## ABSTRACT

BULLERWELL, LANCE EDWARD. A Multiphysics Coupling Scheme between OpenMC and OpenFOAM for High-Fidelity, Flexible Multi-Physics Analysis of Nuclear Reactors. (Under the direction of Dr. Jason Hou).

The advancement of Generation-IV nuclear reactor designs has demonstrated that the flexible multi-physics modelling capability is of paramount importance for these advanced systems to be deployed in the coming years. The currently available capabilities usually do not allow for the accurate modelling of variant fuel geometries and generation of high-fidelity simulation results to guide the reactor analysis in the design phase. The multi-physics coupling between the Monte Carlo reactor physics calculation and the Computational Fluid Dynamics (CFD) thermal-hydraulics calculations presents a potential solution to this problem. However, the large computational cost associated with the iterative execution of Monte Carlo and CFD, particularly in a Picard iteration, as well as the level of difficulty for efficient data communication between the two systems are the main reasons for this type of coupling not being commonplace.

In this study, a coupling scheme between OpenMC (Monte Carlo) and OpenFOAM (C++ CFD libraries) has been established to facilitate the high-fidelity multi-physics simulation with explicit modelling of complex geometries. A data structure was formulated to exchange information such as temperatures and power densities between the two codes. This data structure operates by reading the output array for the desired value and mapping that value array onto the discretization for the input of the other code, taking advantage of utilities built into OpenFOAM. An advanced iteration strategy was implemented alongside the data structure to reduce the computational time required to reach converged solutions of the coupled system. This iteration strategy relies on a power relaxation strategy on the fission power distribution, which works by

determining the power distribution by an average of all the iterations instead of a single one. This vastly decreases the uncertainty of the Monte Carlo code by increasing the particle sample size. Moreover, an adaptive source size for the input of OpenMC was used, which changes the number of particles per iteration based off the level of convergence of the previous iteration power distribution. These two strategies together will vastly decrease the runtime and increase the stability of the code.

The newly developed coupler was tested on a 2×2 pincell array. A comparison was made between the results of the coupled solver and those of standalone calculations of each code. In addition, this case was used to test the implemented iteration schemes for computational cost and convergence behaviour.

Next, a 17×17 PWR assembly was modelled for initial verification purposes, done by comparing to results from the VERA codes suite for the same problem. In addition a test was done to analyse the effect of refining the tally mesh in the Monte Carlo output on the power distribution mapped to the CFD solver.

The demonstration of a highly flexible, high-fidelity coupling system without prohibitive computational cost, such as the one presented in this work, will be extremely useful in the development of advanced reactor design, as well as in the continued operation and analysis of LWRs.

# A Multiphysics Coupling Scheme between OpenMC and OpenFOAM for High-Fidelity, Flexible Multi-Physics Analysis of Nuclear Reactors.

by
Lance Edward Bullerwell

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

Nuclear Engineering

Raleigh, North Carolina
2020

APPROVED BY:

| | |
|---|---|
| _____ | _____ |
| Dr. Jason Hou<br>Committee Chair | Dr. Maria Avramova |
| _____ | |
| Dr. Kostadin Ivanov | |

## BIOGRAPHY

The author was born in Raleigh, NC to Martha and Arthur Bullerwell in June 1996. In 2014, He began attending North Carolina State University, during which time he found a passion for reactor physics and computational modelling. In May 2018, He graduated with a Bachelor of Science in Nuclear Engineering and continued his studies with pursuit of a Master of Science in Nuclear Engineering.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# BACKGROUND

## Motivation and Introduction

Nuclear reactor design and operation heavily relies on the modelling and simulation (M&S) results. These tools allow for predictive capabilities for reactor behavior in extremely complex governed by realms of physics that are numerically difficult to solve. The current suite of codes used in industry, for the neutronics simulation, mainly utilizes two separate codes: a lattice code to solve for 2D neutron transport in a single lattice and a nodal code to solve a nodal form of the 3D diffusion equation for the entire core. For core Thermal-Hydraulics (T-H) simulation, a subchannel code is typically used.

However, there is a desire for many features in modelling and analysis tools for nuclear reactors that are not met by the current suite of codes used in industry. In the past decade, there has been a large push for the expansion from single-physics codes, which only solve for the neutronics or solve for the T-H problem only, to multi-physics codes, which solve for multiple realms of the physics in a reactor, and couples the effects that they have on each other. This allows for significantly more accurate results. A primary drawback of this approach is that it takes longer to run a multi-physics code than to run each single-physics code, due to the change in conditions in each physics caused by the other facets that have been coupled during the codes execution.

In addition, there is frequently a desire for high-fidelity codes to be used in nuclear reactor modelling and simulation. Higher-fidelity codes produce more accurate results, typically with reduced uncertainty due to using more accurate models with fewer approximations to the physics. This can be very valuable in, particularly in increasing profits in a power plant by reducing the margin caused by less accurate codes. However, higher-fidelity codes have a higher

computational cost, which is a significant hurdle to their wider implementation. With high-fidelity is the desire for highly flexible tools, which can model a larger variety of reactor designs with varying scales in the reactor, and thus remove the need for multiple codes for each reactor. However, an increase in flexibility means that creating and running models with the tool becomes more difficult, as many things aren't assumed as defaults. Also, more flexible tools will typically have longer runtimes, as they will not have been optimized for the specific model and scenario.

This work presents a multi-physics coupling scheme between the open-source Monte Carlo neutronics code OpenMC and the open-source Computational Fluid Dynamics (CFD) code OpenFOAM. This system represents a high-fidelity, highly flexible multi-physics tool, that also has the benefit of utilizing two open-source codes. Efforts were made to yield the most accurate results with this solver, while decreasing runtime through various acceleration and stability algorithms.

**Monte Carlo to CFD Coupling Experience**

The work of Novak et al. [Novak et al.] shows a preliminary work in a transient coupling of the Monte Carlo code OpenMC to the CFD code Nek5000. Buffalo, which solves for the thermal mechanical behavior in the fuel, is also included. This coupling was performed within the Multiphysics Object Oriented Simulation Environment (MOOSE) finite element framework [Gaston et al.]. MOOSE acts as an external driver for the two codes in this work, where the existing OpenMC and Nek5000 codes are "wrapped" to act as MultiApps, with one acting as the Master app. In this system the wrapped version of OpenMC is the MasterApp, as it is the only portion that solves over the entire domain.

This coupling system is run based off of a Picard iteration by running codes in order of Nek5000, OpenMC, and then Buffalo. Once each code runs in an iteration, convergence is evaluated. If the time step is converged, the solver moves to the next time step. If the time step is not converged, each code is rerun in the Picard step until convergence is reached. This method is not ideal, as there are issues with unrelaxed and unaccelerated Picard iteration strategies. Also, the explicit time-stepping method implemented can require extremely small time steps to maintain stability and accuracy of the solution, which significantly adds to the already large computational cost in running Monte Carlo coupled to CFD.

It is important to note that this coupling scheme represents an internal coupling, as the individual solvers are given internal access to data from the other codes, as opposed to relying on output and input files to perform data transfer.

A major issue discussed in this work is the issue of the number of cells in the Monte Carlo and the CFD simulations. The CFD simulation needs very fine mesh to accurately capture turbulence and boundary layers effects. Conversely, Monte Carlo requires very few cells to solve, only enough to accurately represent the geometry. Increasing the number of cells in a Monte Carlo code can greatly increase the memory requirements and the computational cost of a simulation. This causes data transferring to be difficult between Monte Carlo and CFD codes without significant inefficiencies or inaccuracies being induced.

The implemented method in this work was to use homogeneous temperature regions in the solved pincell. This limits the accuracy of the neutronics results as the thermal feedback is not calculated locally. There was discussion of work on OpenMC to alleviate this issue by allowing the code to utilize heterogeneous temperatures and densities without slowing the

calculation significantly by direct solve of the distance to the next collision in the Monte Carlo solver.

To get accurate fission power info from a Monte Carlo solver, typically a fine tally mesh is needed. To alleviate this issue, finite element tallies (FETs) were used. The FETs work to develop an expression for the fission power distribution as sums of orthogonal polynomials. This helps to solve the memory and computational issues with large numbers of tallies in the Monte Carlo solver, which are similar to the issues with a large number of Monte Carlo cells. The main issue with this approach is that these FETs cannot be constructed for arbitrary geometries, which limits the applicability of the method.

Also, the data that was analyzed to determine convergence was the temperature and k-eigenvalue data. The use of k-eigenvalue to determine convergence is less ideal than using power distribution data. The eigenvalue is not a local value, so it is possible for the eigenvalue to be converged when the local power distributions in the case have not fully converged.

In [Cardoni et al.], a coupling between the Monte Carlo code MCNP5 and the CFD code STAR-CCM+ is shown. The coupled solver is referred to as MULTINUKE. MULTINUKE is the Perl script that governs the data transfer and iteration algorithm, as well as the additional utility scripts needed and the execution of both MCNP5 and STAR-CCM+. This coupling is an external coupling, where the only data the individual solvers can exchange is in the form of existing input and output structures.

MULTINUKE starts running by creating and executing an isothermal MCNP5 case, which is used to initialize the power distribution for STAR-CCM+. From there, MULTINUKE proceeds according to a simple Picard scheme. STAR-CCM+ is run, and the temperature and density data is output to MCNP5, which is run again and convergence is checked. Convergence

for this code is checked similar to [Novak et al.], the temperature and the k-eigenvalue is checked. As in that case the use of the k-eigenvalue as the only convergence value for the neutronics simulation should be avoided to yield the most accurate and well converged results.

It is important to note that there is no stability or acceleration strategy discussed in this work. A simple Picard iteration is typically undesirable in multi-physics solvers, particularly when a Monte Carlo code is used to solve for neutronics. This is due to the iteration scheme being naturally unstable. This issue is exacerbated with Monte Carlo codes, due to the stochastic nature of the code and the inherent statistical noise and inaccuracy that causes.

The results in this work are shown for a portion of a pincell and appear to show good convergence. However, it should be mentioned that the model is very simple and small. The CFD model is also fairly low fidelity, with 9,984 cells. Also, even on a small, and lower fidelity, model, this case took 8 hours to run on 4 cores. This shows the need for both large computational power and accelerated iteration schemes to make the running of full-scale Monte Carlo to CFD multi-physics simulations feasible.

The work in [Tuominen et al.] shows a coupling between the Monte Carlo neutronics code Serpent 2 and the C++ library OpenFOAM used for CFD calculations. These codes were coupled externally, based on the existing Serpent multi-physics interface that has been developed. This interface allows for refined temperature and density data to be utilized in the Monte Carlo solver without slowing down the simulation excessively. This provides the solution to the temperature transfer issue raised in [Novak et al.], the coupling of OpenMC to Nek5000 within the MOOSE framework. In addition this interface also allows for tallies for fission power to be counted directly on the CFD mesh.

This iteration scheme for this system is based off of a Picard iteration, however it implements relaxation strategies to increase stability of the code and subsequently decrease the runtime. The first relaxation scheme discussed is a stochastic relaxation scheme which works to remove the statistical uncertainty in the Monte Carlo results. This scheme, often referred to as power relaxation, will be discussed in detail later. The second strategy was a fixed under relaxation scheme, detailed in Equation 1,

$$P_{rel}^{n+1} = (1-\alpha)P_{rel}^{n} + \alpha P^{n+1} \quad (1)$$

where $P_{rel}^{n+1}$ and $P_{rel}^{n}$ are the relaxed power distributions for iteration n+1 and n respectively, $P^{n+1}$ is the iteration-wise power distribution for iteration+1, and $\alpha$ is the under-relaxation factor. To test the coupled solver, a 5x5 fuel assembly was modelled. The solver showed good convergence results. Of most interest, the stochastic relaxation scheme performed significantly better than the fixed under-relaxation method. The fixed under-relaxation scheme appeared to have a limit to convergence, related to the statistical noise of the Monte Carlo noise. The stochastic relaxation removes this effect and is able to achieve a much tighter convergence.

**Monte Carlo Coupling Experience**

While the previous experience with coupled Monte Carlo to CFD work is the most valuable to this work, there is still much to be gained from analyzing other work in reactor multi-physics. Past coupling of Monte Carlo neutronics codes to any T-H solver are of interest, in large part due to the nature of Monte Carlo simulation.

Monte Carlo codes do not solve the neutron transport equation, instead using probability to yield the same results by modelling the behavior of large numbers of particles. Due to this, the flux and power distribution for one iteration are not influenced by the next. This means that

different iteration strategies are needed to yield stable convergence of a multi-physics code which utilizes Monte Carlo for neutronics.

The coupling of Monte Carlo neutronic code OpenMC to subchannel code Cobra-EN is presented in [Mylonakis et al.]. This paper gives a proper description of a Picard iteration and the benefits and issues with it.

A Picard iteration multi-physics solver can be described by two solutions, **x** and **y**, which are the output results of the neutronics and T-H solvers respectively. The representation of the T-H and neutronics solvers and post-processers are shown in Equation 1 and 3,

$$\boldsymbol{y}^{k+1} = T(\boldsymbol{x}^{k}) \tag{2}$$

$$\boldsymbol{x}^{k+1} = N(\boldsymbol{y}^{k+1}) \tag{3}$$

where T(**x**) represents the T-H solver and its output processing, N(**y**) represents the neutronics solver and its output processing, and k is the iteration number. A full Picard iteration (PI) can be described by Equation 4 by combining Equation 2 and 3.

$$\boldsymbol{x}^{k+1} = N(T(\boldsymbol{x}^{k})) \tag{4}$$

A main benefit of Picard iterations is that they are fairly simple to implement for independent solvers. However, there is an issue with Picard iterations not having guaranteed convergence, which means under-relaxation is often implemented to aid in convergence.

In place of a Picard iteration, [Mylonakis et al.] discusses implementing a Newton method to ensure convergence and improve stability. A traditional Newton method is not able to be implemented well with Monte Carlo neutronics, as the Newton method requires construction of a Jacobian matrix. The expense of constructing a Jacobian is a limitation in deterministic solvers, but Monte Carlo methods do not solve any actual equations, making construction a Jacobian practically impossible.

Instead of a traditional Newton method, an approximate block Newton method (ABN) is implemented based off [Yeckel et al.]. The work in [Yeckel et al.] describes an ABN method for coupled iteration of nonlinear solvers, which is similar to a Jacobian-free Newton Krylov (JFNK) method.

The approximate block Newton method is based upon a Picard iteration, and convergence is defined upon $\mathbf{x}^*$ as shown in Equation 5.

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k = \boldsymbol{x}^* \tag{5}$$

Equation 4 must be rewritten in a residual form, as shown in equation 6

$$\boldsymbol{x} - N\big(T(\boldsymbol{x})\big) = F(\boldsymbol{x}) \tag{6}$$

Which can be solved via a Newton method as in Equation 7 to find $\delta\mathbf{x}$, which is the update to the solution.

$$\boldsymbol{J}(\boldsymbol{x})\delta\boldsymbol{x} = -F(\boldsymbol{x}) \tag{7}$$

The left hand side of equation 7 can be approximated by equation 8, where ε is a perturbation parameter

$$\boldsymbol{J}(\boldsymbol{x})\delta\boldsymbol{x} = \boldsymbol{J}(\boldsymbol{x})\boldsymbol{v} \approx \boldsymbol{v} - \frac{1}{\varepsilon}\Big(N\big(T(\boldsymbol{x} + \varepsilon\boldsymbol{v})\big) - N\big(T(\boldsymbol{x})\big)\Big) \tag{8}$$

The ABN method starts by a PI-analogous step, solving equations 1 and 2 for a PI solution defined as $\mathbf{y}^{PI}$ and $\mathbf{x}^{PI}$. The difference of $\mathbf{x}^k$ and $\mathbf{x}^{PI}$ is then used to define a vector of residuals, F(**x**), as in Equation 6. Then a matrix-vector solve, utilizing the Generalized Minimal RESidual method (GMRES), is performed on Equation 7, using the approximation presented in Equation 8. Once the solution of Equation 8 is performed, the solution is updated according to Equation 9.

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \delta\boldsymbol{x}^k \tag{9}$$

This process is continued until convergence is reached. The established coupling method showed significant reduction in the number of iterations required to reach convergence of the system when compared to an unrelaxed PI for a pincell. When extended to a full assembly case, and compared to both unrelaxed and under-relaxed PI, the developed coupled solver with ABN implemented showed significant improvement over both PI methods. In addition to showing the increased effectiveness of the ABN method over the PI methods, this work also showed numerical oscillations in the PI scheme, which leads to inefficiencies in the convergence.

The coupling of RMC, a Monte Carlo neutronic code, to CTF, a subchannel code, is shown in [Guo et al.]. This work builds on the past development of the RMC to CTF coupling scheme, and works to increase the stability and speed of the system. The existing coupling strategy was a Picard Iteration, with no major relaxation or stability methods implemented.

A major driver for the addition of stability methods into the existing coupled system is the observation of oscillatory effects between iterations of the solver. When local power peaks occur in the Monte Carlo result in one iteration, this leads to higher temperatures and lower moderator densities. Due to this, the next iteration of the Monte Carlo code gives a depression in the power where a peak occurred on the previous iteration. IT was shown that this result will slow the convergence of the system, and also that the oscillation may prevent convergence altogether.

Two methods of relaxation were discussed and implemented to solve this issue. The first was based off a predictor-corrector method to balance out the oscillation between each iteration step. This method is show in Equation 10,

$$\varphi_{n+1} = \frac{\varphi_n + R(\varphi_n)}{2} \tag{10}$$

where $\varphi_n$ is the relaxed power distribution, and $R(\varphi_n)$ is the output power distribution form RMC for iteration n. This method helps to average out the oscillations between iterations to help stabilize and accelerate the convergence of the system.

The second method is a power relaxation method, similar to the stochastic acceleration discussed in [Tuominen et al.]. This method helps to account for the statistical noise in the Monte Carlo code, and allows for all of the results from the Monte Carlo run to be accounted for, by averaging all of the previous power distributions from the Monte Carlo code. Equation 11 depicts the method as a summation.

$$\varphi_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R(\varphi_i) \tag{11}$$

Equation 11 can be expanded and rearranged to yield a more readily implementable form, as shown in Equation 12.

$$\varphi_{n+1} = \left(1 - \frac{1}{n}\right)\varphi_n + \frac{1}{n}R(\varphi_n) \tag{12}$$

The method detailed by Equations 11 and 12 will show a decreasing error with an increased number of iterations. Also, it can be seen readily from Equations 10 and 12 that the predictor-corrector method and the power relaxation method are both weighted functions of the current power distribution, $R(\varphi_n)$ and the previous power distribution, $\varphi_n$. The only difference between the two methods is the weight given to both of these functions to yield the next power distribution for the T-H input.

This work implemented the predictor-corrector method in equation 10. This method showed good results in stabilizing the oscillatory behavior discussed. However, it should be noted that the convergence results show a similar limitation as to PI results, as discussed in the coupling of Serpent to OpenFOAM in [Tuominen et al.]. That is that there appears to be a limit

to the convergence reached, due to the fact that the uncertainty in the Monte Carlo run is fixed by the number of particles simulated in the Monte Carlo run and that the most recent iteration has a fixed weight on the current relaxed power distribution.

This work also offered a comparison of the results of the fully coupled system to that of an uncoupled run. It showed that there were significant effects in flattening the power distribution, due to the effect of thermal feedback in the system.

In [Bennett et al.] the coupling of the Monte Carlo neutronic code MCNP6 to the subchannel code CTF is shown. The goal of this work was to develop an internally coupled Monte Carlo to subchannel code, and show testing on a larger scale than typical, in this case a full 3D assembly test problem.

An important issue in coupled codes such as these is the method of cross section broadening. In this work the fit_otf routine is used to create functional expansions of cross section with temperature, to remove the need for many large cross section files. This is not possible for thermal scattering data, and as such makxsf is used to create many cross-section files over a distribution of temperatures.

The convergence of the code is based upon the temperature distribution between iterations is compared, as shown in Equation 13.

$$\varepsilon \geq \max_{i,j} \left| \frac{T_{i,j}^{current} - T_{i,j}^{previous}}{T_{i,j}^{previous}} \right| \tag{13}$$

The maximum temperature change between iterations for any node in the T-H calculation is determined, and convergence is checked based off that value. This code was coupled internally, as significantly less read and write time is needed, reducing the computational cost.

When running coupled Monte Carlo and T-H simulation, there is a statistical noise that affects the power distribution. This noise affects equation 13, as shown in equation 14, where ξ is the statistical noise present in the Monte Carlo simulation.

$$\varepsilon \geq \max_{i,j} \left| \frac{T_{i,j}^{current} - T_{i,j}^{previous} + \xi_{i,j}^{current} - \xi_{i,j}^{previous}}{T_{i,j}^{previous} + \xi_{i,j}^{previous}} \right| \quad (14)$$

The magnitude of the statistical noise acts to limit the convergence of the system. A technique to account for the statistical noise must be implemented to avoid this limit, which was seen in RMC to CTF, the PI results in Serpent to OpenFOAM, and MCNP5 to STAR-CCM+ [Guo et al., Tuominen et al., Cardoni et al.]. The method implemented to avoid this limit is the power relaxation strategy discussed in [Tuominen et al., Guo et al.]. This method, detailed in Equation 12, ensures the error decreases with more iterations due to the flux in the next iteration being the mean value of all the previous iterations.

The convergence for this code system with relaxed flux was compared to the unrelaxed flux. The relaxed flux showed much better convergence than the unrelaxed case, and the unrelaxed case showed the same issue of a limit on the convergence seen in previous coupled iterations.

The Simulator using MCNP with Integrated Thermal-Hydraulics for Exploratory Reactor Studies (SMITHERS) [Richard et al.] is an external coupling between MCNP and a multi-channel T-H solver. A primary goal of its development was to create a flexible external coupling framework to integrate thermal-hydraulics solvers into MCNP, with a focus on the capability to perform burnup calculation through the MCNP6/MONTEBURNS code. In this work, MCNP was coupled to a Multi-channel analysis solver.

It should be noted that this scheme is able to be extended to higher-fidelity (subchannel, CFD) T-H methodologies. This is due to the modular software configuration that limits the

interaction between the main SMITHERS script and the T-H solver. SMITHERS maps the power from the MCNP combinatorial geometry to a generalized nodal form of the power. The fidelity of this nodal form can be adjusted readily by the user. To adjust for different fidelity T-H solvers.

