ORNL/TM-2020/1824

Assessment of the BISON Metallic Fuel Performance Models

Jacob A. Hirschhorn
Jeffrey J. Powers

January 2021

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2020/1824

Nuclear Energy and Fuel Cycle Division

ASSESSMENT OF THE BISON METALLIC FUEL PERFORMANCE MODELS

Jacob A. Hirschhorn
Jeffrey J. Powers

January 2021

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

CONTENTS

EXECUTIVE SUMMARY

The US Department of Energy is leading a project to design and construct a fast spectrum test reactor called the Versatile Test Reactor (VTR). The BISON nuclear fuel performance code will be used to model VTR driver fuel, including looking at the effects of differences between the VTR driver fuel element design and the legacy fuel designs and experiments on which it is based. Simulations will be conducted to help determine whether the design's behavior and performance are properly understood and to assess the margins to cladding failure and fuel melting relative to those predicted for past metallic fuel experiments. These predictions are expected to streamline VTR design and operation by helping inform the VTR driver fuel element design and by providing supplemental information for the fuel design safety basis.

In this work, a critical review of the metallic fuel models available in BISON was conducted to improve the accuracy and reliability of BISON's predictions for VTR applications. Two new approaches for modeling metallic fuel performance were defined by using BISON's existing capabilities, and how these approaches improve the accuracy of BISON's predictions was demonstrated by simulating an irradiation experiment conducted in the Experimental Breeder Reactor-II (EBR-II). The first approach uses existing BISON models to successfully eliminate about one third of the error in the code's axial fuel elongation predictions for the EBR-II fuel element.

The second approach implements a vented porosity closure (i.e., hot pressing) model by using an approximation in the BISON input file to more realistically account for the effects of solid swelling on the volume of low-burnup metallic fuel. This approach eliminates nearly all the remaining error in the code's fuel elongation predictions for the EBR-II fuel element. Despite these improvements, BISON still underpredicts radial cladding dilation, suggesting that cladding creep or fuel-cladding mechanical interaction is not being captured correctly. The success of the second approach and its consistency with experimental observations and theory suggest that revisions to BISON's swelling models are necessary.

Based on these findings, the authors identified several issues that require further investigation and made recommendations for continued BISON use and code development. Studies will be conducted to identify optimal swelling parameters, define best practices for the treatment of fast neutron flux, determine appropriate meshing and solver options, and test various models for cladding wastage and damage. Recommended code developments include modifying swelling models to account for vented porosity closure and revisions that balance functionality between BISON's automatic differentiation (AD) and non-AD models. The metallic fuel modeling approach will continue to be refined with the results of these investigations, and BISON will be monitored for new developments with updates incorporated as they are made available.

FIGURES

TABLES

ACRONYMS

AD	automatic differentiation
AFC	Advanced Fuels Campaign
ANL	Argonne National Laboratory
CDF	cumulate damage fraction
EBR-II	Experimental Breeder Reactor-II
FCCI	fuel-cladding chemical interaction
FCMI	fuel-cladding mechanical interaction
FFTF	Fast Flux Test Facility
FGR	fission gas release
INL	Idaho National Laboratory
IFR	Integral Fast Reactor
LANL	Los Alamos National Laboratory
MOOSE	Multiphysics Object Oriented Simulation Environment
NEAMS	Nuclear Energy Advanced Modeling and Simulation
ORNL	Oak Ridge National Laboratory
PJFNK	Preconditioned Jacobian Free Newton Krylov
SFR	sodium-cooled fast reactor
VTR	Versatile Test Reactor
2D	two-dimensional

ACKNOWLEDGMENTS

This work was sponsored by the US Department of Energy's Office of Nuclear Energy Versatile Test Reactor project. The authors would like to acknowledge the BISON team at Idaho National Laboratory for its continued support.

1. INTRODUCTION

The US Department of Energy is currently engaged in a multiyear project to design and construct a fast-spectrum test reactor called the Versatile Test Reactor (VTR) [1]. Participants in the project include researchers, engineers, and other technical professionals from several national laboratories, universities, and private industries. VTR will be a sodium-cooled fast reactor (SFR) fueled by a U-Pu-Zr metallic fuel alloy. VTR's flexible experimental test capabilities and high fast neutron flux will provide the tools needed to study the viability of next-generation reactor concepts and long-term material degradation due to irradiation.

Modern SFR and U-Pu-Zr driver fuel designs are largely based on those developed during the Integral Fast Reactor (IFR) program of the 1980s and 1990s [2]. Thousands of binary U-Zr and ternary U-Pu-Zr fuel elements were irradiated in Experimental Breeder Reactor-II (EBR-II) during the IFR program [3]. Iterative design improvements yielded robust fuel elements capable of achieving high burnups without failure [4]. These achievements—as well as the wealth of operational experience acquired from EBR-II, the Fast Flux Test Facility (FFTF), and other reactors—are expected to help inform VTR design, construction, and operation.

The increasing availability of advanced modeling tools and computational resources in recent years is expected to further streamline VTR design and operation. Computer codes—such as LIFE-METAL [5], BISON [6], ALFUS [7], and others [8]–[10]—have been developed to quantitatively model and predict metallic fuel performance. Using these codes will enable designers to evaluate the effects of differences between designs, helping ensure that safety and performance criteria can be satisfied. BISON is the most modern and flexible of these tools, and it is under continuous development by the BISON team at Idaho National Laboratory (INL) and researchers from several other national laboratories, universities, and industry partners. As such, BISON will be used to model VTR driver fuel. Other codes will be used to help interpret and support BISON's predictions, as necessary.

A thorough understanding of the accuracy of BISON's predictive capabilities is needed to confidently apply its findings to inform VTR fuel element designs. Assessments for several metallic fuel elements irradiated at EBR-II and the Transient Reactor Test Facility are currently available in the BISON repository [11], and researchers at Oak Ridge National Laboratory (ORNL) are actively developing many more [12], [13]. These assessments are used to evaluate BISON's accuracy and to help identify possible improvements by directly comparing its predictions with the results of the irradiation experiments. Assessments are maintained within the BISON repository so that they can be used to evaluate the effectiveness of code improvements over time. The results of these assessments show that the accuracy of BISON's predictions is generally comparable with those of other metallic fuel performance codes, but there is room for improvement.

Assessments such as these are useful for quantifying the accuracy of a fuel performance code, but the source of inaccuracies often remains ambiguous. Numerical analysis techniques, such as sensitivity analysis, can be applied to help identify the responsible models and parameters [14]. Unfortunately, the complexity and nonlinearity of multiphysics fuel performance modeling often make the results difficult to interpret. Therefore, expert evaluation and judgement are often needed to track down, identify, and resolve these issues. The objective of this work is to perform a critical review of the material-specific properties, constitutive relations, and behavioral models in BISON used to predict metallic fuel performance.

This document begins by briefly examining results obtained from an EBR-II metallic fuel performance assessment conducted in BISON to determine what types of results require improvement. Next, additional background information needed to frame the discussion is reviewed, and the inputs necessary to conduct a metallic fuel performance simulation in BISON are summarized. Then, each model used in the

assessment and any alternatives available in BISON are reviewed, focusing on the potential of each model to contribute to the observed inconsistencies. Finally, an approach is recommended for modeling metallic fuel performance by using BISON's existing capabilities, and priorities for short-term and long-term code development are suggested.

2. OBSERVATIONS FROM THE T654 FUEL PERFORMANCE ASSESSMENT

A team of ORNL researchers recently conducted a BISON assessment of the T654 fuel element irradiated during Experiment X430 in EBR-II [12]. An updated frictionless-contact version of that assessment was selected for further examination in the current work. This fuel element consisted of U-19Pu-10Zr[*] fuel clad with the HT9 (Fe-12Cr-1Mo) steel alloy, and it was irradiated to an average burnup of 10.4 at. %.

The primary goal of applying BISON to simulate metallic fuel performance is to determine whether the proposed fuel element design can be operated safely throughout its lifetime. Reactor safety criteria for this type of fuel element are typically established to prevent cladding rupture and fuel melting during normal and off-normal operation. Maintaining the physical integrity of the cladding ensures that it can contain the fission products produced by reactor operation, limiting the spread of contamination. Preventing fuel melting also helps maintain cladding integrity by limiting chemical interactions between the fuel and cladding and helps avoid unexpected reactivity changes due to fuel relocation.

The dimensional stability of the cladding can also be a significant safety concern. Dimensional changes in the cladding can result from internal fission gas pressure, mechanical interactions with the fuel, irradiation-induced swelling, and thermal and irradiation-induced creep. The nature of these changes depends on the fuel and cladding materials and the operational history of the reactor. Although dimensional instabilities cannot be avoided entirely, they must be anticipated and accounted for in the core design to ensure that they do not disrupt coolant channel flow or promote adverse mechanical interactions between the fuel elements and other core components.

For a given set of materials and core design, the temperatures, stresses, and strains likely to promote fuel melting, cladding rupture, and other adverse mechanical effects can be identified and used to establish reactor safety criteria. Fuel performance models can then be applied to simulate the thermo-mechanical response of the fuel element. These predictions can then be used to evaluate whether the proposed design satisfies the safety criteria.

Simulation results from the T654 assessment are presented in Figure 1 with experimental data from X430 for comparison [15]. These results give some indication of the levels of accuracy and certainty that can be expected from metallic fuel performance simulations. The fission gas release (FGR) and temperature results agree fairly well. Variations in the predicted fuel temperatures are on the same order as the uncertainties expected to arise from power and coolant temperature inputs [16]. Temperature variations could significantly impact calculations, such as neutronics, but they are unlikely to create a serious safety issue because the margins to melting are substantially larger. Still, they are worth investigating because many fuel and cladding properties depend on temperature. The predicted mechanical responses of the fuel and cladding require the most improvement. These observations are kept in mind while discussing BISON's inputs and assessing its models.

[*]All compositions in this work are given in weight-percent unless otherwise specified.

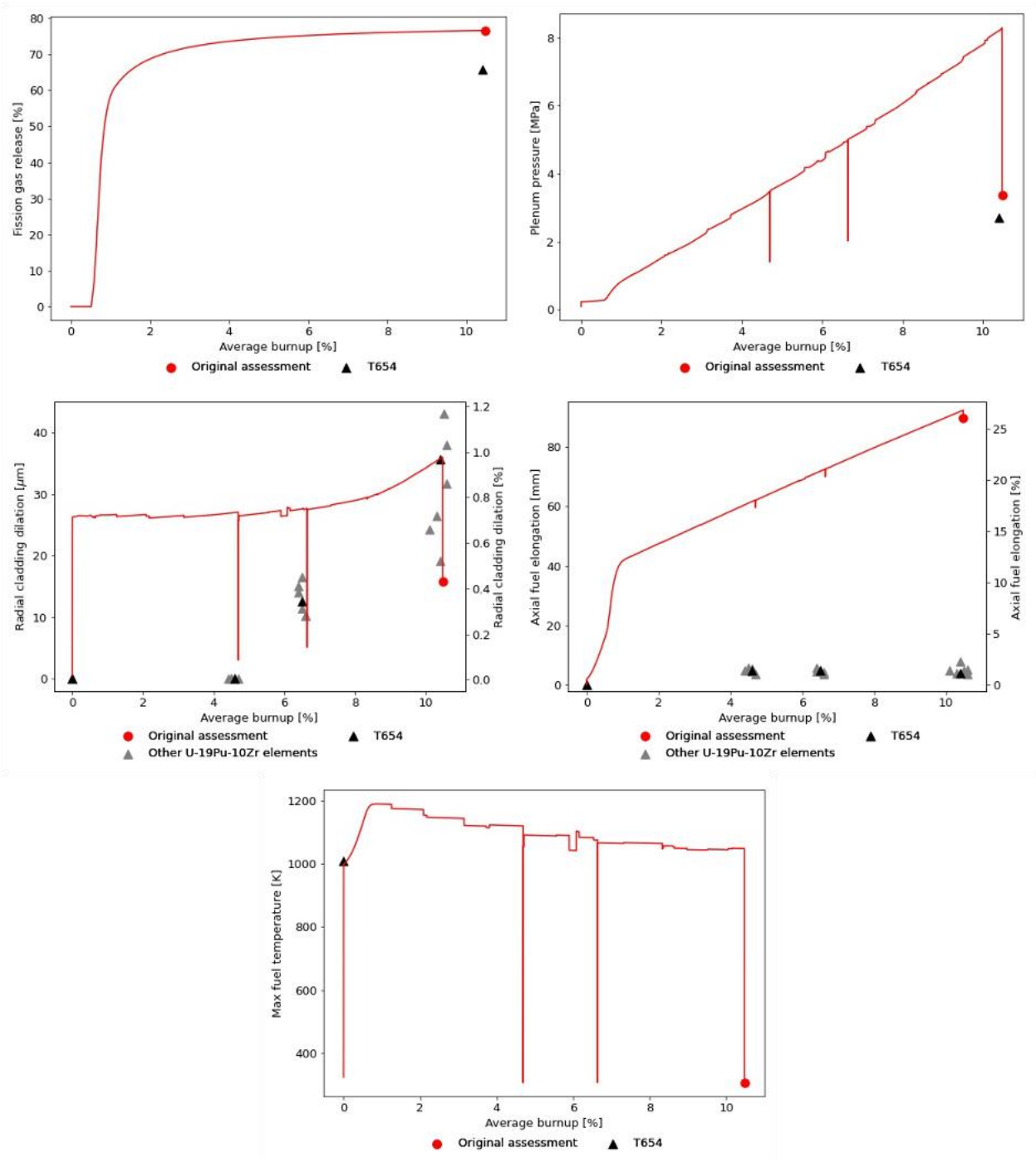

Figure 1. FGR (top left), plenum pressure (top right), peak radial cladding dilation (mid left), axial fuel elongation (mid right), and maximum fuel temperature (bottom) results obtained from the original T654 assessment. Experimental data from T654 and other U-19Pu-10Zr fuel elements irradiated during X430 are provided for comparison [15].

3. ADDITIONAL BACKGROUND

This section reviews additional background information needed to provide context for the model assessment. A broad overview of BISON's thermo-mechanics models is presented first, followed by an introduction to the metallic fuel irradiation behaviors likely to be observed in U-Pu-Zr fuels, such as T654.

3.1 BISON THERMO-MECHANICS

The foundation of any BISON fuel performance simulation is the multibody thermo-mechanics problem created to predict temperatures, stresses, and strains within the fuel-cladding system. A brief introduction to the thermal and mechanical systems available within BISON is provided as follows. For more details, readers can refer to the well-documented Multiphysics Object Oriented Simulation Environment (MOOSE) framework upon which BISON's thermo-mechanical models are primarily based [17], [18]. The first variable in a BISON thermo-mechanics problem is the temperature T, which is solved for by using the heat equation:

$$\rho c_p \frac{\partial T}{\partial t} = -\nabla k \nabla T + q, \tag{1}$$

where ρ is the density, c_p is the isobaric specific heat capacity, t is the time, k is the thermal conductivity, and q is the volumetric heat generation rate. User inputs are needed to define the coordinate system, geometry, initial and boundary conditions, and heat generation rate. Density, specific heat capacity, and thermal conductivity are material-specific properties, some of which are already available within BISON for U-Pu-Zr and HT9.

A general discussion of the tensor mechanics tools available in BISON is beyond the scope of this work. However, a brief review of how mechanical interactions are modeled in BISON helps frame the discussion of the inputs needed to calculate the stress and strain throughout the fuel element. By assuming that the acceleration throughout the system is zero and that there are no additional sources of stress, the authors arrive at a simple form of the stress divergence equation:

$$\nabla \cdot \boldsymbol{\sigma} = 0, \tag{2}$$

where $\boldsymbol{\sigma}$ is the stress tensor. Solving this equation yields a steady-state mechanics solution at each time step. A body force due to gravity, which is aligned with the axis of the fuel element, is also typically included in BISON fuel performance simulations.

The stress is obtained from the strain by using physics- and material-specific constitutive relations of the form:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon} - \boldsymbol{\epsilon_0}), \tag{3}$$

where $\boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon_0}$ are the total strain and stress-free strain tensors, respectively. Many of the constitutive relations needed to model elastic deformation, creep, swelling, and thermal expansion are already available in BISON for U-Pu-Zr and HT9. Others must be constructed manually in the input file by using generic constitutive relations.

Strain components are obtained from displacements d_i (where i denotes the direction), which form a vector that describes how far each material point has moved from its original position. The displacements—not the stresses or strains—are the variables used in BISON. The stress divergence

equation is solved by identifying the displacements that minimize spatial gradients in the stress while satisfying the system's constitutive relations.

3.2 METALLIC FUEL IRRADIATION BEHAVIORS

Metallic fuels exhibit several irradiation behaviors that are responsible for departures from the typical thermo-mechanical response of a nonnuclear system. A thorough examination of the irradiation behaviors exhibited by U-Pu-Zr fuels is beyond the scope of this work. Many detailed discussions have been published on these topics in the open literature [19]–[21]. Only a limited overview is included here to provide context for this review.

A fresh metallic fuel element comprises a cast fuel slug within a cylindrical cladding tube with welded end caps. The narrow radial gap between the fuel and cladding is filled with liquid sodium to promote heat transfer. The fuel slug is about half the height of the cladding tube and rests near the bottom of the fuel element. The remaining interior volume near the top of the fuel element, referred to as the *plenum*, is filled with an inert gas during fabrication.

Solid and gaseous fission products deposit within the fuel lattice, causing it to swell outward toward the cladding [22]. The gaseous fission products diffuse to form bubbles, which coalesce into channels. The channels finally interconnect and release the fission gas into the gap and plenum at about the same time that the fuel swells to contact the cladding. Thereafter, new fission gases are vented to the plenum, limiting further gaseous swelling within the fuel.

The timing associated with contact and FGR is not a coincidence. This behavior is achieved by ensuring that the cross-sectional space within the cladding is sufficient to allow for fuel expansion before large-scale fission gas channel interconnection. The ratio of the fuel cross-sectional area to the inner cladding cross-sectional area is often referred to as the *smear density*. A smear density of ~75% helps to limit fuel-cladding mechanical interaction (FCMI). However, solid fission products continue to promote fuel swelling after FGR, so the risks associated with FCMI increase with burnup.

