t_{H.C.} = 20\Delta t_{nekRS}$).

For schemes (C) and (D), the temperature calculation is outside the Richardson iteration, which means neutronics transport solution is fully converged at every time step of heat conduction solve. This is the opposite of (A) and (B). One might set the maximum number of Richardson iterations as some small value to have it gradually converged over to the end of calculation, but one needs to

have a fairly clear understanding on the neutronics convergence of the problem of interest. For example, one can set the maximum number of Richardson iterations per heat conduction solve by roughly estimating the number of iterations required for neutronics solve and the number of time steps required for heat conduction solve. Figure 3.6 shows the convergence history of the same pin-cell problem for the coupling scheme (D). Richardson iteration was converged fully to the tolerance of 1E-3 at the beginning of the calculation and temperature calculations are performed with one neutronics transport update per heat conduction time step. To terminate the calculation, the steady-state detection function of the transient executioner can be used. However, the L2-norm of solution difference in two successive time steps (steady-state differential norm) didn't decrease over time for the pin-cell problem solved as shown in Figure 3.6, regardless of whether the solution of nonlinear system (solid temperature) or aux system (heat source, heat flux, surface temperature) was used. In this case, the simulation can be terminated by using the *Terminator UserObject* by manually defining a convergence metric such as fluid temperature change over time. To make the coupling scheme (D) more efficient to use, it would be desirable to have a MOOSE capability to choose a set of time steps at which a certain sub application is performed. In this way, even NekRS can become the main application to drive the rest of physics codes. The coupling scheme (C) was not explored since there is no compelling reason to have the neutronics solve sandwiched between solid and fluid temperature solves.

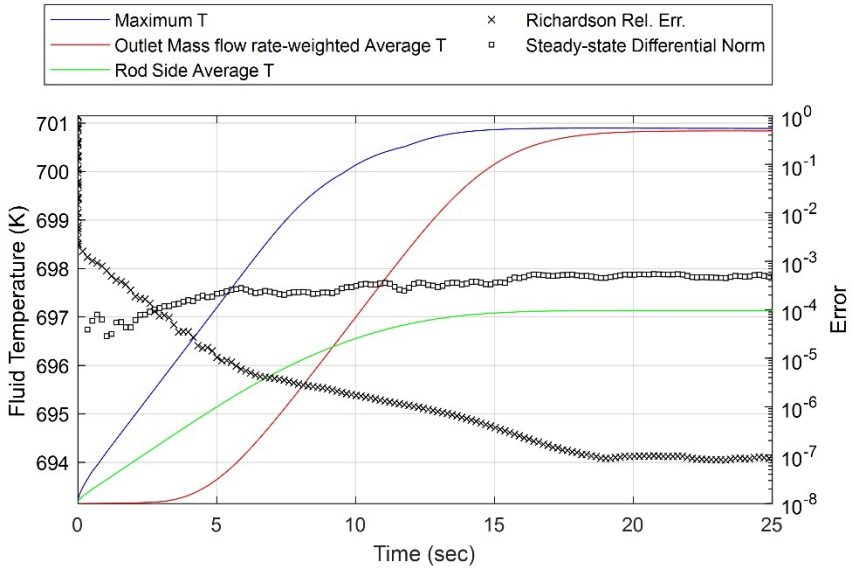

Figure 3.6. Temperature evolution and neutronics convergence with the coupling scheme (D) ($\Delta t_{H.C.} = 20\Delta t_{nekRS}$).

Even though (A) and (D) coupling schemes are suggested with some updates in either Griffin and MOOSE, only (A) was actively used in this work. Data transfers for (A) are summarized in Table 3.1. There are a few noteworthy remarks.

- For power density, two input parameters *from_postprocessors_to_be_preserved* and *to_postprocessors_to_be_preserved* are specified to preserve the total power produced only in the solid region. Even though Griffin calculates power in the fluid region via non-fission

heating, it is not counted (though Cardinal and NekRS do support sending volumetric power distributions to the NekRS fluid domains for this use case).

- The L2-norm of fluid temperature is calculated on heat conduction's fine fluid mesh and transferred to Griffin for use as convergence criteria. While Griffin has the data to calculate this L2-norm, the Griffin mesh in the fluid domain is coarser due to low coolant mesh density requirements for a fast reactor neutronics problem. The fluid mesh in heat conduction can be as fine as needed for high L2-norm accuracy, without impacting computational time. The fluid domain is not a part of its solve and is only used for storing fluid bulk temperature transferred from NekRS.
- The solid-fluid wall temperature is transferred between Griffin and heat conduction solely for backup purpose in heat conduction. This is purely specific to the coupling scheme (A). When a sub application is called in every fixed-point iteration, variables in the aux system are not restored even with the input parameter *keep_solution_during_restore* in *FullSolveMultiApp* being turned on. The input parameter keeps variables only in the nonlinear system.
- Solid-fluid wall heat flux integral is transferred to NekRS to avoid loss of energy purely from data transfer between different meshes. Since integral of transferred heat flux on the NekRS mirror mesh is not equal to that of original heat flux from MOOSE in general due to the different discretization orders and quadrature rules, the transferred heat flux values are normalized to make its integral value match the transferred integral value. Even though the actual heat flux integral calculated by the heat conduction module can be transferred, the total heat source in the solid domain is transferred owing to inevitable mesh discretization error in calculating the heat flux. This is only valid when the sum of all interfaces where transfer takes place and those of insulated boundary is the same as the sum of all outer surfaces of heated regions.
- Solid-fluid wall temperatures and fluxes are transferred separately for rod and duct.
- The sum of heat flux integrals in rod and duct is transferred to NekRS for normalization of heat flux.

Table 3.1. List of data transfers for coupling scheme (A)

Data	Transfer Type	Direction*	Purpose
Power Density	MultiAppMeshFunctionTransfer	G → H	For volumetric heat source in heat conduction
Solid Temperature	MultiAppInterpolationTransfer	G ← H	For Doppler feedback in neutronics
Fluid Temperature	MultiAppNearestNodeTransfer	H ← N	As an intermediate pass to Griffin
	MultiAppInterpolationTransfer	G ← H	For fluid density feedback in neutronics
L2-norm of Fluid Temperature	MultiAppPostprocessorTransfer	G ← H	For convergence check in Griffin
Solid-Fluid Wall Temperature	MultiAppNearestNodeTransfer	G ↔ H	Just for back-up purpose
		H ← N	For Dirichlet boundary condition in heat conduction
Solid-Fluid Wall Heat Flux	MultiAppNearestNodeTransfer	H → N	For Neumann boundary condition in NekRS
Solid-Fluid Wall Heat Flux Integral	MultiAppPostprocessorTransfer	H → N	For normalization of heat flux to be used in NekRS
Sync Flag	MultiAppPostprocessorTransfer	H → N	For efficient transfer only when needed

* G: Griffin, H: Heat conduction, N: NekRS

3.2 MultiApp Hierarchy – Perturbed Condition

For calculations of perturbed cases, STM is added to the top level of the MultiApp system in the coupling scheme of (A). Thus, Griffin, heat conduction and NekRS become the first, second and third sub applications of STM. In this work, *MultiAppSamplerControl* was used to propagate perturbations down to sub applications without any transfer as shown in Figure 3.7. Perturbing fuel conductance is shown as an example. A file-local variable that does not belong to any MOOSE object named "fuel_conductance" is declared in a Griffin input and is perturbed by the stochastic tool in the upper level of the MultiApp system using *MultiAppSamplerControl*. This parameter can be assigned in any MOOSE object including the MultiApp block where the "cli_args" feature of Python is used to pass command line arguments to the sub application. "cli_args" can replace the value of a file-local variable as well as that in a certain MOOSE object. Thus, the same file-local variable can be declared in the second sub application and sent to its (third) sub application (NekRS). In this way, perturbation can be propagated to a sub application of any level.

At the time of writing, NekRS cannot be integrated with the MOOSE stochastic tools module in this procedure. Several limitations currently exist that need to be resolved.

1. As discussed earlier, NekRS uses a custom mesh format (.re2) stored on disk. For geometric perturbations, NekRS needs to support directly reading meshes provided from MOOSE, such as Exodus II formats.
2. A measure needs to be devised to propagate perturbations to parameters in native NekRS inputs. For instance, with "native" MOOSE applications, many parameters such as the "fuel_conductance" in Figure 3.7 are directly exposed in the MOOSE input files. However, to change the fluid thermal conductivity in NekRS, an intermediate layer in the Cardinal input file needs to instruct that changes should be made to however the fluid thermal conductivity is represented in NekRS (which is an array member of the nrs_t::cds_t pointer with one array value per GLL point).
3. Since NekRS is an independent code from MOOSE, the NekRS driver needs to be fully compatible with the way the STM drives multiple runs of sub applications in a single run of the tool. There are three operation modes: normal, batch-reset, and batch-restore. The normal mode creates one sub application for each perturbed case and the others create one sub application for each processor. For example, for generating 10 perturbed cases, the normal mode instantiates 10 Cardinal sub application objects simultaneously. The other two modes depend on how many processors are used and how many perturbation cases there are. If 2 processors are used, 2 Cardinal sub application objects are instantiated simultaneously and destroyed after each of its calculation for 5 times each. If one wants to use multiple processors with only one sub application instance at a time, "min_procs_per_app" in *SamplerFullSolveMultiApp* and "min_procs_per_row" in a sampler needs to be specified equal to the number of processors being used.

   Currently, however, NekRS does not support running more than one NekRS case on the same MPI communicator due to the extensive use of global variables and the writing of just-in-time compiled OCCA code to a fixed-directory location (the .cache/ directory within the working directory). These cache files must be separate for each unique NekRS run, but there is currently no way to control unique NekRS perturbed runs from overwriting this data needed by other concurrent runs.

   In addition, the batch-restore mode requires that Cardinal allow full backup and restore of NekRS. This means that Cardinal must add functions to save and restore the entire "live" state of a NekRS simulation. This functionality could be added after identifying all the aspects of a NekRS simulation that must be "alive," and is mostly a book-keeping effort.

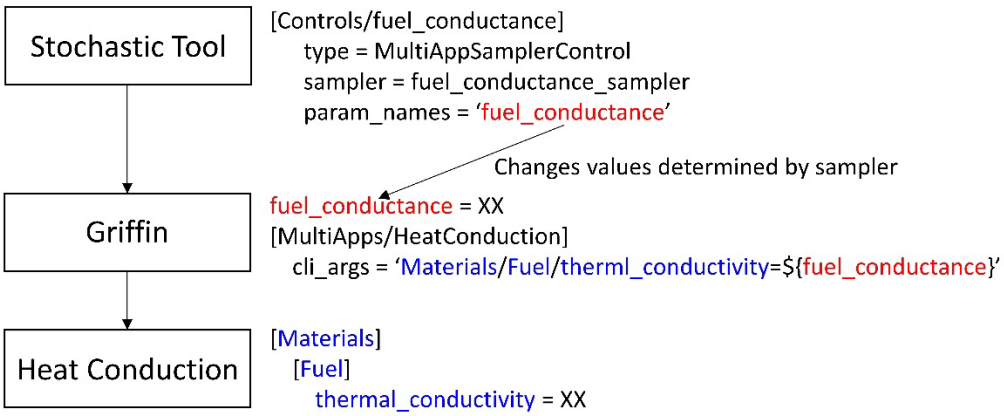

Figure 3.7. Example of input perturbations in Griffin and heat conduction module using the stochastic tools module. Placeholder numerical values are represented as "XX".

3.3 Code Installation and Environment

For the coupled code system, Cardinal and Griffin were compiled in an HPC cluster using the same environment by loading identical modules of gcc, openmpi, cmake, hdf5, and python. Griffin used the same MOOSE framework directory that Cardinal refers to as its submodule. In this setting, two codes could be linked dynamically without any difficulties by just specifying the shared library of the other code in an input of the main application. Because Griffin and Cardinal are both compiled with MOOSE's heat conduction module enabled, either can be used to run the heat conduction problem. Relevant input parameters in the MultiApp block of each application are listed below.