The iteration strategy implemented in the SMITHERS code is a typical Picard Iteration, with no details on relaxation being discussed. There is also no in depth discussion of the convergence behavior of the code. The results of the test cases showed good agreement with expected results.

**CFD Coupling Experience**

In addition to the above mentioned multi-physics projects, many useful multi-physics projects in terms of CFD codes being coupled to non-Monte Carlo neutronics code. Of particular interest is the work of internal coupling OpenFOAM to a Diffusion solver for neutronics in GeN-FOAM.

GeN-FOAM, developed by the Paul Scherrer Institute, is a multi-physics code based on OpenFOAM, with the addition of a multi-group diffusion sub-solver to perform neutronics solution. The diffusion sub-solver was built to import multi-group cross sections generated by the Monte Carlo code Serpent2 [Fiorina et al.].

An important feature of GeN-FOAM is the capability to perform transient simulations. It is discussed in [Fiorina et al.] that the simulation of transients in reactors typically is done by coupling neutron diffusion codes and system or subchannel T-H codes. These transient couplings typically rely on explicit time-schemes, which require extremely short timesteps to maintain stability and accuracy of the simulation. The transient time-scheme in GeN-FOAM is an implicit,

first-order, Euler scheme which should allow for larger timesteps compared to an explicit scheme.

GeN-FOAM has the ability, in the neutron diffusion solver, to perform calculations for local delayed neutron precursor concentrations, taking into account flow of delayed neutron precursors in the instance of a fluid fuel.

# METHODOLOGY

## Single Physics Codes

OpenMC [Romano et al.] is an open-source Monte Carlo neutronics code. It was originally developed at the Massachusetts institute of Technology (MIT) in the Computational Reactor Physics Group (CRPG).

OpenMC uses constructive solid geometry for constructing system geometry. Regions are defined by the union and/or intersection of half-spaces created by bounding surfaces. The structure of the constructive solid geometry in OpenMC supports universe-based geometry, similar to MCNP and Serpent.

OpenMC currently uses surface-tracking for its particle tracking algorithm. Surface tracking for particle transport relies on the independence of the interaction probability on the particle's history. Thus, when the distance to the next surface is less than the sampled distance to the next interaction, then the particle track is stopped at the surface and a new track is started at the new surface, or any boundary conditions on the surface are applied [Romano et al., Lepänen].

It should be noted that surface tracking has efficiency drawbacks in complex geometries, where there are large numbers of surfaces to track across. If individual cells are comprised of large numbers of surfaces, it can be expensive to determine the distance to the nearest surface. The method of tracking interactions across surfaces by restarting particle tracks grows in computational expense significantly when the mean free path is significantly longer than the distances between surfaces [Lepänen].

Cross section data used in OpenMC must be stored in an HDF5 format. The HDF5 data files can be generated by converting the ACE format that Serpent and MCNP use. OpenMC can treat temperature dependence of cross section data in multiple ways. The nearest temperature

library points can be used, an interpolation between the closest temperature libraries above and below the input temperature can be used, or a windowed multipole library can be used. The windowed multipole approach allows for on-the-fly Doppler broadening, and can greatly decrease the required memory to store cross-section data [Romano et al.].

OpenFOAM is a C++ library developed to perform field operation and manipulation (FOAM) for the solution of partial differential equations (PDEs). It is an open-source library that is frequently used in the implementation of computational fluid dynamics (CFD) solvers. This includes CFD solvers for compressible and incompressible flow, conjugate heat transfer, and turbulence modelling via Reynolds Averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) [Weller et al.].

Of most interest in this work is the OpenFOAM solver *chtMultiRegionSimpleFoam*. This solver is a steady-state, compressible CFD solver that allows for conjugate heat transfer between multiple regions, including solid regions. This solver was used in the coupled solver developed in this work. The only change made to the solver was the addition of a volumetric heat generation term to the energy equation solver [Weller et al.].

In addition to the *chtMultiRegionSimpleFoam* solver, many data utilities included in OpenFOAM were used. All meshes used in *chtMultiRegionSimpleFoam* in this work were generated using gmsh, and the OpenFOAM utility *gmshToFoam* can be used to convert meshes from gmsh into the correct format for OpenFOAM. In addition, the OpenFOAM *blockMesh* utility allows for generation of Cartesian meshes. The OpenFOAM *mapFields* utility allows for the mapping of data fields from one OpenFOAM mesh onto a similar mesh.

**Data Structure**

As in other instances of coupling Monte Carlo neutronics codes to T-H solvers [Tuominen et al., Bennett et al., Cardoni et al., Richard et al.], a primary concern in coupling OpenMC to OpenFOAM is the methodology of mapping data from the constructive solid geometry methods typically employed in Monte Carlo codes and the discrete mesh of the T-H solver. This issue is of particular interest when the T-H solver uses unstructured mesh which is typically used in CFD solvers such as OpenFOAM.

The method developed for this coupled solver was to first develop a Cartesian mesh in OpenFOAM that matches the tallies mesh defined in the OpenMC *tallies.xml* file, covering the entire domain of the system. This was done using the OpenFOAM *blockMesh* utility. It's important to note now that OpenMC uses centimeters to measure length, while OpenFOAM uses meters, so appropriate conversions must be made in case setup.

Once the OpenFOAM *blockMesh* is made, a conversion from the method OpenMC uses for storing coordinates in ordered triplets to the way OpenFOAM stores its *polyMesh*. *PolyMesh* defines the physical location of all the points used to build the mesh, these points are used to define the faces of the mesh elements, and cells are defined based off the faces they are contained by. A script, *mesh_convert.exe* had to be made to perform the conversion between the two grid layouts, and to output a file for mapping the tallies output for fission power on to the OpenFOAM Cartesian mesh.

Once the power distribution can be mapped from the OpenMC tallies to the OpenFOAM Cartesian mesh, the method of mapping the power distribution form the Cartesian mesh that covers the entire domain to the unstructured mesh for each region to be solved by

*chtMultiRegionSimpleQFoam* must be determined. This is done in this solver by utilizing the *mapFields* utility of OpenFOAM.

The *mapFields* utility maps fields in OpenFOAM from a source geometry onto the corresponding fields of a target geometry. In this solver, the Cartesian mesh is the source geometry and the unstructured mesh for each region is the target [Weller et al.]. It is important to note that a reasonably fine tally mesh will be required to capture and translate the behavior of non-Cartesian boundaries, such as cylindrical fuel rods, without interpolation losses.

The second mesh consideration is the mapping of the unstructured temperature results in OpenFOAM for the CFD solve onto the cells defined in the OpenMC. This was done by, first, using the *topoSet* utility in OpenFOAM to create *cellZoneSets* of cells in the mesh that correspond to the cells created in the *geometry.xml* input for OpenMC. Then the average temperature in each of these *cellZoneSets* is output at the end of each OpenFOAM iteration via the *volFieldValue* option in OpenFOAM [Weller et al.].

Once the method of translating the data between the meshes of the two codes has been developed, the process of actual data transmission must be determined and implemented. As this solver is an externally coupled code, the format of input and output files must be accounted for in this process.

For the Temperature read and write, the read step is performed on the output fields for the *volFieldValue* data for each OpenMC cell. This is read from the *postProcessing* folder generated from OpenFOAM. The average temperatures by cell are output by region in subfolders of the *postProcessing* folder. Each of these subfolders contains folders for each *volFieldValue* zone specified that is in that region.

To read this data correctly, this solver's data structure must know which of these zones correspond to which OpenMC cell and which region (fuel, clad…) each zones is in. In addition, this solver's data structure has to have the data for the xml input lines for each cell in OpenMC. This is because these lines must be rewritten to account for a different temperature value being applied after each OpenFOAM iteration. All of this info must be supplied in this solver's input file, *coupler.inp*.

After each OpenFOAM run in this solver, the *volFieldValue* data is read in from output, and then a new geometry.xml file is opened. The geometry info is rewritten to this new file, with the temperature values for each cell being updated during writing. Thus OpenMC can be rerun with the updated temperature values from OpenFOAM. The method of reading and writing density values in nearly identical.

For the power distribution read and write, the read step is performed on the *tallies.out* file output from OpenMC. The *tallies.xml* input is read at the start of this solver's run, and the info for which tally contains the kappa fission output is determined, as well as verifying the tally mesh. Then this solver reads the data from *tallies.o*ut and stores it in the *power_dist* array. As OpenMC outputs the tallies in terms of electron volts per source particle and OpenFOAM takes heat generation input in units of Watts per cubic meter, the tallies output values must be normalized based off of the power input read by this solver.

Once the power has been read and normalized to the correct form, the *real_power* variable must be determined. This is done by the relaxation and acceleration methods discussed in the next section.

Once the *real_power* array has been calculated and filled according to the relaxation and acceleration algorithms, it must be output. This data is output onto the Cartesian mesh developed

for OpenFOAM that matches the tally mesh. The mapping function for the OpenMC ordered triplets to the correct cell numbers is used to write the OpenFOAM Q file correctly. The directory where this mesh and file are stored are input by the user.

Once the heat generation file has been written to the Cartesian mesh directory, *mapFields* is run to map that data onto the unstructured mesh for each region that OpenFOAM will solve over.

**Iteration Scheme**

The iteration scheme in this solver is based off of a Picard Iteration. The power distribution is initialized in OpenMC by running an isothermal case. This initial power distribution is then passed to OpenFOAM and the iteration loop is started. OpenFOAM is then run with this initialized power distribution for a certain number of steady-state iteration steps. After that, the temperature and density data are read and passed to the OpenMC input files. OpenMC is then run again and the new power distribution is determined. Convergence of this power is then checked. If convergence is reached, iteration and final outputs are written. Otherwise, iteration outputs are written and this solver loops back to the start of the OpenFOAM calculation. This process is continued until convergence is reached.

As Discussed in many previous works, a Picard iteration does not guarantee convergence, and has particular stability issues when used with Monte Carlo codes [Tuominen et al., Bennett et al., Cardoni et al.]. Because of this, various strategies have been employed to improve the convergence of a Picard Iteration. A common method is to perform fixed under-relaxation, where the previous iteration relaxed flux and the current unrelaxed flux are combined to yield a new relaxed flux. This method improves the result but as seen in [Tuominen et al.], it has a similar issue to a Picard Iteration as it does not guarantee convergence. JFNK solvers and other

similar solver, such as the ABN scheme presented in [Yeckel et al.] can greatly improve over a Picard Iteration. However there implementation is often very complex and with Monte Carlo codes can be particularly tricky.

The other main option used is so-called stochastic relaxation or power relaxation. Power relaxation, unlike fixed under-relaxation, does guarantee convergence, as shown in the in-depth derivation shown in [Ivanov et al.]. Power relaxation is typically defined as in equation 12. This works to mitigate the effect of the statistical noise of the Monte Carlo simulation on the convergence of the solver. However, for this solver, a slightly different approach was taken. This approach, as mentioned in [Tuominen et al.], is based on the fact that the statistical noise that the relaxation strategy is trying to eliminate is dependent upon the number of particles run in the Monte Carlo simulation, resulting in Equation 15.

$$\varphi_{n+1} = \left(1 - \frac{np_n}{\sum_{i=1}^{n} np_1}\right)\varphi_n + \frac{np_n}{\sum_{i=1}^{n} np_1} R(\varphi_n) \tag{15}$$

Where $np_n$ is the number of particles run in iteration n of OpenMC and $\sum_{i=1}^{n} np_1$ represents the total number of particles run in the first n iterations of OpenMC. It can be readily shown that when the number of particles run in each iteration is held constant, Equation 15 simplifies to Equation 12 [Tuominen et al.].

As was discussed in [Richard et al.], the temperature in coupled Monte Carlo simulations typically converges much quicker than the power converges. Once the temperature converges, the value in running more iterations of the T-H solver decrease greatly. Also, early Monte Carlo solves on the unconverged temperature distribution result in less accurate power distributions. These less accurate power distributions are still included in the final power distribution and can slow convergence.

Thus in typical power relaxation, early Monte Carlo iterations improve the rate of the temperature convergence, by providing more accurate power profiles, but slow the convergence of the power distribution. Also, the T-H iterations after the temperature has converged to the current power distribution do not provide significant benefit as the Monte Carlo solution is still converging.

To mitigate the effects of this behavior, a strategy of adaptive batching of the Monte Carlo code was implemented in this solver, based off of Equation 15. The way adaptive batching is currently implemented is by requesting multiple convergence states from the user. Each of these states has a corresponding number of batches to be run in OpenMC. Once a certain convergence level has been reached, then the number of batches, and subsequently particles, to be run on the next iteration of OpenMC is changed. By using adaptive batching, the early iterations of the OpenMC use fewer total particles to get a rough power distribution for OpenFOAM. This distribution is accurate enough to get the temperature profiles close to the final result. Once the code converges more, more batches, and particles, are run in OpenMC.

The effect of this strategy is that, due to fewer particles being run, the early iterations of OpenMC have significantly less effect on the final power result according to Equation 15, but do not slow the convergence of the temperature distribution significantly. Once the temperature distribution is more converged and more particles are being run in OpenMC, more of the codes time is being spent on the Monte Carlo calculation and less on the T-H calculation. Thus, the effect of running the T-H calculation on an already converged temperature distribution is significantly less.

The convergence of this solver is currently based upon the relaxed power distribution of the tallies mesh. This is done by calculating the RMS error on the power distribution in each cell. The total amount of power in each cell in watts, not the power density, is used.

To do this, the difference between the power in each cell in the current and previous iteration must first be calculated as in Equation 16,

$$\Delta p_n^{i,j,k} = \Delta V \cdot \left| \varphi_n^{i,j,k} - \varphi_{n-1}^{i,j,k} \right| \tag{16}$$

where $\varphi_n^{i,j,k}$ is the power density on iteration n of the i[th] cell in the x-direction, j[th] cell in the y-direction, and k[th] cell in the z-direction, $\Delta V$ is the volume of the cell, and $\Delta p_n^{i,j,k}$ is the difference in power for the cell between iterations, in watts. Once the difference in each cell is calculated, the RMS value can be calculated according to Equation 17,

$$RMS = \sqrt{\frac{\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} \left( \Delta p_n^{i,j,k} \right)^2}{I \cdot J \cdot K}} \tag{17}$$

where I is the number of cells in the i-direction, J is the number of cells in the j-direction and K is the number of cells in the k-direction.

## 2×2 PINCELL CASES

The first test run to verify the functionality of this solver was a 2×2 PWR pincell case. This case was run to verify the behavior of this solver, and to serve as a test for the various iterative strategies tested.

### Pincell Description

The pincell specifications used for the 2×2 pincell test case are shown below. The fuel pin consists of only fuel region, no gap or clad. The boundary conditions used in both the T-H and neutronic simulations are reflective radially. The full specifications used to develop the case are given in Table 1.

Table 1: 2×2 Pincell Specifications

| | |
|---|---|
| Fuel pin radius (cm) | 0.4572 |
| Pincell pitch (cm) | 1.26 |
| Fuel height (cm) | 365.76 |
| Power (kW) | 267.781 |
| Inlet temperature (K) | 565 |
| Flow rate (kg/s) | 1.1097 |
| Fuel enrichment (%) | 3.1 |
| Boron concentration (ppm) | 1300 |

The defined case has an analytic solution for the channel temperature change ($\Delta T$) of 40.9 K. The result from the converged simulation was a channel $\Delta T$ of 40.57 K, showing strong agreement with the analytic result.

The mesh for the moderator region in OpenFOAM was 128,700 cells, and the mesh for the fuel region in OpenFOAM was 34,590 cells. Both of these meshes were generated using Gmsh. 2D slices of the mesh for the fuel and moderator regions are shown on the left and right, respectively, in Figure 1.

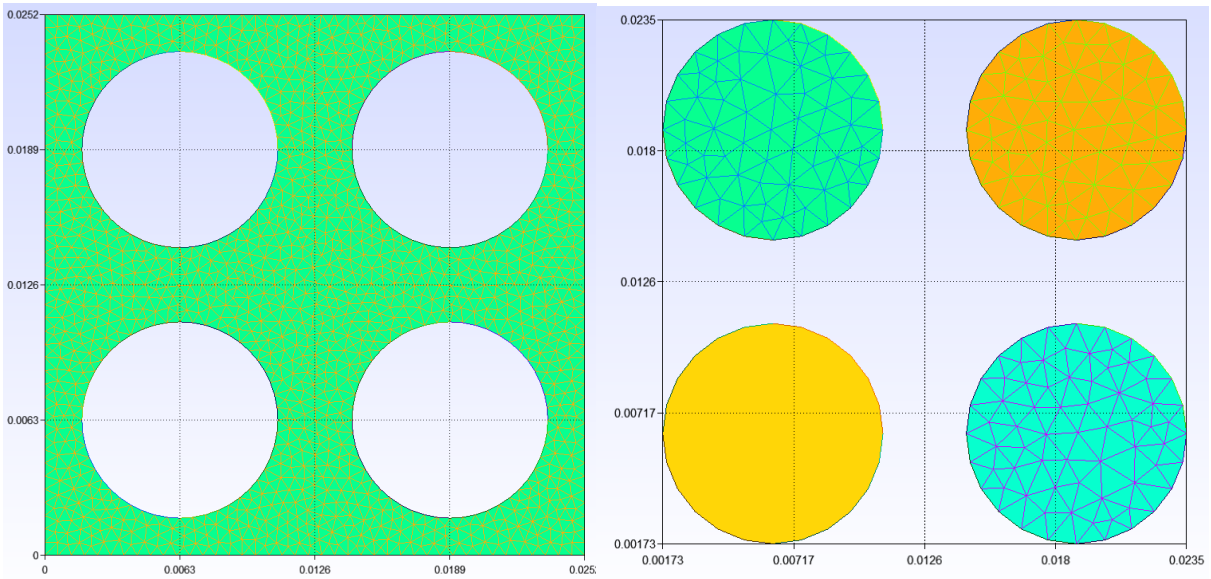


Figure 1: 2×2 OpenFOAM Meshes

The tally mesh in OpenMC for this case had 50 cells in the axial direction and a 75×75 mesh in the radial direction. This tally mesh results in cells that are 7.3152×0.0036×0.0036 cm for a volume of 0.008259 $cm^3$

**Standalone Cases**

To show the benefit of the coupled case, standalone cases for the 2×2 pincell case were run to provide a comparison. The models for the coupled case and the standalone were identical except for the values that would be provided by the other code. Assumptions of average values were used in place of the heat generation for the T-H calculation and the temperature and densities for the neutronic calculations.

To run the standalone case for OpenFOAM, the volumetric heat generation in the fuel was set to the average value of 278.96 MW/m$^3$ in place of using a power distribution from OpenMC.

The plot of the fuel temperature axially in the channel is shown in Figure 2. The fuel temperature is mostly uniform axially, with a steady increase that corresponds to the increased coolant temperature at higher elevations. The average fuel temperature in each rod is 725.59 K, with a standard deviation of 0.6543 K between each rod. The peak fuel temperature observed in this case is 888.38 K and occurs at the centerline at the channel outlet height

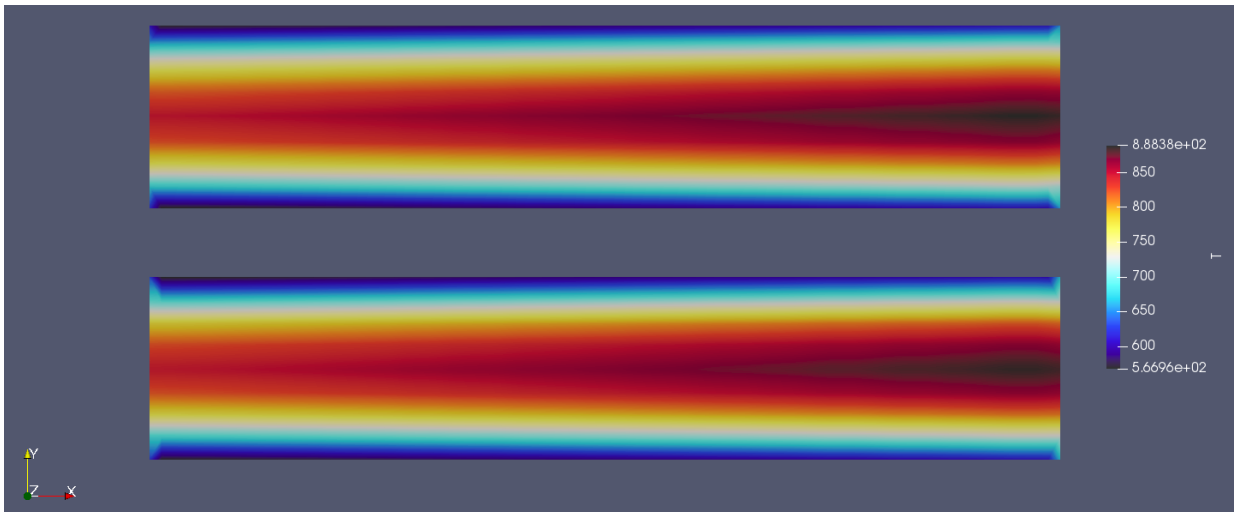


Figure 2: 2×2 Standalone Fuel Temperature, Axial Distribution (K)

The plot of the moderator temperature axially in the channel is shown in Figure 3. The average moderator temperature was 587.75 K. The bulk moderator channel ΔT was 40.41 K.