Low-conductivity fission gases degrade heat transport within the fuel, but the interconnected channels they form provide paths for high-conductivity liquid sodium to infiltrate the fuel from the radial gap between the fuel and cladding. These behaviors substantially impact temperatures throughout the fuel. Studies have been conducted to characterize the average fission gas and sodium contents of irradiated fuels [23] and estimate the magnitude of their effects on fuel temperature [16]. Unfortunately, the temporal and spatial dependencies of these behaviors are still largely unknown.

Contact also enables fuel-cladding chemical interaction (FCCI) [24]. FCCI involves the diffusion of the cladding's constituents into the fuel, which forms low melting temperature phases, decreasing the margin to fuel melting. FCCI also involves the diffusion of lanthanides from the fuel into the cladding, which can react to form brittle phases that increase the likelihood of cladding rupture [25]. FCCI produces a wastage region at the inner surface of the cladding, reducing its effective thickness and degrading its ability to support mechanical loading [26].

Numerous crystalline phases can form in U-Pu-Zr fuels under normal operating conditions. The temperature gradient within the fuel sets up chemical potential gradients between the phases, promoting the interdiffusion of fuel constituents in a process called *constituent redistribution* [27]–[30]. In typical fuel alloys, U and Zr tend to interdiffuse, whereas Pu remains largely immobile. Constituent redistribution can promote macroscale phase change, which can influence the fuel's porosity development and mechanical properties. Unfortunately, the complexity of these processes makes them difficult to model. As such, models for constituent redistribution and phase change are still under development, and most metallic fuel properties are correlated only to bulk constituent composition, temperature, and/or burnup.

Irradiation can also cause the cladding to swell, and the fuel and cladding are susceptible to thermal- and irradiation-induced creep. Creep helps limit stress buildup within the fuel element, which can be beneficial in some cases. However, changes in the cladding dimensions can lead to interference between the fuel element and other core structures, and fuel growth—particularly axial elongation—can introduce undesirable changes in core reactivity. The cladding's mechanical behavior under irradiation, chemical compatibility with the fuel, and neutronic properties are important considerations during alloy selection.

4. INPUTS NEEDED TO SIMULATE METALLIC FUEL PERFORMANCE

The T654 assessment serves as a convenient baseline, which illustrates the inputs needed to set up a metallic fuel performance simulation in BISON. These inputs are grouped into four categories and are discussed in this section within the context of identifying likely sources of inaccuracy in BISON's predictions.

4.1 COORDINATE SYSTEM AND GEOMETRY INPUTS

Typical simulations involve a single metallic fuel element and are conducted in cylindrical coordinates. The fuel element's behavior is normally assumed to not vary in the azimuthal direction, allowing a two-dimensional (2D), axisymmetric R-Z domain to be used. This assumption is thought to be valid during typical operating conditions, and it substantially reduces the computational cost of the simulation. The necessary geometry inputs are summarized in Table 1.

Table 1. Geometry inputs needed to simulate metallic fuel performance in axisymmetric R-Z coordinates.

Component	Input
Fuel	Radius
	Height
Gap	Thickness
Cladding	Outer radius
	Height
	Upper plug thickness
	Lower plug thickness
Coolant channel	Pitch

The exact dimensions of fuel elements can differ due to variations in the fabrication process. Quality control is conducted to verify that fuel elements conform to established tolerances, and those that fail are rejected. Although simulations are often conducted by using only the nominal fuel element geometry, there could be value in using tools such as BISON to quantify how fabrication variations might impact metallic fuel performance and safety. Tolerances to which safety and performance are less sensitive might be able to be relaxed, potentially yielding cost savings and increasing fabrication throughput. The availability of detailed, statistically representative quality control data would maximize the effectiveness of these studies.

The spatial discretization of the problem domain is also an important consideration that could reduce prediction accuracy. The discretization scheme chosen for a fuel performance simulation will depend on the geometry and physics involved, and it will always represent a compromise between accuracy and computational cost. It is difficult to articulate specific requirements for discretization because the amount of error that can be tolerated varies from case to case. Other criteria, such as the element aspect ratio and numerical convergence, can often be used as guides.

4.2 THERMAL INPUTS

A uniform temperature equal to room temperature or reactor ambient temperature is typically applied throughout the problem domain to define the initial condition. One or more startup steps, during which the heat generation rate and coolant inlet temperature can be increased, are often used at the beginning of a simulation to allow the fuel element to reach normal operating conditions. These steps, which are on the order of hours and days, ensure that the effects of the heat-up can be included in the simulation while improving numerical convergence. Using startup steps is unlikely to impact the accuracy of metallic fuel performance simulations that involve normal operating conditions because the behaviors of interest should occur months or years into the simulation. More realistic initial conditions and startup schemes could be necessary for simulations that involve off-normal operating conditions, which might cause the fuel element to evolve much more quickly.

Boundary conditions for the temperature are normally applied to the axisymmetric R-Z domain by using a zero-flux Neumann condition on the fuel element axis and a convective cooling condition on the outer cladding surface. Zero-flux Neumann conditions are often also applied to the top and bottom of the fuel element. The impact of this approximation will likely be negligible because radial heat transfer is much greater than axial heat transfer. The convective cooling condition requires information about the temperature and mass flux of the coolant, which can be difficult to accurately calculate for individual fuel elements due to the sparsity of instrumentation within a reactor core. Finally, the relationship between the temperatures at the fuel surface and cladding inner surface is governed by a thermal contact model, which also accounts for heat transport through the liquid sodium-filled gap.

Heat generation can vary with time due to short-term transients, such as reactor startup and long-term changes in reactivity due to burnup. It can also be a function of space due to the shape of the neutron flux profile, which varies axially and radially within the core. Radial variations in heat generation within a single fuel element are often neglected because a fuel element's length is much greater than its radius. Accurate and detailed operational logs provide a valuable input for heat generation calculations. However, instrumentation within the core might be insufficient to reconstruct a realistic power distribution for fuel performance simulations. Neutronic simulations can be used to supplement sparse heat generation data, but these often require accurate temperatures to produce reliable results. As such, using coupled thermal-neutronic simulations is a promising approach.

Lastly, density, specific heat capacity, and thermal conductivity are material properties that are different for each material and could vary with temperature. Material-specific, temperature-dependent correlations that describe these properties are often drawn from the literature and included in BISON. When necessary, these correlations are developed into more complex behavioral models to account for the irradiation effects. Material-specific properties and behavioral models are discussed in Section 4.4.

4.3 MECHANICAL INPUTS

Initial conditions are not required for the displacements because the stress divergence equation does not contain a time derivative term. Stresses that develop early in fuel performance simulations involving fresh fuel elements and normal operating conditions are relatively small. Deformation during this time results primarily from thermal expansion. During off-normal operation when larger changes in displacement can be expected, the validity of the no-acceleration assumption might require reevaluation.

Zero-value Dirichlet boundary conditions are typically applied to the radial displacements of the fuel and cladding at the fuel element axis and to the axial displacements of the fuel and the cladding at the bottom of each. The fuel and cladding are positioned so that there is a small gap between the bottom of the fuel and the top of the lower cladding plug, which eliminates the need to model contact between these surfaces. No boundary conditions associated with support structures are routinely used. The common practice of omitting external mechanical constraints allows the cladding to expand and contract freely.

This should be sufficient to capture the two mechanical behaviors likely to impact metallic fuel performance during normal operation at low burnup: FCMI and FGR. Displacement boundary conditions might require reevaluation for higher burnup simulations in which excessive swelling and creep could make buckling a concern.

The mechanical relationship between the fuel surface and cladding inner surface is governed by a contact model, which can be applied with or without friction. ORNL researchers recently conducted a study to evaluate the performance of the contact model with and without friction and to assess its potential to impact metallic fuel performance simulation results [13]. Neither option yielded consistent improvements in mechanical predictions, but the inclusion of friction increased computational cost. The implications of these findings are still being investigated.

Finally, differential pressure across the cladding is determined with the help of a plenum pressure model and a coolant pressure boundary condition. Many SFR coolant systems are operated at near-atmospheric pressure. Therefore, most of the pressure exerted onto the cladding exterior by the coolant is due to hydrostatic head and pump head. Internal pressure increases over time due to FGR into the plenum, decrease in plenum volume, increase in temperature, and FCMI. Increases in internal pressure contribute to tensile stresses in the cladding.

4.4 BEHAVIORAL MODELS, MATERIAL-SPECIFIC PROPERTIES, AND CONSTITUTIVE RELATIONS

Separate behavioral models are used to supplement thermo-mechanics models for complex situations, such as multibody thermal and mechanical contact, convective cooling, and irradiation effects. These are supplemented with material-specific properties and constitutive relations to make the generic physics specific to U-Pu-Zr and HT9 and to allow BISON to simulate impactful irradiation behaviors. The distinction between the three categories is sometimes ambiguous. Fortunately, this ambiguity impacts code development and input file generation, not simulation accuracy and performance. For the remainder of this work, these categories are collectively referred to as *models*.

5. MODEL ASSESSMENT

The continuous development of BISON and MOOSE makes it impractical for their developers to regularly release numbered versions of the codes. This can sometimes make it difficult to determine which versions of the codes are being discussed. To minimize confusion, discussions in this work will be based on versions of the codes obtained from the INL repository on September 11, 2020. The hashes associated with the codes' latest Git commits are:

- BISON: 3ea1757dfa94d5b8776c1f91a830466d5dd841a2
- MOOSE: eedf525c82d05db4f883b1bc325a04fbadb0afdd

BISON documentation is publicly available online, but this version is not updated as frequently as the code itself [31]. As of this writing, the online version was last updated on May 18, 2020 (Git hash e0abf174a4862d763a8b8e7aba3e0f931bb708c9). A current version of the BISON documentation, which can be compiled from the code itself, was used for reference in the current work.

First, the models used in the T654 assessment were reviewed. Then, the review was expanded to the other models available in BISON, and their potential for inclusion in future metallic fuel performance simulations was evaluated. Next, different combinations of models were tested to identify those most appropriate for this type of work. Finally, the results of these tests were used to justify recommendations for the use of BISON's existing capabilities and further code development.

BISON thermo-mechanics models and the MOOSE models on which they are based have been successfully applied to a large variety of material systems. Given these successful applications, the authors assumed that the thermo-mechanics physics are correct and focused their attention on the models specific to U-Pu-Zr and HT9. For example, the authors evaluated the technical basis of the thermal conductivity correlations and how porosity models are used to account for irradiation effects, not how the heat conduction equation is implemented and solved.

5.1 MODELS USED IN THE T654 ASSESSMENT

Most models used in the T654 assessment were already available in BISON, but a few were constructed manually in the input file with the help of generic models. The thermal, mechanical, and other models used in the assessment are listed in Table 2 [32]. Some models require additional input parameters not covered in previous sections. These are discussed in the models' critical review. Each model listed in Table 2 is assessed in the following sections.

Table 2. BISON's thermal, mechanical, and other models used in the T654 assessment.

Component	Physics	Model Name	Description
Fuel	Thermal	Density[a]	Calculates the density of the fuel
		ThermalUPuZr	Calculates the specific heat capacity and thermal conductivity of the fuel
	Mechanical	UPuZrElasticityTensor	Calculates the elasticity tensor of the fuel
		ComputeThermalExpansionEigenstrain[a]	Calculates the thermal expansion behavior of the fuel
		UPuZrGaseousEigenstrain	Calculates gaseous swelling and porosity development in the fuel
		BurnupDependentEigenstrain	Calculates solid swelling in the fuel
		UPuZrCreepUpdate	Calculates the creep behavior of the fuel
	Other	UPuZrFissionRate	Calculates the fission rate density in the fuel
		UPuZrBurnup	Calculates the burnup of the fuel
		FgrUPuZr	Calculates the FGR behavior of the fuel
Gap	Thermal	ThermalContact	Governs heat transfer between the fuel and cladding
	Mechanical	Contact	Governs mechanical contact between the fuel and cladding
		PlenumPressure	Calculates the pressure inside the plenum
Cladding	Thermal	Density[a]	Calculates the density of the cladding
		ThermalHT9	Calculates the specific heat capacity and thermal conductivity of the cladding
	Mechanical	ComputeIsotropicElasticityTensor[a]	Calculates the elasticity tensor of the cladding
		ComputeThermalExpansionEigenstrain[a]	Calculates the thermal expansion behavior of the cladding
		HT9VolumetricSwellingEigenstrain	Calculates gaseous swelling in the cladding
		HT9CreepUpdate	Calculates the creep behavior of the cladding
	Other	FastNeutronFlux	Calculates the fast neutron flux in the cladding
		FailureCladHT9	Predicts cladding failure
		MetallicFuelWastage	Calculates the thickness of the wastage region at the inner surface of the cladding
Coolant channel	Thermal	CoolantChannel	Calculates the surface temperature of the cladding
	Mechanical	Pressure	Calculates the coolant pressure applied to the outer surface of the cladding

[a]Generic models used to manually construct material-specific models in the input file.

5.1.1 Density

Two instances of Density were used to calculate the densities of the fuel and cladding by using original user-provided values. Displacements are automatically applied to adjust for effects such as thermal expansion. As such, the user-provided densities should be the densities of the materials at their thermal expansion reference temperatures.

5.1.2 ThermalUPuZr

ThermalUPuZr calculates the specific heat capacity of the fuel as a function of temperature. Two correlations are included: the Savage correlation, which was based on data collected from U-15Pu-10Zr [33], and the Karahan correlation, which was later derived from the Savage correlation [34]. The BISON code documentation points out that the BISON implementation of the Karahan correlation introduces composition dependencies that are not supported by experimental data and therefore increase uncertainties. Savage noted that Pu content does not substantially impact the specific heat capacity of U-Pu-Zr alloys, suggesting that the U-15Pu-10Zr data might be a good approximation for the behavior of other U-Pu-Zr fuels. Therefore, the Savage correlation, which is valid between 25 and 1,150°C, is the most appropriate choice for the current work.

ThermalUPuZr also calculates the thermal conductivity of the fuel as a function of temperature, local constituent composition, and porosity, which is obtained from UPuZrGaseousEigenstrain. Four correlations are included: the Billone et al. correlation, for which the reference could not be located[*]; the Kim et al. correlation [35]; the Galloway et al. correlation [30]; and the Los Alamos National Laboratory (LANL) correlation, which has not yet been published in the open literature. Each correlation yields the thermal conductivity of the unirradiated fuel to which a correction is applied to account for the insulating effects of porosity. Only the Billone et al. correlation provides a method to account for sodium infiltration. The BISON documentation states that the LANL correlation provides the best fit to the available experimental data, and thus it is the most appropriate choice for the current work. Unfortunately, the BISON documentation does not specify the ranges of temperature and composition for which it is applicable.

5.1.3 UPuZrElasticityTensor

UPuZrElasticityTensor calculates the elasticity tensor of the fuel as a function of temperature, local constituent composition, and porosity, which is obtained from UPuZrGaseousEigenstrain. The model uses correlations from the *IFR Metallic Fuels Handbook*, which was first compiled in 1989 and released as a Argonne National Laboratory (ANL) report in 2019, to calculate the elastic modulus and Poisson's ratio of the fuel [36]. The uncertainties associated with these correlations are particularly high because, as the BISON documentation states, the data from which they were derived are extremely limited. The correlations were derived by using data collected from pure U, to which mixture rules, ceramic porosity corrections, and simplified models for phase change were applied. The handbook does not specify the temperature range for which the correlation is valid. Improved descriptions of the fuel's elastic properties are expected to substantially affect the code's mechanical predictions.

[*]Listed in the BISON documentation as: M. C. Billone, Y. Y. Liu, E. E. Gruber, T. H. Hughes, and J. M. Kramer, "Status of Fuel Element Modeling Codes for Metallic Fuels," in *Proceedings American Nuclear Society International Conference on Reliable Fuels for Liquid Metal Reactors*, Tucson, Arizona, September 7–11, 1968.

5.1.4 ComputeThermalExpansionEigenstrain

Two instances of ComputeThermalExpansionEigenstrain were used to calculate the eigenstrains in the fuel and cladding due to thermal expansion by using user-provided constant thermal expansion coefficients and reference temperatures.

5.1.5 UPuZrGaseousEigenstrain

UPuZrGaseousEigenstrain calculates the fuel porosity and the associated eigenstrain due to the accumulation of gaseous fission products. It is based on a semi-empirical model derived by Olander, which was published in 1976 [37]. The simplified model assumes that all new fission gas atoms are created within preexisting, evenly spaced, spherical bubbles that then grow and interconnect. The model depends on temperature and fission rate density, which is obtained from UPuZrFissionRate, but it does not account for the diffusion of gas atoms, resolution, or stresses within the material.

The porosities at which interconnection begins and ends are predefined but can be overridden in the input file. Users must provide a bubble number density, which can vary spatially and with time, and an anisotropy factor, which dictates the ratio of radial swelling to axial swelling. Unfortunately, appropriate values for these parameters are still being determined.

Other metallic fuel performance codes, such as ALFUS, have demonstrated favorable results by using gaseous swelling models that have fewer assumptions and simplifications [7]. Gaseous fission products and porosity development will likely substantially impact the thermal and mechanical response of metallic fuel elements. Several modeling approaches involving viscoplastic swelling are being developed for use in BISON. These models are reviewed in Section 5.2.5.

5.1.6 BurnupDependentEigenstrain

BurnupDependentEigenstrain calculates the eigenstrain in the fuel due to solid fission product swelling. It uses burnup, which is obtained from UPuZrBurnup, and a constant swelling factor with a default value of 1.5. This value produces a linearly increasing eigenstrain of 1.5% per atom-percent burnup. The default value is based on a fission yield analysis of U-Pu-10Zr fuel (the exact Pu content was not specified), which was conducted to support ALFUS development [7]. Other studies suggest that swelling due to solid fission products could range from 1 to >1.5% per atom-percent burnup [23]. Uncertainties in the swelling factor might need to be investigated before conducting metallic fuel performance simulations involving burnups greater than 10 at. % because the accumulation of solid fission products increases the risks associated with FCMI at high burnup.