1. Griffin calls Cardinal (The other way round is possible.)
   - library_path = "{path of parent directory of Cardinal}/lib"
   - library_name = "libcardinal-opt.la" (This input card can be omitted unless one needs debugging or profiling library.)
   - app_type = "CardinalApp"
2. Griffin or Cardinal calls itself.
   - Nothing needs to be specified.

3.4 Summary of Multiphysics Approach

The coupling assessment in this chapter was used to select the following setup for the LFR pin calculation:

- Coupling Scheme (A) from Figure 3.2 which places Griffin as the main application, Heat Conduction as the child application, and NekRS (Cardinal) as the grandchild application.
  - (A) converges temperature first and then neutronics solution. The temperature calculation loop is inside the neutronics calculation. Convergence criteria for temperature needs to be tight to avoid false convergence since only convergence on neutronics solution is checked in the outer iteration in Griffin.

  - (D) converges neutronics solution first at the beginning and then temperature. Neutronics solver needs to be called at every time step of temperature solver. This would be computationally expensive when the time step size of heat conduction needs to be small.

- Codes can be coupled dynamically using instructions in Section 3.3.
- For perturbed calculations, Stochastic Tools Module becomes the main application, calling Griffin as its child application and so forth. However, NekRS (via Cardinal) cannot receive data properly from STM due to current implementation limitations.

4 Lead-Cooled Fast Reactor: Single Pin

A single pin-cell problem with and without duct were solved first to demonstrate the workflow and test the coupling hierarchy in the simplest problem setting. While LFRs place fuel pins in a hexagonal lattice, we simply model a square duct around this single-pin model since the purpose is only to test the coupling setup. Table 4.1 shows the geometric and operating condition for pin cell case. Figure 4.1 shows the sketch of the square pin cell model and the radial view of its MOOSE-based mesh. Both Griffin and the heat conduction module used a MOOSE mesh generated by *PolygonConcentricCircleMeshGenerator* in the Reactor Module.

Table 4.1. Key parameters of pin cell model

Parameter	***Value***	***unit***
Fuel pin diameter	0.8638	cm
Duct flat to flat distance	1.2	cm
Fuel length	10	cm
Upper and lower reflector length	5 (each)	cm
Upper and lower reflector material	Clad (rod) + Lead (coolant)	-
Lead density	10,401	kg/m^3
Lead specific heat	144.19	J/kg·K
Reynolds number	500	-
Fuel thermal conductivity	1.882	W/m·K
Clad and duct thermal conductivity	21.6	W/m·K
Lead inlet temperature	693.15	K
Lead inlet speed	1.0	cm/s
Total power	10.0	W

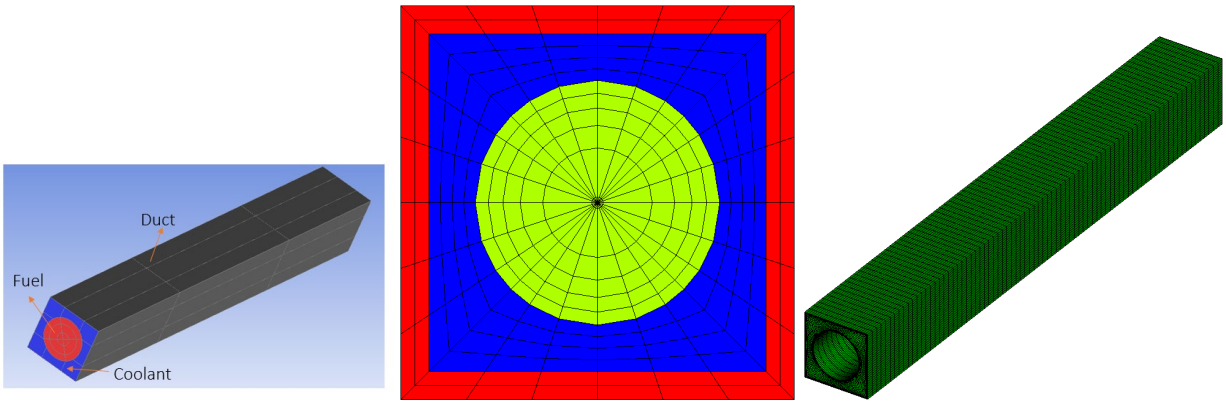

Figure 4.1. Sketch of pin cell model (left), radial view of its MOOSE mesh (middle), and NekRS fluid mesh (right).

4.1 Griffin and Heat Conduction Sensitivity Study

The sensitivity study for a Griffin-heat conduction coupled calculation with a fixed fluid-solid boundary condition (constant wall temperature) was conducted. First, the sensitivity of power solution to mesh and angular quadrature and sensitivity of temperature and heat flux solutions to mesh were examined for Griffin and heat conduction standalone calculations, respectively. Figure 4.2 shows the result of Griffin standalone calculations with a uniform fuel temperature: maximum relative errors of fuel powers in 100 meshes (5 radial meshes × 20 axial meshes) with respect to the reference solution obtained at the tightest condition for different angular quadratures in DFEM-SN and different number of radial (coolant) and azimuthal mesh divisions. As shown, power distribution is quite insensitive to angular quadrature and mesh mainly because local heterogeneity effect is weak in a fast reactor problem and power is calculated using scalar flux, which is the integrated angular flux over angle. The radial mesh division test for the fuel rod region was performed for a Griffin-heat conduction coupled calculation shown later.

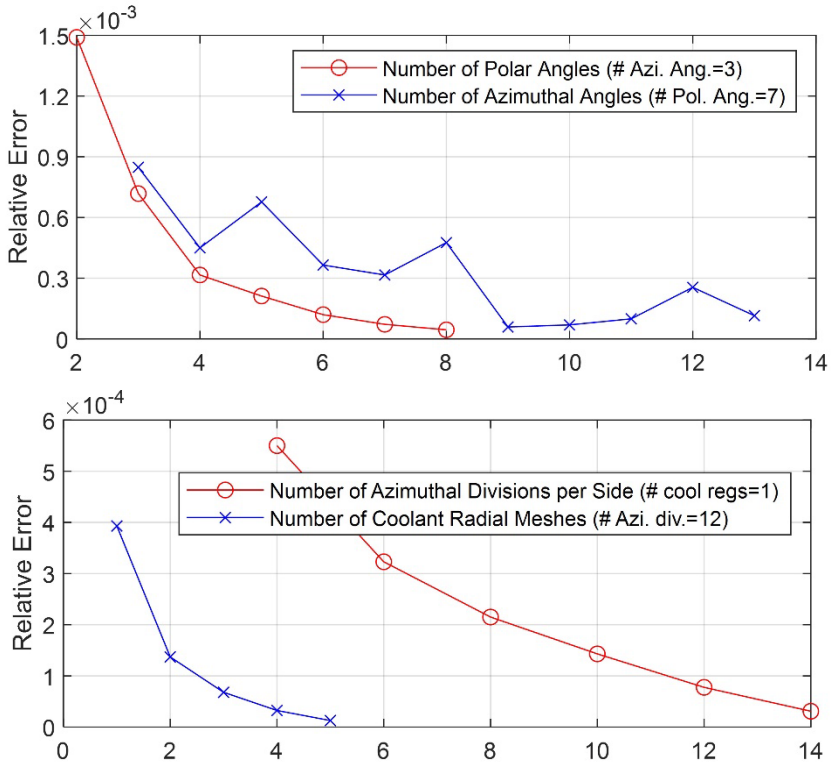

Figure 4.2. Power convergence studies in Griffin DFEM-SN standalone calculation with increasing angular (top) and mesh discretization (bottom). The figure shows maximum relative errors of power for different angular quadrature and increasing mesh refinement.

Figure 4.3 shows the results of heat conduction standalone calculations with uniform power: maximum relative errors of temperature and heat flux integral at fuel rod surface with respect to the reference solution obtained at the tightest condition for different number of radial and azimuthal divisions in a mesh. The heat flux integral is compared to its theoretical value, which is the total power produced inside the surface that the heat flux is integrated over. Note that heat fluxes are

always underestimated at the boundary owing to mesh discretization, resulting in negative error. The results in Figure 4.3 indicate that at least 30 radial meshes are needed to achieve less than 1% error in heat flux integral and 0.1% error in temperature distribution. Even though heat conduction solutions are not sensitive to the number of azimuthal divisions in a mesh, a sufficiently refined mesh is required in the azimuthal direction due to data transfer considerations at the fluid-solid boundary with Cardinal.

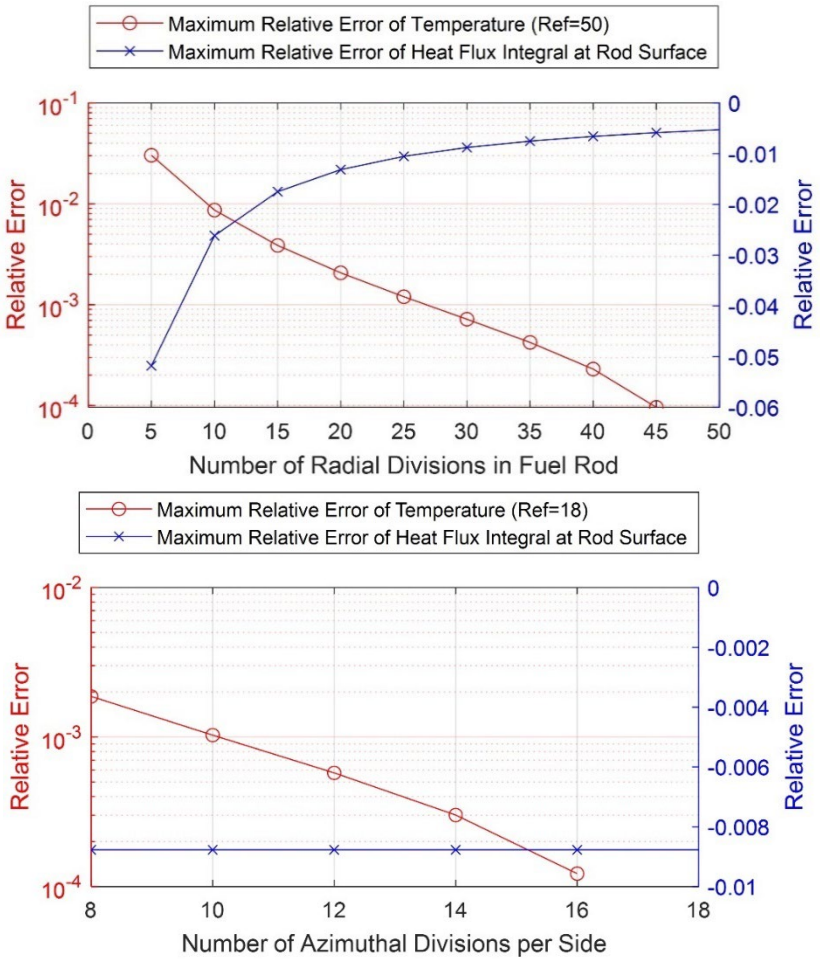

Figure 4.3. Temperature and heat flux integral convergence studies in heat conduction standalone calculation with (top) varying radial mesh (# Azimuthal Divisions = 8) and (bottom) varying azimuthal mesh (Radial Divisions = 30). The figure shows maximum relative errors of temperature and heat flux integral for different mesh refinement.

It should be emphasized that the Richardson iteration error check (neutron angular flux convergence) within Griffin is the driving force determining whether the coupled simulation is considered "converged". Error in the fixed point iteration loop, which includes the calls to sub-applications including temperature solution in this example, is not considered when checking the Richardson iteration error and deciding whether to continue to another Richardson iteration or to terminate the simulation. Since the solid temperature solution (which is part of the sub-application

data inside the fixed point iteration) needs to meet a desired accuracy, some understanding is needed about how the neutronics and temperature solutions evolve with Richardson iterations. The evolution of neutronics and temperature solutions with Richardson iterations was examined to estimate the relative error in neutron angular fluxes between successive Richardson iterations which result in a desired accuracy in solid temperature under a Griffin (main) – heat conduction (sub) coupled calculation. Again, only the neutron angular flux convergence is checked outside the fixed-point iteration loop to determine the termination of simulation. Figure 4.4 shows that the relative error of 0.05 for neutron angular fluxes yields nearly-converged power and temperature solutions within 0.1% error. Thus, the relative Richardson error tolerance of 1E-3 was used in all subsequent coupled calculations.