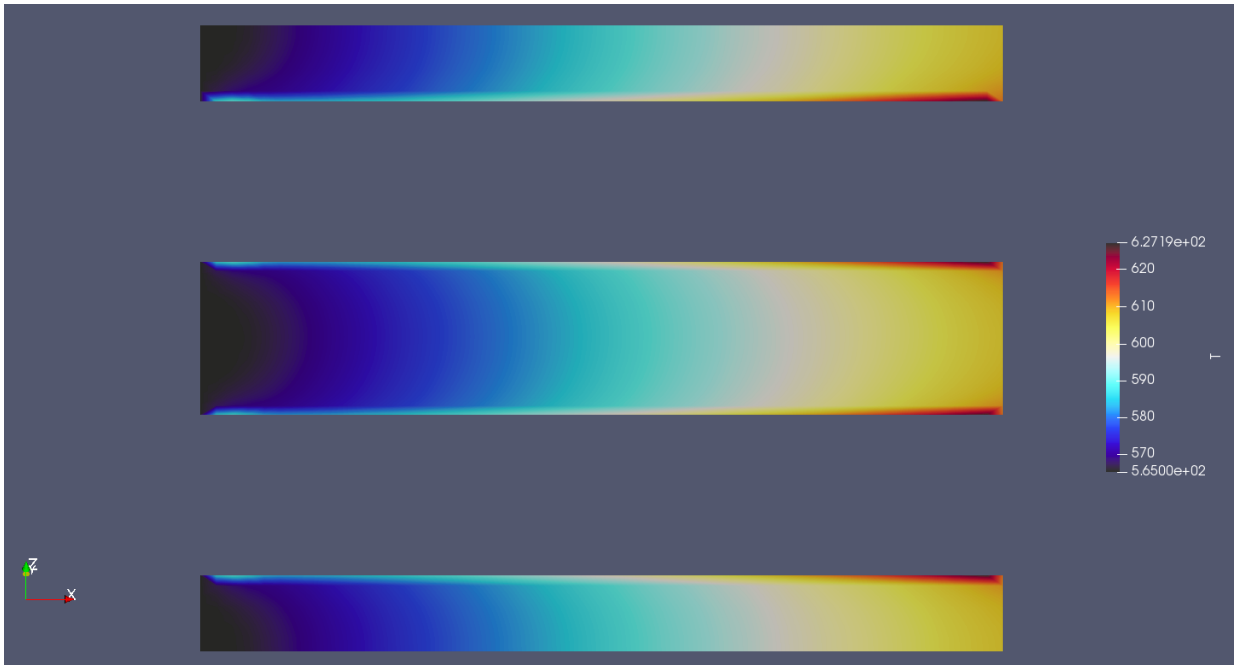


Figure 3: 2×2 Standalone Moderator Temperature, Axial Distribution (K)

The plot of the moderator velocity radially in the channel is shown in Figure 4. The slice was take at the center of the channel axially. The peak moderator velocity occurred in the subchannel center and was 4.9904 m/s. No-slip boundary conditions on the fuel rod surface cause boundary layer development. The symmetry boundaries along the radial boundaries appear to capture the same velocity profile as in the subchannel center.

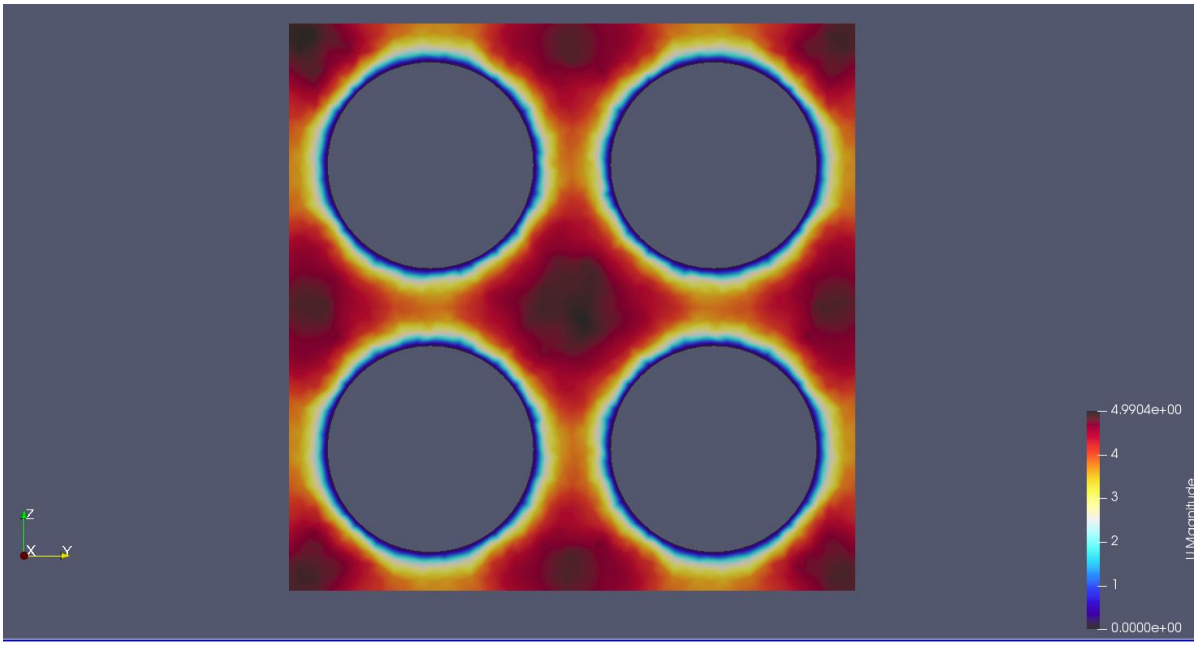


Figure 4: 2×2 Standalone Moderator Velocity, Radial Distribution (m/s)

To run the standalone case for OpenMC, the temperature and density values were set for hot zero power (HZP) conditions, where all materials were set at the inlet temperature of 565k. The simulation was run for 10,000 batches with 250 inactive batches. 25,000 particles were run in each batch, totaling 250 million particles.

The eigenvalue for the standalone neutronic calculation was determined to be 1.19436 with a standard deviation of ±6 pcm.

Figure 5 and 6 shows plots of the power Distribution from OpenMC. Figure 5 shows the axial distribution and Figure 6 shows the radial distribution. The peak power density is 445.87 $MW/m^3$.

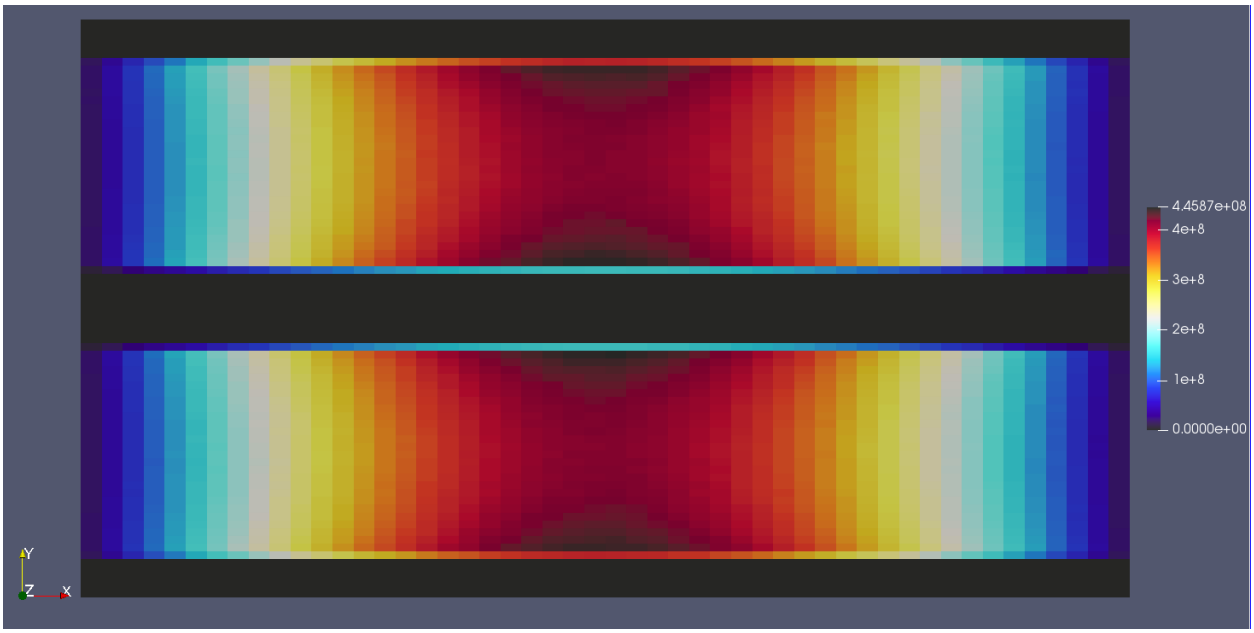


Figure 5: 2×2 Standalone Power Density, Axial Distribution (W/m$^3$)

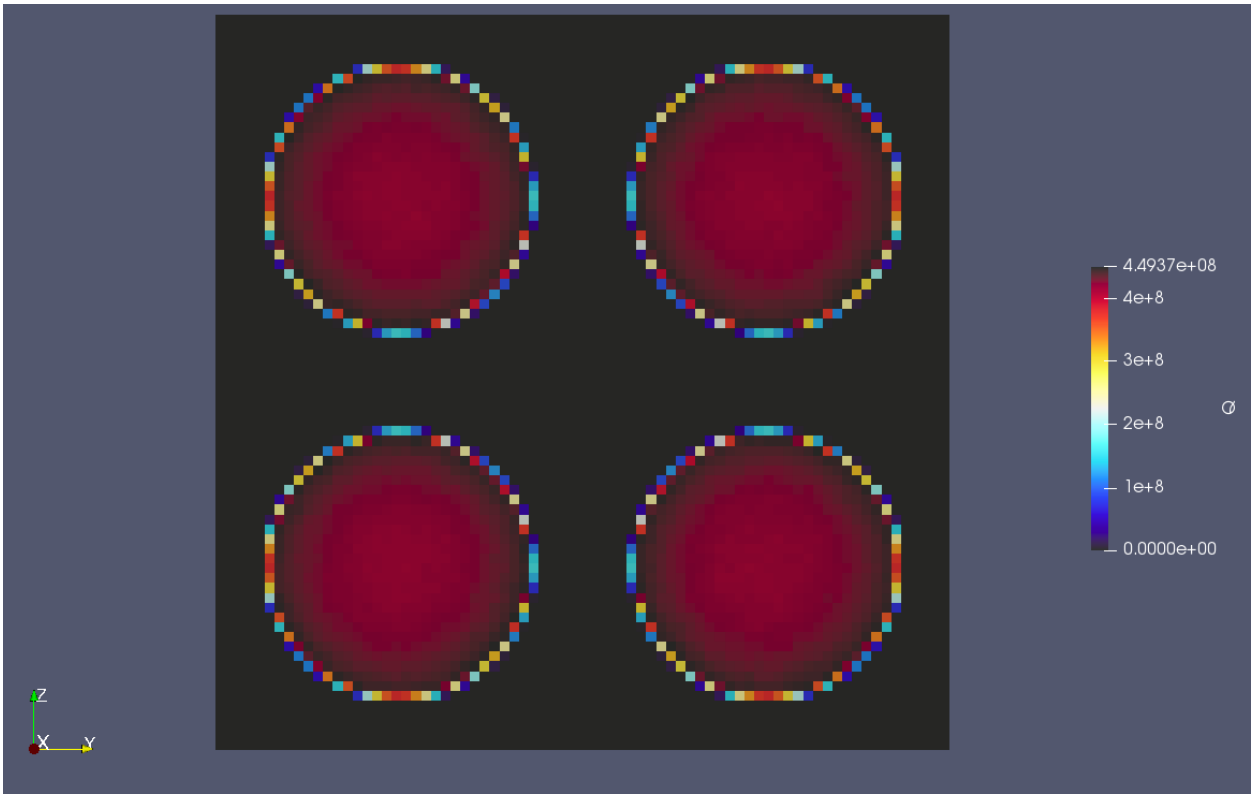


Figure 6: 2×2 Standalone Power Density, Radial Distribution (W/m$^3$)

**Coupled Calculation Results**

The plot of the power distribution axially in the fuel along the rod centerline is shown in Figure 7. The maximum fission density occurs in the center of the fuel axially, and the fuel rod surface radially. The peak power density that occurs locally is 437.08 MW/m$^3$, and the average value is 278.96 MW/m$^3$. The eigenvalue for this case was determined to be 1.18403.

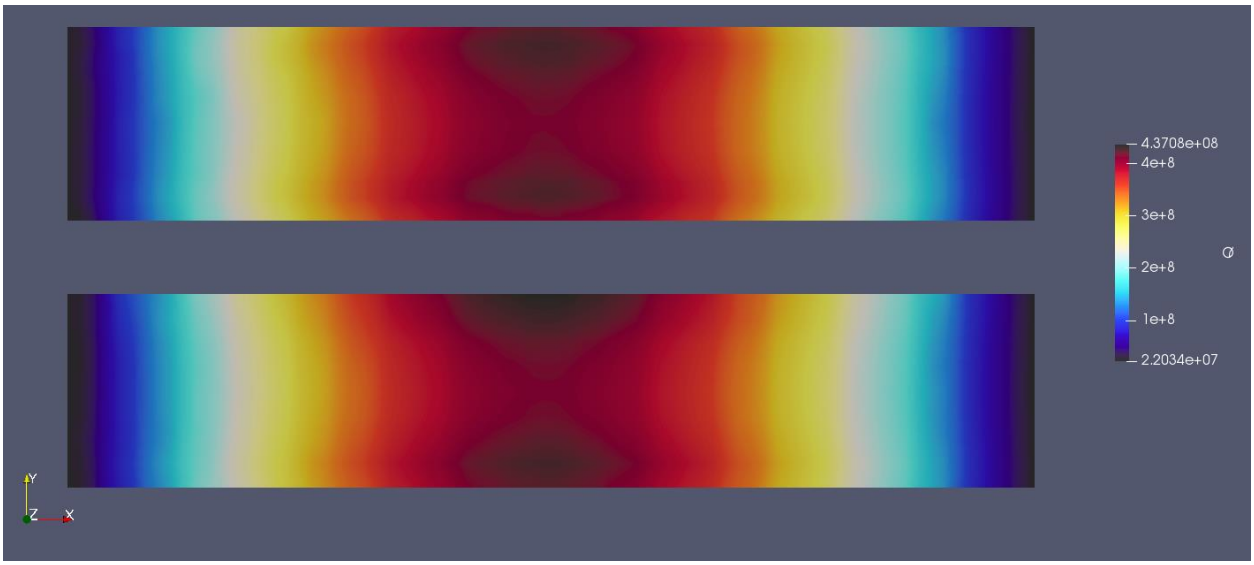


Figure 7: 2×2 Coupled Power Density, Axial Distribution (W/m$^3$)

The plot of the fuel temperature axially in the channel is shown in Figure 8. The fuel temperature is significantly higher in the center axially than at either the inlet or outlet, due to a much greater fission rate in the center of the pincell. The average fuel temperature in each rod is 742.66 K, with a standard deviation of 0.7734 K between each rod. The peak fuel temperature observed in this case is 1036.1 K, and occurs at the centerline at the channel outlet height.

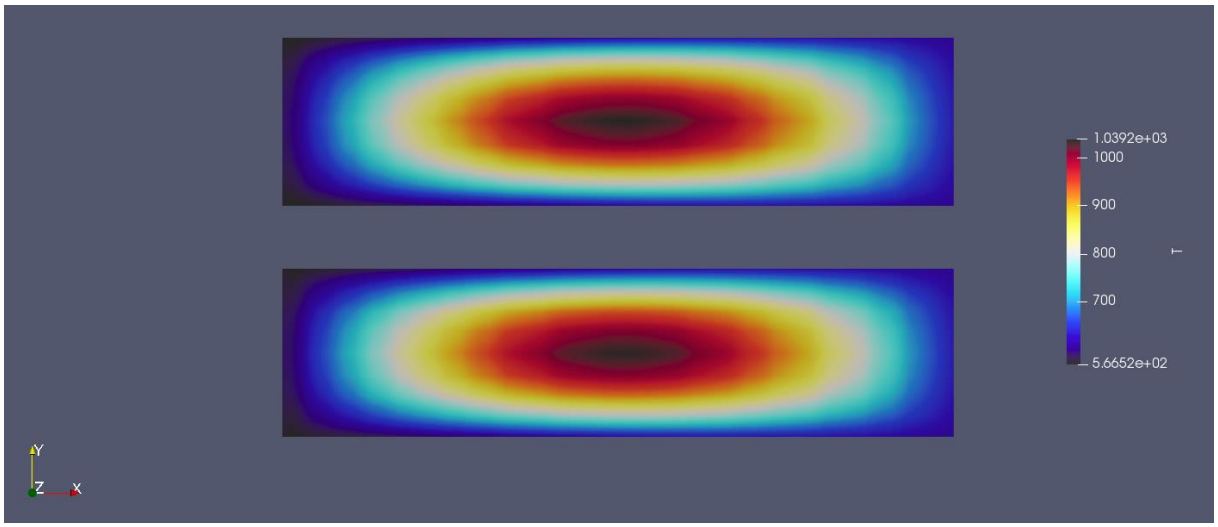


Figure 8: 2×2 Coupled Fuel Temperature, Axial Distribution (K)

The plot of the moderator temperature axially in the channel is shown in Figure 9. The moderator temperature rises consistently along the channel height, with a peak temperature of 624.86 K. The average outlet temperature in this case is 609.79 K, representing a channel ΔT is 44.79 K.

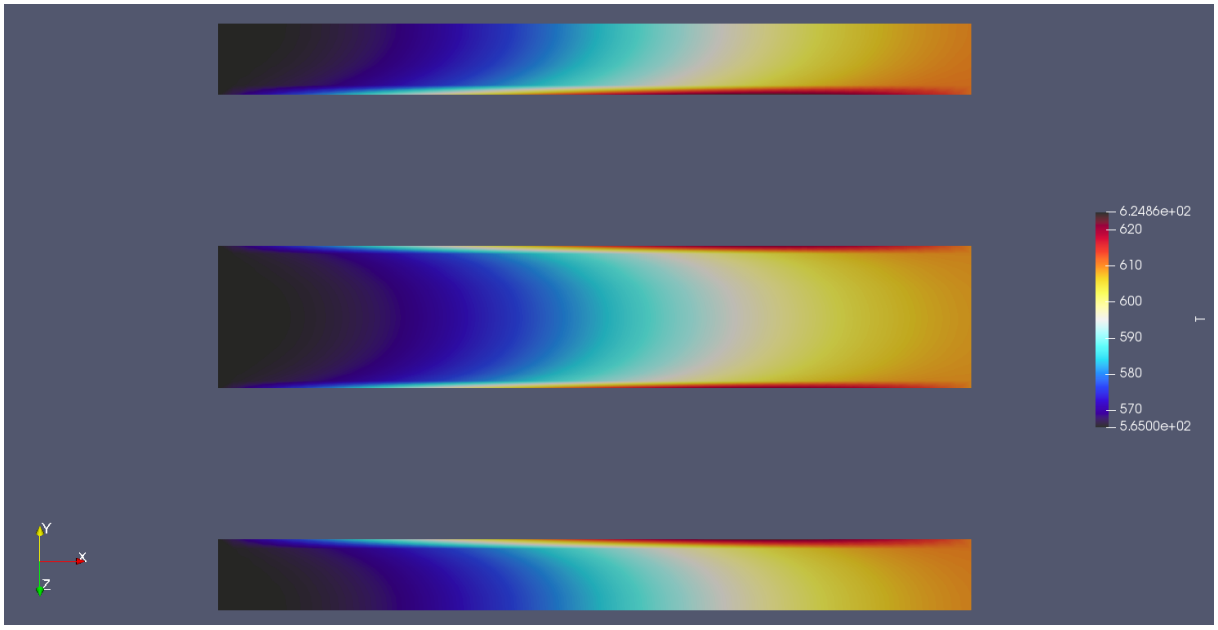


Figure 9: 2×2 Coupled Moderator Temperature, Axial Distribution (K)

The plot of the moderator velocity radially in the channel is shown in Figure 10. The slice was take at the center of the channel axially. No-slip conditions are imposed on the fuel rod surface, leading to zero velocities there. A boundary layer developed leading to the center of the channel. The peak velocity in the channel is 4.5474 m/s.

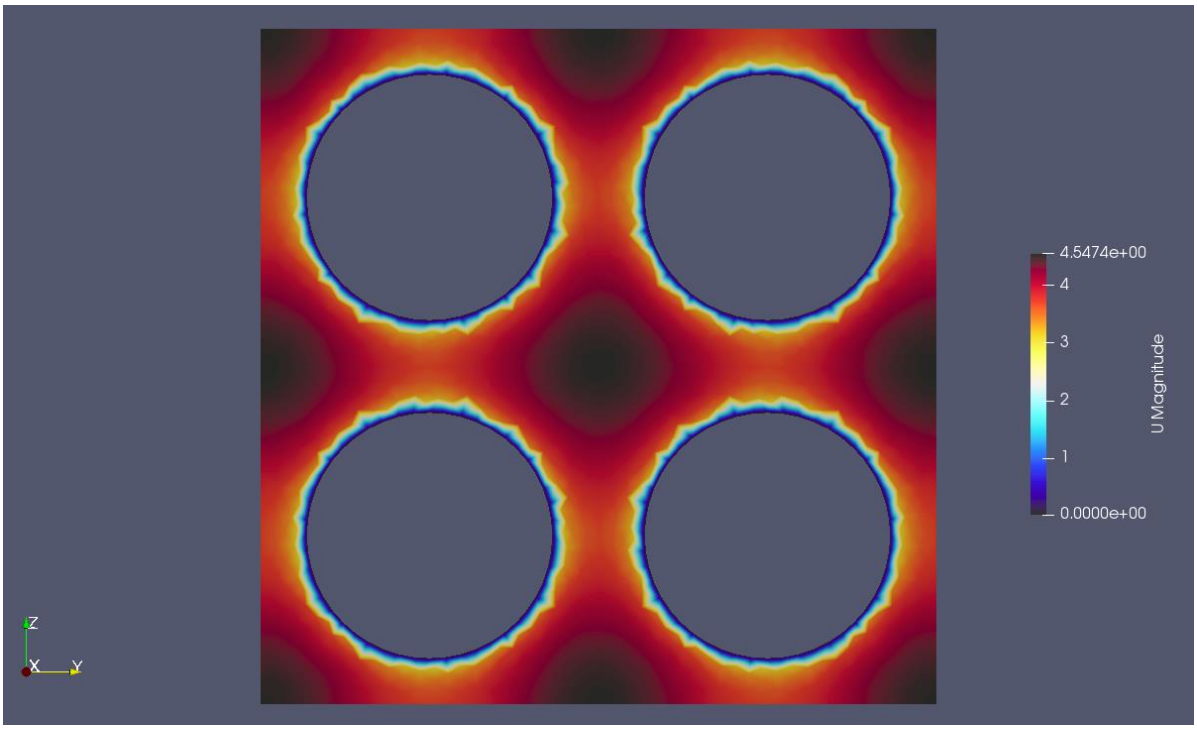


Figure 10: 2×2 Coupled Moderator Velocity, Radial Distribution (m/s)

**Comparison of Standalone and Coupled Calculations**

The eigenvalue for the standalone calculation was 1.19436 with an uncertainty of ±6 pcm. For the coupled calculation, the eigenvalue was 1.18396 with an uncertainty of ±4 pcm. The result is a difference of 878 pcm when accounting for full T-H feedback. The comparisons below take the difference of the coupled calculation from the standalone calculation.