5.1.7 UPuZrCreepUpdate

UPuZrCreepUpdate calculates fuel deformation due to irradiation-induced creep and steady-state thermal creep by using correlations from the *IFR Metallic Fuels Handbook*. The creep calculation uses the porosity obtained from UPuZrGaseousEigenstrain, fission rate density calculated by UPuZrFissionRate, and temperature. The ranges of temperature and stress for which the correlations are valid are not explicitly stated in the *IFR Metallic Fuels Handbook*. However, the correlations are expected to be applicable to typical U-Pu-Zr fuels because they were derived by using data from alloys that contained the same phases as should be stable in those fuels during normal operation. As with UPuZrElasticityTensor, the uncertainties associated with these correlations are high due to the sparsity of the data on which they were based. The uncertainties in these models are expected to place an upper limit on how accurately BISON can predict the deformation of metallic fuels.

5.1.8 UPuZrFissionRate

UPuZrFissionRate calculates the local fission rate density in the fuel from the local constituent composition, average linear heat rate, and axial power profile. The calculation uses a constant energy per fission parameter with a default value of 3.28×10^{-11} J/fission (about 205 MeV/fission), which can be overridden in the input file. When desired, an empirical correlation can also be applied to account for the effects of radial U and Zr redistribution [30]. Fission energy yields vary by isotope and with incident neutron energy. The energy per fission value should be chosen to account for fuel composition, operational conditions, and the effects of breeding.

5.1.9 UPuZrBurnup

UPuZrBurnup calculates the local burnup of the fuel in fissions per initial heavy-metal atom from its initial density, its initial constituent composition, and the fission rate density calculated by UPuZrFissionRate.

5.1.10 FgrUPuZr

FgrUPuZr applies a simple model based on broad experimental observations to simulate FGR from the fuel by using the fission rate density calculated by UPuZrFissionRate and porosity obtained from UPuZrGaseousEigenstrain. All fission gas is assumed to be retained within the fuel until a critical porosity is reached, after which a user-specified fraction of the accumulated and newly produced fission gas is released to the plenum. This behavior is based on the theory that large-scale porosity interconnection leads to FGR, which generally marks the end of gaseous swelling. Several parameters can be overridden in the input file to finetune the FGR behavior.

The critical porosity should be set between the porosities at which interconnection begins and ends; these parameters are set in UPuZrGaseousEigenstrain. This requirement is necessary for ensuring that the two models behave consistently, but it is not enforced by BISON, partly because the two models are not coupled. FGR has no direct effect on fission gas bubble behavior in UPuZrGaseousEigenstrain because it does not impact the bubble number density and because the number of gas atoms in each bubble is not permitted to decrease.

5.1.11 ThermalContact

ThermalContact is an action used to model heat transfer from the fuel to the cladding across the unmeshed gap. Actions are objects that can be used to automatically set up variables, models, auxiliary systems, and more, simplifying the user experience. The BISON documentation for this system is currently limited. ThermalContact can be applied in the GapHeatTransfer and GapConductance modes, and the former appears to be the most appropriate choice for the current work. Users must provide a constant or temperature-dependent conductivity for liquid sodium and should specify cylindrical gap geometry.

A minimum gap thickness can also be specified to limit the effect of temperature discontinuities on the code's performance. Currently, the minimum gap thickness is often set to the original gap thickness. This approximation improves convergence but is not expected to significantly impact temperature predictions because the high conductivity of liquid sodium limits temperature variations across the gap, even when the gap is relatively large. As discussed in Section 6, tests were conducted to identify the optimal ThermalContact settings for use in the current work. These tests also evaluated whether a temperature-dependent sodium thermal conductivity correlation should be used to capture the effects of axial temperature variations on gap conduction.

5.1.12 Contact

Contact is an action that interfaces with MOOSE's contact module to model mechanical contact between the fuel and cladding. Generally, contact problems can be very difficult to solve, and the approaches available for their solution are complex. As such, Contact has a wide variety of configurable options and parameters. The optimal settings for metallic fuel performance simulations are not known with certainty at this time, but it might be possible to use the material properties of the fuel and cladding to inform these selections. For example, frictionless contact might be sufficient to model contact between a weak, easily deformed fuel and a much stronger cladding. On the other hand, frictional contact could more accurately capture bonding between the fuel and cladding due to FCCI. Tests were conducted in Section 6 to begin to identify optimal Contact settings, and different options will continue to be explored in future work.

5.1.13 PlenumPressure

PlenumPressure is an action used to calculate the pressure within the plenum and apply appropriate boundary conditions to the inner surface of the cladding. It uses the ideal gas law to calculate the initial number of moles of gas inside the plenum given its initial pressure and temperature. Released fission gas, which is calculated by FgrUPuZr, can be supplied to PlenumPressure through its material input parameter to model its effect on the internal pressure of the fuel element. Temperature can also be coupled into PlenumPressure by supplying an average internal fuel element temperature, which is calculated by a postprocessor or action by using one of several averaging techniques.

5.1.14 ThermalHT9

ThermalHT9 calculates the specific heat capacity and thermal conductivity of the cladding as functions of temperature. The BISON documentation states that the specific heat capacity is calculated by using a linear correlation derived from the data presented in a 1992 publication [38], which in turn references a publication from 1976. The latter correlation could not be located[*]. The thermal conductivity is calculated by using a correlation from the *IFR Metallic Fuels Handbook*. The correlation was derived from data collected between 127 and 927°C, which should make it sufficient for the current work. Within that range, the thermal conductivity of HT9 varies by less than 10%.

5.1.15 ComputeIsotropicElasticityTensor

ComputeIsotropicElasticityTensor calculates the elasticity tensor of the cladding by using two user-provided constant elastic properties. Users typically specify the elastic modulus along with the shear modulus or Poisson's Ratio. This model cannot account for changes in the cladding's elastic properties with temperature, which could be significant. Temperature-dependent mechanical properties available in BISON will be reviewed and evaluated for use in Section 5.2.

5.1.16 HT9VolumetricSwellingEigenstrain

HT9VolumetricSwellingEigenstrain calculates the eigenstrain in the cladding due to irradiation-induced swelling by using a correlation from the *IFR Metallic Fuels Handbook*. The correlation is valid for temperatures between 380 to 700°C, which should be sufficient for the current work. HT9VolumetricSwellingEigenstrain uses the fast neutron flux and fluence calculated by FastNeutronFlux. A typo in the BISON documentation was identified by comparing it with the *IFR Metallic Fuels Handbook* and the BISON source code. Specifically, Eq. (5) of the HT9VolumetricSwellingEigenstrain documentation contains a minus sign where a plus sign should be. The authors recommend that this typo be corrected in the next update.

[*]The reference for the correlation is listed as: Y. Sanokawa and T. Hiraoka, in *Genshiryiku Hand Book*, ed K. OHM, Tokyo, 1976, p 853.

The *IFR Metallic Fuels Handbook* states that, "HT9 may never show significant swelling, regardless of fluence," and that the correlation might be just as accurate as not modeling the swelling at all. The BISON documentation also claims that the correlation "is expected to over-predict the swelling" for fast fluence values greater than 2×10^{22} neutrons/cm^2, with fast neutrons defined as those having energies >0.1 MeV. Based on these discussions, the model's predictions should be regarded as the upper limit of cladding deformation due to irradiation-induced swelling.

The results in Figure 1 show that BISON underpredicted radial cladding dilation. Cladding deformation is influenced by FGR, cladding swelling and creep, and fuel swelling and creep through FCMI. Unfortunately, it is difficult to determine the source of the inconsistencies from these results. Using code comparisons or additional assessments of irradiation experiments might provide additional insights into what mechanisms and models are responsible for these discrepancies.

5.1.17 HT9CreepUpdate

HT9CreepUpdate calculates cladding deformation due to irradiation-induced creep and steady-state thermal creep by using correlations from the *IFR Metallic Fuels Handbook*. The correlations are valid for temperatures between 350 and 750°C and stresses between 0 and 250 MPa, which should be sufficient for the current work. HT9CreepUpdate uses the fast neutron flux calculated by FastNeutronFlux, with fast neutrons defined as those having energies >0.1 MeV.

5.1.18 FastNeutronFlux

FastNeutronFlux calculates the fast neutron flux in neutrons per square meter per second, which is used to calculate the fluence in neutrons per square meter. The model provides options that allow the neutron flux to be specified directly or calculated from the linear heat rate. The resulting flux can be constant or a function of time and/or space. When using the linear heat rate, users must specify a factor that relates the flux to the linear heat rate in Watts per meter. The BISON documentation does not provide any information regarding how the factor should be calculated. When calculating the factor, only neutrons with energies >0.1 MeV should be included to ensure that the resulting flux is compatible with the correlations used in HT9VolumetricSwellingEigenstrain and HT9CreepUpdate. Variations between how the fast neutron flux is calculated could significantly affect irradiation-dependent behaviors, such creep and swelling.

5.1.19 FailureCladHT9

FailureCladHT9 uses correlations compiled from several sources to predict failure in the cladding. Long-term failures due to burnup are modeled by using a cumulate damage fraction (CDF), which should be appropriate for normal operating conditions. Options used to predict short-term failures due to transients, such as reactivity insertion accidents, are also available. These options incorporate correlations for the CDF and constrained cavity growth and could be applied to model off-normal operations. Both options rely on the temperature and hoop stress within the cladding. The basis of these correlations and the validity of their implementation were not evaluated at this time, but future work should include this scope.

5.1.20 MetallicFuelWastage

MetallicFuelWastage calculates the thickness of the wastage region that forms at the inner surface of the cladding due to FCCI. It includes several correlations, some of which were developed by researchers at ANL and/or are used in LIFE-METAL. Some correlations were calibrated by using EBR-II data from the EBR-II Fuels Irradiation and Physics Database [39]. The correlations predict wastage region growth as a function of temperature, the fast neutron flux calculated by FastNeutronFlux, and/or the burnup calculated

by UPuZrBurnup. Only one model can account for gap closure. Doing so requires the cladding penetration depth, which is obtained from Contact.

The basis of these correlations and the validity of their implementation were not evaluated because the availability of information regarding their creation and calibration is very limited. No original references for the correlations are listed in the BISON documentation. The authors recommend testing to evaluate the models' predictions against data from irradiated fuel elements to identify the most appropriate model. The wastage region thickness and the effects of its formation on cladding stress will likely be important factors when establishing metallic fuel element safety limits. Further work must assess ongoing BISON developments targeted at modeling the effects of wastage on cladding stress.

5.1.21 CoolantChannel

CoolantChannel is an action that sets up a convective cooling condition on the outer surface of the cladding. It contains the tools needed to model liquid sodium within a triangular coolant channel given its inlet temperature, pressure, mass flux, and geometry. The correlations that describe the properties of the liquid sodium are defined within MOOSE's FluidProperties module [40]. CoolantChannel's heated perimeter and hydraulic diameter calculations make it most appropriate for simulations involving fuel elements at the interior of the fuel bundle. Modifications might be necessary for fuel elements on the periphery of the bundle due to variations in effective flow area around those pins. Otherwise, CoolantChannel and its associated models are believed to be appropriate for the current work.

5.1.22 Pressure

The Pressure boundary condition specifies the coolant pressure applied to the outer surface of the cladding. An input is normally supplied by using a constant or a time-dependent function or postprocessor. This model has been successfully applied for numerous other material systems and is expected to be sufficient for typical metallic fuel performance applications.

5.2 OTHER MODELS AVAILABLE IN BISON

This section extends the assessment to evaluate other models available in BISON. The authors' definition of *models* is expanded to include solution approaches, solver options, more general actions, and more. Some of these models can be used to supplement those covered in the previous sections. Others could replace those covered in the previous sections entirely. Tests were conducted in Section 6 to identify which combinations of parameters and models yield the best results.

5.2.1 Automatic Differentiation

The Newton method is one of the most common and straightforward approaches available for solving systems of nonlinear differential equations. Unfortunately, the efficient application of the Newton method requires a full and accurate Jacobian, which can be difficult and time-consuming to calculate and code, particularly for multiphysics systems with complex material properties and constitutive relations. Even minor inaccuracies in a model's Jacobian contributions can degrade numerical convergence or prevent the problem from converging entirely. These issues are partly why most BISON fuel performance problems are solved via Preconditioned Jacobian Free Newton Krylov (PJFNK) methods, which do not require an explicit Jacobian.

It is difficult to say which of the two methods—Newton or PJFNK—is best for metallic fuel performance simulations because their effectiveness can vary with the number of equations, the coupling between them, the mesh size, and other factors. Metallic fuel performance problems seem amenable to solution using the Newton method, but the full and accurate Jacobian would be needed. Fortunately, the automatic differentiation (AD) system recently added to MOOSE can be used to automatically form the Jacobian

symbolically [41]. AD use requires additional overhead, but it might improve convergence and reduce overall computational cost.

AD functionality uses different data types than those used in traditional non-AD models. As such, it has become a common practice to retain two versions of a model in BISON. For example, users can select from a non-AD model, such as UPuZrBurnup, and its AD counterpart, ADUPuZrBurnup. Almost all AD models follow this naming convention. Most AD and non-AD models provide the same functionality, differing only in how the derivatives are treated. However, some models are available in only one version or are offered in AD and non-AD versions with different capabilities. Coupling between AD and non-AD models is usually prohibited due to their use of different data types. As described in Section 6, tests were conducted to determine whether AD can be successfully applied to solve this type of problem by using the Newton method and whether this approach yields any computational advantages over using non-AD models and PJFNK.

5.2.2 ADUPuZrThermal and ADUPuZrSodiumLogging

ADUPuZrThermal began as the AD version of ThermalUPuZr but subsequently diverged slightly from its non-AD counterpart. In addition to calculating specific heat capacity and thermal conductivity, ADUPuZrThermal can be used to apply a porosity correction that includes the effects of both fission gas-filled and liquid sodium-filled porosity [23]. These are calculated from the overall porosity and porosity interconnectivity by using ADUPuZrSodiumLogging. Currently, the authors do not have access to any results that demonstrate improved accuracy via these models, but the inclusion of sodium infiltration in fuel performance simulations is consistent with experimental observations and seems to be a step in the right direction.

5.2.3 Other U-Pu-Zr Swelling Models

Several other swelling models are available for the fuel in BISON, including UPuZrGaseousSwelling, UPuZrPorosityEigenstrain, UPuZrLowTemperatureSwelling, UPuZrAnisotropicSwellingEigenstrain, UPuZrVolumetricSwellingEigenstrain, UPuZrVolumetricSwellingEigenstrainLM, and their AD counterparts. However, the authors believe that UPuZrGaseousEigenstrain and BurnupDependentEigenstrain are most appropriate for the current work because they conveniently separate the effects of solid and gaseous swelling, are used in the accepted metallic fuel assessment cases, and appear to combine the best of what the other models have to offer. AD versions of the preferred models are also available.

5.2.4 Other FGR Models

There are several other FGR models available in BISON, including FgrUPuZrLM, its AD counterpart, and ADUPuZrFissionGasRelease. FgrUPuZrLM and ADFgrUPuZrLM implement the FGR correlation used in LIFE-METAL. The authors recommend using FgrUPuZr at this time because it is used in the accepted metallic fuel assessment cases. ADUPuZrFissionGasRelease is essentially the AD equivalent of FgrUPuZr, except it does not allow users to specify the final amount of fission gas released.

5.2.5 Viscoplasticity-Based Gaseous Swelling and FGR Models

In addition to the eigenstrain-based gaseous swelling and FGR models described in previous sections, a newer and fundamentally different approach is implemented in ADSimpleFissionGasViscoplasticityStressUpdate and ADCoupledFissionGasViscoplasticityStressUpdate. These models use viscoplasticity methods to calculate the inelastic strain due to gaseous swelling, which is then coupled to other inelastic strain contributions, such as creep, to simultaneously model porosity development, interconnection, and FGR. The models assume that the concentration of fission gas bubbles remains constant with time, but it is

permitted to vary with space. Unlike earlier approaches, these models differentiate between fission gas atoms that are still dissolved in the fuel matrix, those that have reached bubbles or pores, and those that have been released from the fuel.

Fission gas production, porosity interconnection, and FGR are all modeled within a common parent class, ADFissionGasViscoplasticityStressUpdateBase. ADSimpleFissionGasViscoplasticityStressUpdate and ADCoupledFissionGasViscoplasticityStressUpdate are then used to model gas atom diffusion, absorption, and the volumetric response of the fuel by using various methods. The former, like ADUPuZrGaseousEigenstrain, is based on the semi-empirical model derived by Olander, which assumes that gas atoms are created within bubbles and that the material is not under stress [37]. Both models are more sophisticated than earlier methods in that they couple gaseous swelling, porosity development and interconnection, and FGR.

On the other hand, ADCoupledFissionGasViscoplasticityStressUpdate adds another substantial layer of sophistication by modeling the behavior of dissolved gas atoms and the force balance at the bubble surface [42]. Despite these advances, the models are very new and have not been widely used in realistic fuel performance simulations. They currently do not have non-AD counterparts, and the impact of their use on computational cost and robustness is currently unknown. For these reasons, these models were not included in the metallic fuel assessments at this time. The models should be reevaluated at a later date as BISON development continues.

5.2.6 UPuZrThermalExpansionEigenstrain

UPuZrThermalExpansionEigenstrain calculates the eigenstrain in the fuel due to thermal expansion by using linear correlations that span three different temperature zones. The boundaries between the three zones lie at 595 and 665°C, and the slopes of the correlations differ significantly. These features suggest that a single, constant thermal expansion coefficient, such as the one used with ComputeThermalExpansionEigenstrain, might be insufficient to capture the fuel behavior. The correlations are valid between 0 and 940°C, which should make them sufficient for the current work [43]. A typo in the BISON documentation was identified by comparing the original reference and the BISON source code. Specifically, the second term in the second equation of the UPuZrThermalExpansionEigenstrain documentation should be 1.003×10^{-2} instead of 1.0003×10^{-2}. The authors recommend that this typo be corrected in the next update.