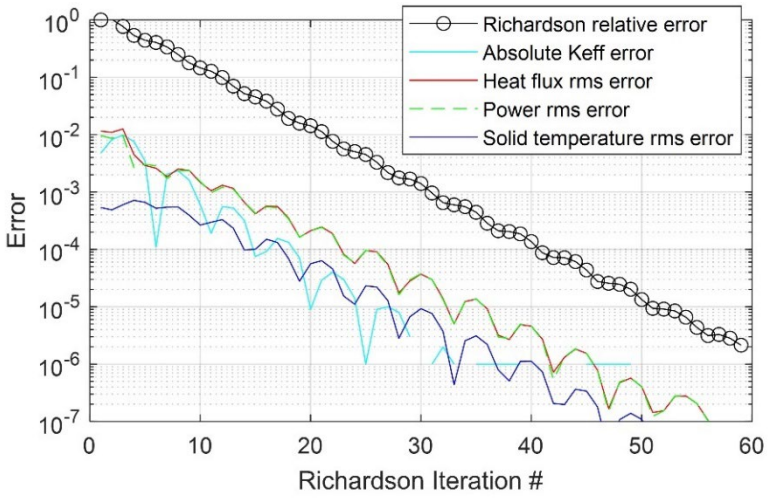

Figure 4.4. Convergence behavior of neutronics and heat conduction variables over Richardson iterations in a Griffin (main)-heat conduction (sub) coupled calculation.

Another test was performed to check the optimum number of fixed-point iterations and GMRES inner iterations per Richardson iteration in a Griffin-heat conduction coupled calculation. Note that a CMFD calculation not only accelerates transport solution updates but also updates power, and thus, temperature via subsequent heat conduction solve inside the fixed-point loop. Having a tight convergence on low order flux and temperature may or may not reduce the number of Richardson iterations. The overall performance is determined by the balance of additional computation burden from CMFD + heat conduction solve and reduction of outer Richardson iterations. The same story holds for inner iterations of linear solve per Richardson iteration. These behaviors depend on the problem of interest.

Table 4.2 shows the runtime and computational performance results. First, having more fixed-point iterations does not decrease the number of Richardson iterations, indicating that a single CMFD calculation per Richardson iteration is enough and having more than one CMFD calculations just imposes more burden than good. The major cause of computational time increase is from the larger number of heat conduction solves. Use of a boundary layer mesh near the fluid-solid boundary instead of uniformly remaining the entire solid domain may help reduce the computational cost of

the heat conduction solver. Boundary layer meshing is now available through the Reactor module and can be attempted in future work.

Similar computational times were seen between 2 and 3 maximum fixed-point iterations because the temperature solution converged within 2 fixed-point iterations per Richardson iteration. Larger number of inner iterations only serves to increase the calculation time for neutron transport. In conclusion, just a single iteration for each of fixed-point and linear (GMRES) solve is enough for our problem to have an optimum convergence in power and temperature solutions in the solid domain under the fixed fluid-solid boundary condition.

Table 4.2. Computational time comparison for Griffin → heat conduction coupling scheme A

# of max fixed point iteration/ Richardson iteration	# of max. GMRES inner iteration	# of Richardson iteration	Computational Time (sec)			
			Transport Update	CMFD	Heat Conduction	Total
1	1	31	96.6	57.4	272.1	426.1
	2	31	106.3	59.0	272.0	437.3
	4	31	121.4	57.3	268.8	447.5
2	1	33	115.8	97.5	482.1	695.4
	2	33	121.4	98.7	471.7	691.8
	4	33	137.4	95.7	469.5	702.6
3	1	34	115.9	98.7	491.9	706.5
	2	34	123.8	98.7	485.0	707.5
	4	34	143.6	98.1	486.7	728.3

Richardson relative tolerance = 1E-3
Number of MPI processors = 360

Coupling scheme (C), which employs a heat conduction (main) – Griffin (sub) coupled calculation, was tested to compare results against scheme (A). In scheme (A), the Richardson iteration drives CMFD + temperature calculations, whereas in coupling scheme (C) the temperature calculation drives Richardson + CMFD calculations.

Based on the result of Table 4.2, the number of linear iterations and CMFD solve per Richardson iteration was set to 1 for coupling scheme (C). Compared to the best result of 426 seconds in Table 4.2 (the first row), the performance is much better at 195 seconds. This is mainly because of many fewer heat conduction solves in scheme C. Only 26.5 seconds was spent in heat conduction solve under scheme C vs. 272 seconds in scheme A. The computational time per heat conduction solve at each time step was 9 seconds, whereas only 4 seconds was required per transport sweep + CMFD in Griffin. In Griffin (main) – heat conduction (sub) coupled calculation (scheme A), more computationally expensive heat conduction solve was unnecessarily involved per Richardson iteration during the inevitable convergence process of the neutronics solution itself. It would have not been a problem if the computational burden of heat conduction solve was negligible compared to neutron transport sweep + CMFD solve. For a coupled simulation with NekRS involved, the neutronics solution convergence process can be distributed along the time period needed to

converge fluid temperature in NekRS, during which heat conduction solve needs to be involved anyway to update heat flux boundary condition to NekRS.

Table 4.3. Computational time for heat conduction → Griffin coupling scheme C

# of Richardson Iteration	# of Fixed Point Iteration	Computational Time (sec)			
		Transport Update	CMFD	Heat Conduction	Total
43	3	100.5	67.8	26.5	194.8

Richardson relative tolerance = 1E-3
Max. GMRES inner iterations = 1
Max. fixed point iterations = 1
Number of MPI processors = 360

4.2 NekRS Mesh Sensitivity Study for Energy Conservation

The NekRS mesh sensitivity study was performed with MOOSE heat conduction-NekRS coupled calculation to ensure that the power source imposed in the solid domain does not experience losses due to mesh structure, boundary condition and model configuration. The tests are performed with different coolant (water and lead), different power source, different solid mesh, different fluid mesh and different model configuration.

In general, three different metrics can be easily checked to ensure energy conservation and proper resolution of boundary heat fluxes. The heat flux at the fluid-solid interface (1) in the solid mesh and (2) in the fluid mesh should both approximately match the total imposed power. Some error will be present because finite element methods only weakly impose flux boundary conditions. And (3), the energy applied to the fluid should match the enthalpy rise in the fluid domain as

$$\begin{aligned} 0 &= \int_{V_{\text{solid}}} \left(\dot{q}'''_{\text{solid}} - \vec{\nabla} \cdot \dot{\vec{q}}''_{\text{solid}\to\text{fluid}} \right) dV, \\ \int_{V_{\text{fluid}}} \rho c_p u_z \frac{\partial T}{\partial z} dV &= \int_{V_{\text{fluid}}} \left(\dot{q}'''_{\text{fluid}} - \vec{\nabla} \cdot \dot{\vec{q}}''_{\text{fluid}\to\text{solid}} \right) dV = \int_{\text{fluid-solid interface}} \dot{\vec{q}}''_{\text{solid}\to\text{fluid}} \cdot d\vec{S}, \\ \dot{m} c_p (\overline{T}_{out} - \overline{T}_{in}) &= \int_{V_{\text{solid}}} \dot{q}'''_{\text{solid}} dV, \end{aligned} \tag{1}$$

where $\dot{\vec{q}}''$ is a surface heat flux, $\dot{q}'''$ is a volumetric heat source, $\dot{m}$ is the mass flow rate, c_p is the heat capacity of coolant, $\overline{T}_{in}$ and $\overline{T}_{out}$ are the mass flow rate-weighted average temperature at inlet and outlet, respectively. In above equations, the divergence theorem, the opposite sign of heat fluxes in equations of the first and second rows are used. It is assumed that heat source in the fluid domain is ignored and top and bottom boundaries are insulated. All three of these conservation checks are performed in this section and will be referenced as #1, #2, and #3 for clarity.

Table 4.4 shows the results for the enthalpy rise check (#3) with different power source in solid domain. The magnitude of the power source has negligible impact on the error of energy conservation. This is convenient because the energy conservation study can be performed with

small power to speed up the calculation, since the case with small power requires less time for temperature field to develop.

Table 4.4. Energy conservation study with different power source in solid domain

Case	Small power	Big power
Volume integral of power source in solid domain (W)	11.71	117.1
Heat flux integral on interface (W)	11.69	116.91
Cp (J/g·K)	4.2	4.2
Mass flow rate (g/s)	0.85	0.85
Average outlet temperature (K)	283.25	312.55
Energy calculation from enthalpy rise (W)	11.66	116.65
Energy conservation error, #3 (%)	0.23	0.22

The mesh density near the interface in solid domain can affect the calculation of temperature gradient and thus the heat flux on the interface. Energy conservation studies using different solid meshes shown in Figure 4.5 were performed. The results are summarized in Table 4.5. Mesh density in the *solid* domain had little impact on the energy conservation in the fluid domain (#2), though a finer solid mesh does give improved representation of the heat flux computed on the solid surface (#1).

By using a refined mesh near the fluid-solid interface, the energy conservation between the heat flux on the interface and the volume integral in solid domain can reach 99.8%. In order to loosen the requirement on solid mesh, the total power produced in the solid domain can be transferred from MOOSE to NekRS for energy conservation since upper and lower boundaries are assumed to be insulated. In other words, a Receiver postprocessor can be used with a default value of the imposed power, as opposed to a SideDiffusiveFluxIntegral to actually compute the heat flux. The heat fluxes from MOOSE heat conduction module can be adjusted inside Cardinal to ensure that the heat flux used in NekRS can be made consistentwith the flux integral in the MOOSE heat conduction module.

a) coarse mesh b) Refined mesh

Figure 4.5. Meshes with different density in solid domain

Table 4.5. Energy conservation study with different mesh density in solid domain

Case	Coarse mesh	Refined mesh
Volume integral of power source in solid domain (W)	11.70	11.71
Heat flux integral on interface (W)	11.37	11.69
Energy conservation error, #1 (%)	**2.82**	**-0.17**
Cp (J/g·K)	4.20	4.20
Mass flow rate (g/s)	0.85	0.85
Average outlet temperature (K)	283.19	283.28
Energy calculation from enthalpy rise (W)	11.33	11.65
Energy conservation error, #3 (%)	**0.34**	**0.35**

Different mesh structures in the fluid domain can lead to different post-processing error and resolution error. To understand the impact of the mesh structure in the fluid domain on energy conservation, different mesh structure shown in Figure 4.6 are used. The detailed structures for different meshes are listed in Table 4.6. From mesh1 to mesh3, the mesh density is increased in three dimensions. The fourth mesh uses same configuration as mesh3 but with higher polynomial order. Table 4.7 lists the results with different meshes in the fluid domain. The error of energy conservation decreases with increase of mesh density. The mesh density has slight impact on the calculation of mass flow rate and average outlet temperature.