Shown in Figure 11 is the change of the power distribution in the fuel from the standalone case to the coupled case axially. The lower portion of the core has lower power in the standalone case compared the coupled case, and the upper portion has higher power in the

standalone case compared to the coupled case. The difference between the two is up to a 5.45 $MW/m^3$ increase in power or a 4.9297 $MW/m^3$ decrease in the power locally. The average absolute difference over the fuel is a 1.67 $MW/m^3$ change.

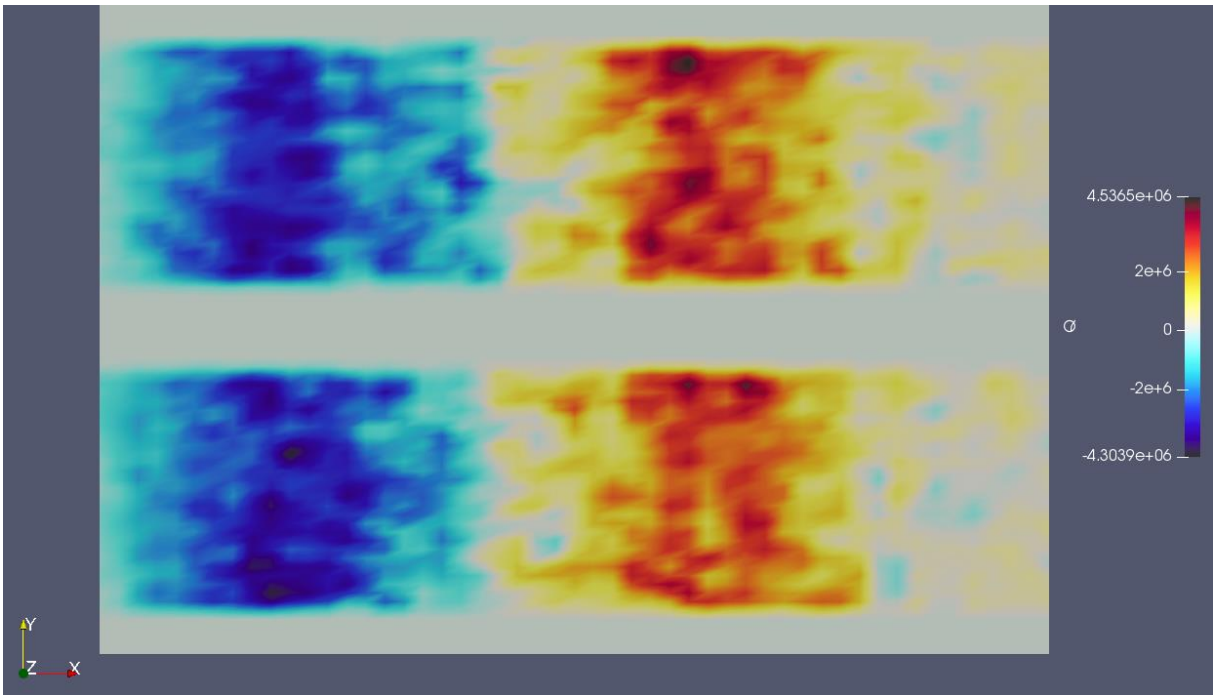


Figure 11: 2×2 Difference in power density, coupled vs. uncoupled ($W/m^3$)

Shown below in Figure 12 is the change in the fuel temperature axially from the standalone case to the coupled case. There was a maximum increase of 257 K and a maximum decrease of 170 K when running OpenFOAM coupled as opposed to standalone. The average absolute change in the fuel temperature was 66 K. The fuel temperature increased in running the standalone calculation near the inlet and outlet, particularly at the centerline, mainly due to higher heat generation in the standalone case. Conversely, the fuel temperature was significantly lower in the center of the channel, as there was much lower heat generation there in the standalone case versus the coupled case.

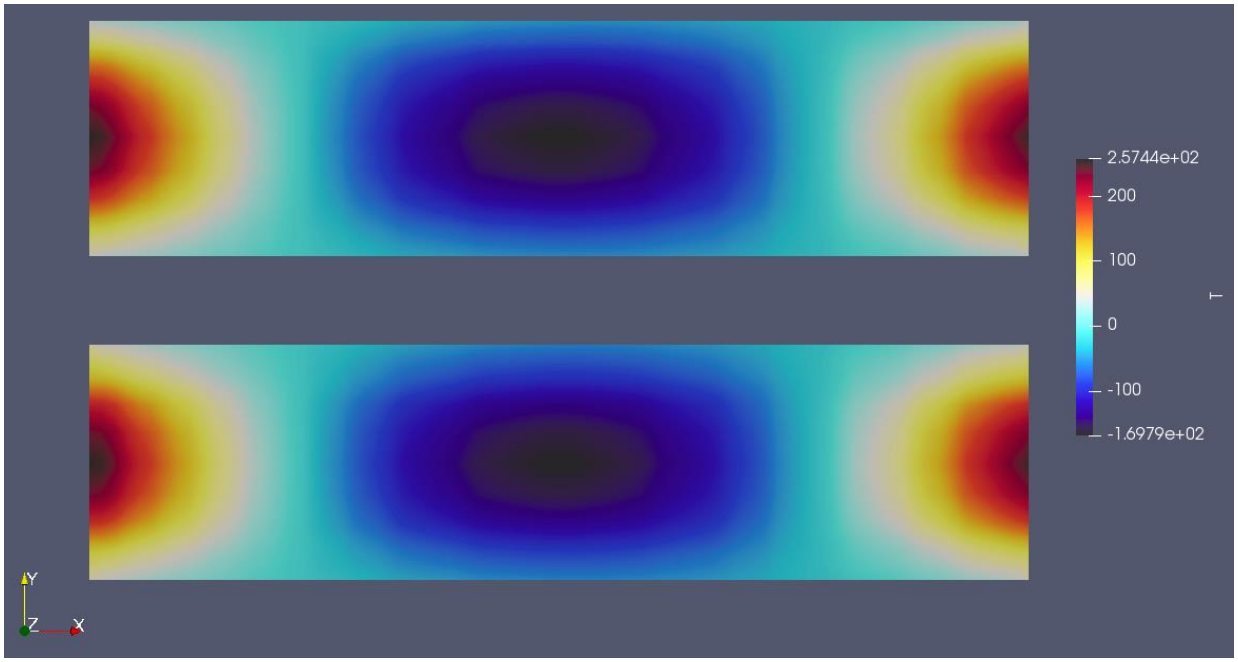


Figure 12: 2×2 Difference in Fuel Temperature, coupled vs. uncoupled (K)

Shown below in Figure 13 is the change in the moderator temperature axially from the standalone case to the coupled case. There was a maximum increase of 7.86 K and a maximum decrease of 6.72 K when running OpenFOAM coupled as opposed to standalone. The average absolute change in the moderator temperature was 1.65 K. The standalone case sees an increase in the moderator temperature at the bottom of the channel, due to the fuel temperature being higher than the coupled case. This changes in the middle of the channel, where the fuel temperature would be higher in the coupled case. The difference appears to propagate away from the wall along the channel, due to the heat being transferred into the moderator at the wall, and then that heat transfer slowly to the bulk fluid as it flows along the channel.

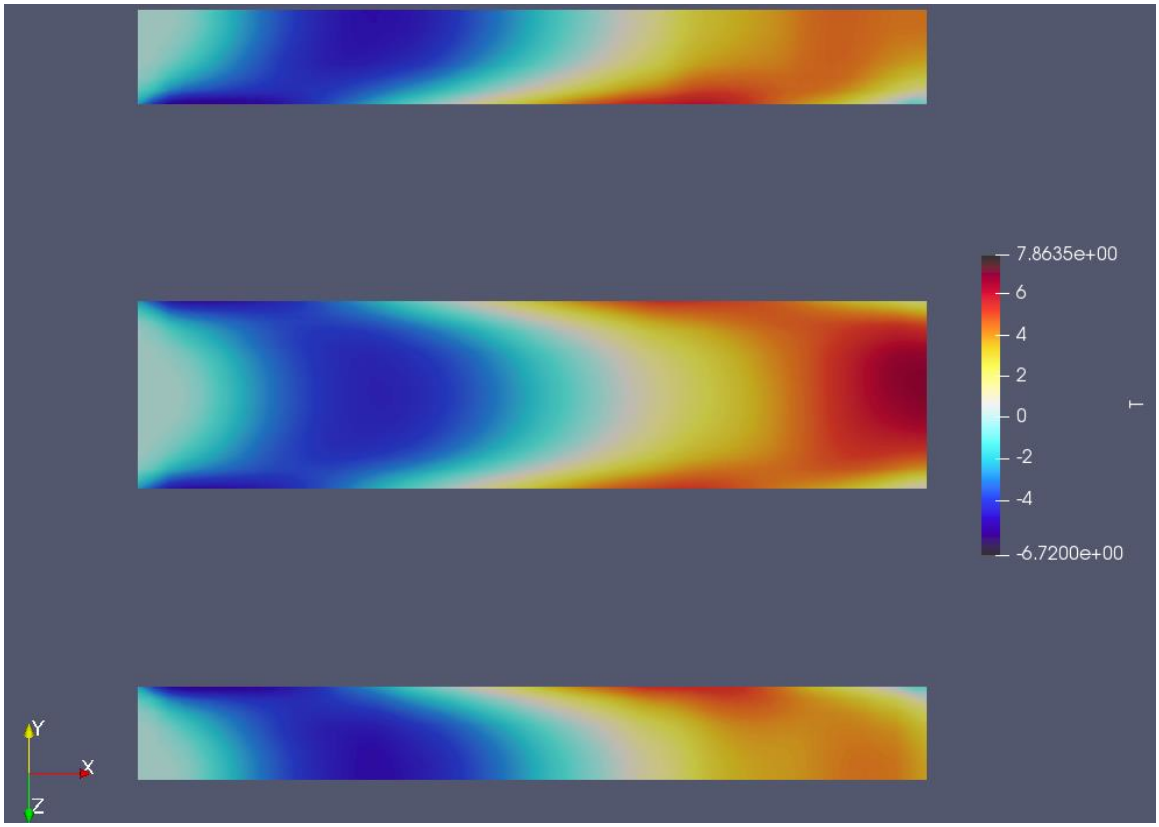


Figure 13: 2×2 Difference in Moderator Temperature, Coupled vs. Uncoupled (K)

**Comparison of Power Relaxation to Adaptive Batching**

In addition to comparing the coupled code to the uncoupled simulation results, the 2x2 pincell case was used to test the effectiveness of the iteration schemes implemented. A comparison was done between the 2x2 pincell case convergence for unrelaxed Picard Iterations, power relaxation, and the adaptive batching method. This comparison was done for convergence on the RMS error of 1E-3 and of 5E-4.

Table 2 shows the computational time taken to run each case. The Picard iteration simulations took longer than 10 days to complete, even for 1E-3 convergence. In both cases run, adaptive batching offers a 10-15% decrease in runtime over power relaxation. Both power relaxation and adaptive batching offer greater than an order of magnitude improvement in computational time when compared to the unrelaxed Picard iteration.

Table 2: 2×2 Computational Cost, Algorithm Comparison

| Convergence Criterion | Picard Iteration (hh:mm:ss) | Power Relaxation (hh:mm:ss) | Adaptive Batching (hh:mm:ss) |
|---|---|---|---|
| 1E-3 | > (240:00:00) | 19:05:29 | 17:33:40 |
| 5E-4 | > (240:00:00) | 39:28:43 | 33:57:58 |

The further comparisons of convergence behavior show the data from the 1E-3 case. Figure 14 shows the comparison of the RMS power error, the max normalized power error, and the RMS temperature error. The x-axis in this plot is the number of particles that have been simulated and the y-axis is the error value. The RMS power error and the maximum normalized power error converge at a similar rate to each other, while the RMS temperature error converges at a significantly quicker rate. The RMS power converges to 1E-3 in 168000 particles, while the RMS temperature converges to 1E-5 in 60000 particles. Convergence of the system is done on the power distribution to ensure that convergence of both the T-H and neutronics are reached, due to the power distribution converging significantly slower than the temperature.

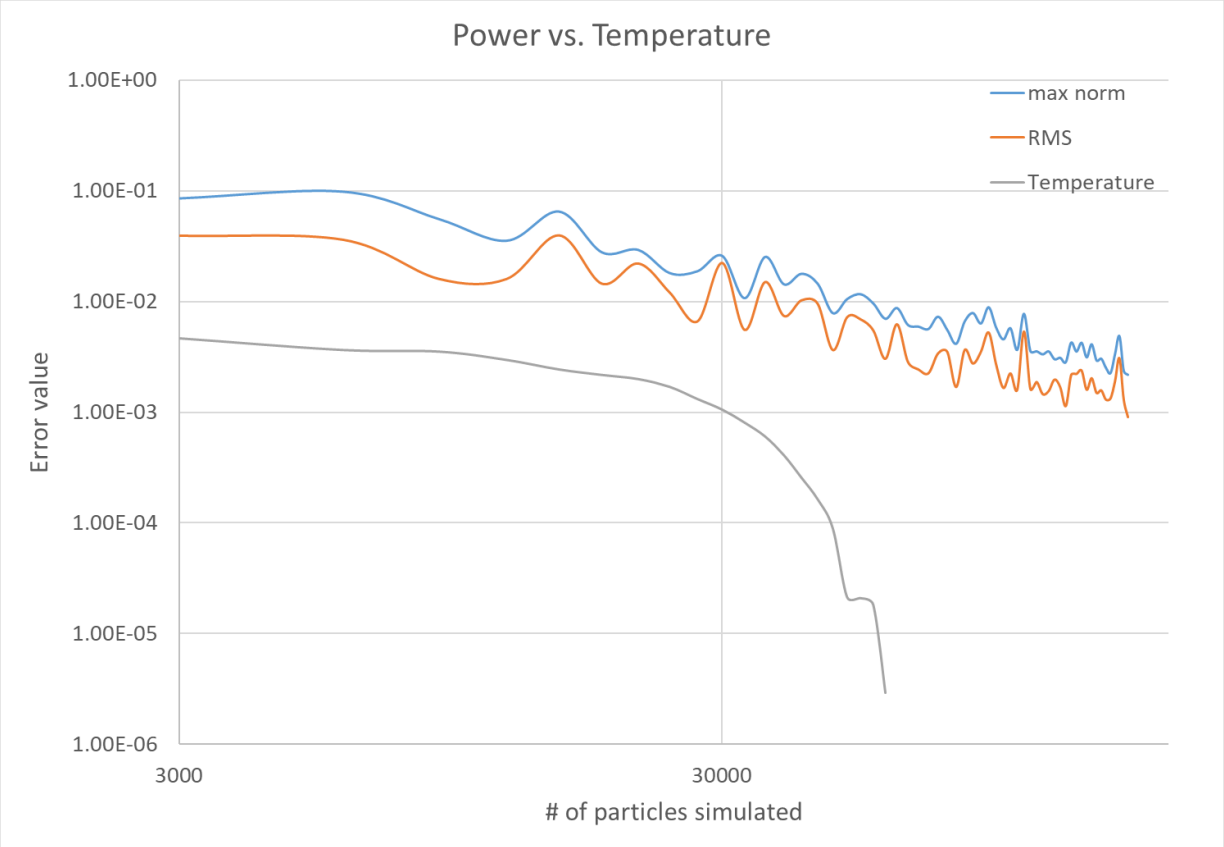


Figure 14: 2×2 Convergence, Power vs. Temperature

Figures 15 shows the convergence on the maximum normalized error for the power relaxation case against the adaptive batching case for a convergence criterion of 1E-3. The x-axis is the number of particle simulated over the coupled simulation and the y-axis is the maximum normalized error value at that point. It appears from this plot that the power relaxation method converges quicker on a per-particle basis. However, it should be noted that in adaptive batching, more particles are run once the system is more converged. Thus the weight of the most recent iteration is typically higher in adaptive batching later in the solver run than in power relaxation, based on Equation 15.

When accounting for the weights of the most recent iteration on the relaxed power, the error plot in Figure 16 is found. The y-axis in Figure 16 is the rate of convergence of the

maximum normalized error per a fixed number of particles simulated. In this figure, we see that the adaptive batching, once the number of particles per iteration becomes greater than the fixed value in the power relaxation, converges quicker.

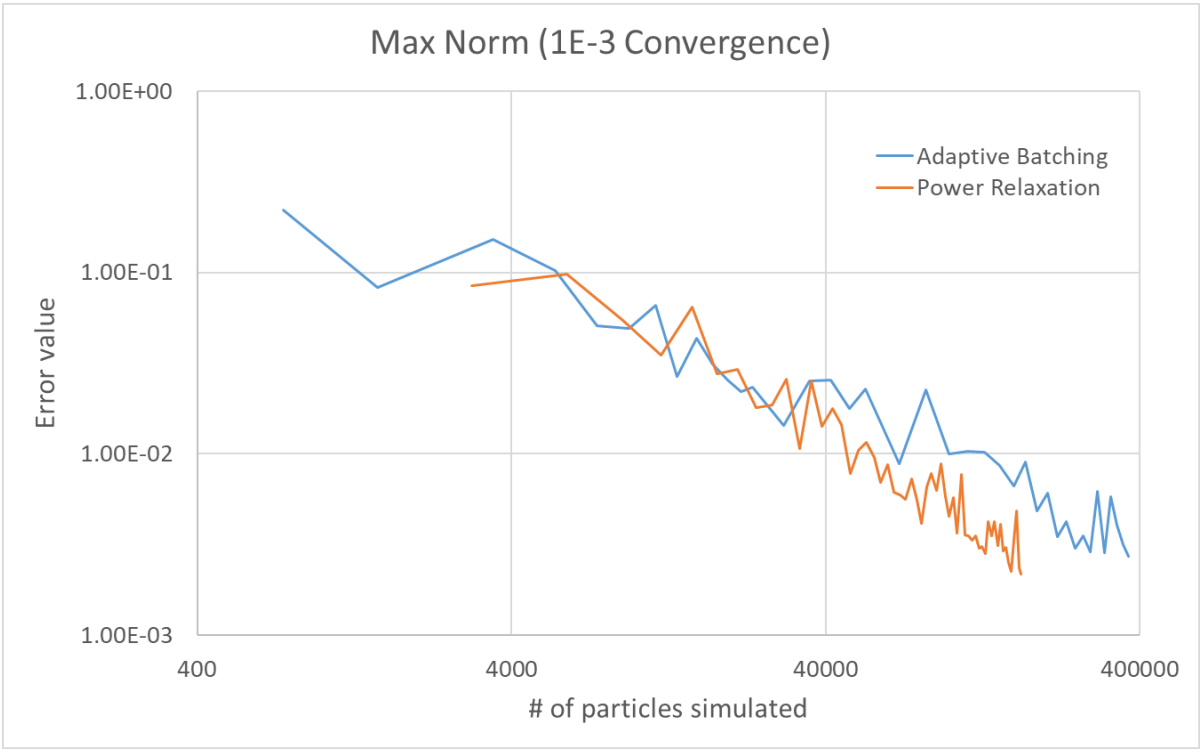


Figure 15: 2×2 Convergence, Max Norm Algorithm Comparison

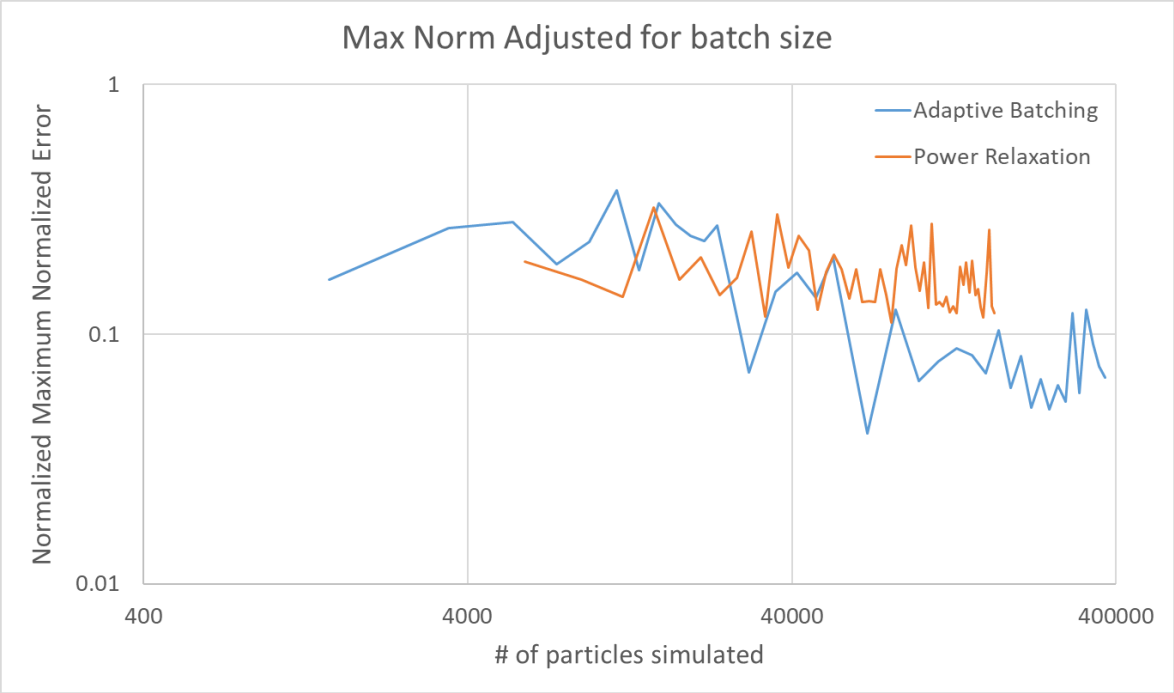


Figure 16: 2×2 Convergence, Max norm Algorithm Comparison, fixed Weight

The RMS error for the power relaxation case and the adaptive batching case are shown in Figure 17. The same behavior is seen as in the maximum normalized error, where the power relaxation appears to converge quicker. And as is shown in Figure 16, the same effect is seen as in Figure 18; the rate of convergence when the weight of the previous iteration is normalized is faster in adaptive batching.