5.2.7 Constituent Redistribution Models

Two separate systems are available for modeling constituent redistribution in BISON, each of which is implemented by using multiple thermodynamic and kinetic models and a transport equation. The two systems are centered around PhaseUPuZr and ADUPuZrPhaseLookup. Unfortunately, both models make approximations or suffer from limitations that make them inappropriate for use in metallic fuel assessments at this time. PhaseUPuZr does not include all the phases expected to be stable in U-Pu-Zr fuels, and ADUPuZrPhaseLookup has only been shown to produce accurate results for U-Zr fuels.

Even without modeling constituent redistribution, ADUPuZrPhaseLookup could be used to visualize how the fuel phase composition varies spatially and over time in response to temperature. These calculations would not contribute much to computational cost and could provide valuable insights for postirradiation examinations of the fuel and further model development. ADUPuZrPhaseLookup does not have a non-AD counterpart. The authors recommend this approach if AD models are selected for use in the current work and future metallic fuel assessments. They also recommend the periodic reevaluation of the constituent redistribution models as BISON development continues.

5.2.8 NuclearMaterial Actions

NuclearMaterialUPuZr and NuclearMaterialHT9 are actions used to simplify the input syntax by setting up models for U-Pu-Zr and HT9, respectively. The authors do not recommend their use at this time because the actions have a rigid structure and because it is still uncertain what models are best for this application. The authors might recommend revising the actions at a later time once the optimal models and parameters have been identified.

5.2.9 PlenumTemperature

PlenumTemperature is an action that can be used to estimate the temperature of the plenum from the volume-weighted temperatures at the surface of the fuel and inner surface of the cladding. Tests were conducted in Section 6 to evaluate its suitability for use in the current work and future metallic fuel assessments.

5.2.10 HT9ElasticityTensor

HT9ElasticityTensor and its AD counterpart calculate the elasticity tensor of the cladding by using temperature-dependent correlations for its elastic and shear moduli [44]. The authors recommend using these models in the current work because they are believed to better represent the cladding's mechanical properties and their temperature dependencies.

5.2.11 HT9ThermalExpansionEigenstrain

HT9ThermalExpansionEigenstrain and its AD counterpart calculate the eigenstrain in the cladding due to thermal expansion by using a nonlinear correlation, which is valid up to 777°C [45]. The authors recommend its use over ComputeThermalExpansionEigenstrain for metallic fuel assessments.

5.2.12 UPuZrFastNeutronFlux

UPuZrFastNeutronFlux and its AD counterpart calculate the fast neutron flux from the isotopic composition of the fuel, the isotopes' fission cross sections, the EBR-II flux spectrum, and the fission rate density provided by UPuZrFissionRate or ADUPuZrFissionRate. Users can specify ^{235}U enrichment and ^{240}Pu content, but no options are available to distinguish between ^{239}Pu and ^{241}Pu, which might vary significantly between weapons and reactor-grade Pu. The calculated fast neutron flux is adjusted to include only neutrons with energies >0.1 MeV for compatibility with HT9VolumetricSwellingEigenstrain and HT9CreepUpdate. Fast neutron flux values from databases or neutronics simulations would be ideal, but these are not always available. As described in Section 6, limited tests were conducted to evaluate the best approach at this time, and these issues will continue to be investigated in the future.

5.2.13 Other FCCI Models

There are several other FCCI models available in BISON, including EutecticThicknessFCCI, DiffusionalEutecticThicknessFCCI, and ThicknessLayerFCCI. Each model is based on a different approach and draws on correlations from several sources. The best model is unknown at this time, and further modifications to account for the effects of FCCI on cladding stress are in progress. The authors recommend further testing of the existing models and evaluating new models as they are made available.

5.2.14 Automatic Scaling

Fuel performance simulations are inherently multiphysics problems, and the magnitudes of the variables often differ drastically (i.e., the variables have different scales). Poor scaling can make it difficult for the system to invert the Jacobian or select an appropriate differencing parameter when applying the Newton

and PJFNK methods, respectively. This can introduce truncation errors and degrade convergence. It can also make it difficult to determine when satisfactory convergence has been reached because the convergence criteria are compared with the L2 norm of the variable's residuals, not with the residual of each variable individually.

ReferenceResidualProblem is often used to compensate for poor scaling in BISON simulations. This approach involves applying the convergence criteria to each variable's residual separately, which prevents variables with larger scales from disproportionally influencing the result. The recent addition of Automatic Scaling to MOOSE offers another approach to address this issue. Automatic Scaling can be applied at each time step to automatically scale the variables individually or in groups based on their Jacobian contributions, residuals, or a combination thereof. This approach has a simple input syntax, and its flexibility might offer additional advantages over ReferenceResidualProblem. As described in Section 6, tests were conducted to determine whether this approach should be applied to metallic fuel assessments.

5.2.15 MeshGenerator System

BISON and MOOSE are slowly transitioning away from simple mesh objects in favor of the MeshGenerator system, which offers a more modular set of tools that can be used to generate multiple blocks, define connections between them, create and name node sets, and more. The authors recommend transitioning to SmearedPelletMeshGenerator for metallic fuel performance simulations to minimize the number of input files that must be converted when traditional mesh objects are eventually deprecated.

6. MODEL EVALUATION AND SELECTION

Now that the models used in the T654 assessment have been examined and other potentially applicable BISON models have been identified, they can be evaluated for use. Although the assessment yielded several concrete recommendations, the efficacy of many models could not be determined without testing. Some models and options must be changed in groups because many of BISON's systems are numerically coupled or otherwise interrelated. For example, simulations must be conducted by using AD or non-AD models due to their use of different data types. This, combined with the large number of models to be evaluated, made testing each combination of models impractical at this time.

Instead, the authors leveraged past modeling experience to form larger groups of compatible models and settings to be tested together. Although not exhaustive, the authors believe this type of testing effectively identified the most promising modeling approach within the constraints imposed by VTR's metallic fuel benchmarking timeline. The following criteria were used to guide the evaluation of different approaches. These criteria were chosen to account for the structural and behavioral complexity of the fuel element materials and accommodate sensitivity analysis and uncertainty quantification, which could be applied to inform metallic fuel element design.

- Temperature-dependent, material-specific models that capture all of the materials' irradiation behaviors should be applied whenever possible.
- Models that deliver accurate results while being computationally robust (i.e., able to converge when using perturbed parameters) and minimizing computational expense should be used whenever possible.
- Models and techniques that minimize the complexity of the input file should be employed to improve usability, facilitate efficient benchmarking, and enhance quality assurance.

6.1 SIMULATION MODIFICATIONS

Iterative testing was conducted to construct two promising test cases from the original assessment by using non-AD and AD models selected according to the aforementioned criteria. Compared with the original assessment, significant changes in these cases include using SmearedPelletMeshGenerator, a temperature-dependent sodium thermal conductivity in the gap, the PlenumTemperature action, a material time step limit based on fuel creep, and all the temperature- and/or porosity-dependent U-Pu-Zr and HT9 properties available in BISON.

Another significant modification that involved the critical and interconnection porosities of the fuel was made to correct a discrepancy found in the swelling calculation. In the original assessment, the correlations were applied so that a porosity of ~30% would be required to initiate interconnection and FGR. This value corresponds to a gaseous swelling, $\Delta V/V_0 = p/(1-p)$, of ~43%, where V_0 and ΔV are the initial and change in fuel volume, respectively, and p is the porosity. However, an average maximum fuel swelling of ~34% (corresponding to a porosity of ~25%) was calculated from all the fuel elongation and cladding dilation measurements performed on ternary fuel elements irradiated during X430 [15].

Interestingly, the maximum porosity value estimated from the experiment (25%) is the same as the default value used to terminate porosity interconnection in UPuZrGaseousEigenstrain. The authors believe that the value used in the original assessment, which was obtained from an FGR calibration study conducted for U-19Pu-10Zr fuels at INL, might have underestimated the impact on fuel swelling [46]. Therefore, the critical and interconnection porosities were returned to their default values.

The maximum measured fuel elongation for T654 was ~1.4% [15]. The maximum measured fuel elongations from other U-19Pu-10Zr, U-22Pu-10Zr, and U-26Pu-10Zr fuels irradiated during X430 were ~2.3, ~2.2, and ~3.2%, respectively [15]. These observations suggest that these fuels swelled mostly in the radial direction. A swelling anisotropy factor of 0.99 was used in an attempt to capture this behavior for T654. This value is not physically realistic and is not expected to be broadly applicable to other fuel compositions or operational conditions. It was selected for use in the current work because it allows a limiting case that maximizes potential cladding dilation and minimizes potential fuel elongation to be examined. More realistic anisotropy factor values range from ~0.34 to ~0.50 [3]. Optimizing the anisotropy factor may require data and assessments from more than one experiment. This factor must be revisited as model development and benchmarking studies continue.

The FGR model should contain enough tunable parameters to reproduce the experimental observations while using porosities that correspond to physically realistic swelling values. The fission gas bubble density in UPuZrGaseousEigenstrain can be adjusted to expedite or delay FGR, and the fractional release parameters in FgrUPuZr can be modified to control the total amount of FGR. The authors recommend restricting calibration to these parameters and conducting regular recalibrations as existing assessment cases are refined and new ones are added.

BISON's models are intended to be applied to a wide range of metallic fuels. However, the phase composition of binary and ternary metallic fuels varies significantly. These structural differences impact fuel irradiation behavior, so care must be exercised when calibrating models using data from different fuels with different compositions. Furthermore, operating and boundary conditions vary between the experiments used as a basis for assessment cases, which could drive additional differences. Lastly, the irradiation behaviors of these fuels are tightly coupled. As such, no model should be calibrated without examining the effects of the calibration on other models' predictions.

6.2 SIMULATION RESULTS AND DISCUSSION

FGR and plenum pressure simulation results from the original assessment and the two new cases are presented in Figure 2 with experimental data from T654 for comparison [15]. The results show that the

original and non-AD cases predict these behaviors accurately. On the other hand, the AD case predicts too much FGR because ADUPuZrFissionGasRelease does not allow users to specify the final amount of fission gas released.

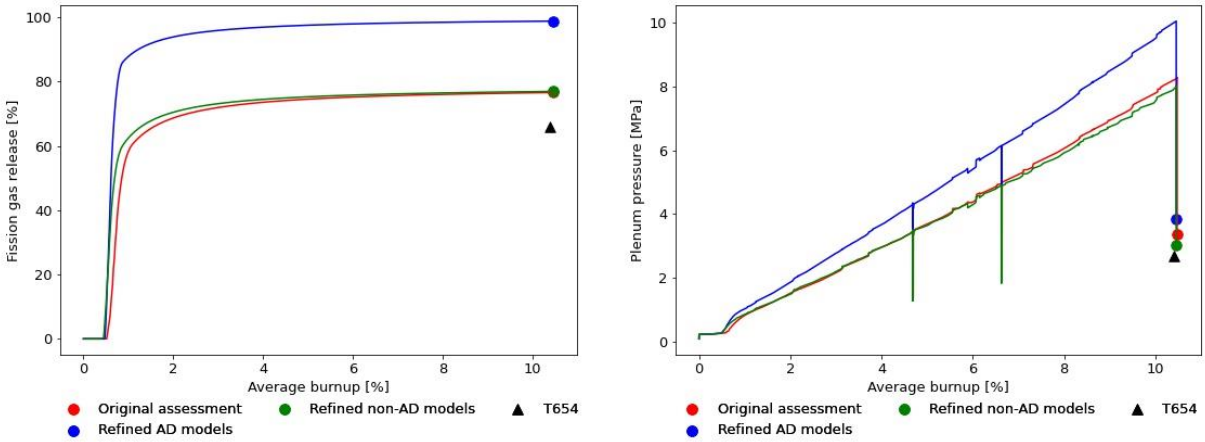

Figure 2. FGR (left) and plenum pressure (right) results obtained from the original T654 assessment compared with results obtained using the two new test cases. Experimental data from T654 are provided for comparison [15].

Peak radial cladding dilation and axial fuel elongation results are shown for the three cases alongside X430 data in Figure 3 [15]. All three cases underpredict radial cladding dilation and greatly overpredict axial fuel elongation. Correcting the critical and interconnection porosities in the non-AD and AD cases improves the predicted fuel elongation significantly but degrades the predicted cladding dilation slightly. All radial cladding dilation measurements obtained from U-19Pu-10Zr fuel elements irradiated during X430 are on the order of microns and vary widely, particularly with increasing burnup. Fuel elongation measurements were all on the order of millimeters, and were much more consistent with one another, considering the overall fuel aspect ratio. Therefore, the authors believe that the improved accuracy of the axial predictions represents an overall improvement in BISON's ability to model the fuel element's mechanical behavior.

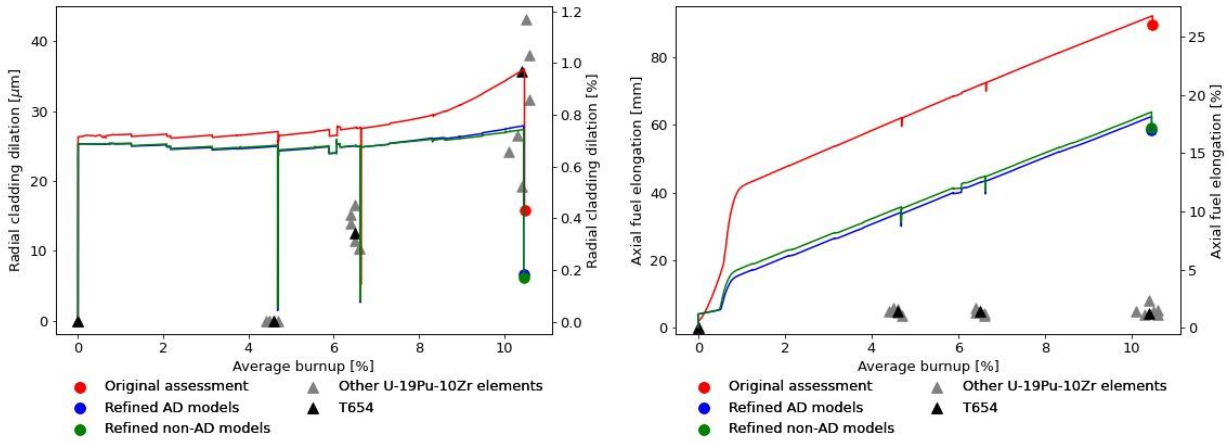

Figure 3. Peak radial cladding dilation (left) and axial fuel elongation (right) results obtained from the original T654 assessment compared with results obtained using the two new test cases. Experimental data from T654 and other U-19Pu-10Zr fuel elements irradiated during X430 are provided for comparison [15].

Finally, fuel temperatures and simulation timing results are shown for the three cases in Figure 4 [15]. The results show that the combined effects of the new models tend to decrease the predicted fuel temperature. Like the original assessment, the non-AD case does not include sodium infiltration because it cannot be applied with the LANL thermal conductivity correlation. The AD models do provide this capability, and the lowest fuel temperatures were obtained for this case, as expected. The results also show that the computational cost of the non-AD case is comparable with that of the original assessment, but the AD case requires about twice the amount of time to run on the same number of processors. Using Automatic Scaling did not significantly improve these results but should be reevaluated periodically as BISON continues to develop.

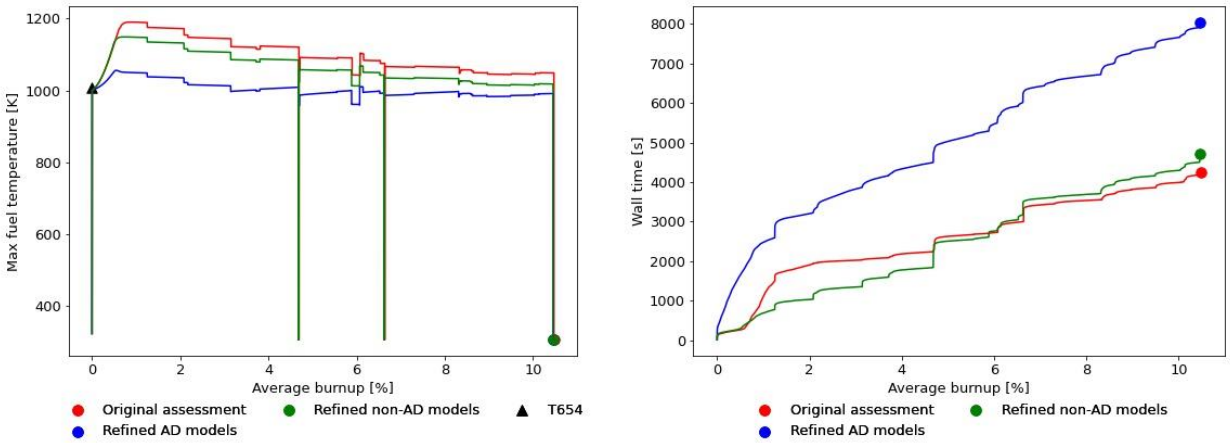

Figure 4. Maximum fuel temperature (left) and wall time (right) results obtained from the original T654 assessment compared with results obtained using the two new test cases. Experimental data from T654 are provided for comparison [15].

The AD case generally required fewer iterations to converge at each time step, and advanced models, such as viscoplastic swelling and constituent redistribution, are available only in AD. Unfortunately, the computational cost of AD simulations is higher overall, and the advanced models have not yet been widely adopted or thoroughly vetted through use in assessments. There are also no AD models for thermal expansion in the fuel or swelling in HT9, and its FGR predictions cannot be optimized as in non-AD models. Finally, the CoolantChannel and ThermalContact actions are not written to interface directly with AD properties.

Workarounds can be constructed manually within the input file to overcome some of these shortcomings by using models such as MaterialConverter and DerivativeParsedMaterial, but this is not an acceptable long-term solution. In light of these current limitations and despite promising recent developments, the authors believe that using non-AD models is most compatible with VTR's short-term metallic fuel benchmarking goals. The authors recommend reevaluating AD models at a later date. The input file syntax used in the non-AD case is provided in Appendix A.

The X430 data did not include experimental measurements associated with cladding failure or FCCI with which to compare BISON predictions and contrast different models. The authors recommend that multiple models for these behaviors be applied within each simulation. This will allow the results to be compared, inspected for consistency, and used to establish trends and expected ranges for these behaviors.