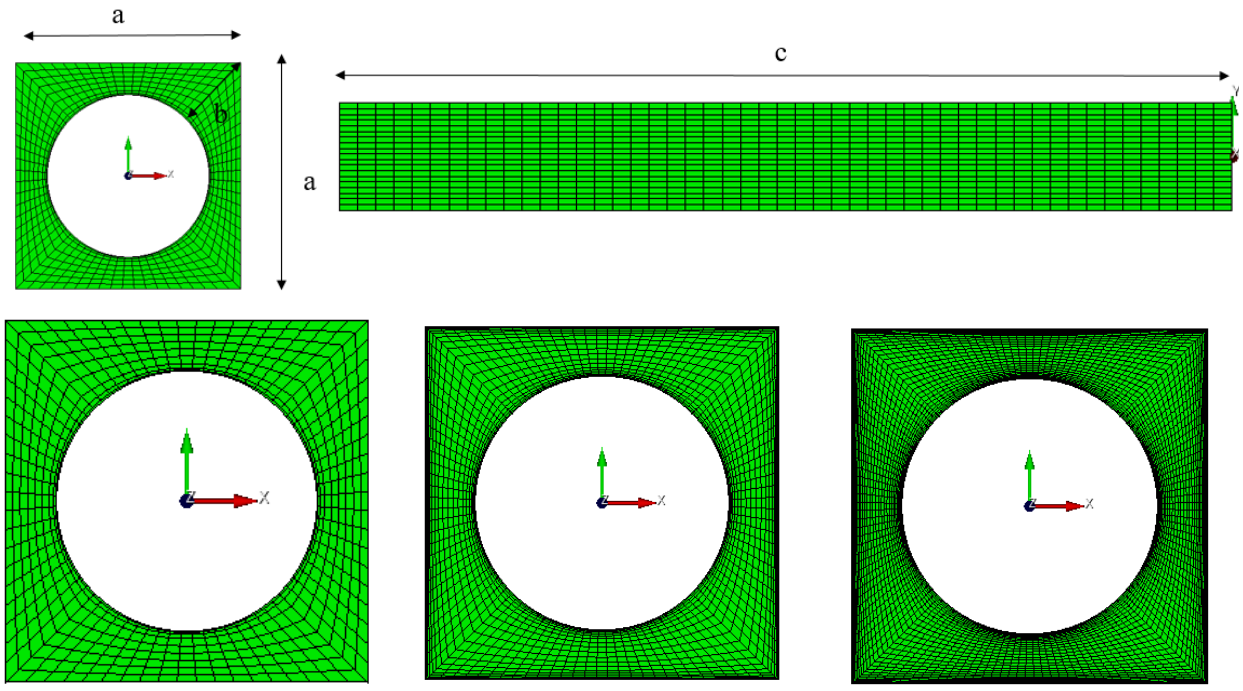

Figure 4.6. Meshes with different density in fluid domain

Table 4.6. Meshes with different structures in fluid domain

Case	Polynomial order	Mesh configuration (a x b x c)	Element number

Mesh 1	5	20 x 10 x 50	40,000
Mesh 2	5	30 x 20 x 80	192,000
Mesh 3	5	40 x 30 x 100	480,000
Mesh 3(high order)	7	40 x 30 x 100	480,000

Table 4.7. Energy conservation study with different fluid meshes with water flow

Case	Mesh1	Mesh2	Mesh3	Mesh3 (high order)
Volume integral of power source in solid domain (W)	11.71	11.71	11.71	11.71
Solid heat flux integral on interface (W)	11.69	11.69	11.69	11.69
Cp (J/g·K)	4.20	4.20	4.20	4.20
Mass flow rate (g/s)	0.846	0.853	0.853	0.854
Average outlet temperature (K)	283.28	283.25	283.26	283.26
Energy calculation from enthalpy rise (W)	11.651	11.664	11.677	11.680
Energy conservation error, #3 (%)	**0.35**	**0.23**	**0.12**	**0.10**

The mesh sensitivity study was further performed with different inlet mesh density in fluid domain (Figure 4.7). The results (Table 4.8) indicates that the energy conservation error (#3) significantly reduces to 0.064% by using refined inlet mesh in fluid domain. If we assign a constant velocity (which is unrealistic) at inlet, the velocity will need some length to develop to a parabolic velocity profile with zero velocity at the wall, which makes the velocity gradient very big near inlet. Refining the mesh near the inlet can make the postprocessing more accurate which impact the energy losss. This also explains why the refined mesh (Table 4.7) in fluid domain can reduce the error of energy conservation. To some extent, it's due to the refinement of the inlet mesh.

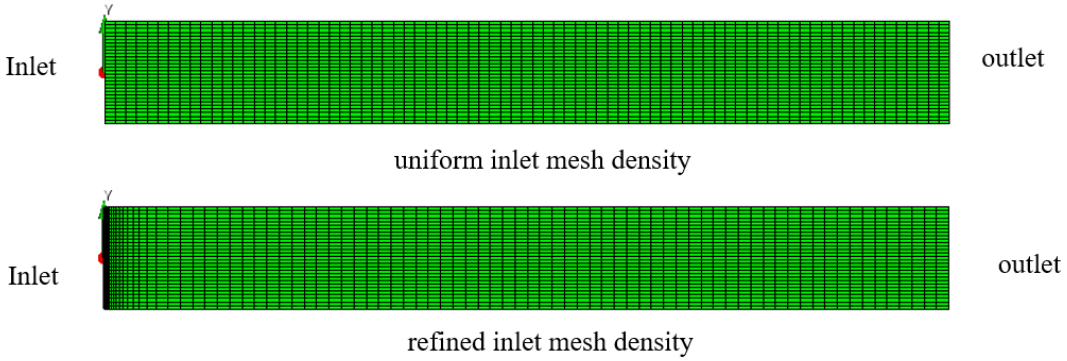

Figure 4.7. Meshes with different inlet density in fluid domain

Table 4.8. Energy conservation study with different mesh in fluid domain with water flow

Case	uniform inlet mesh density	refined inlet mesh density
Volume integral of power source in solid domain (W)	11.707	11.707
Heat flux integral on interface (W)	11.691	11.691

Cp (J/g·K)	4.2	4.2
Mass flow rate (g/s)	0.853	0.853
Average outlet temperature (K)	283.254	283.260
Energy calculation from enthalpy rise (W)	11.664	11.684
Energy conservation error, #3 (%)	**0.235**	**0.064**

It is not necessary to refine the mesh near the inlet when *water* is used as coolant. The recycling boundary condition or a non-heated region in front of inlet can also be considered as a good solution for providing a realistic velocity profile at inlet to ensure the energy conservation. However, when *lead* is coolant, simply refining the inlet meshing in fluid domain has limited improvement on energy conservation, due to different fluid properties between water and lead. Heavy liquid metal flow, such as lead, has a much smaller Prandtl number than water. The thicker thermal boundary layer leads to a more significant entrance effect when applying uniform velocity boundary condition. The study shows that an unheated entrance domain can successfully reduce the error of energy conservation to 0.0017% (Table 4.9). An unheated entrance is when a portion of the domain is designated as an unheated region to let the flow develop. As the flow enters the heated regions, the velocity profile is more parabolic or more realistic.

Table 4.9. Energy conservation study with model configuration in fluid domain for lead flow

Case	Inlet refined mesh + constant velocity	Uniform inlet mesh + unheated entrance domain
Volume integral of power source in solid domain (W)	11.707	5.854
Heat flux integral on interface (W)	11.691	5.846
Cp (J/g·K)	0.144	0.144
Mass flow rate (g/s)	8.876	8.876
Average outlet temperature (K)	702.021	697.567
Energy calculation from enthalpy rise (W)	11.545	5.846
Energy conservation error, #3 (%)	**1.249**	**0.002**

The mesh sensitivity study on energy conservation reveals the following:

- Energy conservation studies can be performed with small power to speed up the calculation.
- By using refined mesh near the interface in solid domain, the energy conservation between the heat flux on the interface and the volume integral of power source can reach 99.8% for this application. Even finer meshes can be used to achieve tighter agreement.
- The error of energy conservation decreases with the increase of fluid mesh density, especially when the mesh near the inlet is refined. When the coolant is water, as long as the mesh near inlet is refined, the error of energy conservation is negligibly small.
- When the coolant is lead, the entrance effect for lead is more significant than that for water. Simply refining the inlet meshing in fluid domain has limited improvement on energy

conservation. An unheated entrance domain or recycling boundary condition are necessary for energy conservation.

4.3 Coupled Calculation Results

Full coupled calculations were performed for the single pin-cell problem with duct shown in Figure 4.1. For meshes, the exact one shown in Figure 4.1 was used for Griffin and 31, 20, and 6 radial meshes in the fuel rod, coolant and duct regions were used for heat conduction. Mesh1 in Figure 4.6 was used for the problem without duct in the NekRS calculation, and mesh1 with boundary layers near duct was used for the problem with duct. Both NekRS meshes used the 4^{th} polynomial order. Griffin and heat conduction calculations used 1 cm-height meshes and NekRS calculation used 0.4 cm-height meshes.

The numbers of polar and azimuthal angles in DFEM-SN for Griffin were 2 and 3, respectively, and the first Legendre order was used for anisotropic scattering. The relative error tolerance of Richardson iterations was set as 0.001, and that of fixed point iterations was set as 0.0001. The L2-norm of fluid temperature computed in the heat conduction solve was set as the convergence metric of fixed-point iterations. (User-specified tolerance can be used only for fixed-point iteration loop, not for Richardson iteration loops.) For heat conduction, relative and absolute nonlinear tolerances of 1E-6 and 1E-8 were used, respectively, with relative linear tolerance of 1E-9 and 50 maximum linear iterations.

The initial coupled calculation consists of two calculations. The first coupled calculation solved for energy conservation with NekRS, with the fluid velocity simply set to a fixed value to approximate the actual velocity distribution. This approximation is possible for incompressible, constant-property flows. The time step of NekRS was 8.638 milli-seconds and that of heat conduction solve was 172.75 milli-seconds (20 NekRS time steps). The NekRS time step size was adequate since the Courant number was 0.12. The second coupled calculation turned on the velocity calculation in NekRS and used the final solution of the first calculation as the initial condition. In the second calculation, time step sizes of both NekRS and heat conduction solve were reduced by 10 times, resulting in further reduction in the Courant number to 0.01. This two-stage approach was used to reduce overall computational time by providing a reasonable temperature initial condition to the full solve, while also providing a fast-running initial problem for debugging.

Figure 4.8 shows the NekRS temperature evolution over time together with the fixed-point and Richardson iteration errors. In the first calculation, a small oscillation in the maximum fluid temperature was observed in the almost converged time range. This oscillation was not captured in the convergence check of fixed-point iterations since only the every end point of this oscillation was checked and its contribution to L2 norm of entire fluid temperature was small. In the second calculation, the velocity distribution was updated using the converged temperature distribution as the initial condition. While the Griffin calculation quickly converged, temperature distribution didn't change, resulting in termination of calculation in a short time. One noteworthy observation in the second calculation was that the small oscillation in the maximum fluid temperature disappeared.

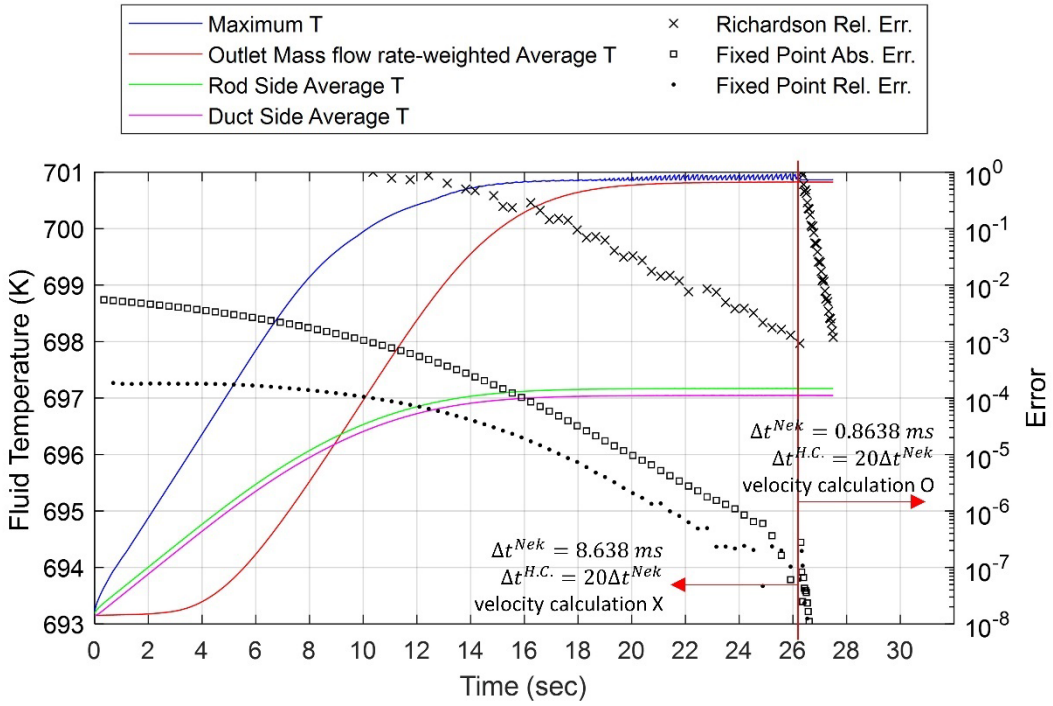

Figure 4.8. NekRS temperature convergence and Richardson/fixed point errors over time for the single pin-cell problem with duct.