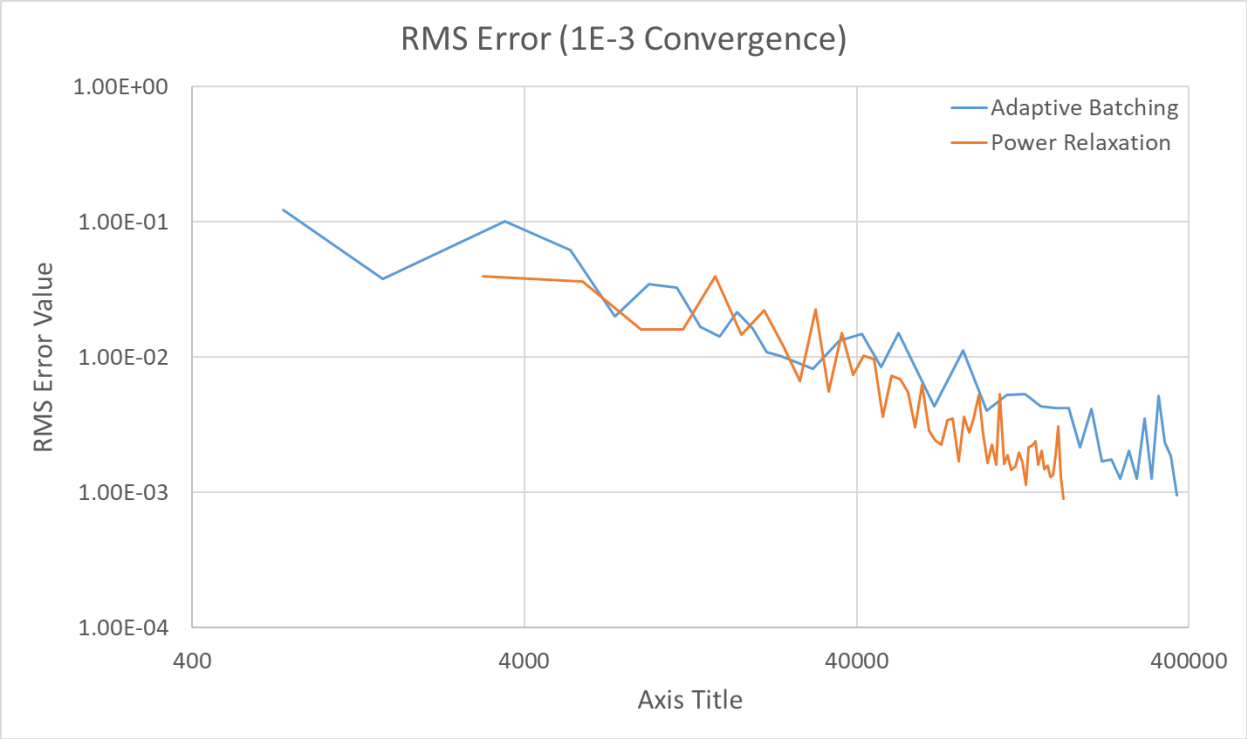


Figure 17: 2×2 Convergence, RMS error Algorithm Comparison

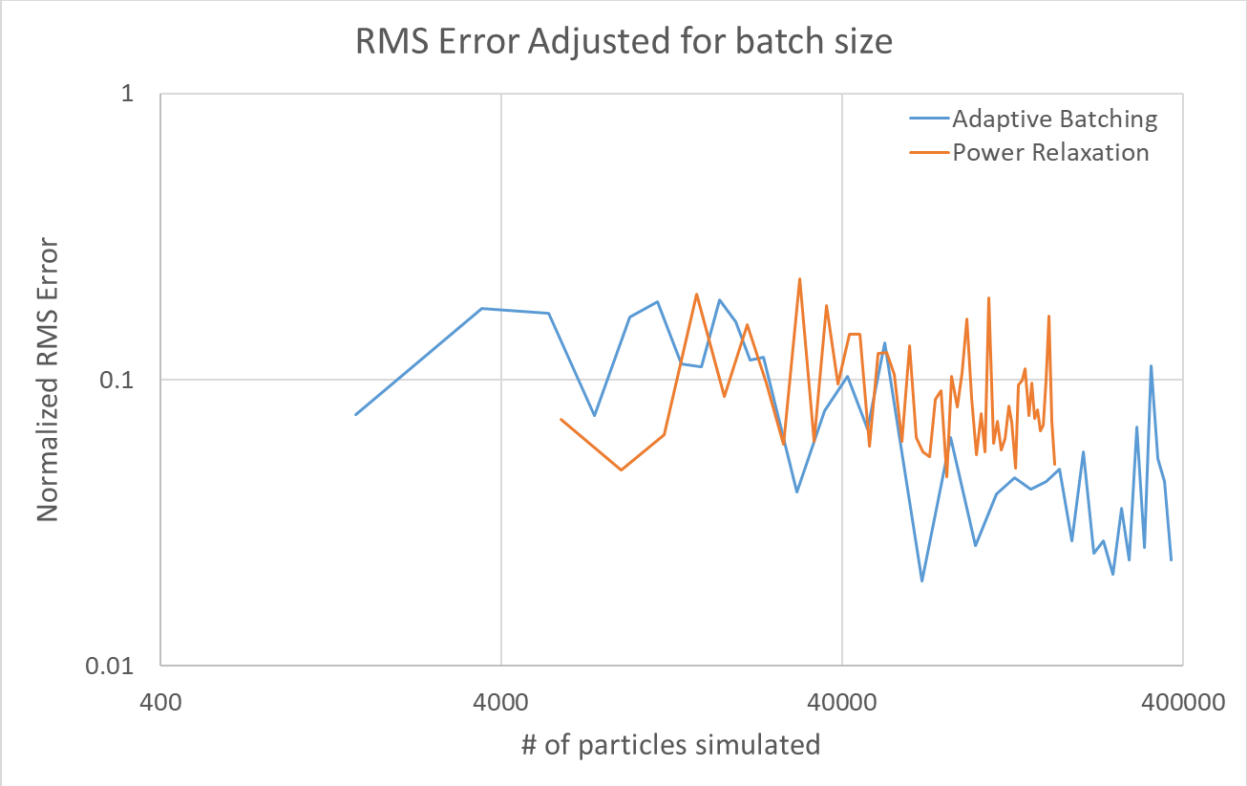


Figure 18: 2×2 Convergence, RMS Error Algorithm Comparison, Fixed Weight

It is important to note that while adaptive batching in these cases appears to converge slower, this is mainly due to the increased weight of later iteration compared to a traditional power relaxation method. Also, even though more particles were simulated to reach the same level of convergence in adaptive batching, the overall runtime of the code was less due to more efficient use of the T-H solver.

Figure 19 shows the convergence of the eigenvalue results over the course of the coupled solver's run. The error is given as a pcm change of the eigenvalue between iterations.

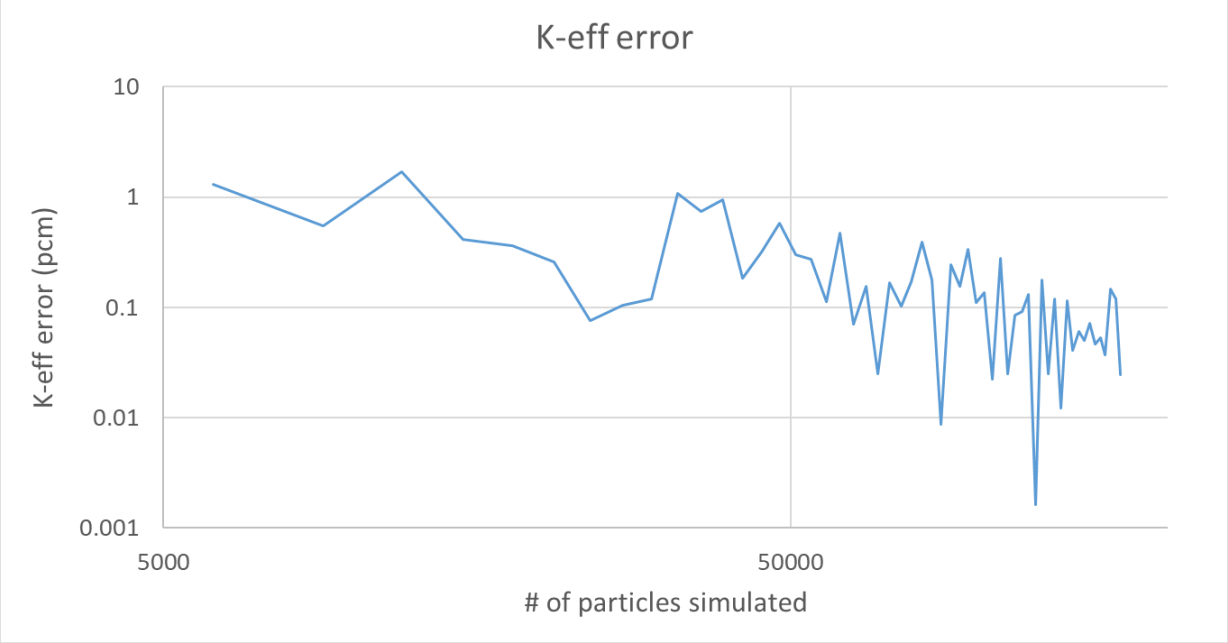


Figure 19: 2×2 Convergence, Eigenvalue Error

## ASSEMBLY RESULTS

For full testing of the coupled code, as well as verification purposes, a 17×17 PWR fuel assembly was tested. This assembly was based largely upon problem 6 in the VERA Core Physics Benchmark Progression Problems [Godfrey et al.], without spacer grids. The results of this case were compared against results for the same model from VERA for cross-code verification.

The general problem specifications for the model are given below in Table 3. The composition and densities of the fuel and structural material can be found in [Godfrey et al.]. The radial geometry is shown in Figure 20 with quarter symmetry being shown. Fuel is represented in red, Moderator in blue, Structure in green and fill gas in yellow. The model is setup with symmetry boundaries on the radial boundaries, both in the neutronics and T-H simulations. The inlet and outlet are modelled as vacuum boundaries in the neutronics

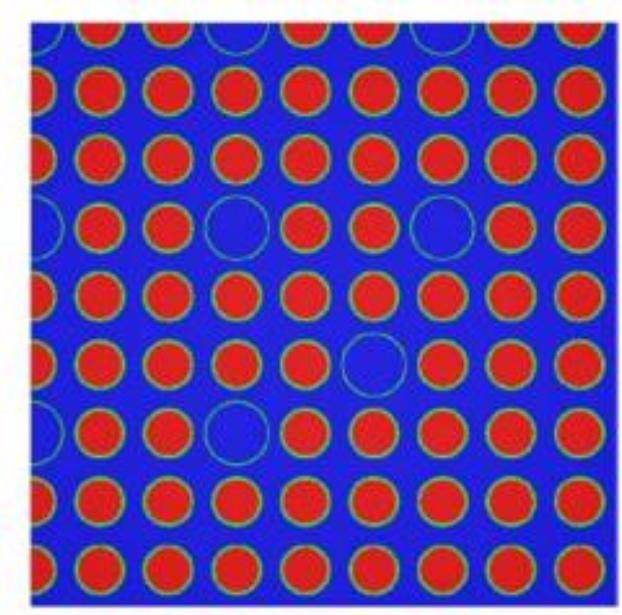

Figure 20: 17×17 PWR Assembly Geometry, Quarter Symmetry [Godfrey et al.]

Table 3: 17×17 PWR Assembly Specifications

| | | | |
|---|---|---|---|
| Pellet radius (cm) | 0.4096 | Fuel Density (g/cc) | 10.257 |
| Inner clad radius (cm) | 0.418 | Fuel Enrichment (%) | 3.1 |
| Outer clad radius (cm) | 0.475 | Inlet Coolant Temperature (K) | 565 |
| Rod pitch (cm) | 1.26 | System Pressure (MPa) | 15.5132 |
| Fuel stack height (cm) | 365.76 | Boron Concentration (ppm) | 1300 |
| Inner guide tube radius (cm) | 0.561 | Pellet material | $UO_2$ |
| Outer guide tube radius (cm) | 0.602 | Clad material | Zircaloy-4 |
| Rated Power (MW) | 17.67 | Fill gas material | Helium |
| Rated Coolant Mass Flow (kg/s) | 85.968 | Guide tube material | Zircaloy-4 |

The rod surfaces are modelled as no-slip velocity boundaries. The heat transfer between materials in the assembly is performed using the *turbulentTemperatureCoupledBaffleMixed* boundary condition, which enables conjugate heat transfer between the CFD regions. A fixed inlet velocity is specified based on the mass flow rate and moderator density at inlet conditions. The outlet pressure is fixed at the outlet. Turbulence modelling in OpenFOAM is performed using the k-Epsilon model for Reynolds Averaged Simulation (RAS).

All of the meshes used in OpenFOAM were generated with Gmsh and are made of tetrahedra extruded from a 2D radial mesh of the assembly. The moderator region has 1,369,400 cells and the CFD mesh for the fuel region has 705,400 cells.

**Coupled Calculation Results**

The average fuel temperature in the system was determined to be 765 K, and the maximum fuel temperature was 1103.7 K.

Figure 21 and 22 show the axial fuel temperature distribution in Kelvin. Figure 21 is taken at a slice along the assembly center and Figure 22 is taken at a slice along the assembly edge. The peak fuel temperature along the assembly centerline is 1095.2 K. The peak fuel temperature along the assembly edge is 1042.0 K. The highest temperatures occur at the fuel centerline, at the midchannel height. The rods along the centerline are noticeably hotter than those along the assembly edge.

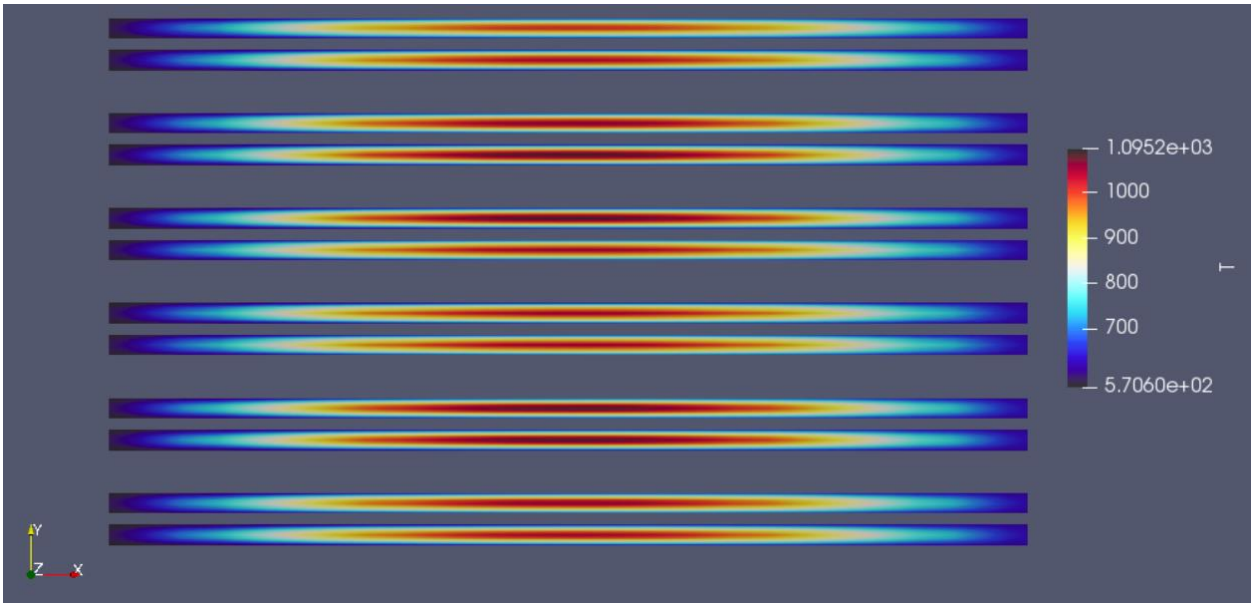


Figure 21: 17×17 PWR Assembly Fuel Temperature Axial Distribution along Assembly Centerline (K)

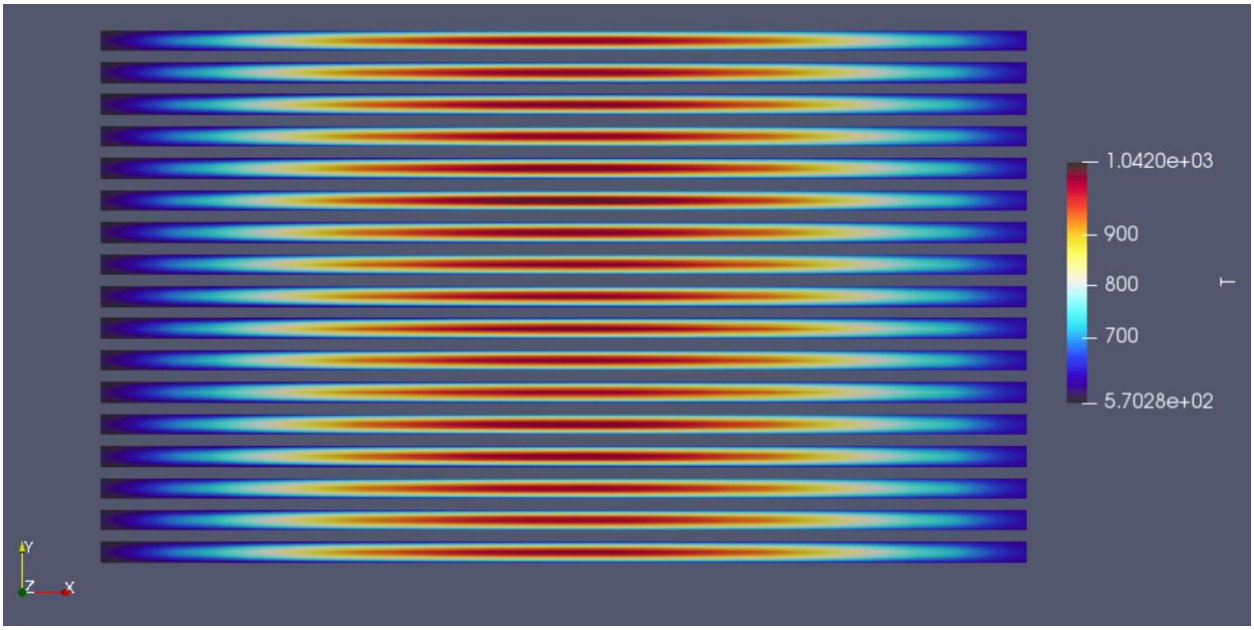


Figure 22: 17×17 PWR Assembly Fuel Temperature Axial Distribution along Assembly Edge (K)

Figure 23 shows the radial fuel temperature distribution, taken at the midchannel height in Kelvin. The peak fuel temperature along this slice is 1103.5 K. The highest fuel temperatures occur at the rod centerline, and in the rods nearest to the guide tubes. The rods along the assembly periphery have significantly lower temperatures, particularly at the rod centerline.

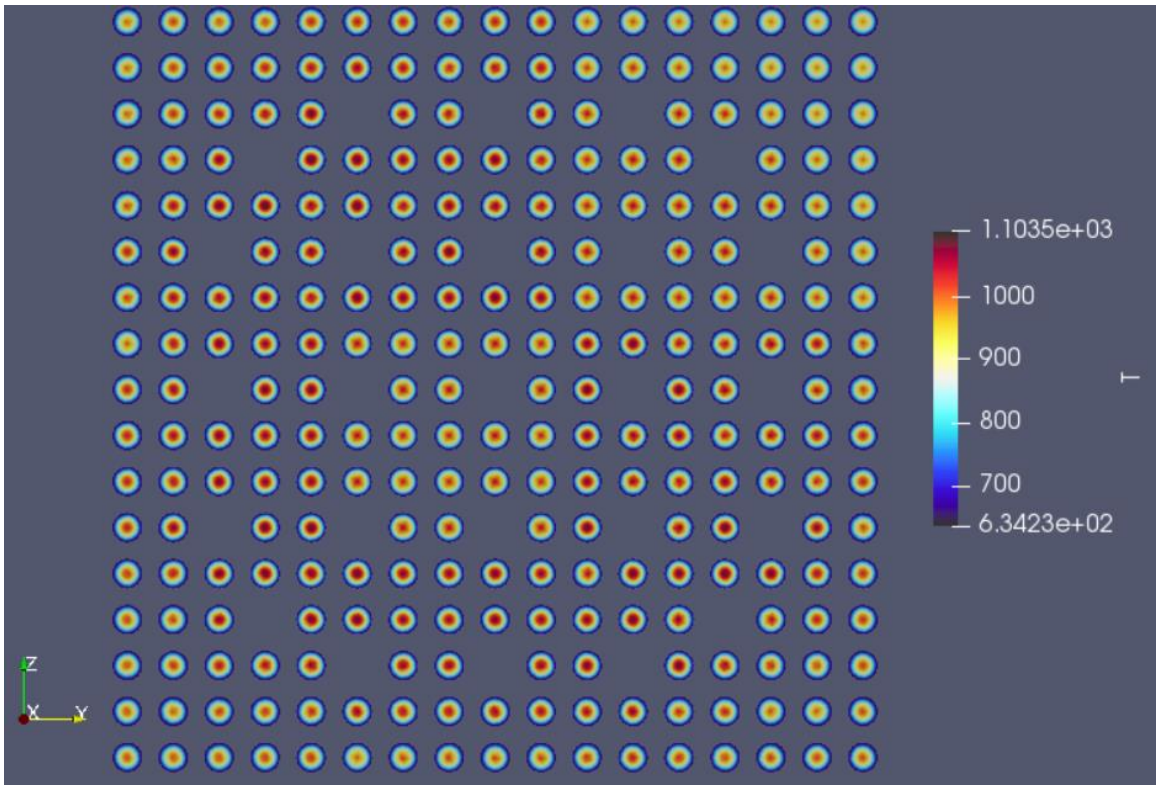


Figure 23: 17×17 PWR Assembly Fuel Temperature, Radial Distribution at Midchannel Height (K)

The average moderator temperature was found to be 587.05 K. The peak moderator temperature was 634.05 K. The inlet moderator temperature was 565 K and the average outlet moderator temperature was 608.58 K, giving a channel $\Delta T$ of 43.58 K.

Figures 24 and 25 show the axial moderator temperature distribution in Kelvin. Figure 24 is taken along the assembly centerline and Figure 25 is taken along the assembly edge. The peak moderator temperature along the assembly centerline slice is 633.52 K and along the assembly edge slice is 626.45 K. The moderator temperature is highest in channels that run along the hotter fuel rods, which occur more towards the center of the assembly and closer to guide tubes. The channel temperature is overall lower in channels that border guide tubes, as heat is only being added on one side of the channel.