6.3 AXIAL FUEL ELONGATION

Comparing the FGR results from Figure 2 and the axial fuel elongation results from Figure 3 reveals another curious behavior in the predicted fuel swelling. Figure 2 suggests that the fuel has swollen,

reached interconnection porosity, and released its fission gas by ~0.5 at. % burnup. The new results in Figure 3 show that the fuel length is relatively constant during this time, aside from some initial elongation due to thermal expansion. This is consistent with a swelling anisotropy factor of 0.99.

The new results in Figure 3 show an abrupt elongation of the fuel after contact at ~0.5 at. % burnup. This is likely due to axial fuel creep being driven by contact pressure. Thereafter, the fuel swells primarily in the axial direction and continues to elongate at a near-constant rate. During this time, gases are vented to the plenum through interconnected porosity, so all of this elongation is due to the accumulation of solid fission products.

Experimental observations suggest that total volumetric swelling remains essentially constant up to ~10 at. % burnup and that volumetric swelling due to the accumulation of solid fission products should be accommodated by decreases in the vented porosity [23]. If these observations are correct, then BISON's current inability to capture porosity closure (sometimes referred to as *hot pressing*) due to solid swelling might be responsible for it overpredicting axial fuel elongation. By using the non-AD case from the previous section as a base, two additional test cases were constructed and run to investigate this possibility further.

Both test cases assumed that the accumulation of solid fission products does not contribute to fuel elongation/dilation. This assumption was implemented by omitting eigenstrains associated with solid swelling. The second test case took this assumption one step further by assuming that solid swelling contributes solely to pore closure. The second case is more realistic because it redirects the effect that solid swelling would have on fuel elongation/dilation to pore closure rather than neglecting solid swelling entirely.

In the second test case, a new porosity was calculated from the porosity provided by UPuZrGaseousEigenstrain by reducing it by $p = (\Delta V/V_0)/(1 + \Delta V/V_0)$, where $\Delta V/V_0$ was taken to be the 1.5% per atom-percent burnup that would result from solid swelling [7]. The new porosity was then provided to ThermalUPuZr, FgrUPuZr, UPuZrElasticityTensor, and UPuZrCreepUpdate to capture the effects of porosity closure on the thermo-mechanical behavior of the fuel. These modifications were made by using a ParsedMaterial in the input file—no BISON source code changes were needed.

Figure 5 compares the axial fuel elongation and fuel temperature results from the two new cases with the non-AD case from the previous section [15]. The radial cladding dilation and FGR results are not included here because all three cases predicted similar results. Figure 5 shows that accounting for porosity closure improves agreement between the predicted and observed axial fuel elongations. Furthermore, the fuel temperature results show that porosity closure might significantly impact BISON's temperature predictions (up to ~80°C toward the end of the simulations). This approach assumed that the thermal conductivity of solid fission products is the same as that of unirradiated U-19Pu-10Zr.

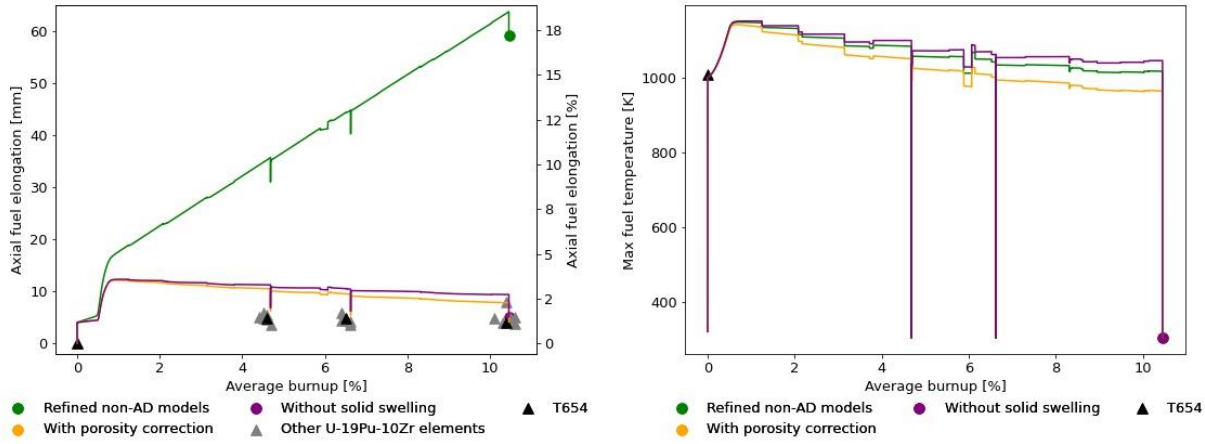

Figure 5. Axial fuel elongation (left) and maximum fuel temperature (right) results obtained for T654 using the non-AD case from the previous section and two new cases conducted to investigate the effects of solid swelling and porosity closure. Experimental data T654 and other U-19Pu-10Zr fuel elements irradiated during X430 are provided for comparison [15].

There are several notable limitations to the modeling approach used in this section. First, this approach assumes that solid swelling promotes the closure of all porosity, not just that which is interconnected and depressurized by venting to the plenum. Furthermore, UPuZrGaseousEigenstrain calculates interconnection based on its own porosity rather than the new one. These factors introduce errors during the first ~0.5 at. % burnup, the time at which significant FGR occurs. Second, this approach cannot account for the production of additional gaseous fission products, which would stabilize porosity until they are vented to the plenum. Continued production of solid and gaseous fission products and their competing effects on porosity evolution would likely establish a steady-state porosity—the minimum amount necessary to vent new fission gases. After this porosity is reached, the accumulation of additional solid fission products would tend to increase the length and/or diameter of the fuel. This places an upper limit on the burnup for which this approach can be considered realistic. Finally, solid swelling likely contributes to porosity closure and fuel elongation/dilation simultaneously, rather than one followed by the other.

These limitations aside, the authors believe that the second test case involving porosity closure is consistent with experimental observations and that it more accurately represents the behavior of low-burnup U-Pu-Zr fuels. Using this modeling approach drastically improves BISON's axial fuel elongation predictions for X430. The authors believe that the results and discussions in this section provide sufficient justification to motivate further investigation and the refinement of swelling and porosity models in BISON. In the meantime, the authors will continue to evaluate the effectiveness of this approach by applying it to other metallic fuel assessments. The input file syntax used in the porosity closure case is included in Appendix B.

7. RECOMMENDATIONS

All the models and settings currently recommended for use in metallic fuel assessments are provided in Appendices A and B. This section summarizes other recommendations from the previous sections based on what the authors consider to be the most impactful tasks from the perspective of the VTR project. These recommendations are grouped into three categories: optimizing BISON's existing capabilities, short-term code development, and long-term code development. These specific, actionable recommendations are included in the following sections.

7.1 BISON USAGE

The authors recommend using the models and input file syntax shown in Appendices A and B. The following additional recommendations are based on BISON's existing capabilities and are expected to improve its accuracy and computational performance for metallic fuel assessments.

- Calculate and document appropriate values for the energy per fission, fission gas bubble density, solid swelling factor, and swelling anisotropy factor.
- Investigate the various approaches for calculating fast neutron flux in BISON. Determine whether it should be specified by the user or calculated from the linear heat rate, whether the linear heat rate-based calculations are appropriate for the isotopic compositions under consideration, and what code modifications might be needed to address any issues identified in these studies.
- Conduct studies to identify the optimal models and parameters for contact (with or without friction), cladding failure, and FCCI. Run simulations with multiple cladding failure and FCCI models to help establish trends and expected ranges for these behaviors.
- Conduct studies to identify the optimal meshing, PETSc, quadrature, damping, and time-stepping options for metallic fuel performance simulations.
- Evaluate whether the no-acceleration assumption is appropriate for simulations involving off-normal conditions.
- Periodically reevaluate AD models, advanced swelling models, constituent redistribution models, sodium infiltration models, models that couple FCCI to cladding stress, and Automatic Scaling as BISON continues to develop.

7.2 SHORT-TERM BISON DEVELOPMENT

The authors recommend the following short-term code development projects, which should require only minor time investments. Many of these projects involve balancing functionality between AD and non-AD models, which the authors believe will ease the transition between the two systems and maximize BISON's flexibility for future metallic fuel applications. Other projects involve developing and testing new capabilities, such as revised swelling models and the models needed to couple FCCI to cladding stress. These projects would benefit from the support of the Nuclear Energy Advanced Modeling and Simulation (NEAMS) Program and/or the Advanced Fuels Campaign (AFC) and would yield impactful benefits to the VTR program.

- Resolve inconsistencies in the names of AD and non-AD objects, such as ThermalUPuZr/ADUPuZrThermal and ThermalHT9/ADHT9Thermal.
- Investigate, develop, and implement a method to account for vented porosity closure (i.e., hot pressing) due to solid swelling in the fuel.
- Continue developing the models needed to couple FCCI to cladding stress.
- Reproduce all ADUPuZrThermal functionality within ThermalUPuZr and create non-AD versions of ADUPuZrSodiumLogging, ADSimpleFissionGasViscoplasticityStressUpdate, and ADCoupledFissionGasViscoplasticityStressUpdate to make these impactful options available for non-AD applications.
- Create AD versions of UPuZrThermalExpansionEigenstrain and HT9VolumetricSwellingEigenstrain to make them available for AD applications.
- Revise the ThermalContact and CoolantChannel actions to make them directly compatible with AD material properties.
- Implement checks to warn about or enforce consistency between the porosity interconnectivity thresholds used in UPuZrGaseousEigenstrain, FgrUPuZr, and their AD counterparts.
- Correct the typos identified in the HT9VolumetricSwellingEigenstrain and UPuZrThermalExpansionEigenstrain documentation.

7.3 LONG-TERM BISON DEVELOPMENT

The authors recommend the following long-term code development projects, which would also benefit from the support of NEAMS and/or AFC.

- Once the models and settings most appropriate for use in metallic fuel assessments are identified, incorporate them into actions to simplify the input syntax. This would substantially improve usability for designers, engineers, and regulators.
- Update all U-Pu-Zr and HT9 material properties and constitutive relations as new data are collected and correlations are made available.
- Expand these recommendations to develop and refine similar models for D9 and 316 stainless-steel claddings and U-10Zr fuels to increase the number of irradiation experiments that can be developed into BISON benchmarks.

8. CONCLUSION

The BISON nuclear fuel performance code will be used to model VTR driver fuel, including looking at the effects of differences between the VTR driver fuel element design and the legacy fuel designs and experiments on which it is based. Simulations will be conducted to help determine whether the design's behavior and performance are properly understood and to assess the margins to cladding failure and fuel melting relative to those predicted for past metallic fuel experiments. These predictions are expected to streamline VTR design and operation by helping inform the VTR driver fuel element design and providing supplemental information for the fuel design safety basis.

In this work, a critical review of the metallic fuel models available in BISON was conducted to improve the accuracy and robustness of BISON's predictions for VTR applications. Two new approaches were defined for modeling metallic fuel performance by using BISON's existing capabilities, and how the use of these approaches improves the accuracy of BISON's predictions was demonstrated by simulating an irradiation experiment conducted in EBR-II. Based on these approaches, several issues were identified that will require further investigation and recommendations were made for continued BISON use and code development. The authors will continue to refine the metallic fuel modeling approach with the results of these investigations and monitor BISON for new developments, incorporating updates as they are made available.

9. REFERENCES

[1] "Versatile Test Reactor." https://www.energy.gov/ne/nuclear-reactor-technologies/versatile-test-reactor (accessed Sep. 15, 2020).

[2] Y. I. Chang, L. C. Walters, J. J. Laidler, D. R. Pedersen, D. C. Wade, and M. J. Lineberry, "Integral Fast Reactor Program annual progress report FY 1994 - ANL-IFR-246," Argonne, 1994.

[3] R. G. Pahl, D. L. Porter, C. E. Lahm, and G. L. Hofman, "Experimental studies of U-Pu-Zr fast reactor fuel pins in the Experimental Breeder Reactor-II," *Metall. Trans. A, Phys. Metall. Mater. Sci.*, vol. 21 A, no. 7, pp. 1863–1870, 1990, doi: 10.1007/BF02647233.

[4] R. Seidel *et al.*, "A decade of advances in metallic fuel," 1990.

[5] A. M. Yacout and M. C. Billone, "Current Status of the LIFE Fast Reactors Fuel Performance Codes," 2013.

[6] J. D. Hales *et al.*, "BISON Theory Manual: The Equations Behind Nuclear Fuel Analysis - BISON Release 1.3 - INL/EXT-13-29930," Idaho Falls, Idaho, 2016. doi: 10.2172/1107264.

[7] T. Ogata and T. Yokoo, "Development and validation of ALFUS: an irradiation behavior analysis code for metallic fast reactor fuels," *Nucl. Technol.*, vol. 128, no. 1, pp. 113–123, 1999, doi: 10.13182/NT99-A3018.

[8] A. Karahan and J. Buongiorno, "A new code for predicting the thermo-mechanical and irradiation behavior of metallic fuels in sodium fast reactors," *J. Nucl. Mater.*, vol. 396, no. 2–3, pp. 283–293, 2010, doi: 10.1016/j.jnucmat.2009.11.022.

[9] W. Hwang, C. Nam, T. S. Byun, and Y. C. Kim, "MACSIS: A metallic fuel performance analysis code for Simulating IN-reactor behavior under Steady-state conditions," *Nucl. Technol.*, vol. 123, no. 2, pp. 130–141, 1998, doi: 10.13182/NT98-A2887.

[10] T. Kobayashi *et al.*, "Development of the sesame metallic fuel performance code," *Nucl. Technol.*, vol. 89, no. 2, 1990, doi: 10.13182/NT90-A34345.

[11] S. R. Novascone, A. Casagranda, P. G. Medvedev, C. Matthews, and A. X. Zabriskie, "Summary and Assessment of Metallic Fuel Capabilities in Bison - INL/EXT-18-51399," Idaho Falls, Idaho, 2018.

[12] I. Greenquist and J. Powers, "Metallic Fuel Benchmark Simulations Based on the X430 Experiments - ORNL/TM-2020/1565," Oak Ridge, Tennessee, 2020.

[13] I. Greenquist, K. Cunningham, J. Hu, and J. Powers, "A metallic fuel performance benchmark problem based on the IFR-1 experiment - ORNL/TM-2020/1534," Oak Ridge, Tennessee, 2020.

[14] G. Pastore *et al.*, "Uncertainty and sensitivity analysis of fission gas behavior in engineering-scale fuel modeling," *J. Nucl. Mater.*, vol. 456, pp. 398–408, 2015, doi: 10.1016/j.jnucmat.2014.09.077.

[15] S. Hayes, D. Crawford, and R. Pahl, "Test design description and postirradiation examination of the HT9 advanced driver fuel test (X430) - ANL-IFR-225," Argonne, Illinois, 1994.

[16] A. M. Yacout, W. S. Yang, G. L. Hofman, and Y. Orechwa, "Average irradiation temperature for the analysis of in-pile integral measurements," *Nucl. Technol.*, vol. 115, no. 1, pp. 61–72, 1996, doi: 10.13182/NT96-A35275.

[17] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandie, "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nucl. Eng. Des.*, vol. 239, no. 10, pp. 1768–1778, 2009, doi: 10.1016/j.nucengdes.2009.05.021.

[18] C. J. Permann *et al.*, "MOOSE: Enabling massively parallel multiphysics simulation," *SoftwareX*, vol. 11, p. 100430, 2020, doi: 10.1016/j.softx.2020.100430.

[19] R. G. Pahl, D. L. Porter, D. C. Crawford, and L. C. Walters, "Irradiation behavior of metallic fast reactor fuels," *J. Nucl. Mater.*, vol. 188, no. C, pp. 3–9, 1992, doi: 10.1016/0022-3115(92)90447-S.

[20] G. L. Hofman, L. C. Walters, and T. H. Bauer, "Metallic fast reactor fuels," *Prog. Nucl. Energy*, vol. 31, no. 1/2, pp. 83–110, 1997.

[21] A. Aitkaliyeva, M. Tonks, J. Hirschhorn, J. Powers, I. Greenquist, and B. Beeler, "Research Needs for Uranium-Zirconium- Based Metallic Fuels - INL/EXT-20-58719," Idaho Falls, Idaho, 2020.