The oscillation occurred due to the use of too large time step size in the heat conduction solve even though the heat conduction equation didn't have a time-derivative kernel, but only the time-dependent boundary condition. This was confirmed by performing an additional calculation in which the time step size of the heat conduction solve was decreased by 10 times while NekRS time step size was kept constant. The black and red data in Figure 4.9 show the result. It was confirmed that the oscillation disappeared with the use of smaller time step size in the heat conduction solve. In the real physical system, all physics are intimately connected. However, in the computational model, Picard iteration executes each physics domain one after the other, introducing non-physical "freezing" of the multi-physics simulation while each other physics domain is solved. If data is transferred too infrequently between applications, the coupled solve can diverge. The notion of "too infrequently" is very difficult to quantify and depends on the "strength" of the coupled physics feedback as well as intrinsic time scales of the system (such as thermal diffusivity). Oscillations have been observed in previous Cardinal simulations [24] and the technique of increasing the physics communication frequency was also successful in this work in eliminating the oscillations.

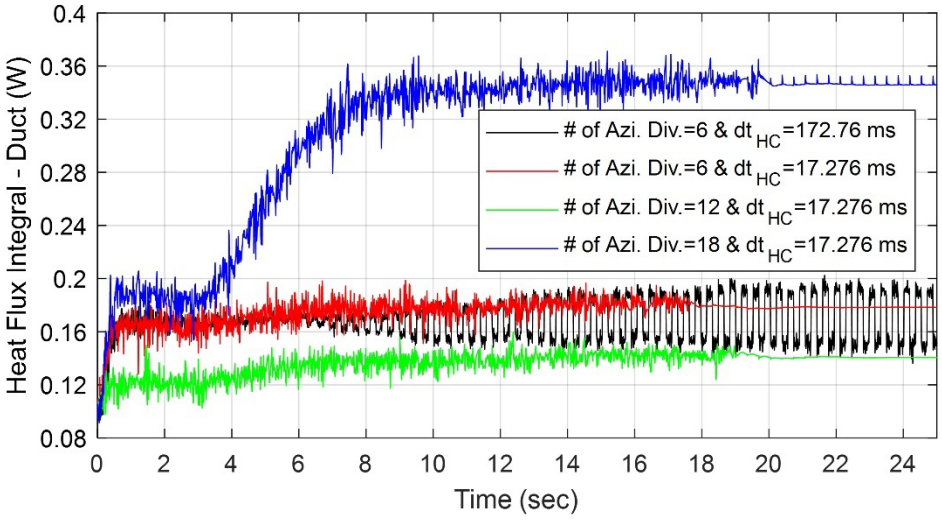

Figure 4.9. Flux integral values at the duct inner surface computed by NekRS when different meshes in the heat conduction module and different time steps were used.

The converged red value for heat flux integral on the duct inner surface in Figure 4.9, computed by NekRS, turned out to be significantly different from the computed value from the heat conduction module (refer to 40.02% error on row 2 of Table 4.10, which will be discussed shortly). Thus, two additional calculations (green and blue lines in Figure 4.9) were performed by increasing the number of azimuthal divisions in both Griffin and heat conduction meshes to better align the heat conduction mesh and NekRS mesh. Figure 4.10 shows the upper-right portion of mesh regions near duct (pink-duct, white-coolant) for different azimuthal mesh divisions: 6 (top), 12 (middle), and 18 (bottom) per side. Heat conduction mesh lines are black and NekRS mirror mesh lines are red. As illustrated, using 6 azimuthal divisions per side seems to be problematic. The coarser the mesh points are, the more likely *MultiAppNearestNodeTransfer* picks up values at too far points of the other mesh from the target point, resulting in large error.

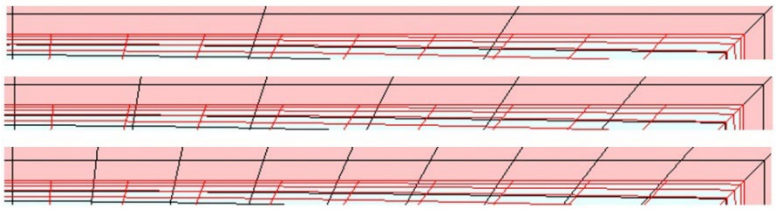

Figure 4.10. Upper-right portion of mesh near duct (pink-duct, white-coolant) for different azimuthal mesh divisions: 6 (top), 12 (middle), and 18 (bottom) per side. Black lines are HC and red lines are NekRS.

Table 4.10 shows heat flux integrals of heat conduction scaled to match the total power in the solid domain and integrated values of transferred heat flux in NekRS mirror mesh for different azimuthal divisions. The rightmost column is the output of a *NekHeatFluxIntegral* postprocessor.

Once heat flux values are transferred to NekRS, their integrated values over the NekRS mirror mesh will not match the integrated values in the heat conduction mesh. Thus, the scaling factor is multiplied to the transferred values to match the integral value in the heat conduction mesh. As explained in Section 3.1, however, the total power in the solid domain was used instead of the heat flux integral value in calculating the scaling factor. In this way, the total power will be used in NekRS to heat up fluid. However, energy in each region cannot be preserved since only a single scaling factor is used to preserve the total power only. Thus, each of heat flux integral values on rod surface and duct inner surface has an error with the same absolute magnitude of opposite sign as shown in the table. Errors shown indicate how well data was transferred between the heat conduction mesh and the NekRS mirror mesh. Note that the sum of integrated heat flux in NekRS (right two columns) is the same as the sum of original rod and duct powers (sum of columns 2 and 3), ensuring the total energy conservation regardless of heat flux errors distributed over different solid-fluid interfaces.

We refer back to the large error in duct heat flux using the mesh with 6 azimuthal divisions per side for heat conduction. The result was improved with the use of 12 azimuthal divisions per side (8.43% error in Table 4.10) but was much more aggravated with the use of 18 (159.40% error in the same table). This could occur because there are actually two transfers occurring (heat flux to NekRS, wall temperature to MOOSE heat conduction). A large mismatch in resolution in either direction (finer for NekRS vs. finer for MOOSE) could cause the nearest node transfer to pick up values "far" from the correct value. *MultiAppInterpolationTransfer* was tried instead, but integrals of transferred values in two meshes were again not consistent. With the nearest node transfer, it seems that having an approximate match in the mesh density is ideal for reducing mismatches in the energy conservation (#1 and #2 as discussed earlier).

Table 4.10. Heat flux integrals (heat conduction) scaled to match the total power and integrated values of transferred heat flux in NekRS mirror mesh for different meshes.

Azi. Div.	Scaled heat flux integral (W) in heat conduction mesh		Fluid wall heat flux (W)	
	Rod	Duct	Rod	Duct
6[a)]	9.85453	0.12982	9.77783 ~ 9.84578 (-0.09% ~ -0.78%)	0.13614 ~ 0.20342 (+4.86% ~ +56.69%)
6[b)]	9.85668	0.12766	9.79933 (-0.58%)	0.17875 (+40.02%)
12[b)]	9.85457	0.12978	9.84005 (-0.15%)	0.14072 (+8.43%)
18[b)]	9.85114	0.13321	9.63633 (-2.18%)	0.34555 (+159.40%)

a) $dt_{H.C.} = 172.76\ ms$, b) $dt_{H.C.} = 17.276\ ms$

Table 4.11. Comparison of power, heat flux integral in heat conduction module and NekRS temperature for meshes of different azimuthal divisions

Azi. Div.	Max. Fluid T (K)	Avg. Outlet Fluid T (K)	Power (W)		Heat Flux Integral MOOSE (W)	
			Rod	Duct	Rod	Duct
6[a)]	707.81	707.68 (0.0006%)	9.86405	0.12030	9.77771 (-0.88%)	0.12881 (+7.08%)
6[b)]	707.70	707.45 (-0.033%)	9.86405	0.12030	9.77770 (-0.88%)	0.12664 (+5.27%)
12[b)]	707.71	707.55 (-0.018%)	9.86406	0.12029	9.77773 (-0.88%)	0.12877 (+7.05%)
18[b)]	707.73	707.60 (-0.012%)	9.86407	0.12028	9.77773 (-0.88%)	0.13222 (+9.93%)

a) $dt_{H.C.} = 172.76\,ms$, b) $dt_{H.C.} = 17.276\,ms$

Table 4.11 summarizes errors in fluid temperature rise, and heat flux integrals computed by the MOOSE heat conduction module. First of all, note that the sum of rod and duct powers is not equal to the total power (10 W) because of heat generated from the coolant region, though small. Based on the total power produced in rod and duct, the mass flow rate-weighted averages of outlet temperature were accurately predicted with -0.0006% for the initial calculation (6 azimuthal mesh divisions with $dt_{H.C.} = 172.76\,ms$) and -0.012 ~ -0.033% for calculations using $dt_{H.C.} = 17.276\,ms$. The low -0.0006% error of the first calculation seems to arise from error cancellation. It is noteworthy that the impact of errors in rod and duct heat fluxes shown in Table 4.10 on maximum temperature was negligible – less than 0.1 K - and thus, those errors are not of a significant concern. It should also be noted that such errors didn't affect the energy conservation either since the coupling scheme was designed to preserve the total power produced in the solid domain. Right two columns show heat flux integral values computed by MOOSE. They cannot exactly match the power produced in each of region due to mesh discretization error. As already seen from the sensitivity test of Figure 4.3, increasing azimuthal divisions in a mesh doesn't reduce the heat flux solution errors, but increasing the radial or axial refinement could be used to drive these errors lower. Using smaller times step size or azimuthally finer meshes does not reduce errors much.

Figure 4.11 to Figure 4.13 are axial distributions of heat flux on the fuel rod and duct inner surfaces, those of power and temperature in each region, respectively. Since the up and bottom boundaries in the neutronics calculation were reflective to avoid large leakage in this simple test problem, power shape is almost flat in the axial direction and symmetric due to negligible temperature changes and negligible feedback effects accordingly.

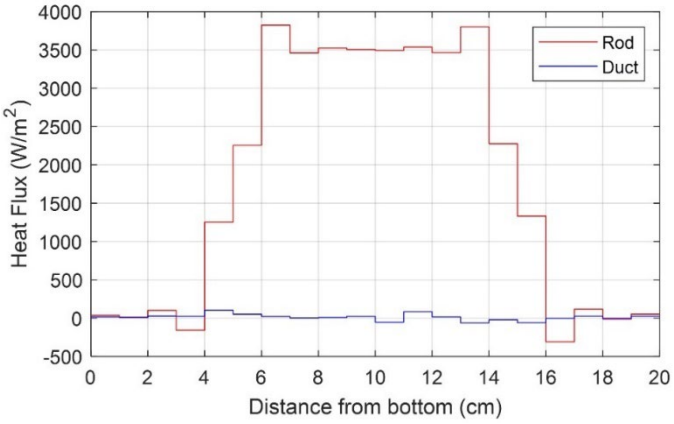

Figure 4.11. Axial distribution of heat flux on fuel rod and duct inner surfaces for single pin-cell problem.

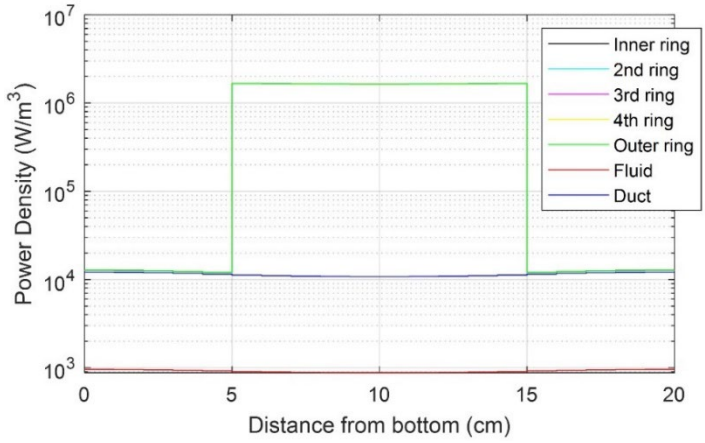

Figure 4.12. Axial distribution of power in each radial region for single pin-cell problem.