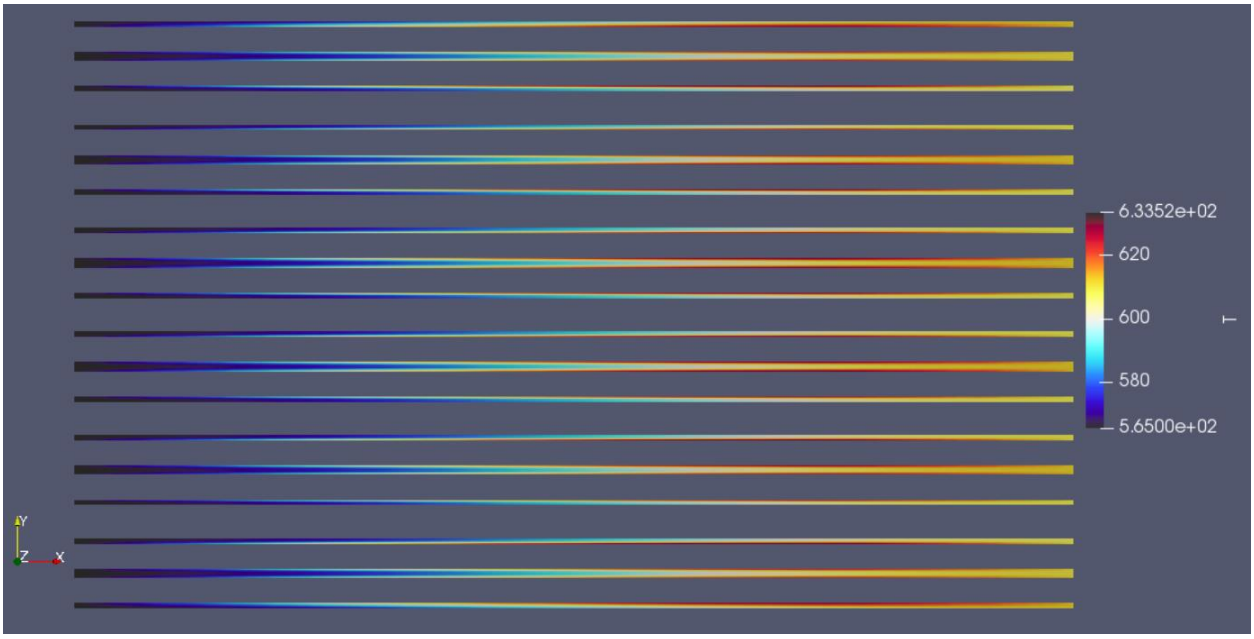


Figure 24: 17×17 PWR Assembly Moderator Temperature Axial Distribution along Assembly Centerline (K)

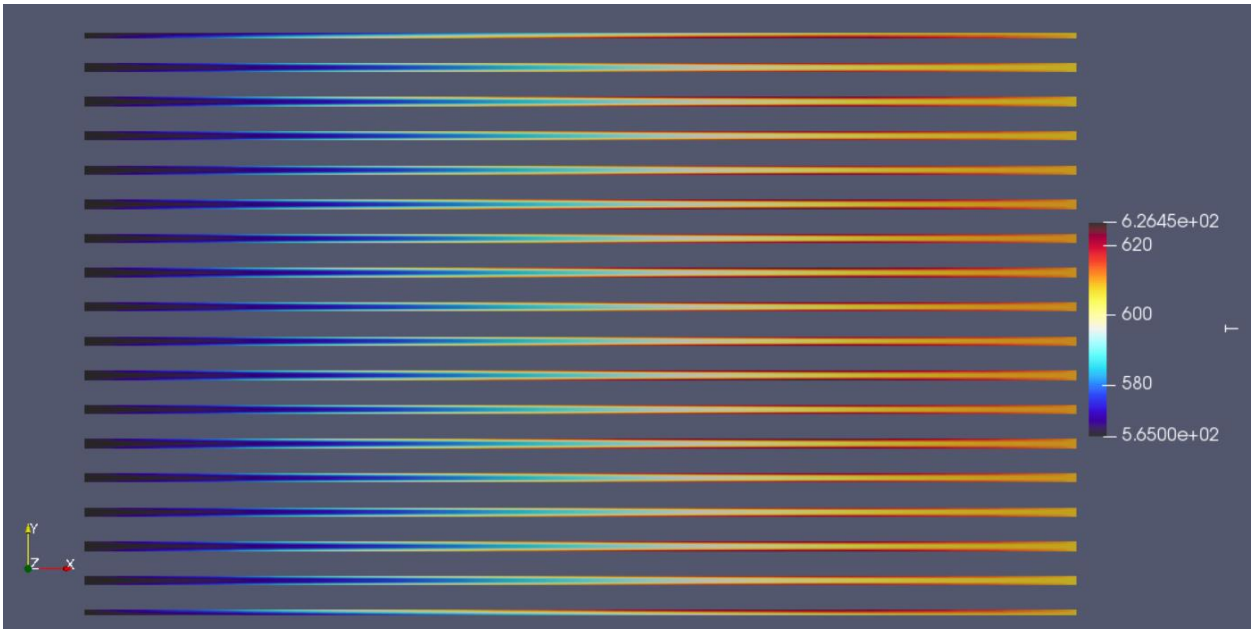


Figure 25: 17×17 PWR Assembly Moderator Temperature, Axial Distribution along Assembly Edge (K)

Figures 26 shows the radial moderator temperature distribution taken at mid-channel height in Kelvin. The peak moderator temperature at this height is 627.52 K. There is a sharp temperature gradient between the rod surface and the bulk fluid temperature at this height. The guide tube locations can be clearly seen by lower local temperatures in the subchannels surrounding them.

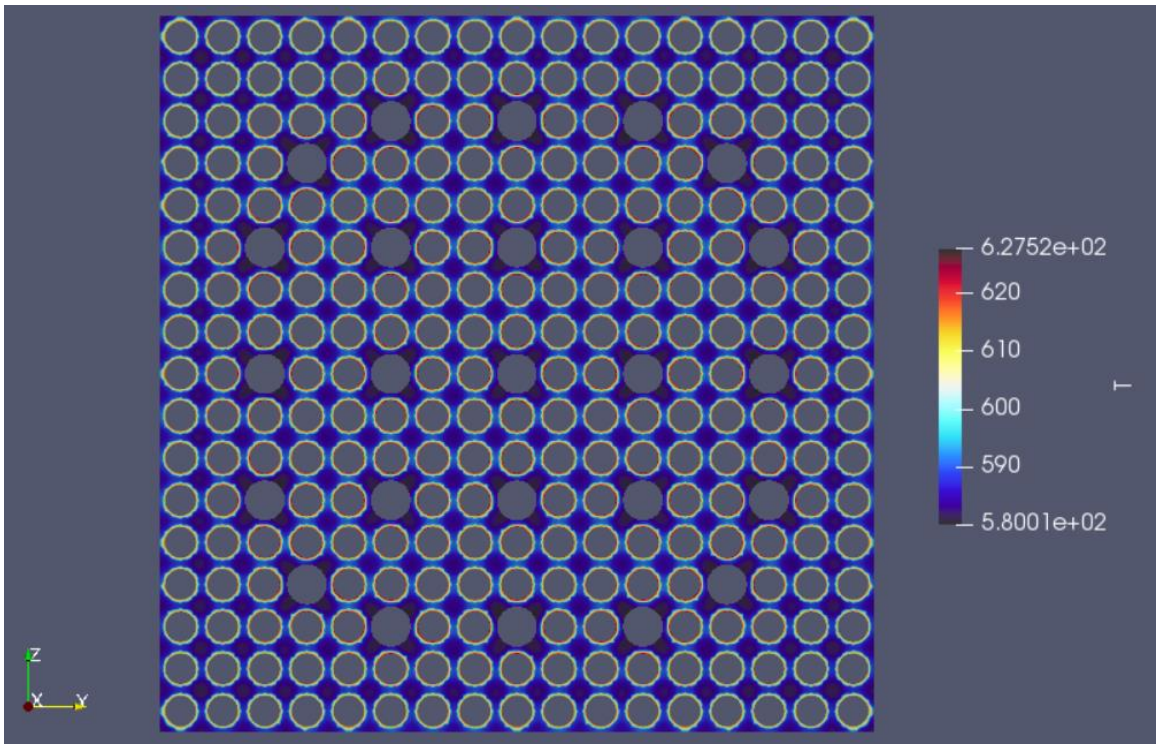


Figure 26: 17×17 PWR Assembly Moderator Temperature, Radial Distribution at Midchannel Height (K)

Figures 27 shows the radial moderator temperature distribution taken at channel outlet in Kelvin. The peak moderator temperature at this height is 618.05 K. The moderator temperature on the rod surface is lower at this height, due to lower fuel temperatures. However, the moderator temperature is much higher on average, and the heat is much more dispersed away from the rod surface than it was at the mid-channel height.

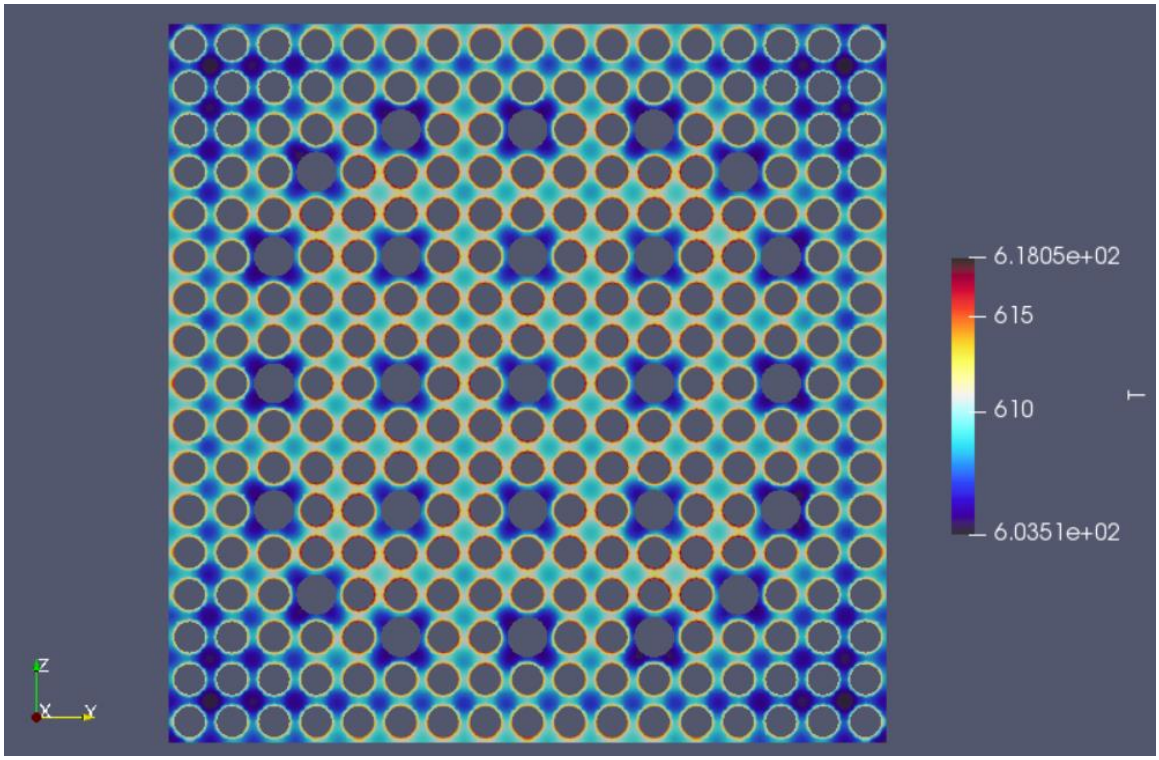


Figure 27: 17×17 PWR Assembly Moderator Temperature, Radial Distribution at Channel Outlet (K)

In the fuel, the peak power density was found to be 556.91 MW/m$^3$. The radial peaking factor was determined to be 1.1031 and the axial peaking factor was found to be 1.4594.

Figure 28 shows the radial power distribution at mid-channel height in W/m$^3$. The peak power density is 556.36 MW/m$^3$. The power is lowest on the assembly corners, and highest nearest the guide tubes. Rods next to guide tubes show large peaks on the side nearest the guide tube.

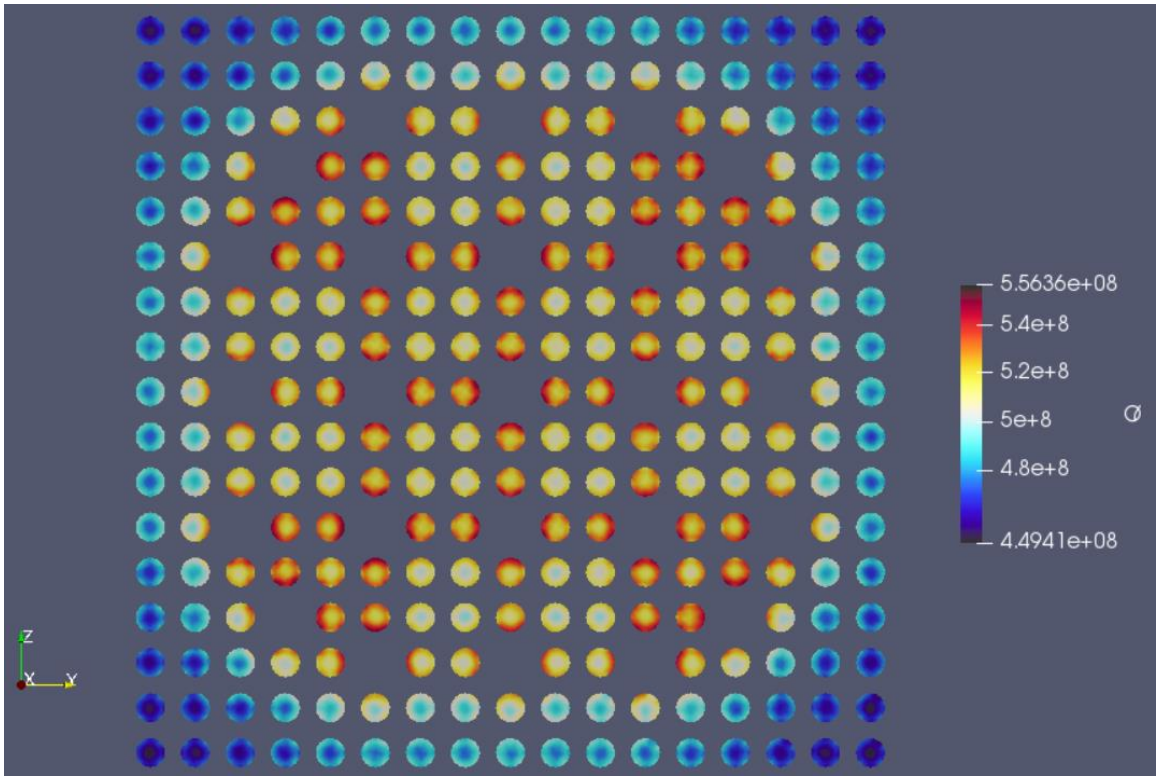


Figure 28: 17×17 PWR Assembly Power Density, Radial Distribution at Midchannel Height ($W/m^3$)

Figures 29 shows the axial power distribution along the assembly centerline in $W/m^3$. The peak power density is 546.92 $MW/m^3$ along this section. The power peaks in the center of the assembly both axially and radially. The radial distribution at any height is fairly flat due to the reflective boundary conditions, while there is a sharp drop in the power density closer to the inlet or outlet, due to the vacuum boundaries on those surfaces.

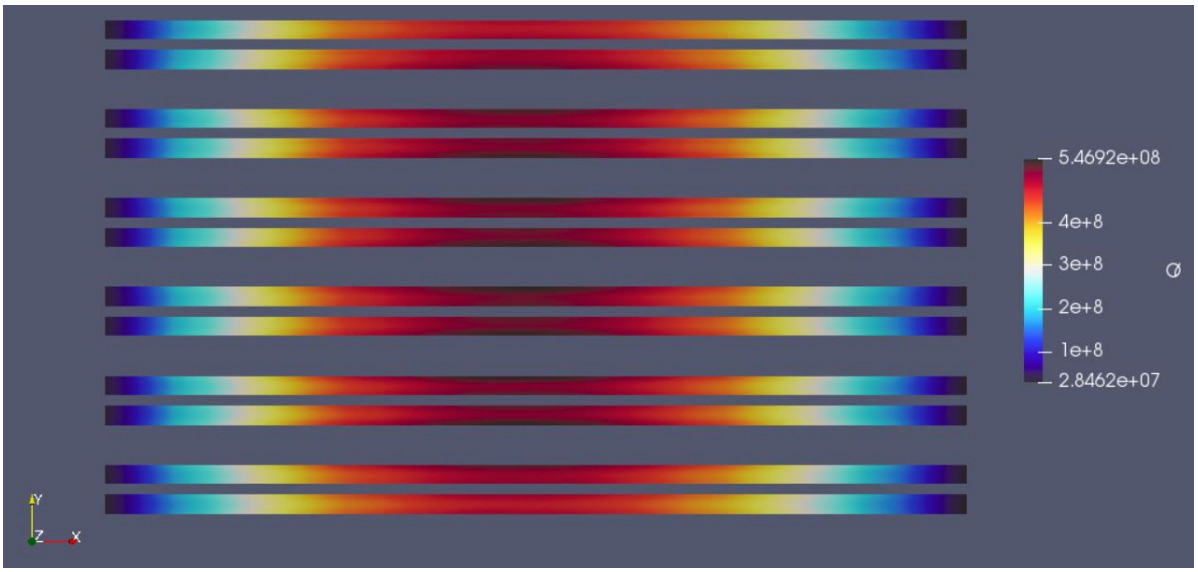


Figure 29: 17×17 PWR Assembly Power Density, Axial Distribution at Assembly Centerline ($W/m^3$)

Figure 30 shows the axial power distribution along the assembly edge in $W/m^3$. The peak power density is 495.03 $MW/m^3$ along this section. There is a similar sharp power peak axially as seen along the assembly centerline, however there is very little change in the power distribution radially compared to the centerline case.

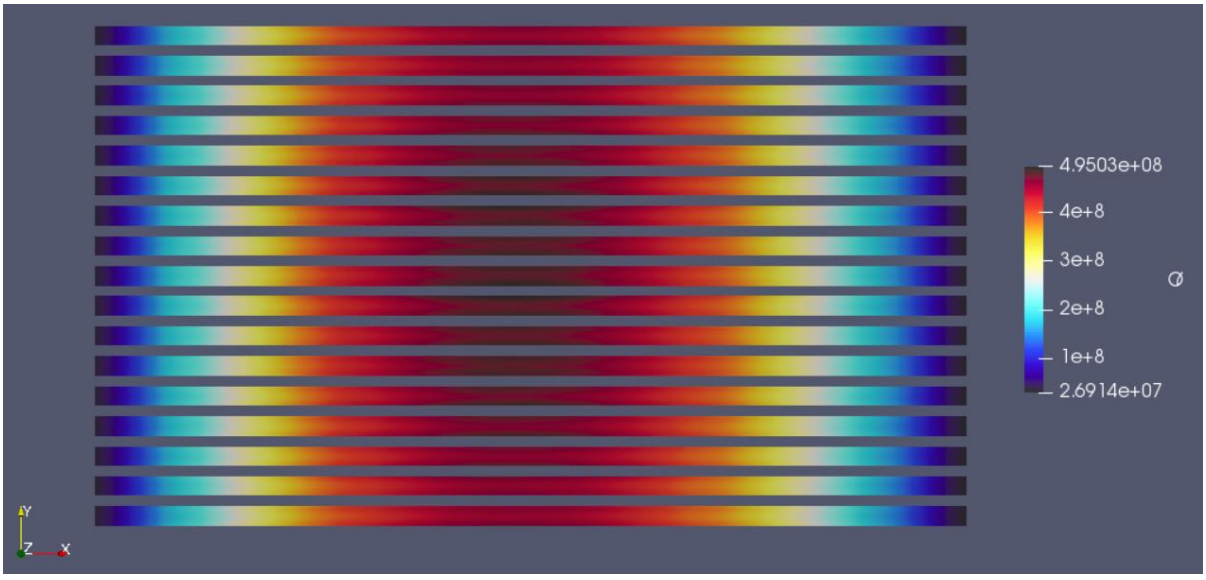


Figure 30: 17×17 PWR Assembly Power Density, Axial Distribution at Assembly Edge ($W/m^3$)

Figures 31 and 32 show the radial velocity profile at mid-channel height and at core outlet respectively. There is very little difference between the two distributions, as the boundary layer should be fully developed by mid-channel height. The peak velocity is 4.7518 m/s, and the average velocity is 4.08 m/s. The channel velocity is lowest around the guide tubes, as the channel diameter is smallest there.