[22] G. L. Hofman, R. G. Pahl, C. E. Lahm, and D. L. Porter, "Swelling behavior of U-Pu-Zr fuel,"

Metall. Trans. A, vol. 21, no. 2, pp. 517–528, 1990, doi: 10.1007/BF02671924.
[23] T. H. Bauer and J. W. Holland, "In-Pile Measurement of the Thermal Conductivity of Irradiated Metallic Fuel," *Nucl. Technol.*, vol. 110, no. 3, pp. 407–421, 1995, doi: 10.13182/nt95-a35110.
[24] C. Matthews, C. Unal, J. Galloway, D. D. Keiser, and S. L. Hayes, "Fuel-cladding chemical interaction in U-Pu-Zr metallic fuels: A critical review," *Nucl. Technol.*, vol. 198, no. 3, pp. 231–259, 2017, doi: 10.1080/00295450.2017.1323535.
[25] R. D. Mariani, D. L. Porter, T. P. O'Holleran, S. L. Hayes, and J. R. Kennedy, "Lanthanides in metallic nuclear fuels: Their behavior and methods for their control," *J. Nucl. Mater.*, vol. 419, no. 1–3, pp. 263–271, 2011, doi: 10.1016/j.jnucmat.2011.08.036.
[26] R. G. Pahl, C. E. Lahm, and S. L. Hayes, "Performance of HT9 clad metallic fuel at high temperature," *J. Nucl. Mater.*, vol. 204, no. C, pp. 141–147, 1993, doi: 10.1016/0022-3115(93)90210-P.
[27] M. Ishida, T. Ogata, and M. Kinoshita, "Constituent migration model for U-Pu-Zr metallic fast reactor fuel," *Nucl. Technol.*, vol. 104, no. 1, pp. 37–51, 1993, doi: 10.13182/NT93-A34868.
[28] Y. S. Kim, G. L. Hofman, S. L. Hayes, and Y. H. Sohn, "Constituent redistribution in U-Pu-Zr fuel during irradiation," *J. Nucl. Mater.*, vol. 327, no. 1, pp. 27–36, 2004, doi: 10.1016/j.jnucmat.2004.01.012.
[29] J. Hirschhorn, M. Tonks, A. Aitkaliyeva, and C. Adkins, "A study of constituent redistribution in U–Zr fuels using quantitative phase-field modeling and sensitivity analysis," *J. Nucl. Mater.*, vol. 523, pp. 143–156, 2019, doi: 10.1016/j.jnucmat.2019.05.053.
[30] J. Galloway, C. Unal, N. Carlson, D. Porter, and S. Hayes, "Modeling constituent redistribution in U-Pu-Zr metallic fuel using the advanced fuel performance code BISON," *Nucl. Eng. Des.*, vol. 286, pp. 1–17, 2015, doi: 10.1016/j.nucengdes.2015.01.014.
[31] "BISON: A Finite Element-Based Nuclear Fuel Performance Code," 2020. https://mooseframework.inl.gov/bison/ (accessed Oct. 27, 2020).
[32] J. Hu, J. Powers, A. Oaks, and F. Heidet, "Preliminary fuel performance assessment of proposed versatile test reactor driver fuel concept for normal operating conditions using BISON - ORNL/SPR-2019/1302," Oak Ridge, Tennessee, 2020.
[33] H. Savage, "The heat content and specific heat of some metallic fast-reactor fuels containing plutonium," *J. Nucl. Mater.*, vol. 25, no. 3, pp. 249–259, 1968, doi: 10.1016/0022-3115(68)90168-2.
[34] A. Karahan, "Modeling the thermo-mechanical and irradiaton behavior of metallic and oxide fuels for sodium fast reactors," Massachusetts Institute of Technology, 2009.
[35] Y. S. Kim, T. W. Cho, and D. S. Sohn, "Thermal conductivities of actinides (U, Pu, Np, Cm, Am) and uranium-alloys (U-Zr, U-Pu-Zr and U-Pu-TRU-Zr)," *J. Nucl. Mater.*, vol. 445, pp. 272–280, 2014, doi: 10.1016/j.jnucmat.2013.11.018.
[36] G. Hofman *et al.*, "Metallic fuels handbook - ANL-NSE-3," Argonne, Illinois, 1989.
[37] D. Olander, "Fundamental aspects of nuclear reactor fuel elements - solutions to problems," Berkely, California, 1976.
[38] N. Yamanouchi, M. Tamura, H. Hayakawa, A. Hishinuma, and T. Kondo, "Accumulation of engineering data for practical use of reduced activation ferritic steel: 8%Cr2%W0.2%V0.04%TaFe," *J. Nucl. Mater.*, vol. 191–194, pp. 822–826, 1992, doi: 10.1016/0022-3115(92)90587-B.
[39] "FIPD: EBR-II Fuels Irradiation & Physics Database." https://fipd2.ne.anl.gov (accessed Sep. 17, 2020).
[40] J. K. Fink and L. Leibowitz, "Thermodynamic and transport properties of sodium liquid and vapor - ANL/RE-95/2," Argonne, Illinois, 1995.
[41] A. Lindsay *et al.*, "Automatic Differentiation in MetaPhysicL and its Applications in MOOSE," *Nucl. Technol.*, 2020.
[42] C. Matthews and C. Unal, "Initial implementation of a bubble-surface force-balance fission gas behavior algorithm for metallic nuclear fuel into BISON - LA-UR-19-318314," Los Alamos, New Mexico, 2019.
[43] K. Geelhood and I. Porter, "Modeling and assessment of EBR-II fuel with the US NRC's FAST

fuel performance code," in *Top Fuel*, 2018, no. October.
[44] "Design Properties of HT9 and Russian Ferritic/Martensitic Steels," in *FCRD Materials Handbook - Materials Data for Fast Spectrum Transmutation - LA-CP-14-20070 Rev 6*, Los Alamos., Los Alamos, New Mexico, 2014.
[45] L. Leibowitz and R. A. Blomquist, "Thermal conductivity and thermal expansion of stainless steels D9 and HT9."
[46] A. Casagranda, L. Aagesen, J. Ke, T. Miller, and S. Novascone, "Demonstrate a physics-based, lower-length-scale-informed swelling model that predicts EBR-II observations," Idaho Falls, Idaho, 2020.

APPENDIX A. INPUT FILE SYNTAX WITHOUT POROSITY CLOSURE

INPUT FILE SYNTAX WITHOUT POROSITY CLOSURE

This appendix contains an example of the BISON input file syntax for the non-AD approach without porosity closure.

```
# X430 BENCHMARK PROBLEM
# PIN T654
# Units are in standard SI: J, K, kg, m, Pa, s.

[GlobalParams]
  order = SECOND
  family = LAGRANGE
  elem_type = QUAD8
  density = 15800
  initial_density = 15800
  energy_per_fission = 3.2e-11
  volumetric_locking_correction = false
  displacements = 'disp_x disp_y'
  temperature = T
  initial_X_Zr = 0.226
  initial_X_Pu = 0.160
  X_Zr = 0.226
  X_Pu = 0.160
[]

[Problem]
  type = ReferenceResidualProblem
  coord_type = RZ
  reference_vector = ref
  extra_tag_vectors = ref
  group_variables = 'T disp_x disp_y'
[]

[Mesh]
  [./generated]
    type = SmearedPelletMeshGenerator
    pellet_quantity = 1
    pellet_outer_radius = 2.8410e-03
    pellet_height = 3.4440e-01
    clad_gap_width = 4.3560e-04
    clad_bot_gap_height = 4.0000e-03
    clad_top_gap_height = 3.7725e-01
    clad_thickness = 4.0640e-04
    top_bot_clad_height = 1.5000e-02
    pellet_mesh_density = customize
    clad_mesh_density = customize
    nx_p = 5
    ny_p = 250
    nx_c = 8
    ny_c = 120
    ny_cu = 4
    ny_cl = 4
  [../]
  patch_size = 5
  patch_update_strategy = auto
  partitioner = centroid
  centroid_partitioner_direction = y
[]

[Variables]
  [./T]
    initial_condition = 295
  [../]
[]

```

```
[AuxVariables]
  [./fuel_thermal_strain_magnitude]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_thermal_strain_magnitude]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./fuel_creep_strain_magnitude]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_creep_strain_magnitude]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./fuel_gaseous_strain_magnitude]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./fuel_solid_strain_magnitude]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./gap_conductance]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./element_failed]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./fuel_volumetric_strain]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_hoop_stress]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_hoop_creep_strain]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_hoop_elastic_strain]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_hoop_total_strain]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./local_power]
    block = pellet
    order = CONSTANT
```

```
    family = MONOMIAL
  [../]
  [./T_coolant]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./linear_heat_rate]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
[]

[Functions]
  [./power_history_function]
    type = PiecewiseLinear
    x =  '0 3600 8203212 8206812 13814423 13818023 14428975 14432575 21312419
          21316019 25596874 25600474 26261755 26265355 32714598 32718198 32721798
          32725398 32728998 32896765 32900365 39574695 39578295 42194062 42197662
          43820808 43824408 43895709 43899309 44401212 44404812 47385472 47389072
          48198548 48202148 48205748 48209348 48212948 52079977 52083577 53874489
          53878089 62125235 62128835 62256058 62259658 62620357 62623957 64516928
          64520528 64766586 64770186 67535546 67539146 72155534 72159134 72185697
          72189297 76833647 76837247 77340548 77344148 77738400 77742000 80444447
          80448047 80451647 80455247'
    y = '0.0 44225.3 44225.3 43106.1 43106.1 41403.6 41403.6 41119.9 41119.9
         38881.4 38881.4 38353.3 38353.3 39472.5 39472.5     0.0     0.0     0.0
         33490.2 33490.2 36863.6 36863.6 37123.7 37123.7 32717.8 32717.8 38534.6
         38534.6 38432.1 38432.1 36784.8 36784.8 36036.0 36036.0     0.0     0.0
         0.0 35153.3 35153.3 35153.3 35153.3 35271.5 35271.5 33663.6 33663.6
         34459.7 34459.7 34640.9 34640.9 34428.1 34428.1 34026.2 34026.2 33624.2
         33624.2 33624.2 33624.2 33718.8 33718.8 34057.7 34057.7 34057.7 34057.7
         34215.3 34215.3     0.0     0.0     0.0'
  [../]
  [./coolant_pressure_function]
    type = ConstantFunction
    value = 347702.6
  [../]
  [./T_coolant_in_function]
    type = PiecewiseLinear
    x = '0   3600   32718198 32721798 32725398 32728998 48202148 48205748
         48209348 48212948 80448047 80451647 80455247'
    y = '295 644.15 644.15   305.00   305.00   644.15   644.15   305.00
         305.00   644.15   644.15   305.00   305.00'
  [../]
  [./axial_peaking_factor_function]
    type = PiecewiseBilinear
    xaxis = 1
    yaxis = 0
    y = '0 32725398 48209348 80455247'
    x = '0.018 0.019 0.0534 0.0879 0.1223 0.1567 0.1912 0.2256 0.2601 0.2945 0.3289 0.3634
0.3734'
    z = '0.0000 0.9111 0.9983 1.0625 1.1053 1.1195 1.1053
         1.0696 1.0054 0.9127 0.8129 0.7059 0.0000
         0.0000 0.9056 0.9864 1.0487 1.1006 1.111 1.1006
         1.0695 1.0175 0.9241 0.8306 0.7164 0.0000
         0.0000 0.8961 0.9845 1.0442 1.0939 1.1138 1.1039
         1.074 1.0144 0.9348 0.8354 0.7061 0.0000
         0.0000 0.8954 0.9752 1.0368 1.0882 1.1087 1.0984
         1.0779 1.0163 0.9445 0.8418 0.7289 0.0000'
  [../]
  [./heat_generation_rate_function]
    type = ParsedFunction
    vars = 'lhr_bar p_factor'
    vals = 'power_history_function axial_peaking_factor_function'
    value = 'lhr_bar * p_factor'
  [../]
```

```
    [./gas_volume_function]
      type = ParsedFunction
      vars = 'v_cladding v_fuel'
      vals = 'cladding_volume fuel_volume'
      value = 'abs(v_cladding) - abs(v_fuel)'
    [../]
    [./coolant_flux_function]
      type = PiecewiseLinear
      x = '0 3600  8203212  8206812 13814423 13818023 14428975 14432575 21312419
           21316019 25596874 25600474 26261755 26265355 32714598 32718198 32885965
           32889565 39563895 39567495 42183262 42186862 43810008 43813608 43884909
           43888509 44390412 44394012 47374672 47378272 48187748 48191348 52058377
           52061977 53852889 53856489 62103635 62107235 62234458 62238058 62598757
           62602357 64495328 64498928 64744986 64748586 67513946 67517546 72133934
           72137534 72164097 72167697 76812047 76815647 77318948 77322548 77716800
           77720400 80422847 80426447 80430047 80516447'
      y = '2699.1 2699.1 2699.1 2724.0 2724.0 2697.2 2697.2 2781.0 2781.0 2721.1
           2721.1 2696.9 2696.9 2785.4 2785.4 2793.7 2793.7 2803.5 2803.5 2814.2
           2814.2 2799.6 2799.6 2840.1 2840.1 2839.6 2839.6 2873.7 2873.7 2855.7
           2855.7 2826.4 2826.4 2826.4 2826.4 2788.4 2788.4 2780.6 2780.6 2771.8
           2771.8 2781.5 2781.5 2817.1 2817.1 2807.4 2807.4 2777.1 2777.1 2777.1
           2777.1 2746.4 2746.4 2765.9 2765.9 2765.9 2765.9 2777.1 2777.1 2777.1
           2777.1 2777.1'
    [../]
    [./sodium_conductivity_function]
      type = ParsedFunction
      vars = 'A      B        C         D'
      vals = '124.67 -0.11381 5.5226e-5 -1.1842e-8'
      value = 'A + B * t + C * t^2 + D * t^3'
    [../]
  []

  [Modules/TensorMechanics/Master]
    add_variables = true
    strain = FINITE
    generate_output = 'stress_xx stress_yy stress_zz vonmises_stress hydrostatic_stress
  creep_strain_zz elastic_strain_zz strain_zz'
    [./fuel_mechanics]
      block = pellet
      eigenstrain_names = 'fuel_thermal_strain fuel_gaseous_strain fuel_solid_strain'
      extra_vector_tags = ref
    [../]
    [./cladding_mechanics]
      block = clad
      eigenstrain_names = 'cladding_thermal_strain cladding_gaseous_strain'
      extra_vector_tags = ref
    [../]
  []

  [Kernels]
    [./gravity]
      type = Gravity
      block = 'pellet clad'
      variable = disp_y
      value = -9.81
      extra_vector_tags = ref
    [../]
    [./heat_conduction_time_derivative]
      type = HeatConductionTimeDerivative
      block = 'pellet clad'
      variable = T
      extra_vector_tags = ref
    [../]
    [./heat_conduction]
      type = HeatConduction
      block = 'pellet clad'
      variable = T
```

```
    extra_vector_tags = ref
  [../]
  [./heat_source]
    type = FissionRateHeatSource
    block = pellet
    variable = T
    fission_rate = fission_rate
    extra_vector_tags = ref
  [../]
[]

[AuxKernels]
  [./fuel_thermal_strain_magnitude]
    type = RankTwoScalarAux
    block = pellet
    variable = fuel_thermal_strain_magnitude
    rank_two_tensor = fuel_thermal_strain
    scalar_type = EffectiveStrain
    execute_on = TIMESTEP_END
  [../]
  [./cladding_thermal_strain_magnitude]
    type = RankTwoScalarAux
    block = clad
    variable = cladding_thermal_strain_magnitude
    rank_two_tensor = cladding_thermal_strain
    scalar_type = EffectiveStrain
    execute_on = TIMESTEP_END
  [../]
  [./fuel_creep_strain_magnitude]
    type = RankTwoScalarAux
    block = pellet
    variable = fuel_creep_strain_magnitude
    rank_two_tensor = creep_strain
    scalar_type = EffectiveStrain
    execute_on = TIMESTEP_END
  [../]
  [./cladding_creep_strain_magnitude]
    type = RankTwoScalarAux
    block = clad
    variable = cladding_creep_strain_magnitude
    rank_two_tensor = creep_strain
    scalar_type = EffectiveStrain
    execute_on = TIMESTEP_END
  [../]
  [./fuel_gaseous_strain_magnitude]
    type = RankTwoScalarAux
    block = pellet
    variable = fuel_gaseous_strain_magnitude
    rank_two_tensor = fuel_gaseous_strain
    scalar_type = EffectiveStrain
    execute_on = TIMESTEP_END
  [../]
  [./fuel_solid_strain_magnitude]
    type = RankTwoScalarAux
    block = pellet
    variable = fuel_solid_strain_magnitude
    rank_two_tensor = fuel_solid_strain
    scalar_type = EffectiveStrain
    execute_on = TIMESTEP_END
  [../]
  [./gap_conductance]
    type = MaterialRealAux
    variable = gap_conductance
    property = gap_conductance
    boundary = pellet_outer_radial_surface
  [../]
  [./failed_element]
```

```
    type = MaterialRealAux
    variable = element_failed
    property = failed
    boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
  [../]
  [./fuel_volumetric_strain]
    type = RankTwoScalarAux
    block = pellet
    variable = fuel_volumetric_strain
    rank_two_tensor = total_strain
    scalar_type = VolumetricStrain
    execute_on = TIMESTEP_END
  [../]
  [./cladding_hoop_stress]
    type = RankTwoAux
    block = clad
    variable = cladding_hoop_stress
    rank_two_tensor = stress
    index_i = 2
    index_j = 2
    execute_on = TIMESTEP_END
  [../]
  [./cladding_hoop_creep_strain]
    type = RankTwoAux
    block = clad
    variable = cladding_hoop_creep_strain
    rank_two_tensor = creep_strain
    index_i = 2
    index_j = 2
    execute_on = TIMESTEP_END
  [../]
  [./cladding_hoop_elastic_strain]
    type = RankTwoAux
    block = clad
    variable = cladding_hoop_elastic_strain
    rank_two_tensor = elastic_strain
    index_i = 2
    index_j = 2
    execute_on = TIMESTEP_END
  [../]
  [./cladding_hoop_total_strain]
    type = RankTwoAux
    block = clad
    variable = cladding_hoop_total_strain
    rank_two_tensor = total_strain
    index_i = 2
    index_j = 2
    execute_on = TIMESTEP_END
  [../]
  [./local_power]
    type = FunctionAux
    block = pellet
    variable = local_power
    function = axial_peaking_factor_function
  [../]
  [./T_coolant]
    type = MaterialRealAux
    variable = T_coolant
    property = coolant_temperature
  boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
  [../]
  [./linear_heat_rate]
    type = FunctionAux
    block = pellet
    variable = linear_heat_rate
    function = heat_generation_rate_function
    execute_on = 'INITIAL TIMESTEP_END'
```

```
  [../]
[]

[Contact]
  [./pellet_cladding_mechanical]
    master = clad_inside_right
    slave = pellet_outer_radial_surface
    model = frictionless
    formulation = kinematic
    system = constraint
    tangential_tolerance = 1e-3
    normal_smoothing_distance = 0.1
  [../]
[]

[ThermalContact]
  [./thermal_contact]
    type = GapHeatTransfer
    variable = T
    primary = clad_inside_right
    secondary = pellet_outer_radial_surface
    gap_geometry_type = CYLINDER
    gap_conductivity_function = sodium_conductivity_function
    gap_conductivity_function_variable = T
    quadrature = true
    min_gap = 4.3560e-04
  [../]
[]

[BCs]
  [./fix_disp_x_all]
    type = DirichletBC
    variable = disp_x
    value = 0.0
    boundary = centerline
  [../]
  [./fix_disp_y_all]
    type = DirichletBC
    variable = disp_y
    value = 0.0
    boundary = 'clad_outside_bottom bottom_of_bottom_pellet'
  [../]
  [./Pressure]
    [./coolant_pressure]
      function = coolant_pressure_function
      boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
    [../]
  [../]
  [./PlenumPressure]
    [./plenum_pressure]
      boundary = 'clad_inside_bottom clad_inside_right clad_inside_top'
      startup_time = 0
      initial_pressure = 84000
      volume = gas_volume
      material_input = fission_gas_released
      R = 8.3143
      temperature = plenum_temperature
      output = plenum_pressure
    [../]
  [../]
[]

[PlenumTemperature]
  [./plenum_temperature]
    temp = T
    boundary = 'all_pellet_exterior all_clad_interior'
    inner_surfaces = all_pellet_exterior
```

```
    outer_surfaces = all_clad_interior
  [../]
[]

[CoolantChannel]
  [./convective_cladding_surface]
    variable = T
    inlet_temperature = T_coolant_in_function
    inlet_pressure = coolant_pressure_function
    inlet_massflux = coolant_flux_function
    coolant_material = sodium
    rod_diameter = 7.3660e-03
    rod_pitch = 8.7884e-03
    linear_heat_rate = power_history_function
    axial_power_profile = axial_peaking_factor_function
    subchannel_geometry = triangular
    boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
  [../]
[]

[Materials]
  ###### FUEL ######
  [./fission_rate]
    type = UPuZrFissionRate
    block = pellet
    rod_linear_power = power_history_function
    axial_power_profile = axial_peaking_factor_function
    pellet_radius = 2.8410e-03
  [../]
  [./burnup]
    type = UPuZrBurnup
    block = pellet
    outputs = exodus
    output_properties = burnup
  [../]
  [./fuel_density]
    type = Density
    block = pellet
  [../]
  [./fuel_thermal_properties]
    type = ThermalUPuZr
    block = pellet
    spheat_model = savage
    thcond_model = lanl
    porosity = porosity
  [../]
  [./fuel_gaseous_swelling]
    type = UPuZrGaseousEigenstrain
    block = pellet
    fission_rate = fission_rate
    anisotropic_factor = 0.99
    bubble_number_density = 3.02e+17
    eigenstrain_name = fuel_gaseous_strain
    outputs = exodus
    output_properties = 'gas_swelling porosity'
  [../]
  [./fission_gas_release]
    type = FgrUPuZr
    block = pellet
    fission_rate = fission_rate
    fractional_fgr_initial = 0.512
    fractional_fgr_post = 0.785
  [../]
  [./fuel_solid_swelling]
    type = BurnupDependentEigenstrain
    block = pellet
    eigenstrain_name = fuel_solid_strain
```

```
    swelling_name = solid_swelling
    outputs = exodus
    output_properties = solid_swelling
  [../]
  [./fuel_thermal_expansion]
    type = UPuZrThermalExpansionEigenstrain
    block = pellet
    stress_free_temperature = 295
    eigenstrain_name = fuel_thermal_strain
    outputs = exodus
  [../]
  [./fuel_elasticity_tensor]
    type = UPuZrElasticityTensor
    block = pellet
  [../]
  [./fuel_creep]
    type = UPuZrCreepUpdate
    block = pellet
    porosity = porosity
    max_inelastic_increment = 1e-2
  [../]
  [./fuel_elastic_stress]
    type = ComputeMultipleInelasticStress
    block = pellet
    inelastic_models = fuel_creep
  [../]
  ###### CLADDING ######
  [./fast_neutron_flux]
    type = UPuZrFastNeutronFlux
    pellet_radius = 2.8410e-03
    axial_power_profile = axial_peaking_factor_function
    rod_linear_power = power_history_function
    calculate_fluence = true
    outputs = exodus
    output_properties = fast_neutron_flux
  [../]
  [./cladding_density]
    type = Density
    block = clad
    density = 7771
  [../]
  [./cladding_thermal_properties]
    type = ThermalHT9
    block = clad
  [../]
  [./cladding_gaseous_swelling]
    type = HT9VolumetricSwellingEigenstrain
    block = clad
    fast_neutron_flux = fast_neutron_flux
    fast_neutron_fluence = fast_neutron_fluence
    eigenstrain_name = cladding_gaseous_strain
  [../]
  [./cladding_thermal_expansion]
    type = HT9ThermalExpansionEigenstrain
    block = clad
    eigenstrain_name = cladding_thermal_strain
    stress_free_temperature = 295
  [../]
  [./cladding_elasticity_tensor]
    type = HT9ElasticityTensor
    block = clad
  [../]
  [./cladding_creep]
    type = HT9CreepUpdate
    block = clad
    max_inelastic_increment = 1e-2
  [../]
```

```
  [./cladding_elastic_stress]
    type = ComputeMultipleInelasticStress
    block = clad
    inelastic_models = cladding_creep
  [../]
  [./cladding_failure]
    type = FailureCladHT9
    method = cdf_long
    hoop_stress = stress_zz
    boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
    outputs = exodus
    output_properties = cdf_failure
  [../]
[]

[Preconditioning]
  [./SMP]
    type = SMP
    full = true
  [../]
[]

[Executioner]
  type = Transient
  solve_type = PJFNK
  petsc_options = '-snes_ksp_ew'
  petsc_options_iname = '-pc_type -pc_factor_mat_solver_package -ksp_gmres_restart'
  petsc_options_value = 'lu       superlu_dist                  31'
  line_search = NONE
  l_max_its = 30
  l_tol = 1e-2
  nl_max_its = 30
  nl_rel_tol = 5e-4
  nl_abs_tol = 1e-7
  end_time = 80455247
  dtmin = 1e-2
  dtmax = 1e6
  start_time = 0
  verbose = true
  [./Quadrature]
    order = FIFTH
    side_order = SEVENTH
  [../]
  [./TimeStepper]
    type = IterationAdaptiveDT
    dt = 100
    optimal_iterations = 10
    iteration_window = 4
    growth_factor = 1.25
    cutback_factor = 0.512
    linear_iteration_ratio = 100
    force_step_every_function_point = true
    timestep_limiting_function = power_history_function
    timestep_limiting_postprocessor = creep_timestep
  [../]
[]

[Postprocessors]
  #### FISSION GAS #### (needed for simulation to run)
  [./fission_gas_produced]
    type = ElementIntegralFisGasProduce
    block = pellet
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./fission_gas_released]
    type = ElementIntegralFisGasRelease
```

```
    block = pellet
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./fission_gas_percent]
    type = FGRPercent
    fission_gas_generated = fission_gas_produced
    fission_gas_released = fission_gas_released
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./cladding_volume]
    type = InternalVolume
    boundary = 'clad_inside_bottom clad_inside_top clad_inside_right'
    execute_on = 'INITIAL LINEAR'
    outputs = csv
  [../]
  [./fuel_volume]
    type = InternalVolume
    boundary = 'bottom_of_bottom_pellet pellet_outer_radial_surface top_of_top_pellet'
    scale_factor = -1 # makes the fuel volume positive (the surface normals make it
negative)
    execute_on = 'INITIAL LINEAR'
    outputs = csv
  [../]
  [./gas_volume]
    type = FunctionValuePostprocessor
    function = gas_volume_function
    execute_on = 'INITIAL LINEAR'
    outputs = csv
  [../]
  #### BURNUP ####
  [./max_burnup]
    type = ElementExtremeValue
    block = pellet
    variable = burnup
    value_type = max
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./avg_burnup]
    type = ElementAverageValue
    block = pellet
    variable = burnup
    execute_on = 'INITIAL TIMESTEP_END'
  [../]
  #### TEMPERATURES ####
  [./T_max_fuel]
    type = ElementExtremeValue
    block = pellet
    variable = T
    value_type = max
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./T_max_cladding]
    type = ElementExtremeValue
    block = clad
    variable = T
    value_type = max
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./T_coolant_out]
    type = ElementExtremeValue
    block = clad
    variable = T_coolant
```

```
    value_type = max
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  #### MECHANICAL ####
  [./max_cladding_hoop_strain]
    type = ElementExtremeValue
    block = clad
    variable = strain_zz
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  #### SWELLING ####
  [./growth_cladding_radial]
    type = NodalMaxValue
    variable = disp_x
    boundary = clad_outside_right
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./growth_fuel_axial]
    type = NodalMaxValue
    block = pellet
    variable = disp_y
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./cdf_max]
    type = ElementExtremeValue
    value_type = max
    variable = cdf_failure
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  #### CONVERGENCE ####
  [./l_its]
    type = NumLinearIterations
    outputs = csv
  [../]
  [./total_l_its]
    type = CumulativeValuePostprocessor
    postprocessor = l_its
  [../]
  [./nl_its]
    type = NumNonlinearIterations
    outputs = csv
  [../]
  [./total_nl_its]
    type = CumulativeValuePostprocessor
    postprocessor = nl_its
  [../]
  [./residual_evals]
    type = NumResidualEvaluations
    outputs = csv
  [../]
  [./timestep_size]
    type = TimestepSize
     outputs = csv
   [../]
   #### PERFORMANCE ####
  [./memory]
    type = MemoryUsage
    mem_type = physical_memory
    mem_units = megabytes
  [../]
  [./total_time]
    type = PerfGraphData
```

```
      section_name = Root
      data_type = TOTAL
    [../]
    [./creep_timestep]
      type = MaterialTimeStepPostprocessor
      block = pellet
    [../]
    #### MANUAL SWELLING ####
    [./solid_swelling]
      type = ElementAverageValue
      block = pellet
      variable = solid_swelling
      outputs = csv
    [../]
    [./gas_swelling]
      type = ElementAverageValue
      block = pellet
      variable = gas_swelling
      outputs = csv
    [../]
    [./porosity]
      type = ElementAverageValue
      block = pellet
      variable = porosity
      outputs = csv
    [../]
  []

  [VectorPostprocessors]
    [./clad_disp]
      type = SideValueSampler
      variable = disp_x
      boundary = clad_outside_right
      sort_by = y
    [../]
  []

  [Outputs]
    color = false
    [./exodus]
      type = Exodus
      sync_times = '0 3600 1e6 2e6 3e6 4e6 6e6 7e6 8e6
                    8203212 8206812 13814423 13818023 14428975 14432575 21312419
                    21316019 25596874 25600474 26261755 26265355 32714598 32718198 32721798
                    32725398 32728998 32896765 32900365 39574695 39578295 42194062 42197662
                    43820808 43824408 43895709 43899309 44401212 44404812 47385472 47389072
                    48198548 48202148 48205748 48209348 48212948 52079977 52083577 53874489
                    53878089 62125235 62128835 62256058 62259658 62620357 62623957 64516928
                    64520528 64766586 64770186 67535546 67539146 72155534 72159134 72185697
                    72189297 76833647 76837247 77340548 77344148 77738400 77742000 80444447
                    80448047 80451647 80455247'
      sync_only = true
    [../]
    [./csv]
      type = CSV
      execute_postprocessors_on = 'INITIAL TIMESTEP_END'
      execute_vector_postprocessors_on = FINAL
    [../]
    perf_graph = true
  []
```