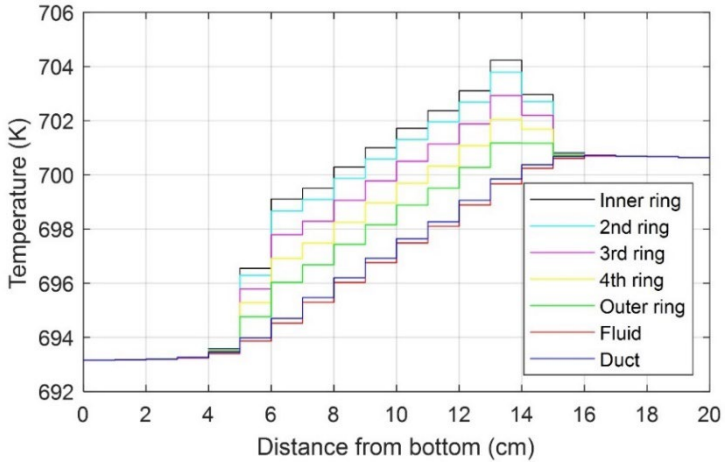

Figure 4.13. Axial distribution of average temperature in each radial region for single pin-cell problem.

5 Lead-Cooled Fast Reactor: 7-Pin Assembly

5.1 7-Pin Assembly Problem Specification

This section describes the geometry for a 7-pin LFR bare bundle. This geometry is a reduced-size version of the fuel bundles provided in the prototype 127-pin fuel bundle design of the Westinghouse LFR [25]. The spacing between the outermost row of pins and the duct matches the nominal design. Relevant dimensions are summarized in Table 5.1. Heat is produced in the annular fuel region and transfers by conduction across the fuel-clad gap to the cladding (Figure 5.1). In order to simplify the model, the gap is not modelled directly in the simulation. The gap thickness is included in the cladding thickness, with the thermal conductivity taken as a mixture of the helium and clad thermal conductivities. (A future approach may include separate, unconnected mesh blocks for fuel and cladding which would allow MOOSE HCM to impose heat transfer across an unmeshed gap.) Helium gas is present in the empty regions. Heavy liquid metal flows around the pins to remove the fission heat.

Table 5.1. Geometric dimensions of the 7-pin cell problem

Parameter	*Value*	*unit*
Fuel inner diameter	0.404	cm
Fuel outer diameter	0.864	cm
Clad outer diameter	1.0807	cm
Pin pitch	1.34	cm
Assembly flat-to-flat (inner duct)	3.7164	cm
Duct wall thickness	0.353	cm
Fuel length	106.072	cm
Upper and lower reflector length	20 (each)	cm

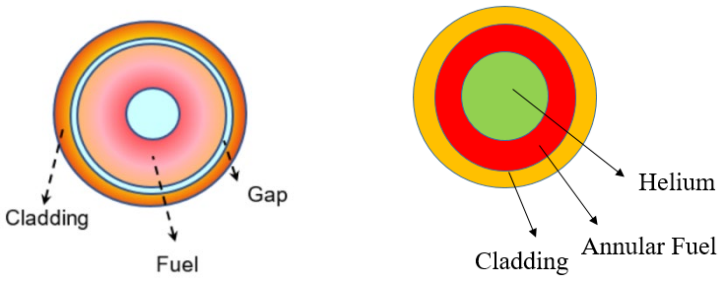

Figure 5.1. Configuration of the annular fuel with and without gap

Specifications and material properties are listed in Table 5.2. The lead coolant flows along the whole length of 1.46272 m from inlet to outlet. The inlet velocity is assumed to be about 0.1 m/s to make the Reynolds number 780. The laminar flow can benefit the test process and the efficiency of the calculation, since the purpose of this calculation is mainly for testing purposes.

Table 5.2. Specifications and material properties of the 7-pin cell problem

Parameter	*Value*	*unit*
Total power	29.134	kW
Reflector material (Cylindrical pin with the same outer radius as fuel pin cladding)	Clad	-
Coolant density (Function of temperature in Griffin)	10,401 @ 810 K	Kg/m^3
Coolant viscosity	0.0171	Pa·s
Coolant specific heat	144.19	J/kg·K
Coolant thermal conductivity	18.09	W/m·K
Fuel thermal conductivity	1.882	W/m·K
Clad thermal conductivity	21.6	W/m·K
Helium thermal conductivity	0.251	W/m·K
Inlet velocity	0.1186	m/s
Inlet temperature	693.15	K
Reynolds number	780	-
Peclet number	106	-

5.2 Griffin-Heat conduction-NekRS model

Radial views of the MOOSE-based meshes of Griffin and heat conduction and the NekRS mirror mesh are shown in Figure 5.2. These meshes are generated inline and NekRS mirror mesh is generated automatically by Cardinal based on the NekRS native mesh of ".re2" type. As shown in the single pin-cell problem, a coarse mesh was used for Griffin calculation since power solution is not sensitive to mesh density. Meanwhile, a very fine mesh was used for heat conduction solve since temperature is our target quantity. Note that the fluid mesh does not participate in the heat conduction solve but is used for storing fluid temperature received from NekRS, calculating the L2-norm of it and transferring it to Griffin.

The inner helium gas region was considered as a solid region in solving the heat conduction equation. The NekRS native mesh used pure hexahedral Hex20 elements. Axially, Griffin used 5 cm-height meshes in the reflector regions and 5.3036 cm-height meshes in the active fuel region, resulting in total 28 axial meshes. The heat conduction module used the axially twice finer mesh of Griffin. For NekRS, the mesh is refined near the solid-fluid interfaces. The polynomial order of the element is 4 for laminar case and will be further increased for turbulent case. The number of layer in axial direction is 100. It's not necessary to refine the mesh near inlet since a non-heated region has been added in the model to have the fully developed velocity profile as the flow enters the heated region.

Two polar and three azimuthal angles were used in Griffin's DFEM-SN solver, , and the first Legendre order was used for anisotropic scattering. The relative error tolerance of Richardson iterations was set as 0.001, and that of fixed-point iterations was set as 0.0001. The L2-norm of fluid temperature computed in the heat conduction solve was set as the convergence metric of fixed point iterations. For heat conduction, relative and absolute nonlinear tolerances of 1E-6 and 1E-8 were used, respectively, with relative linear tolerance of 1E-9 and 50 maximum linear iterations.

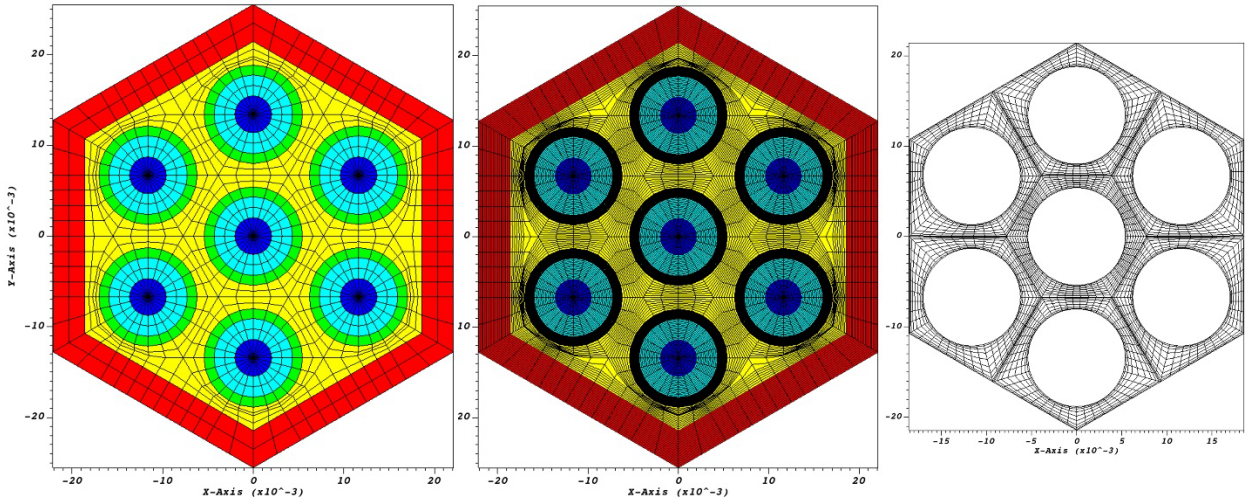

Figure 5.2. Radial views of Griffin mesh (Left), heat conduction mesh (middle), and NekRS mirror mesh (right).

5.3 Results

The calculation procedure consists of three steps. First, a NekRS standalone calculation was performed to provide a good initial temperature/velocity condition to start from. For this, a crude heat flux distribution was obtained from Griffin where just inlet temperature condition was assigned to all regions. In this calculation, dimensionless NekRS time step size of 1E-4 in non-dimensional terms (9.106 micro-seconds) was used. Second, a full coupled simulation started with a frozen velocity calculation using the solution of the NekRS standalone calculation as the initial condition. In this step, the time step size of NekRS was 0.9106 milli-seconds and that of heat conduction solve was 18.212 milli-seconds (20 NekRS time steps). The NekRS time step size was a bit high since the courant number was 2.71. The third coupled calculation was with velocity calculation in NekRS using the final solution of the second calculation as the initial condition. In the second calculation, time step sizes of both NekRS and heat conduction solve were reduced by 10 times, resulting in reduced courant number of 0.27.

Figure 5.3 shows the NekRS temperature evolution over time together with the fixed point and Richardson iteration errors for the second and third calculations explained in the previous paragraph. The first calculation to obtain the initial condition for the second calculation is not shown in the figure. At 11.1 second where “Restart calculation” is remarked, the simulation was terminated in the middle of calculation due to the wall time limit set in a bash file to run the simulation on a HPC cluster. Note that Richardson iteration started at 3 second when the relative error of the L2-norm of fluid temperature went down below 1E-4, continued until 5.3 second when it went back above 1E-4, and restarted at 9.8 second. In the second calculation where the velocity was calculated in NekRS using a smaller time step, the temperature didn’t change and the simulation was terminated in a short time as in the single pin-cell problem.

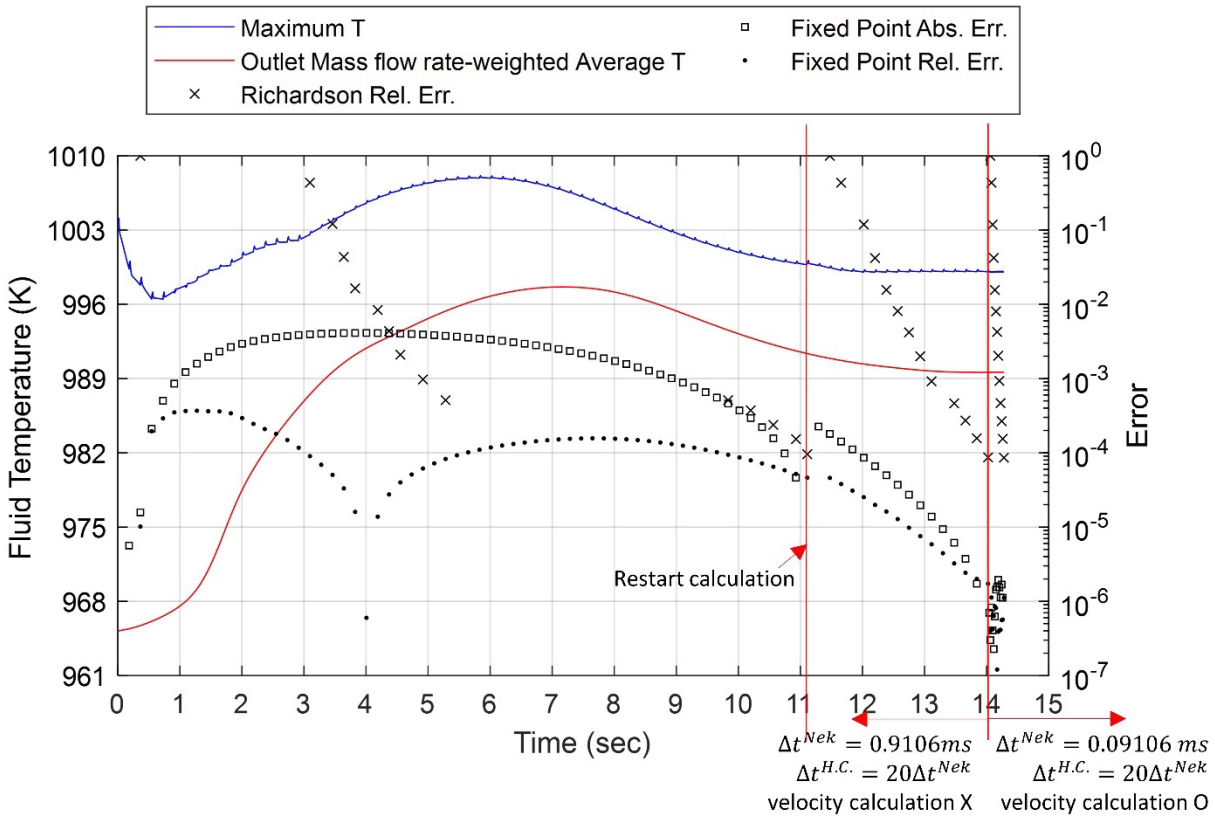

Figure 5.3. NekRS temperature convergence and Richardson/fixed point errors over time for the seven pin-cell problem with duct.