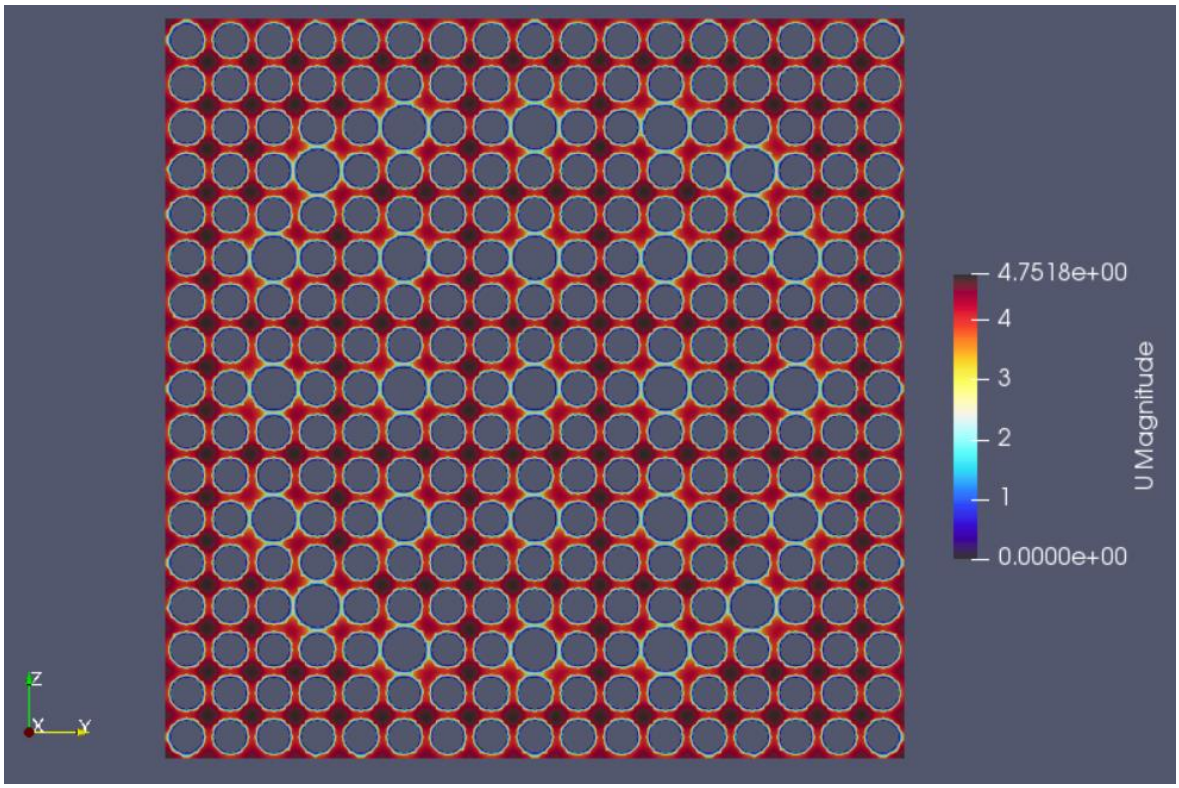


Figure 31: 17×17 PWR Assembly Moderator Velocity, Radial Distribution at Midchannel Height (m/s)

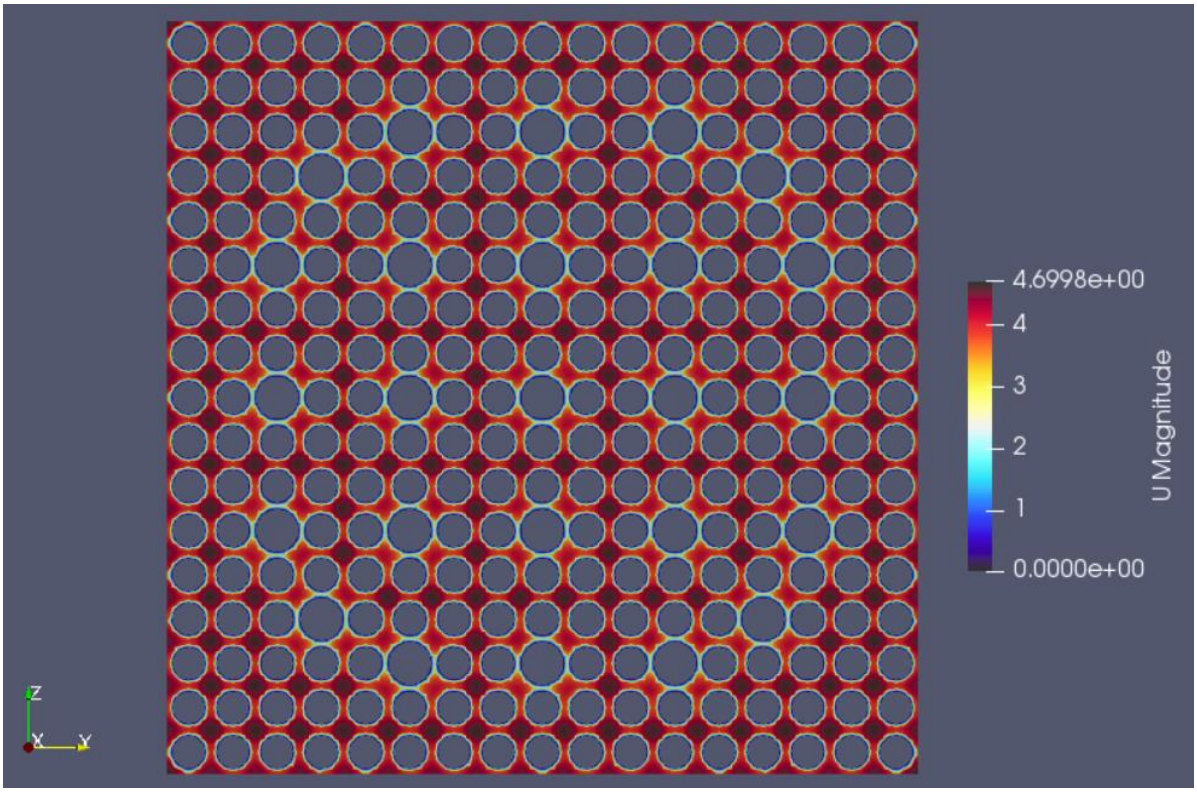


Figure 32: 17×17 PWR Assembly Moderator Velocity, Radial Distribution at Channel Outlet (m/s)

**Comparison to VERA Results**

To perform preliminary verification of the coupled code, the results from the run detailed above was compared to results from VERA. Table 4 shows the results for both codes, and the difference between the two results. The results shown in the table show around 1% or less difference between the two codes.

Table 4: Comparison to VERA Results

| | VERA | This work | Difference |
|---|---|---|---|
| Axial Peaking Factor | 1.45064 | 1.45935 | 0.600% |
| Average Coolant Density (g/cc) | 0.698785 | 0.69572 | 0.439% |
| Max Fuel Temperature (K) | 1117.1 | 1103.5 | 1.218% |
| K-eigenvalue | 1.1619 | 1.1706 | 752 pcm |

**Runtime and Convergence**

An analysis of the convergence behavior of the assembly case of the power distribution was performed. In addition, the convergence of the k-eigenvalue over the course of the solvers run was analyzed. In Figure 33, the convergence of both the maximum normalized error in the system and the RMS error of the power is shown. The plot is on a log-log scale with the number of particles simulated in OpenMC on the x-axis and the error value plotted on the y-axis. Both error plots appear to decrease linearly with respect to the log-log scale. The RMS error converges mostly uniformly, while much more uncertainty can be seen in the maximum normalized error plot. The RMS error reached a convergence level of 2E-3, after simulating 2.745 billion particles. The maximum normalized error reached a convergence of 8.8E-3 over the same number of particles simulated.

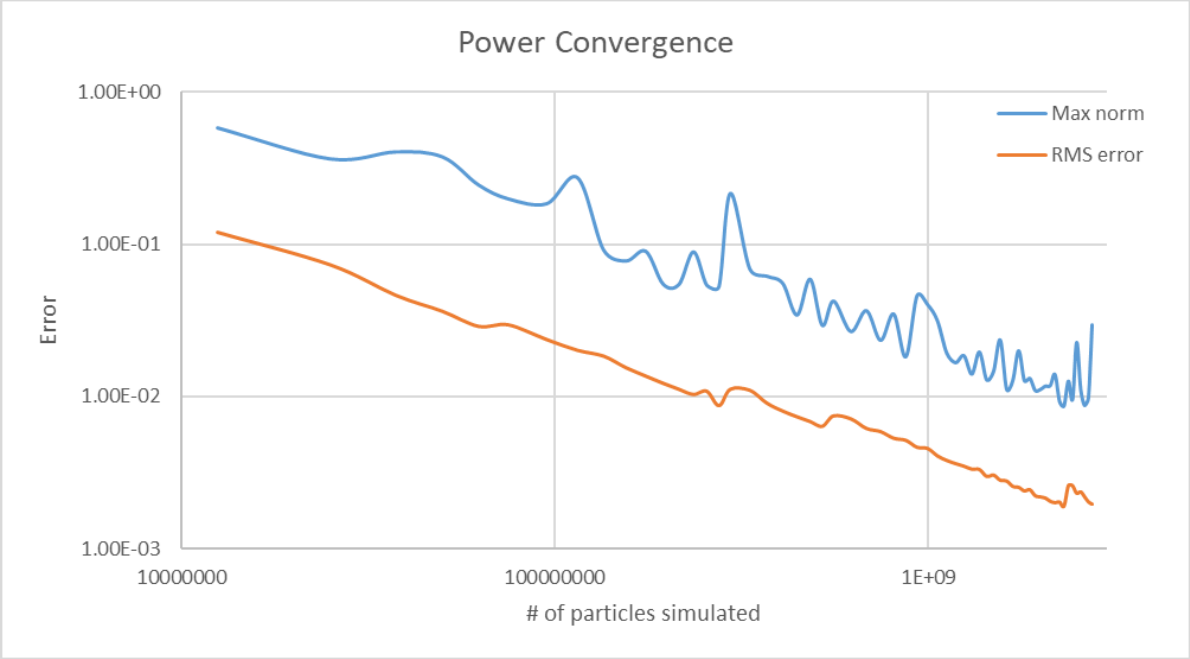


Figure 33: 17×17 PWR Assembly, Power convergence, Max Norm and RMS

Shown in Figure 34 is the convergence of the k-eigenvalue over the course of the simulation. The plot is shown on a log-log scale, with the convergence of the k-eigenvalue in pcm shown on the y-axis and the number of particles on the x-axis. The k-eigenvalue reached a convergence of 0.1 pcm of the course of the simulation.

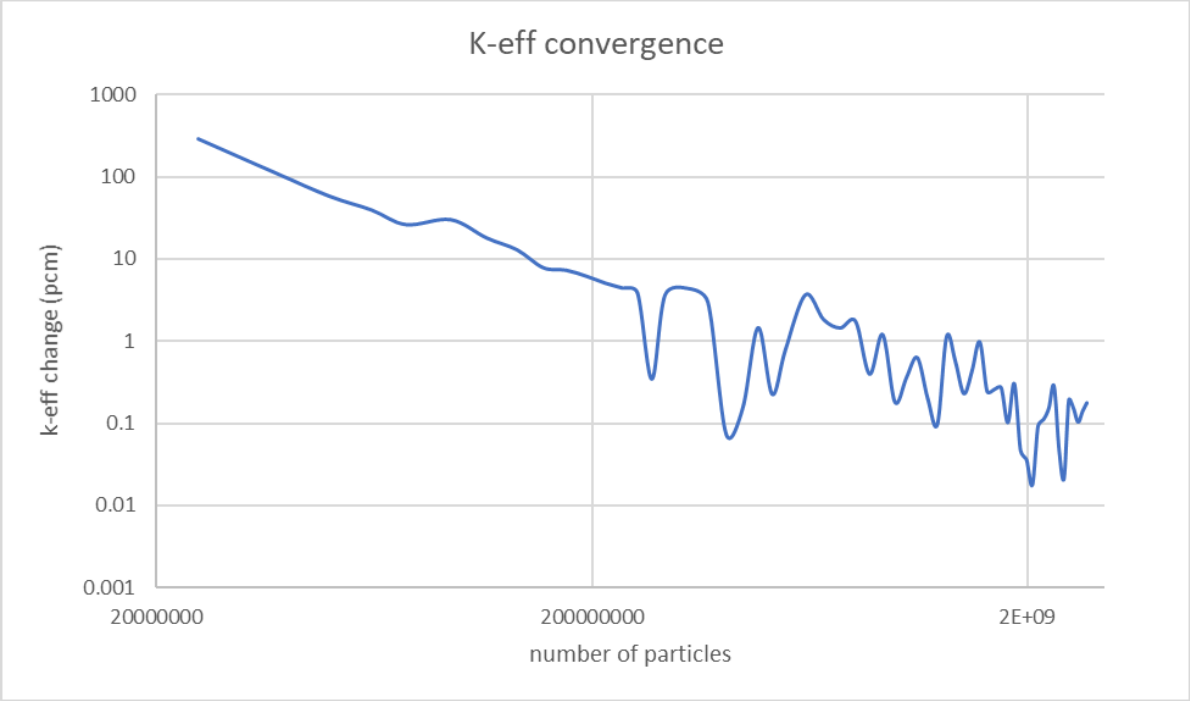


Figure 34: 17×17 PWR Assembly, Eigenvalue Convergence

**Convergence of Power distribution due to Tally Mesh**

A comparison was done to analyze the effect of the fidelity of the tally mesh data that is being mapped on the power distribution after mapFields has been run. This was done by adjusting how fine the radial mesh for the tally is. The three meshes that were tested over were for tally meshes with 40×40, 30×30, and 20×20 cells for each pincell in the assembly. The tally meshes are shown in Figure 35, where blue is the moderator region, orange is represents the clad, white represents the gap, and gray represents the fuel.

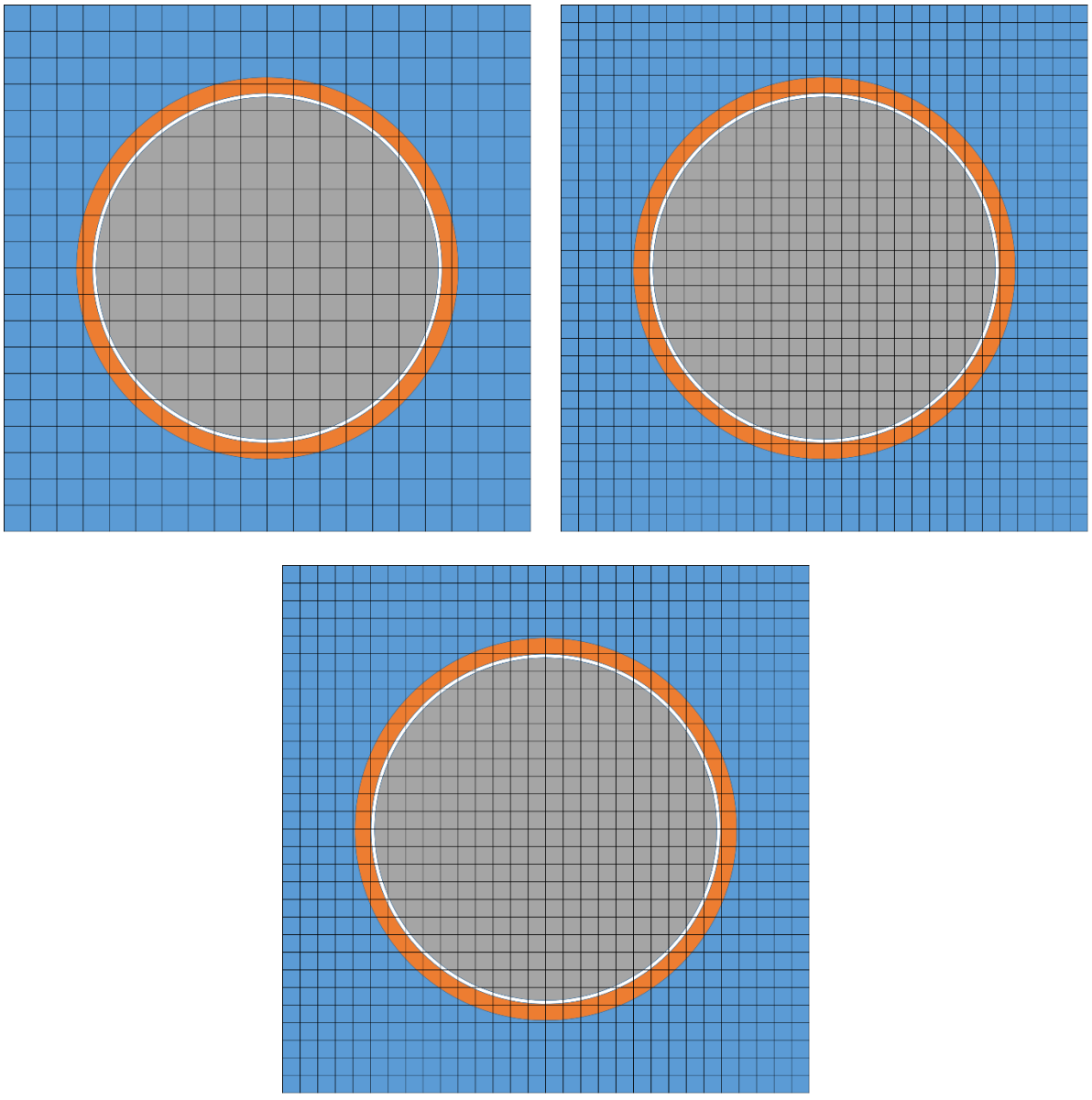

Figure 35: Tally Mesh overlay for Pincell: 20×20, 30×30, 40×40

The power distribution that is mapped from a tally mesh of 40x40 mesh cells radially for each pincell is shown in Figure 36.

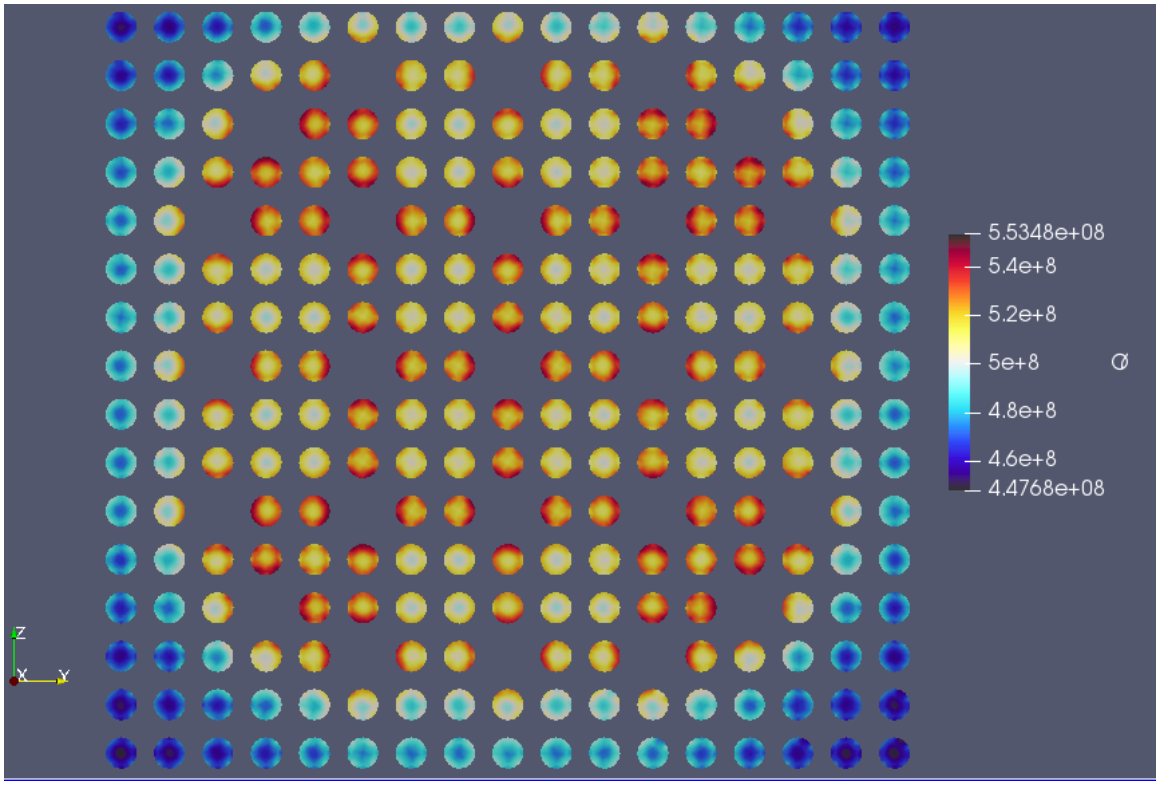


Figure 36: 17×17 PWR Assembly Power, Tally Mesh, 40x40 Radial Mesh ($W/m^3$)

The power distribution that is mapped from a tally mesh of 30×30 mesh cells radially for each pincell is shown in Figure 37.

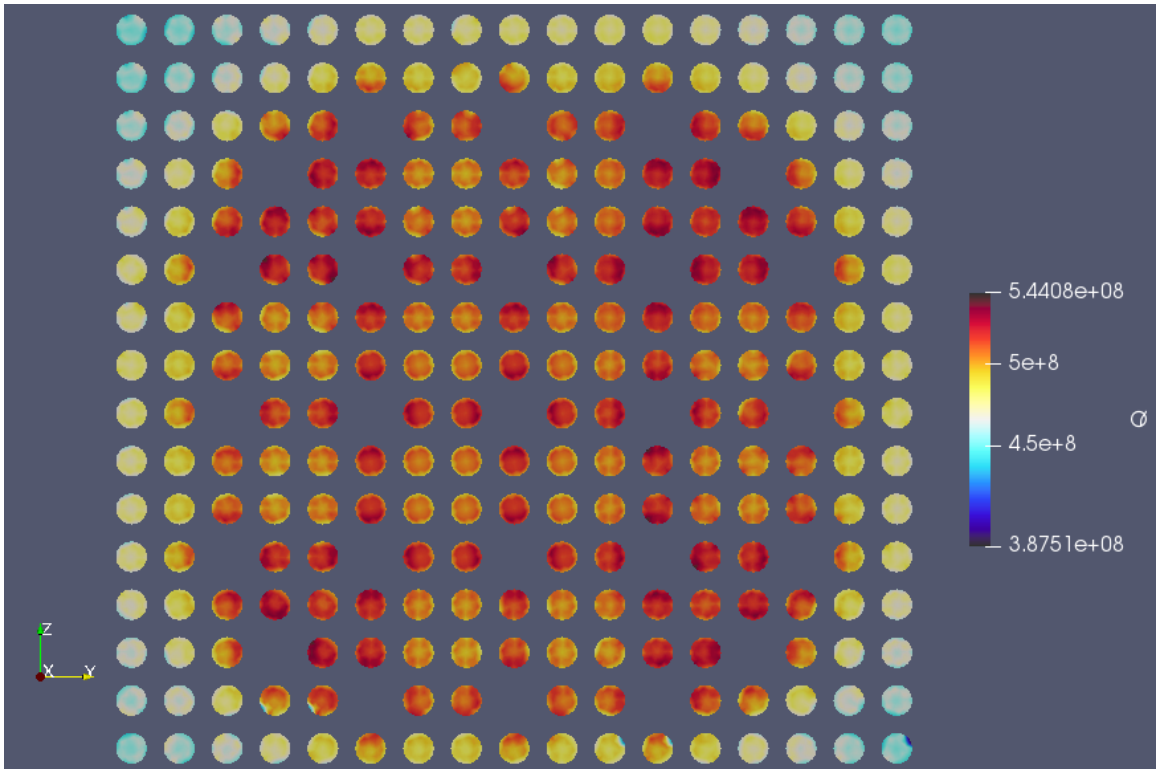


Figure 37: 17×17 PWR Assembly Power, Tally Mesh, 30x30 Radial Mesh ($W/m^3$)

The power distribution that is mapped from a tally mesh of 20×20 mesh cells radially for each pincell is shown in Figure 38.