APPENDIX B. INPUT FILE SYNTAX WITH POROSITY CLOSURE

INPUT FILE SYNTAX WITH POROSITY CLOSURE

This appendix contains an example of the BISON input file syntax for the non-AD approach with porosity closure.

```
# X430 BENCHMARK PROBLEM
# PIN T654
# Units are in standard SI: J, K, kg, m, Pa, s.

[GlobalParams]
  order = SECOND
  family = LAGRANGE
  elem_type = QUAD8
  density = 15800
  initial_density = 15800
  energy_per_fission = 3.2e-11
  volumetric_locking_correction = false
  displacements = 'disp_x disp_y'
  temperature = T
  initial_X_Zr = 0.226
  initial_X_Pu = 0.160
  X_Zr = 0.226
  X_Pu = 0.160
[]

[Problem]
  type = ReferenceResidualProblem
  coord_type = RZ
  reference_vector = ref
  extra_tag_vectors = ref
  group_variables = 'T disp_x disp_y'
[]

[Mesh]
  [./generated]
    type = SmearedPelletMeshGenerator
    pellet_quantity = 1
    pellet_outer_radius = 2.8410e-03
    pellet_height = 3.4440e-01
    clad_gap_width = 4.3560e-04
    clad_bot_gap_height = 4.0000e-03
    clad_top_gap_height = 3.7725e-01
    clad_thickness = 4.0640e-04
    top_bot_clad_height = 1.5000e-02
    pellet_mesh_density = customize
    clad_mesh_density = customize
    nx_p = 5
    ny_p = 250
    nx_c = 8
    ny_c = 120
    ny_cu = 4
    ny_cl = 4
  [../]
  patch_size = 5
  patch_update_strategy = auto
  partitioner = centroid
  centroid_partitioner_direction = y
[]

[Variables]
  [./T]
    initial_condition = 295
  [../]
[]