Figure 5.4 shows the heat flux integral values at the rod and duct inner surfaces computed by NekRS. Oscillations in both values are observed. Considering the observation in a problem without duct which is not explained in this report that this behavior was not shown, it is highly likely that heat flux integral of the duct inner surface oscillates and that of the rod surface is forced to oscillate due to the normalization to the total power. Peaks occur at every time step when Griffin is involved. The impact of this instability in conjugate heat transfer at the duct inner surface seems to be small. As seen in Figure 5.3, the oscillation in the maximum temperature due to oscillation of heat flux at duct inner surface was very small within 0.2 K. Nevertheless, this issue needs to be fixed in the next fiscal year. Our single-pin simulations revealed that choosing more frequent NekRS-heat conduction communication reduced these oscillations. *Because we have changed the problem setup by adding more pins, we need to revisit the choice of 20 heat conduction time steps per NekRS time step.* The domain now has larger radial extent, and it is possible that the oscillations occur due to instabilities in the cross-plane direction that were less likely to appear in the single-pin case.

Despite this issue, Table 5.3 shows that the total energy was preserved. The mass flow-weighted average of outlet temperature was predicted within 0.004% compared to the hand-calculated value based on Eq. (1). As expected, heat fluxes calculated by the heat conduction solve have some errors due to the mesh discretization error. After this heat flux values are transferred to NekRS and scaled to match the total power produced in the solid domain, integrals of those transferred and scaled heat flux values are computed by NekRS for reporting purpose. As explained in Section 4.3, energy produced at each region cannot be produced, but only the sum of them is preserved. One potential

improvement in Cardinal could be to allow individual scaling of heat fluxes in different sidesets, as Cardinal currently only scales the heat flux on the union of all conjugate heat transfer sidesets. Table 5.4 indicates that fluid receives 0.53% more energy from the fuel rods and 40.99% less energy from the duct. This 40.99% error is caused by transfer of heat flux data from the heat conduction module to Cardinal, which needs to be investigated along with the oscillating heat flux issue. However, it should be noted that this error is only 0.5% of the total power, and the high relative error occurs just because the heat flux magnitude is quite small in the duct.

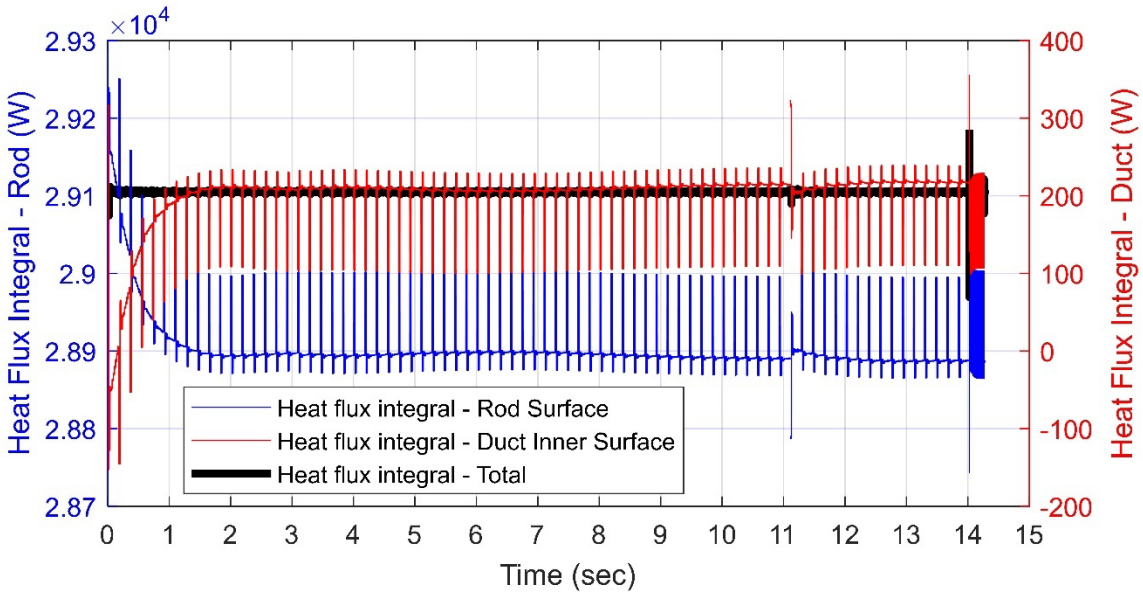

Figure 5.4. Flux integral values at the rod and duct inner surface computed by NekRS.

Table 5.3. Comparison of power, heat flux integral in heat conduction module and NekRS temperature for meshes of different azimuthal divisions

Max. Fluid T (K)	Avg. Outlet Fluid T (K)	Power (kW)		Heat Flux Integral MOOSE (kW)	
		Rod	Duct	Rod	Duct
999.33	989.59 (0.004%)	28.76934	0.33545	27.81683 (-3.31%)	0.35796 (+6.71%)

Table 5.4. Heat flux integrals of heat conduction scaled to match the total power and integrated values of transferred heat flux in NekRS mirror mesh for different azimuthal divisions

Scaled heat flux integral (kW) in heat conduction mesh		Fluid wall heat flux (kW)	
Rod	Duct	Rod	Duct
28.73502	0.36977	28.88630 (+0.53%)	0.21810 (-40.99%)

Figure 5.5 to Figure 5.7 are axial distributions of heat flux on the fuel rod and duct inner surfaces, those of power and temperature in each region, respectively. Since the upper and bottom boundaries in the neutronics calculation were vacuum, power and heat flux have a chopped cosine shape. Even though Doppler and fluid density feedback was considered, their impact in fast reactor was very small so that the shape is almost axially symmetric. Temperature increases almost linearly achieving about 296 K rise. Due to poor heat conductance of helium and fuel, temperatures are significantly higher than the other regions.

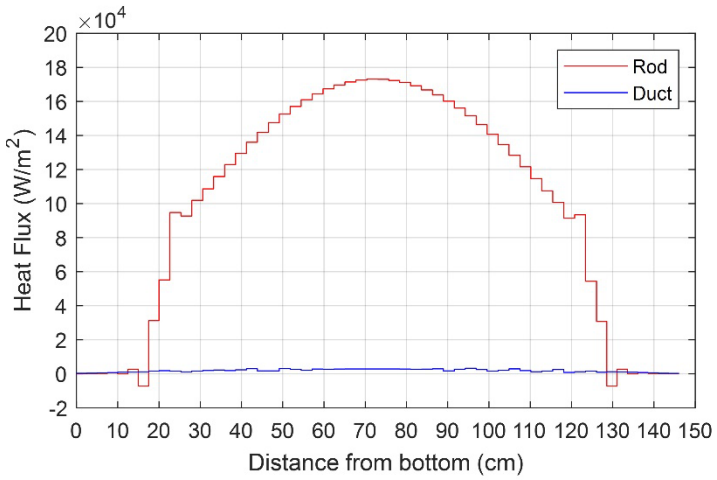

Figure 5.5. Axial distribution of heat flux on fuel rod and duct inner surfaces for 7 pin-cell problem.

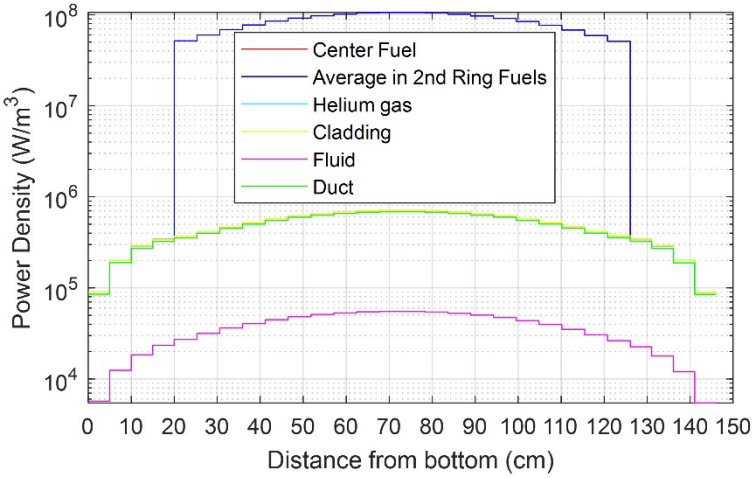

Figure 5.6. Axial distribution of power in each radial region for 7 pin-cell problem.

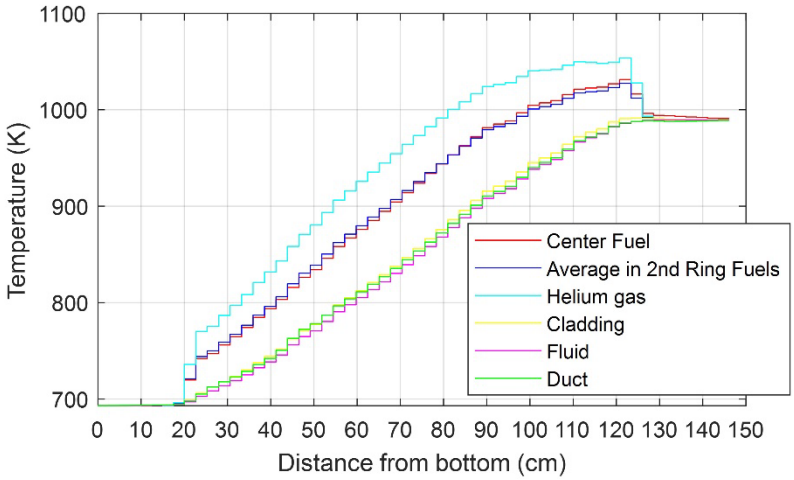

Figure 5.7. Axial distribution of average temperature in each radial region for 7-pin cell problem.

6 Code Development Recommendations to Enable Future Work

While not used in this present work, the list of priority HCFs for LFR are listed in Table 1.1. These HCFs have been evaluated previously with PROTEUS and Nek5000 [11]. Future work will evaluate these with MOOSE-based coupling for a larger assembly problem. However, not all can be evaluated given current limitations in the codes. The following list of recommendations includes code updates recommended for the nominal condition case as well as the stochastic tools workflow case.

Griffin

- The ability to check a user-specified convergence criteria in the Richardson iteration loop would ease the user's responsibility to ensure convergence of sub applications inside the fixed iteration loop for weakly coupled calculations. In this application, neutronics and CFD solutions are very weakly coupled together, so that neutronics convergence does not necessarily mean CFD convergence. Since CFD convergence is checked only inside the fixed iteration loop, the whole simulation ends if neutronics convergence is just satisfied at the outer iteration, resulting in false CFD convergence.