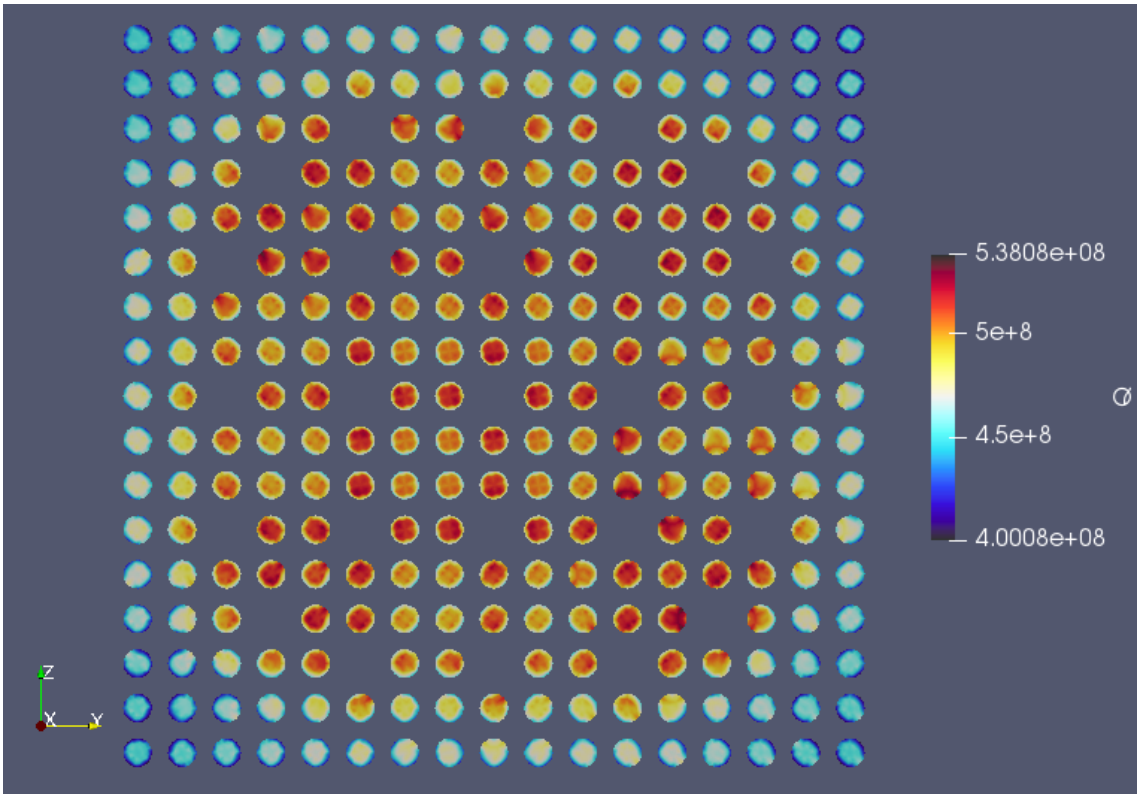


Figure 38: 17×17 PWR Assembly Power, Tally Mesh, 20x20 Radial Mesh ($W/m^3$)

The difference between the 20×20 and 30×30 meshes is shown in Figure 39. The largest increase in any cell is 51.76% when refining the mesh. The largest decrease in any cell is 41.42% when refining the mesh. The average change is 4.19% between the two meshes. The largest changes typically occur near the edges of fuel rods, particularly away from the y or z-axes.

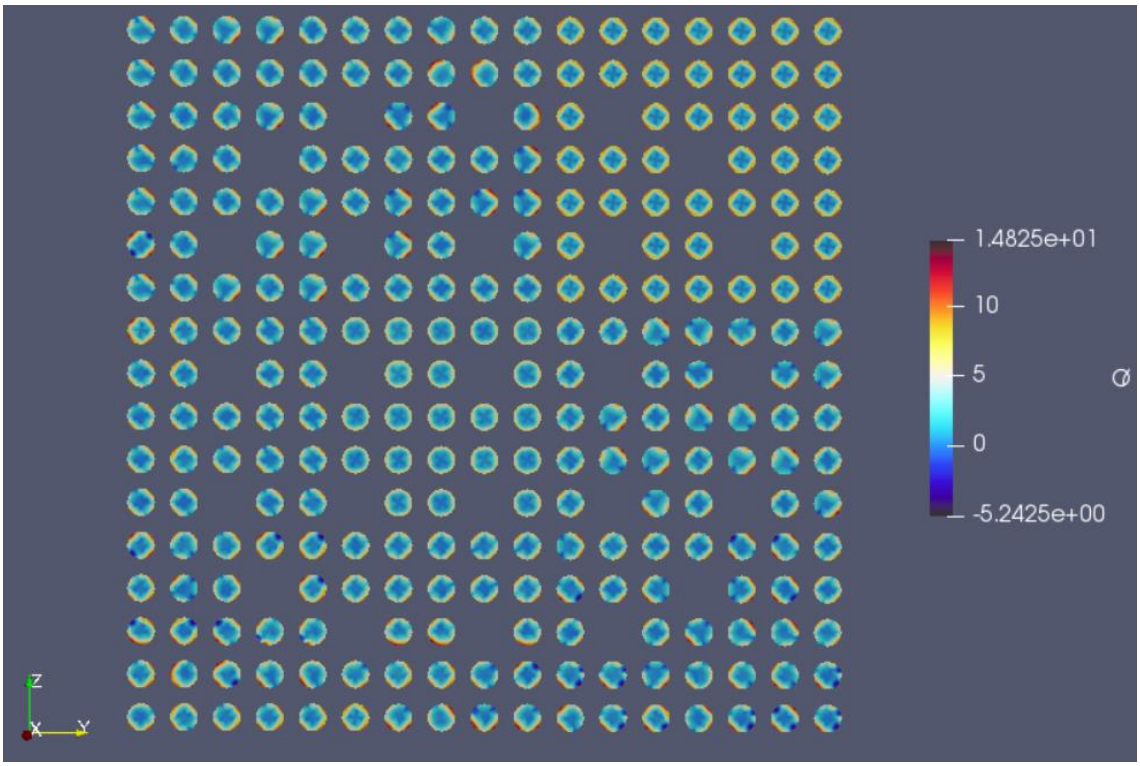


Figure 39: 17x17 PWR Assembly Power Difference, Tally Mesh, 30×30 – 20×20 Radial Mesh (%)

The difference between the 30×30 and 40×40 meshes is shown in Figure 40. The largest increase in any cell is 36.47% when refining the mesh. The largest decrease in any cell is 43.6% when refining the mesh. The average change is 2.52% between the two meshes. The largest changes typically occur near the edges of fuel rods, particularly away from the y or z-axes.

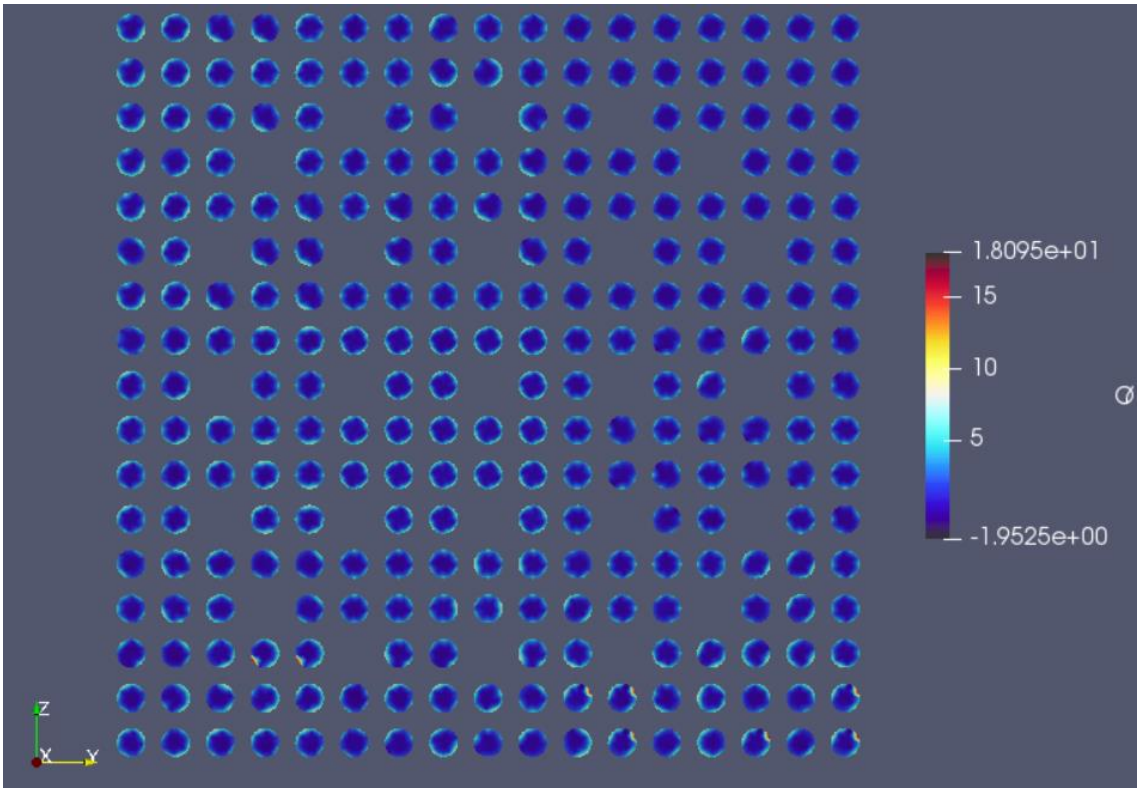

Figure 40: 17×17 PWR Assembly Power Difference, Tally Mesh, 40×40 – 30×30 Radial Mesh (%)

# CONCLUSIONS

In this study, a multi-physics coupling was performed between the open source Monte Carlo neutronics code OpenMC and the open source CFD code OpenFOAM. The purpose of this multi-physics coupling is to be able to perform flexible, high-fidelity simulations of nuclear reactor geometries, such as fuel assemblies. Multi-physics better captures the feedback between the physics in the reactor, such as the effect of temperature on Doppler broadening of cross sections. The use of high-fidelity codes such as a CFD code like OpenFOAM yields more accurate computational results, and results, such as velocity distributions, that cannot be found with lower-fidelity codes.

This coupling was based off a Picard Iteration, and due to the instability and inefficiencies off unrelaxed Picard Iterations, a power relaxation strategy was implemented. In addition to the power relaxation strategy, an adaptive batching scheme was implemented to further decrease the computational cost of the coupler. Due to the high computational cost of both the Monte Carlo and CFD simulations, small efficiency gains in the iteration scheme lead to significant improvements in runtime.

The results of the 2×2 case show that it is important to correctly take into account the feedback by running OpenMC and OpenFOAM in a multi-physics simulation as opposed to being run on their own. The coupled simulation of OpenMC results in a difference of 877 pcm in the eigenvalue calculation and an average difference of 0.8% in the power distribution, with changes of up to 5%. The coupled simulation of OpenFOAM results in a difference of 4.38 K in the outlet temperature distribution. The average absolute difference in the moderator temperature distribution between the coupled and uncoupled simulations is 1.65 K, and the maximum

difference was 7.86 K. The average difference in the fuel temperature distribution was 66 K and the maximum difference was 257 K.

The differences between the results of the standalone simulations and the coupled simulation are significant and give a strong justification for the use of multi-physics simulations, particularly due to the high-fidelity of the results.

The convergence results from the 2×2 case show a great deal about the behavior of the various algorithms discussed in this work. First, the utility of relaxing a Picard iteration with a stochastic based relaxation technique is clear, in the order of magnitude faster runtime of both power relaxation and adaptive batching when compared to the unrelaxed Picard.

Second, the convergence results on the 2×2 case show that the addition of an adaptive batching scheme to the power relaxation method offers a noticeable improvement in the runtime of the simulation. The adaptive batching case ran 8% and 16% on the 2 comparison cases run on the 2×2 case when compared to the traditional power relaxation results.

Of particular interest in the behavior of the adaptive batching when compared to the power relaxation case was the apparent convergence rate per particle. Adaptive batching appeared to have a slower rate of convergence than power relaxation after a certain point. However, this was caused by the fact that the convergence in adaptive batching has a higher weight as the number of particles increase. The result is that it appears that, relative to the weights of the most recent iteration, adaptive batching yields a faster convergence rate. This behavior should be studied more in depth in the future, both from a mathematical perspective and a computational perspective. Also, even though the convergence rate appeared slower per particle, adaptive batching still had a noticeably faster runtime than traditional power relaxation.

A more detailed comparison of the adaptive batching method to the traditional power relaxation method should be performed. Testing over a wider variety of test cases, in addition to over a variety of convergence criterions, should be performed, such as a full fuel assembly or a research reactor with plate fuel. While the tests performed here show that adaptive batching outperforms power relaxation, there is not enough data to definitively say which method performs better. In addition to addition computational testing of the algorithms, a full mathematical derivation of the convergence behavior, similar to [Ivanov et al.] should be performed.

On adaptive batching, an optimization of the method of determining batch size, and thus number of particles per iteration is needed. Currently the batch size is determined based off comparison of the RMS power error to a user input table of batch sizes and convergence values. However, two main changes could improve the functioning of the adaptive batching: changing to checking the error on the temperature as opposed to on the power and having the batch size determined based off the convergence of the power, the temperature, and the current batch size and CFD runtime.

The goal of adaptive batching is to run fewer Monte Carlo particles when the temperature is poorly converged, so this result does not end u weighted to heavily in the final power distribution. Once the temperature converges, many Monte Carlo particles are to be run, as the temperature won't change significantly in further CFD runs. This has a double effect of running fewer CFD runs after this point. While the power convergence should follow the temperature convergence somewhat, the temperature is what adaptive batching is based upon.

The current implementation of the user input table for control of adaptive batching is useful and offers a great deal of control of the simulation run, but it is not the most optimal

method. The development of some model to guide the batching size based off the current behavior of the simulation could be very useful and could offer a further improvement on the increased efficiency already seen in adaptive batching.

In addition to this improvement in the adaptive batching performed on the Monte Carlo calculation, a similar method applied to the CFD calculation in OpenFOAM could be applied. An adjustment of the number of timesteps simulated in OpenFOAM based on the convergence of the temperature and power could increase the benefit seen in adaptive batching by decreasing the number of timesteps when more converged. A similar method for developing a model to control the number of timesteps run based off the convergence of the system as discussed for the Monte Carlo could be adopted.

The results of the assembly case show the high-fidelity of results that can be produced with this coupled solver. Highly detailed local results for the temperature, power density, and velocity can be determined. However, there are issues with the simulation that can be improved. Two features of OpenMC could be improved on to improve the efficiency of the coupler as well as greatly decreasing the memory requirements for the code. These are implementing a method of inputting temperature distribution as opposed to cell-by-cell temperatures in each cell and by outputting the tally results in a method other than the current fixed cartesian method.

If each material region were able to have a temperature distribution, then in the assembly case there would be 289 clad regions, 264 fuel regions, and 26 moderator regions. Thus, 579 cells would be needed to accurately model the system. In the current method, over 3000 cells are used to model a 17×17 fuel assembly, and this only offers around 12 cells for each pincell to model the fuel, clad, and moderator. The accuracy of the simulation would increase significantly,

with less constraints on memory. As discussed in [Novak et al. Tuominen et al.], the implementation of delta tracking could offer a way to implement this strategy.

Use of a fixed Cartesian tally mesh can cause inaccurate mapping of the power distribution in a multi-physics simulation. This can be seen in the results of the tally mesh convergence study performed, where refining the tally mesh has a significant effect on the power distribution mapped to OpenFOAM. Two methods have been discussed previously that could alleviate this issue. The first was discussed in [Tuominen et al.], using the final mesh for the T-H as the mesh on which tallies are accumulated, which prevents an interpolation or mapping issues. The second was discussed in [Novak et al.], using finite element tallies (FET) to expand the power distribution from OpenMC. Either of these methods will offer a significant improvement in both the accuracy of the fission power results in OpenFOAM, as well as in the efficiency of the calculation, both due to decreased runtime and decreased memory requirements.

The comparison of the assembly case results to the results from VERA show strong agreement and form the start to verification for this new solver. The difference in the axial power peaking factor, average moderator density, and the peak fuel temperature were all at or less than 1%, and the difference in the eigenvalue from the 2 codes was approximately 700 pcm. More work needs to be done to verify the behavior and results of this coupled solver, but this initial comparison to VERA gives a strong basis for verification.

The convergence comparison on the tally mesh show that the fineness of the tally mesh has a strong effect on the power distribution and how it gets mapped onto the OpenFOAM mesh. This refinement appears to cause lower powers in the center of fuel rods and higher powers on the rod periphery. This is most likely due to the Cartesian cells of a coarser mesh on the

periphery not capturing the circular shape of the fuel rod, causing an interpolation issue. This can be seen in the fission power distribution in Figure 38.

Increasing from 20×20 pincells radially to 30×30 pincells causes large changes in the power distribution. The 20×20 case consistently appears to have a higher power in the center of fuel rods and a lower value on the rod periphery. When refining from 30×30 to 40×40, a similar effect is seen, with refinement causing the power in the center of the rod to decrease, and some locations on the rod periphery increase.

Future work on this system will focus mainly on increasing the efficiency of the system or increasing the flexibility of the system.

A large focus for increasing efficiency of the system is to optimize the implementation of the adaptive batching scheme. Developing a more advanced strategy or model for determining the number of particles for the next OpenMC iteration than the current strategy of checking the temperature convergence is needed. A model that takes into account the convergence of the temperature and power distributions, as well as the rates of convergence of both, to determine the optimum batch size would be ideal. Also, determining ways to minimize the amount of data transfer and read and write operations needed between iterations of the two codes is important, as there is significant data overhead at this point in the system's execution.

In addition to increasing the efficiency of the iteration and data transfer of the coupled system, work on increasing the efficiency of the individual codes can offer significant computational cost gains. For OpenMC, the implementation of delta-tracking in such a way as to allow for temperature distributions, as opposed to discrete cells at constant temperature, would decrease the memory requirements of the system geometry and increase the efficiency of the simulation. Also, the ability to utilize an unstructured tally mesh, which can be matched exactly

to the CFD mesh, would allow for a coarse tally mesh that will yield an increase in the accuracy of the transferred data. An unstructured tally mesh would also remove the need for the OpenFOAM Cartesian mesh and the need for performing the field mapping between the Cartesian mesh and the final OpenFOAM mesh. In addition, the ability to transfer the exact distributions of variables, onto the same mesh, will remove much of the current data transfer steps that are needed to accurately map the data.

In terms of increasing the flexibility of the coupled system, the ability to develop a transient solver is desirable. The development of a time stepping scheme that is stable, and does not exponentially increase the runtime, is necessary for any transient solver to be feasible in implementation. Particularly of interest is the method of determining the power distribution, and being able to utilize the computational time spent on previous iterations while accounting for the transient changes as well.

Other potential gains in flexibility that can be made are the implementation of more accurate models for thermal-mechanics that can account for the expansion of fuel and structural elements. The implementation of a delayed neutron precursor drift model would be advantageous as it would allow for modelling of liquid-fueled reactors.

## REFERENCES


Avramova, M.N. et al. "CTF User's Manual," CASL/NCSU (2017).

Bennett, A. et al. "Coupled MCNP6/CTF Code: Development, Testing, and Application," *Annals of Nuclear Energy*, **96** 1-11 (2016).

Cardoni, J.N. et al. "Nuclear Reactor Multi-Physics Simulations with Coupled MCNP5 and STAR-CCM+," University of Illinois at Urbana-Champaign, (2011).

Fiorina, C. et al. "GeN-FOAM: a Novel OpenFOAM Based Multi-Physics Solver for 2D/3D Transient Analysis of Nuclear Reactors," *Nuclear Engineering and Design*, **294**, 24-37 (2015).

Gaston, D. et al. "Physics-Based Multiscale Coupling for Full Core Nuclear Reactor Simulation," *Annals of Nuclear Energy*, **82**, 90-97 (2015).

Godfrey, A.T. "VERA Core Physics Benchmark Progression Problem Specifications," ORNL (2014).

Guo, J.J. et al. "Versatility and Stabilization Improvements of full Core Neutronics/Thermal-Hydraulics Coupling between RMC and CTF," *Nuclear Engineering and Design* **332** 88-98 (2018)

Ivanov, A. et al. "Internal Multi-Scale Multi-Physics Coupled System for High Fidelity Simulations of Light Water Reactors," *Annals of Nuclear Energy*, **66**, 104-112 (2014).

Kochunas, B. et al. "Overview of Development and Design of MPACT," *Proceedings of M&C 2012*, Sun Valley, ID, May 5-9 2013 (2013).

Leppänen J. “On the use of Delta-Tracking and the Collision Flux Estimator in the Serpent 2 Monte Carlo Particle Transport Code,” *Annals of Nuclear Energy,* **105**, 161-167 (2017).

Mylonakis, A.G. et al. “A Newton-based Jacobian-free Approach for neutronics Monte-Carlo/thermal-hydraulic static coupled analysis,” *Annals of Nuclear Energy* **110** 709-725, (2017)

Novak A. et al. “Preliminary Coupling of OpenMC and Nek5000 within the MOOSE Framework,” *Proceedings of PHYSOR 2018,* Cancun, Mexico, April 22-26 (2018).

Richard, J.G. et al. “SMITHERS: An Object-oriented Modular Mapping Methodology for MCNP-based Neutronic - Thermal Hydraulic Multiphysics,” *Annals of Nuclear Energy*, **81** (2015)

Romano, P.K. et al. “OpenMC: A State-of-the-art Monte Carlo Code for Research and Development,” *Annals of Nuclear Energy*, **82**, 90-97 (2015).

Tuominen, R. et al. “Coupling Serpent and OpenFOAM for Neutronics-CFD Multi-Physics Calculations,” *Proceedings of PHYSOR 2016.*

Weller, H.G. et al. “A Tensorial Approach to Computational Continuum Mechanics using Object-Oriented Techniques,” *Computers in Physics,* **12**(6) (1998).

Yeckel, A. et al. “An Approximate Block Newton Method for Coupled Iterations of Nonlinear Solvers: Theory and Conjugate Heat Transfer Applications,” *Journal of Computational Physics*, **228**, 8566-8588.