```

```
[AuxVariables]
  [./fuel_thermal_strain_magnitude]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_thermal_strain_magnitude]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./fuel_creep_strain_magnitude]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_creep_strain_magnitude]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./fuel_gaseous_strain_magnitude]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./fuel_solid_strain_magnitude]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./gap_conductance]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./element_failed]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./fuel_volumetric_strain]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_hoop_stress]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_hoop_creep_strain]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_hoop_elastic_strain]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./cladding_hoop_total_strain]
    block = clad
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./local_power]
    block = pellet
    order = CONSTANT
```

```
    family = MONOMIAL
  [../]
  [./T_coolant]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./linear_heat_rate]
    block = pellet
    order = CONSTANT
    family = MONOMIAL
  [../]
[]

[Functions]
  [./power_history_function]
    type = PiecewiseLinear
    x =  '0 3600 8203212 8206812 13814423 13818023 14428975 14432575 21312419
          21316019 25596874 25600474 26261755 26265355 32714598 32718198 32721798
          32725398 32728998 32896765 32900365 39574695 39578295 42194062 42197662
          43820808 43824408 43895709 43899309 44401212 44404812 47385472 47389072
          48198548 48202148 48205748 48209348 48212948 52079977 52083577 53874489
          53878089 62125235 62128835 62256058 62259658 62620357 62623957 64516928
          64520528 64766586 64770186 67535546 67539146 72155534 72159134 72185697
          72189297 76833647 76837247 77340548 77344148 77738400 77742000 80444447
          80448047 80451647 80455247'
    y = '0.0 44225.3 44225.3 43106.1 43106.1 41403.6 41403.6 41119.9 41119.9
         38881.4 38881.4 38353.3 38353.3 39472.5 39472.5     0.0     0.0     0.0
         33490.2 33490.2 36863.6 36863.6 37123.7 37123.7 32717.8 32717.8 38534.6
         38534.6 38432.1 38432.1 36784.8 36784.8 36036.0 36036.0     0.0     0.0
         0.0 35153.3 35153.3 35153.3 35153.3 35271.5 35271.5 33663.6 33663.6
         34459.7 34459.7 34640.9 34640.9 34428.1 34428.1 34026.2 34026.2 33624.2
         33624.2 33624.2 33624.2 33718.8 33718.8 34057.7 34057.7 34057.7 34057.7
         34215.3 34215.3     0.0     0.0     0.0'
  [../]
  [./coolant_pressure_function]
    type = ConstantFunction
    value = 347702.6
  [../]
  [./T_coolant_in_function]
    type = PiecewiseLinear
    x = '0   3600   32718198 32721798 32725398 32728998 48202148 48205748
         48209348 48212948 80448047 80451647 80455247'
    y = '295 644.15 644.15   305.00   305.00   644.15   644.15   305.00
         305.00   644.15   644.15   305.00   305.00'
  [../]
  [./axial_peaking_factor_function]
    type = PiecewiseBilinear
    xaxis = 1
    yaxis = 0
    y = '0 32725398 48209348 80455247'
    x = '0.018 0.019 0.0534 0.0879 0.1223 0.1567 0.1912 0.2256 0.2601 0.2945 0.3289 0.3634
0.3734'
    z = '0.0000 0.9111 0.9983 1.0625 1.1053 1.1195 1.1053
         1.0696 1.0054 0.9127 0.8129 0.7059 0.0000
         0.0000 0.9056 0.9864 1.0487 1.1006 1.111 1.1006
         1.0695 1.0175 0.9241 0.8306 0.7164 0.0000
         0.0000 0.8961 0.9845 1.0442 1.0939 1.1138 1.1039
         1.074 1.0144 0.9348 0.8354 0.7061 0.0000
         0.0000 0.8954 0.9752 1.0368 1.0882 1.1087 1.0984
         1.0779 1.0163 0.9445 0.8418 0.7289 0.0000'
  [../]
  [./heat_generation_rate_function]
    type = ParsedFunction
    vars = 'lhr_bar p_factor'
    vals = 'power_history_function axial_peaking_factor_function'
    value = 'lhr_bar * p_factor'
  [../]
```

```
    [./gas_volume_function]
      type = ParsedFunction
      vars = 'v_cladding v_fuel'
      vals = 'cladding_volume fuel_volume'
      value = 'abs(v_cladding) - abs(v_fuel)'
    [../]
    [./coolant_flux_function]
      type = PiecewiseLinear
      x = '0 3600  8203212  8206812 13814423 13818023 14428975 14432575 21312419
           21316019 25596874 25600474 26261755 26265355 32714598 32718198 32885965
           32889565 39563895 39567495 42183262 42186862 43810008 43813608 43884909
           43888509 44390412 44394012 47374672 47378272 48187748 48191348 52058377
           52061977 53852889 53856489 62103635 62107235 62234458 62238058 62598757
           62602357 64495328 64498928 64744986 64748586 67513946 67517546 72133934
           72137534 72164097 72167697 76812047 76815647 77318948 77322548 77716800
           77720400 80422847 80426447 80430047 80516447'
      y = '2699.1 2699.1 2699.1 2724.0 2724.0 2697.2 2697.2 2781.0 2781.0 2721.1
           2721.1 2696.9 2696.9 2785.4 2785.4 2793.7 2793.7 2803.5 2803.5 2814.2
           2814.2 2799.6 2799.6 2840.1 2840.1 2839.6 2839.6 2873.7 2873.7 2855.7
           2855.7 2826.4 2826.4 2826.4 2826.4 2788.4 2788.4 2780.6 2780.6 2771.8
           2771.8 2781.5 2781.5 2817.1 2817.1 2807.4 2807.4 2777.1 2777.1 2777.1
           2777.1 2746.4 2746.4 2765.9 2765.9 2765.9 2765.9 2777.1 2777.1 2777.1
           2777.1 2777.1'
    [../]
    [./sodium_conductivity_function]
      type = ParsedFunction
      vars = 'A      B        C         D'
      vals = '124.67 -0.11381 5.5226e-5 -1.1842e-8'
      value = 'A + B * t + C * t^2 + D * t^3'
    [../]
  []

  [Modules/TensorMechanics/Master]
    add_variables = true
    strain = FINITE
    generate_output = 'stress_xx stress_yy stress_zz vonmises_stress hydrostatic_stress
  creep_strain_zz elastic_strain_zz strain_zz'
    [./fuel_mechanics]
      block = pellet
      eigenstrain_names = 'fuel_thermal_strain fuel_gaseous_strain'
      extra_vector_tags = ref
    [../]
    [./cladding_mechanics]
      block = clad
      eigenstrain_names = 'cladding_thermal_strain cladding_gaseous_strain'
      extra_vector_tags = ref
    [../]
  []

  [Kernels]
    [./gravity]
       type = Gravity
       block = 'pellet clad'
       variable = disp_y
       value = -9.81
       extra_vector_tags = ref
    [../]
    [./heat_conduction_time_derivative]
       type = HeatConductionTimeDerivative
       block = 'pellet clad'
       variable = T
       extra_vector_tags = ref
    [../]
    [./heat_conduction]
       type = HeatConduction
       block = 'pellet clad'
       variable = T
```

```
      extra_vector_tags = ref
    [../]
    [./heat_source]
      type = FissionRateHeatSource
      block = pellet
      variable = T
      fission_rate = fission_rate
      extra_vector_tags = ref
    [../]
  []

  [AuxKernels]
    [./fuel_thermal_strain_magnitude]
      type = RankTwoScalarAux
      block = pellet
      variable = fuel_thermal_strain_magnitude
      rank_two_tensor = fuel_thermal_strain
      scalar_type = EffectiveStrain
      execute_on = TIMESTEP_END
    [../]
    [./cladding_thermal_strain_magnitude]
      type = RankTwoScalarAux
      block = clad
      variable = cladding_thermal_strain_magnitude
      rank_two_tensor = cladding_thermal_strain
      scalar_type = EffectiveStrain
      execute_on = TIMESTEP_END
    [../]
    [./fuel_creep_strain_magnitude]
      type = RankTwoScalarAux
      block = pellet
      variable = fuel_creep_strain_magnitude
      rank_two_tensor = creep_strain
      scalar_type = EffectiveStrain
      execute_on = TIMESTEP_END
    [../]
    [./cladding_creep_strain_magnitude]
      type = RankTwoScalarAux
      block = clad
      variable = cladding_creep_strain_magnitude
      rank_two_tensor = creep_strain
      scalar_type = EffectiveStrain
      execute_on = TIMESTEP_END
    [../]
    [./fuel_gaseous_strain_magnitude]
      type = RankTwoScalarAux
      block = pellet
      variable = fuel_gaseous_strain_magnitude
      rank_two_tensor = fuel_gaseous_strain
      scalar_type = EffectiveStrain
      execute_on = TIMESTEP_END
    [../]
    [./fuel_solid_strain_magnitude]
      type = RankTwoScalarAux
      block = pellet
      variable = fuel_solid_strain_magnitude
      rank_two_tensor = fuel_solid_strain
      scalar_type = EffectiveStrain
      execute_on = TIMESTEP_END
    [../]
    [./gap_conductance]
      type = MaterialRealAux
      variable = gap_conductance
      property = gap_conductance
      boundary = pellet_outer_radial_surface
    [../]
    [./failed_element]
```

```
    type = MaterialRealAux
    variable = element_failed
    property = failed
    boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
  [../]
  [./fuel_volumetric_strain]
    type = RankTwoScalarAux
    block = pellet
    variable = fuel_volumetric_strain
    rank_two_tensor = total_strain
    scalar_type = VolumetricStrain
    execute_on = TIMESTEP_END
  [../]
  [./cladding_hoop_stress]
    type = RankTwoAux
    block = clad
    variable = cladding_hoop_stress
    rank_two_tensor = stress
    index_i = 2
    index_j = 2
    execute_on = TIMESTEP_END
  [../]
  [./cladding_hoop_creep_strain]
    type = RankTwoAux
    block = clad
    variable = cladding_hoop_creep_strain
    rank_two_tensor = creep_strain
    index_i = 2
    index_j = 2
    execute_on = TIMESTEP_END
  [../]
  [./cladding_hoop_elastic_strain]
    type = RankTwoAux
    block = clad
    variable = cladding_hoop_elastic_strain
    rank_two_tensor = elastic_strain
    index_i = 2
    index_j = 2
    execute_on = TIMESTEP_END
  [../]
  [./cladding_hoop_total_strain]
    type = RankTwoAux
    block = clad
    variable = cladding_hoop_total_strain
    rank_two_tensor = total_strain
    index_i = 2
    index_j = 2
    execute_on = TIMESTEP_END
  [../]
  [./local_power]
    type = FunctionAux
    block = pellet
    variable = local_power
    function = axial_peaking_factor_function
  [../]
  [./T_coolant]
    type = MaterialRealAux
    variable = T_coolant
    property = coolant_temperature
    boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
  [../]
  [./linear_heat_rate]
    type = FunctionAux
    block = pellet
    variable = linear_heat_rate
    function = heat_generation_rate_function
    execute_on = 'INITIAL TIMESTEP_END'
```

```
  [../]
[]

[Contact]
  [./pellet_cladding_mechanical]
    master = clad_inside_right
    slave = pellet_outer_radial_surface
    model = frictionless
    formulation = kinematic
    system = constraint
    tangential_tolerance = 1e-3
    normal_smoothing_distance = 0.1
  [../]
[]

[ThermalContact]
  [./thermal_contact]
    type = GapHeatTransfer
    variable = T
    primary = clad_inside_right
    secondary = pellet_outer_radial_surface
    gap_geometry_type = CYLINDER
    gap_conductivity_function = sodium_conductivity_function
    gap_conductivity_function_variable = T
    quadrature = true
    min_gap = 4.3560e-04
  [../]
[]

[BCs]
  [./fix_disp_x_all]
    type = DirichletBC
    variable = disp_x
    value = 0.0
    boundary = centerline
  [../]
  [./fix_disp_y_all]
    type = DirichletBC
    variable = disp_y
    value = 0.0
    boundary = 'clad_outside_bottom bottom_of_bottom_pellet'
  [../]
  [./Pressure]
    [./coolant_pressure]
      function = coolant_pressure_function
      boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
    [../]
  [../]
  [./PlenumPressure]
    [./plenum_pressure]
      boundary = 'clad_inside_bottom clad_inside_right clad_inside_top'
      startup_time = 0
      initial_pressure = 84000
      volume = gas_volume
      material_input = fission_gas_released
      R = 8.3143
      temperature = plenum_temperature
      output = plenum_pressure
    [../]
  [../]
[]

[PlenumTemperature]
  [./plenum_temperature]
    temp = T
    boundary = 'all_pellet_exterior all_clad_interior'
    inner_surfaces = all_pellet_exterior
```

```
    outer_surfaces = all_clad_interior
  [../]
[]

[CoolantChannel]
  [./convective_cladding_surface]
    variable = T
    inlet_temperature = T_coolant_in_function
    inlet_pressure = coolant_pressure_function
    inlet_massflux = coolant_flux_function
    coolant_material = sodium
    rod_diameter = 7.3660e-03
    rod_pitch = 8.7884e-03
    linear_heat_rate = power_history_function
    axial_power_profile = axial_peaking_factor_function
    subchannel_geometry = triangular
    boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
  [../]
[]

[Materials]
  ###### FUEL ######
  [./fission_rate]
    type = UPuZrFissionRate
    block = pellet
    rod_linear_power = power_history_function
    axial_power_profile = axial_peaking_factor_function
    pellet_radius = 2.8410e-03
  [../]
  [./burnup]
    type = UPuZrBurnup
    block = pellet
    outputs = exodus
    output_properties = burnup
  [../]
  [./fuel_density]
    type = Density
    block = pellet
  [../]
  [./fuel_thermal_properties]
    type = ThermalUPuZr
    block = pellet
    spheat_model = savage
    thcond_model = lanl
    porosity = uncollapsed_porosity
  [../]
  [./fuel_gaseous_swelling]
    type = UPuZrGaseousEigenstrain
    block = pellet
    fission_rate = fission_rate
    anisotropic_factor = 0.99
    bubble_number_density = 3.02e+17
    eigenstrain_name = fuel_gaseous_strain
    outputs = exodus
    output_properties = 'gas_swelling porosity'
  [../]
  [./uncollapsed_porosity]
    type = ParsedMaterial
    block = pellet
    f_name = uncollapsed_porosity
    material_property_names = 'porosity burnup'
    function = 'max(0.0, porosity - (1.5 * burnup) / (1 + 1.5 * burnup))'
    outputs = exodus
  [../]
  [./fission_gas_release]
    type = FgrUPuZr
    block = pellet
```

```
    fission_rate = fission_rate
    fractional_fgr_initial = 0.512
    fractional_fgr_post = 0.785
    porosity = uncollapsed_porosity
  [../]
  [./fuel_solid_swelling]
    type = BurnupDependentEigenstrain
    block = pellet
    eigenstrain_name = fuel_solid_strain
    swelling_name = solid_swelling
    outputs = exodus
    output_properties = solid_swelling
  [../]
  [./fuel_thermal_expansion]
    type = UPuZrThermalExpansionEigenstrain
    block = pellet
    stress_free_temperature = 295
    eigenstrain_name = fuel_thermal_strain
    outputs = exodus
  [../]
  [./fuel_elasticity_tensor]
    type = UPuZrElasticityTensor
    block = pellet
    porosity = uncollapsed_porosity
  [../]
  [./fuel_creep]
    type = UPuZrCreepUpdate
    block = pellet
    porosity = uncollapsed_porosity
    max_inelastic_increment = 1e-2
  [../]
  [./fuel_elastic_stress]
    type = ComputeMultipleInelasticStress
    block = pellet
    inelastic_models = fuel_creep
  [../]
  ###### CLADDING ######
  [./fast_neutron_flux]
    type = UPuZrFastNeutronFlux
    pellet_radius = 2.8410e-03
    axial_power_profile = axial_peaking_factor_function
    rod_linear_power = power_history_function
    calculate_fluence = true
    outputs = exodus
    output_properties = fast_neutron_flux
  [../]
  [./cladding_density]
    type = Density
    block = clad
    density = 7771
  [../]
  [./cladding_thermal_properties]
    type = ThermalHT9
    block = clad
  [../]
  [./cladding_gaseous_swelling]
    type = HT9VolumetricSwellingEigenstrain
    block = clad
    fast_neutron_flux = fast_neutron_flux
    fast_neutron_fluence = fast_neutron_fluence
    eigenstrain_name = cladding_gaseous_strain
  [../]
  [./cladding_thermal_expansion]
    type = HT9ThermalExpansionEigenstrain
    block = clad
    eigenstrain_name = cladding_thermal_strain
    stress_free_temperature = 295
```

```
    [../]
    [./cladding_elasticity_tensor]
      type = HT9ElasticityTensor
      block = clad
    [../]
    [./cladding_creep]
      type = HT9CreepUpdate
      block = clad
      max_inelastic_increment = 1e-2
    [../]
    [./cladding_elastic_stress]
      type = ComputeMultipleInelasticStress
      block = clad
      inelastic_models = cladding_creep
    [../]
    [./cladding_failure]
      type = FailureCladHT9
      method = cdf_long
      hoop_stress = stress_zz
      boundary = 'clad_outside_bottom clad_outside_right clad_outside_top'
      outputs = exodus
      output_properties = cdf_failure
    [../]
  []

  [Preconditioning]
    [./SMP]
      type = SMP
      full = true
    [../]
  []

  [Executioner]
    type = Transient
    solve_type = PJFNK
    petsc_options = '-snes_ksp_ew'
    petsc_options_iname = '-pc_type -pc_factor_mat_solver_package -ksp_gmres_restart'
    petsc_options_value = 'lu       superlu_dist                  31'
    line_search = NONE
    l_max_its = 30
    l_tol = 1e-2
    nl_max_its = 30
    nl_rel_tol = 5e-4
    nl_abs_tol = 1e-7
    end_time = 80455247
    dtmin = 1e-2
    dtmax = 1e6
    start_time = 0
    verbose = true
    [./Quadrature]
      order = FIFTH
      side_order = SEVENTH
    [../]
    [./TimeStepper]
      type = IterationAdaptiveDT
      dt = 100
      optimal_iterations = 10
      iteration_window = 4
      growth_factor = 1.25
      cutback_factor = 0.512
      linear_iteration_ratio = 100
      force_step_every_function_point = true
      timestep_limiting_function = power_history_function
      timestep_limiting_postprocessor = creep_timestep
    [../]
  []
```

```
[Postprocessors]
  #### FISSION GAS #### (needed for simulation to run)
  [./fission_gas_produced]
    type = ElementIntegralFisGasProduce
    block = pellet
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./fission_gas_released]
    type = ElementIntegralFisGasRelease
    block = pellet
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./fission_gas_percent]
    type = FGRPercent
    fission_gas_generated = fission_gas_produced
    fission_gas_released = fission_gas_released
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./cladding_volume]
    type = InternalVolume
    boundary = 'clad_inside_bottom clad_inside_top clad_inside_right'
    execute_on = 'INITIAL LINEAR'
    outputs = csv
  [../]
  [./fuel_volume]
    type = InternalVolume
    boundary = 'bottom_of_bottom_pellet pellet_outer_radial_surface top_of_top_pellet'
    scale_factor = -1 # makes the fuel volume positive (the surface normals make it
negative)
    execute_on = 'INITIAL LINEAR'
    outputs = csv
  [../]
  [./gas_volume]
    type = FunctionValuePostprocessor
    function = gas_volume_function
    execute_on = 'INITIAL LINEAR'
    outputs = csv
  [../]
  #### BURNUP ####
  [./max_burnup]
    type = ElementExtremeValue
    block = pellet
    variable = burnup
    value_type = max
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./avg_burnup]
    type = ElementAverageValue
    block = pellet
    variable = burnup
    execute_on = 'INITIAL TIMESTEP_END'
  [../]
  #### TEMPERATURES ####
  [./T_max_fuel]
    type = ElementExtremeValue
    block = pellet
    variable = T
    value_type = max
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./T_max_cladding]
    type = ElementExtremeValue
```

```
    block = clad
    variable = T
    value_type = max
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./T_coolant_out]
    type = ElementExtremeValue
    block = clad
    variable = T_coolant
    value_type = max
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  #### MECHANICAL ####
  [./max_cladding_hoop_strain]
    type = ElementExtremeValue
    block = clad
    variable = strain_zz
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  #### SWELLING ####
  [./growth_cladding_radial]
    type = NodalMaxValue
    variable = disp_x
    boundary = clad_outside_right
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./growth_fuel_axial]
    type = NodalMaxValue
    block = pellet
    variable = disp_y
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  [./cdf_max]
    type = ElementExtremeValue
    value_type = max
    variable = cdf_failure
    execute_on = 'INITIAL TIMESTEP_END'
    outputs = csv
  [../]
  #### CONVERGENCE ####
  [./l_its]
    type = NumLinearIterations
    outputs = csv
  [../]
  [./total_l_its]
    type = CumulativeValuePostprocessor
    postprocessor = l_its
  [../]
  [./nl_its]
    type = NumNonlinearIterations
    outputs = csv
  [../]
  [./total_nl_its]
    type = CumulativeValuePostprocessor
    postprocessor = nl_its
  [../]
  [./residual_evals]
    type = NumResidualEvaluations
    outputs = csv
  [../]
  [./timestep_size]
    type = TimestepSize
```

```
      outputs = csv
    [../]
    #### PERFORMANCE ####
    [./memory]
      type = MemoryUsage
      mem_type = physical_memory
      mem_units = megabytes
    [../]
    [./total_time]
      type = PerfGraphData
      section_name = Root
      data_type = TOTAL
    [../]
    [./creep_timestep]
      type = MaterialTimeStepPostprocessor
      block = pellet
    [../]
    #### MANUAL SWELLING ####
    [./solid_swelling]
      type = ElementAverageValue
      block = pellet
      variable = solid_swelling
      outputs = csv
    [../]
    [./gas_swelling]
      type = ElementAverageValue
      block = pellet
      variable = gas_swelling
      outputs = csv
    [../]
    [./porosity]
      type = ElementAverageValue
      block = pellet
      variable = porosity
      outputs = csv
    [../]
    [./uncollapsed_porosity]
      type = ElementAverageValue
      block = pellet
      variable = uncollapsed_porosity
      outputs = csv
    [../]
  []

  [VectorPostprocessors]
    [./clad_disp]
      type = SideValueSampler
      variable = disp_x
      boundary = clad_outside_right
      sort_by = y
    [../]
  []

  [Outputs]
    color = false
    [./exodus]
      type = Exodus
      sync_times = '0 3600 1e6 2e6 3e6 4e6 6e6 7e6 8e6
                    8203212 8206812 13814423 13818023 14428975 14432575 21312419
                    21316019 25596874 25600474 26261755 26265355 32714598 32718198 32721798
                    32725398 32728998 32896765 32900365 39574695 39578295 42194062 42197662
                    43820808 43824408 43895709 43899309 44401212 44404812 47385472 47389072
                    48198548 48202148 48205748 48209348 48212948 52079977 52083577 53874489
                    53878089 62125235 62128835 62256058 62259658 62620357 62623957 64516928
                    64520528 64766586 64770186 67535546 67539146 72155534 72159134 72185697
                    72189297 76833647 76837247 77340548 77344148 77738400 77742000 80444447
                    80448047 80451647 80455247'
```

```
    sync_only = true
  [../]
  [./csv]
    type = CSV
    execute_postprocessors_on = 'INITIAL TIMESTEP_END'
    execute_vector_postprocessors_on = FINAL
  [../]
  perf_graph = true
[]
```