INL/EXT-19-55958

September 2019

Summary of Grizzly Development for Advanced Reactor Structural Materials

B. W. Spencer, INL
S. A. Pitts, INL
L. Liu, Utah State University, Temple University
M. Vyas, Utah State University
W. Jiang, INL
A. Casagranda, INL
D. J. McDowell, INL

INL/EXT-19-55958

Summary of Grizzly Development for Advanced Reactor Structural Materials

B. W. Spencer[1]
S. A. Pitts[1]
L. Liu[2]
M. Vyas[3]
W. Jiang[1]
A. Casagranda[1]
D. J. McDowell[1]

September 2019

[1]Computational Mechanics and Materials, Idaho National Laboratory
[2]Utah State University, currently at Temple University
[3]Utah State University

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under U.S. Department of Energy-Idaho Operations Office
Contract DE-AC07-99ID13727

Contents

1 Introduction

This summary report contains an overview of work performed during Fiscal Year 2019 under the Nuclear Engineering Advanced Modeling and Simulation (NEAMS) Fuels Product Line work package entitled "MS-19IN020106 - GRIZZLY - INL".

Grizzly is a code based on the MOOSE framework that has been under development for several years funded by the US Department of Energy's Light Water Reactor Sustainability (LWRS) program. Prior to 2019, Grizzly development was focused on development of capabilities needed to simulate the effects of aging mechanisms in light water reactor (LWR) components. The goal of this LWR-focused work has been to develop tools that can be used to assess the ability of existing LWRs to perform safety in long term operation (LTO) scenarios. This LWR-focused work has primarily focused on the effects of reactor pressure vessel (RPV) embrittlement and aging of reinforced concrete structures.

Grizzly development to support LWR degradation needs is still ongoing, but starting in 2019, Grizzly's capabilities are being expanded to also meet growing needs for a tool to address structural component integrity and degradation in advanced reactors. A simulation capability for advanced reactor components has many of the same basic requirements that one for LWRs does, such as the ability to solve coupled systems of partial differential equations that arise from the fact that many of the problems of interest involve multiple coupled physics. The materials issues also inherently involve multiple length scales, with the response at the engineering scale being strongly affected by mechanisms occurring at atomistic and mesoscopic scales.

This report documents the NEAMS-funded work performed during Fiscal Year 2019 to address advanced reactor structural materials needs. This includes work in the following major areas:

- **XFEM Crack Propagation for 3D Structural Component Models** This builds on the existing extended finite element method (XFEM) capability in MOOSE to allow for propagating cracks in 3D. Cracking, including instantaneous fracture and subcritical crack growth is a failure mode that must be considered for a wide variety of structural components and materials, so this has broad applicability for advanced reactor applications.

- **Integration of a Reduced Order Model for Metals in Grizzly** This is the result of a collaboration with Los Alamos National Laboratory, where reduced-order models have been developed to efficiently incorporate mesoscale-based models of the inelastic response of metals in high-temperature environments such as those that would be experienced in advanced reactors, which generally operate at significantly higher temperatures than LWRs. A constitutive modeling framework that allows the use of these models in MOOSE-based codes has been developed.

- **Develop, Commit, and Test Models for High Temperature Creep and Plasticity in Metals** This is the result of a collaboration with Argonne National Laboratory, where a library of engineering constitutive models targeted at the response of metals under high temperature environments has been under development. This work incorporates this library in Grizzly and BlackBear (the open-source code that contains non nuclear-specific components of Grizzly).

The XFEM development work, which is a Fiscal Year 2019 milestone, is described in detail in Chapter 2. The two activities to incorporate high temperature inelastic models in Grizzly are described in detail in Chapter 3. Finally, Chapter 4 summarizes this work, and outlines future directions for this work.

2 XFEM Crack Propagation for 3D Structural Component Models

A variety of engineering problems require accurately representing propagating discrete cracks. The extended finite element method (XFEM) is a technique that allows for arbitrary mesh-independent cracks to be represented in a finite element model, and has been implemented in INL's open-source MOOSE framework [1]. XFEM development in MOOSE was originally motivated by the need to model discrete fracture in ceramic nuclear fuel [2, 3], and used within the Bison application. There is also a significant need for modeling fracture in structural nuclear power plant components, and those needs have largely driven further development of XFEM in MOOSE, particularly for 3D fracture [4, 5]. The XFEM implementation in MOOSE is particularly powerful because it inherently represents discontinuities caused by the presence of the crack in all solution fields in a multiphysics analysis.

Prior to completion of the present work, the MOOSE XFEM implementation was only able to simulate crack propagation in 2D, and the 3D modeling was limited to stationary crack analysis without the capability of crack propagation. Although it was limited to stationary cracks, the existing 3D XFEM capability was quite useful for Grizzly applications, and has been used extensively for evaluating stress intensity factors, including those for mixed modes, in flaws in LWR RPVs to assess the likelihood of crack initiation during a transient event.

Cracking is an important failure mode of interest for a variety of structural components in both LWRs and advanced reactor concepts, and there are a number of applications for which having the ability to model arbitrary propagation of cracks in 3D is essential for assessing the safety of such components. These include subcritical growth mechanisms such as fatigue and stress corrosion cracking, which are of a concern in components such as boiling water reactor (BWR) core shrouds and steam generator tubes. Many advanced reactors incorporate graphite components, in which fracture is an important mechanism that must be considered. Although the present work is primarily motivated by applications to structural components, modeling propagating cracks in 3D for nuclear fuel applications is also of significant interest.