MOOSE

- In the MultiApp system, sub applications might need to be called just once every several time steps from the main application for a weakly coupled simulation. In this work, this capability would be very useful for the MultiApp hierarchy where the heat conduction module calls Griffin. Even though heat conduction solve needs to be more frequently executed than Griffin, there are good reasons that it is better for the heat conduction solve to be the main application calling neutronics calculation only when needed. In other words, it would be convenient to relax MOOSE's requirement that the sub-applications always run on every main application time step.

- When a sub application is called in every fixed-point iteration of this workflow, variables in the aux system are not restored even with the input parameter *keep_solution_during_restore* in *FullSolveMultiApp* being turned on. The input parameter keeps variables only in the nonlinear system. It would be good to support a new option like *keep_aux_solution_during_restore.*
- Any improvements to help the user debug potential issues in Transfers would be helpful such as user-friendly output to identify IDs that failed to find pairings, or coordinates of pairings themselves. It is difficult to identify the source of transfer discrepancies without additional diagnostic information. Specific issues with Transfers in this work follow:
    - In this work, transfer of a first order monomial elemental variable to a first order Lagrange nodal variable using *MultiAppMeshFunctionTransfer* was not properly done.
    - When using *MultiAppNearestNodeTransfer* in this workflow, transferred values were different depending on how "source_boundary" and "target_boundary" are specified.
    - Another case encountered was that a code died without error message right after 2nd transfer using *MultiAppNearestNodeTransfer* when "fixed_meshes = true." Turning off "fixed_meshes" didn't induce the error. There is a known issue regarding transfer correctness (i.e. not related to this seg fault discovered) when fixed_meshes is true, as documented here: https://tinyurl.com/24ka8dm3. In addition to our discovered seg fault, this issue will need to be resolved to efficiently use this data transfer, since fixed_meshes can dramatically reduce the total transfer cost.

Cardinal/NekRS

- Multiple NekRS instances cannot be populated under the stochastic tool driver. None of the operation modes (normal, batch-reset, and batch-restore) currently are supported because NekRS cannot run multiple instances within the same MPI communicator. Changes are required in NekRS to support this, as well as unique cache directories. To use the batch-restore mode of the MOOSE stochastic tools module, Cardinal must be able to fully backup and restore a full NekRS simulation. This requires identifying which data structures in NekRS to store and creating a function that stores this memory and can recreate a NekRS simulation when called.
- Currently, geometry perturbation cannot be applied to NekRS since a NekRS mesh needs to be generated separately in advance. It is recommended that NekRS be upgraded to directly use an Exodus mesh, which can be created using MOOSE mesh generators.
- Data perturbation cannot reach NekRS native inputs even though they can reach a Cardinal input. It is recommended to let Cardinal be able to control any relevant parameters in NekRS native inputs.

7 Conclusions

This year's effort was focused on establishing a multi-physics coupling workflow among three codes, Griffin, MOOSE heat conduction and NekRS. Since NekRS is not a MOOSE-based application, its MOOSE wrapper Cardinal was coupled. By using the same MOOSE framework and exactly the same environment in Griffin and Cardinal on an HPC cluster, the codes could be coupled through dynamic linking.

Various coupling schemes were investigated. Coupling schemes where NekRS is not on the lowest level in the MultiApps system were ruled out, as were coupling schemes where heat conduction and NekRS were not directly coupled except through Griffin. Two potential schemes remained: Griffin → Heat conduction → NekRS (scheme A) and Heat conduction → Griffin & NekRS as sibling applications on the same level. The latter, however, was not used in this work because the current version of MOOSE does not allow a main application to call its sub application only at specific time steps. If the time step size of the heat conduction solve needs to be small for avoiding numerical instability, the latter scheme would be computationally inefficient since Griffin needs to be called at every time step of heat conduction. If MOOSE is updated to permit more flexible instantiation of sub applications at time steps larger than the parent application, then this scheme would become more attractive than the former scheme. At this time, only the former scheme was used.

For scheme A, convergence studies were performed. The convergence control highly depends on the structure of the executioner, *SweepUpdate*, for DFEM-SN with CMFD calculation, which is the most computationally efficient solver for heterogeneous fine-mesh transport calculation in Griffin. Sub applications are called before and after CMFD calculations for timestep begin and timestep end under the fixed-point iteration loop, which is under the Richardson iteration loop where the DFEM-SN transport sweep is performed after the fixed-point iteration loop. While a user-specified tolerance can be used for convergence check in the fixed-point iteration loop, only the angular flux convergence was checked in the Richardson iteration loop. This means that it would be easy to meet false convergence especially for very weakly coupled simulation since angular flux would easily be converged without sub application's physics quantity being converged. The fact that the sub application is a very slow transient calculation makes it even easier to happen. Since our target quantity is temperature, not neutron angular flux, the convergence should be very carefully controlled under this coupling hierarchy. Thus, the tolerance for temperature convergence in the fixed-point iteration loop needs to be very tight, which means that the first transport sweep is executed after temperature solution of NekRS is almost converged. Cons of this approach is that, after NekRS temperature was almost converged, costly NekRS calculation needs to be performed for iterations that the coupled code system needs to go through for purely neutronics solution convergence process. This inefficiency would be loosened if Griffin is updated to check a user-specified tolerance in the Richardson iteration loop. In this case, by limiting the number of maximum fixed-point iterations per Richardson iteration, neutronics solution can be converged simultaneously together with solutions of sub applications without possibility of their non-convergence.

In this coupling scheme, Griffin receives solid and fluid temperatures from heat conduction for Doppler and fluid density feedback and sends power density to the heat conduction module. The heat conduction module receives solid-fluid interface temperatures from Cardinal as the boundary condition and sends heat flux at those interfaces back to Cardinal. Solid-fluid interface temperatures were defined in the auxiliary system, but they need to be backed up by transfer to Griffin to avoid

the reset of their values at every fixed-point iteration. The heat conduction module also receives fluid temperature from Cardinal and passes them to Griffin. When the heat conduction module sends heat flux to Cardinal, the total power produced in the domain inside the fluid-solid interfaces of interest and insulated boundary was sent together for normalization to ensure energy conservation.

The coupled scheme was tested for a single pin-cell and seven pin-cell problems with duct. For both problems, the energy conservation was confirmed by checking that the temperature rise of fluid is consistent with the total power produced. Some issues were observed in heat flux at duct inner surface: oscillation over time and large errors from the power produced inside the duct. These issues could be resolved by using smaller time step sizes and an azimuthally refined mesh in the heat conduction solve for some cases. For the 7-pin case, we may need to use a finer time step size to further resolve the oscillations, and this issue needs to be investigated in more depth.

Last but not least, some issues were identified in utilizing STM to streamline the HCF evaluation process. The resolution of these issues would involve implementing the ability to handle native NekRS inputs from STM, using a MOOSE mesh generator to build NekRS mirror and native meshes, and enabling multiple runs of NekRS in a single execution of the stochastic tool.

References

1. C. R. Stanek, Overview of DOE-NE NEAMS Program. United States: N. p., 2019. Web. doi:10.2172/1501761
2. C. Lee, J. Ortensi, et al., "Griffin software development plan," ANL/NSE-21/23, INL/EXT-21-63185, June 2021.
3. Paul Fischer, Stefan Kerkemeier, Misun Min, Yu-Hsiang Lan, Malachi Phillips, Thilina Rathnayake, Elia Merzari, Ananias Tomboulides, Ali Karakus, Noel Chalmers, and Tim Warburton. 2021. NekRS, a GPU-Accelerated Spectral Element Navier-Stokes Solver. arXiv:2104.05829 [cs.PF]
4. A. J. Novak, et al, "Coupled Monte Carlo and thermal-fluid modeling of high temperature gas reactors using Cardinal", Annals of Nuclear Energy, Volume 177, 2002, https://doi.org/10.1016/j.anucene.2022.109310..
5. MOOSE Website, "Heat Conduction Module" https://mooseframework.inl.gov/modules/heat_conduction/index.html, Retrieved September 2022.
6. MOOSE Website, "Stochastic Tools Module", https://mooseframework.inl.gov/modules/stochastic_tools/index.html
7. E. R. Shemon, Y. Yu, and T. K. Kim, "Applications of SHARP Toolkit to SFR Challenging Problems: Evaluation of Hot Channel Factors and Demonstration of Zooming Capability," ANL/NSE-18/4, Argonne National Laboratory, September 30, 2018.
8. Y. Yu, E. Shemon, T. K. Kim, "Hot channel factor evaluation for sodium-cooled fast reactors with multi-physics SHARP toolkit", Proc. of NURETH 2019, Portland, OR, Aug 18-23, 2019.
9. E. R. Shemon, Y. Yu, and T. K. Kim, "High Fidelity Multiphysics Strategies with PROTEUS and Nek5000 for Fast Reactor Applications", Proc. of M&C 2019, Portland, OR, Aug 25-20, 2019
10. Y. Yu, E. Shemon, T. K. Kim, and E. Merzari, "Evaluation of Hot Channel Factor for Sodium-Cooled Fast Reactors with Multi-Physics Tooolkit", J. Nucl. Eng & Design. V 365, Aug 15, 2020, DOI: 10.1016/j.nucengdes.2020.110704
11. E. R. Shemon, Y. Yu, and T.K. Kim, "Demonstration of NEAMS Multiphysics Tools for Fast Reactor Applications", ANL/NSE-20/25, Argonne National Laboratory, August 31, 2020.
12. Y. S. Jung and C. H. Lee, "Updates and Verifications of the PROTEUS System in FY19," ANL/NSE-19/34, Argonne National Laboratory, September 30, 2019.
13. P. F. Fischer, J. W. Lottes, and Stefan G. Kerkemier, Nek5000 Web Page, http://nek5000.mcs.anl.gov, 2008.
14. J. Y. Ku, L. K. Chang and D. Mohr, "Analysis of IFR Driver Fuel Hot Channel Factors," Fourth International Topical Meeting on Nuclear Thermal Hydraulics, Operation and Safety, Taipei, Taiwan, April 5-8, 1994.

15. C.J. Permann, et al., "MOOSE: Enabling massively parallel multiphysics simulation", SoftwareX 11 (2020) 100430.
16. E. Shemon, Y. Yu, H. Park, and C. Brennan, "Assessment of Fast Reactor Hot Channel Factor Calculation Capability in Griffin and NekRS", ANL/NSE-21/42, Argonne National Laboratory, September 15, 2021.
17. D. Gaston, et al, "Physics-Based multiscale coupling for full core nuclear reactor simulation", Annals of Nuclear Energy V84, pp 45-54, 2015.
18. Y. Wang, et al., "Performance Improvements to the Griffin Transport Solvers," INL/EXT-21-64272, ANL/NSE-21/51, Argonne National Laboratory and Idaho National Laboratory, September, 2021.
19. Z. Prince, Idaho National Laboratory, Private Communication, September 2022.
20. C. H. Lee and W. S. Yang, "MC2-3: Multigroup Cross Section Generation Code for Fast Reactor Anaysis," ANL/NE-11-41 Rev. 1, Argonne National Laboratory, January, 2012.
21. D. S. Medina, A. St-Cyr, T. Warburton, OCCA: A unified approach to multi-threading languages, preprint arXiv:1403.0968
22. Paul K. Romano, Nicholas E. Horelik, Bryan R. Herman, Adam G. Nelson, Benoit Forget, and Kord Smith, "OpenMC: A State-of-the-Art Monte Carlo Code for Research and Development," *Ann. Nucl. Energy*, **82**, 90–97 (2015)
23. E. Shemon, K. Mo, Y. Miao, Y. S. Jung, . Richards, A. Oaks, and S. Kumar, "MOOSE Framework Enhancements for Meshing Reactor Geometries", Proceedings of PHYSOR, Pittsburgh, PA May 15-20, 2022.
24. A.J. Novak, A. Chaube, D. Shaver, and C. Brooks, "Validation of NekRS-MOOSE Conjugate Heat Transfer Coupling for a 7-Pin Bare Bundle," *Proceedings of the American Nuclear Society* (2022)
25. G. Grasso, A. Levinsky, F. Franceschini, P. Ferroni, "A MOX-fuel core configuration for the Westinghouse Lead Fast Reactor," ICAPP 2019 – International Congress on Advances in Nuclear Power Plants, France, May 12-15 2019.

Nuclear Science and Engineering Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 208
Argonne, IL 60439

www.anl.gov