To enable the 3D crack propagation capability in MOOSE-based codes, this work primarily modified three components of the XFEM module including (1) the element fragmentation algorithm (EFA) which splits regular elements into element fragments at the time of cracking; (2) a previously-developed technique that defines cracks in a 3D domain using 2D surface meshes; and (3) domain integrals used to compute the J integral and stress intensity factors, which are used as criteria for crack propagation. Major changes made to these components include:

- The mesh-based cutter object has been modified to allow it to grow with time.
- The EFA code has been improved to allow correct element fragmentation during 3D propagation.
- The 3D EFA code has been extensively verified by adding more than 20 test cases.
- The system to compute fracture integrals has been refactored, which will allow potentially varying number of front points as the crack grows.

The modifications to the MOOSE XFEM module have been demonstrated using several numerical examples that represent various crack propagation and element cutting conditions. The capability has been further

demonstrated using two other examples that demonstrate the time-dependent growth of a self-similar penny-shaped crack and the burst of a pressurized tube.

2.1 Modifying the 3D Surface Mesh Cutter Object

Cracks in 3D solids generally have arbitrary shapes and may grow into even more complicated shapes as driven by crack propagation laws. Compared with 2D where cracks can be described by multiple connected line segments, the 3D modeling requires an approach that is flexible enough to account for a high level of geometric complications.

Arguably one of the most challenging aspects of XFEM is defining the crack topology. XFEM is very good at representing the discontinuities in the solution field in the presence of a crack, but the crack topology must be provided. Using a topologically 2D surface mesh in 3D space is an ideal option to define crack topology in 3D because its growth is easy to manage — a crack can be propagated simply adding nodes and elements to the mesh. The mesh-based description also makes the crack propagation reasonably arbitrary.

The MOOSE XFEM implementation has an object-oriented architecture, and the classes that define the way that the mesh is cut are an important part of that. A base class defines the interfaces for code that defines how the finite element mesh is traversed by cutting planes (3D) or lines (2D). Multiple cutting classes have been implemented, including cutters based on level set fields, sets of line segments, and simple 3D planes such as rectangles and ellipses.

In addition to these basic 3D planes, a surface mesh based cutter was already developed in MOOSE prior to the present work as demonstrated in Figure 2.1. This example starts with an initially square crack defined by a 2D surface mesh comprised of triangular elements, residing inside a cubic solid object (Figure 2.1a). The crack grows at all boundaries which form the crack front. The direction and speed of crack propagation are governed by prescribed functions in terms of coordinates.

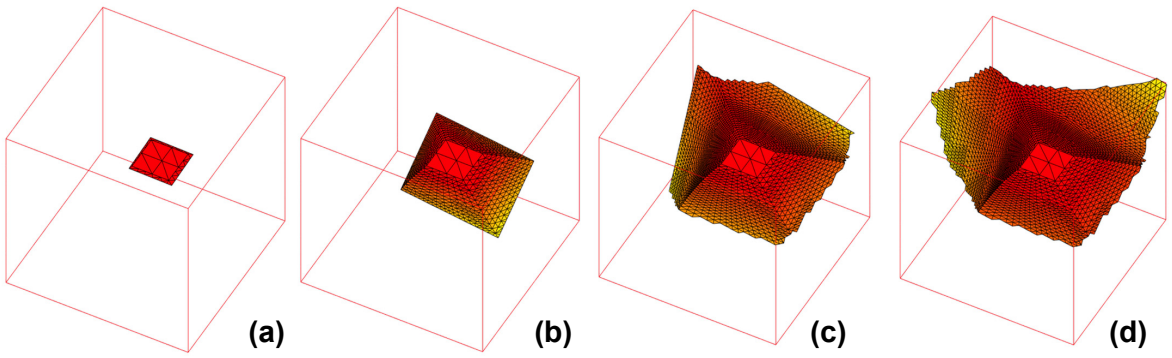


Figure 2.1: Postulated growth of a squared crack inside a 3D cubic domain.

By giving randomized growth functions, the example simulates non-uniform crack growth as shown in Figure 2.1b. As the crack grows to meet geometric boundaries of the solid domain, the surface mesh is made to stop growing. For example, Figure 2.1c shows the state at the point in time when the crack begins to encounter the front, right, and top surfaces of the solid body. The crack stops growing in these locations, while propagation is continued at other parts of the crack front. Figure 2.1d shows the final state where all front points are outside of the solid domain and marked as inactive for growing.

In the version of this cutting code that was merged into the MOOSE code base in 2018, the surface mesh cutter object can in one step grow the crack from the initial to the final state in a single time step. This was an incremental development toward the end goal of an extremely generally capability, and had the purpose

of demonstrating a number of components of this, including (1) insertion of nodes and elements for crack growth, (2) automatic mesh refinement as the crack front becomes longer, and (3) stopping crack growth for points growing outside of the 3D domain. However, it was still lacking the ability to incrementally grow the crack every time step to mimic more physical situations.

The present work modified the mesh-based cutter code to allow crack growth with time. The direction and speed of crack propagation are evaluated at every time steps for front points with an active growing status.

2.2 Improving the 3D Element Fragment Algorithm

The MOOSE XFEM implementation relies on what is known as the phantom node method [6, 7]. This method represents the effects of cracks by replacing the finite elements traversed by a crack with two overlapping elements that each represent the material on one side of the crack, and re-connecting newly-created elements in a manner appropriate to preserve continuity of the solution fields in the uncracked material, and represent the effects of the discontinuity. Figure 2.2 illustrates the high-level flow of the algorithm used to modify the mesh, which is referred to as the element fragment algorithm (EFA).

The bulk of the complexity in a phantom node-based XFEM implementation lies in the EFA. In this algorithm, first it is determined that the element is intersected by a cutting plane. Embedded nodes are then created at intersections of element edges with the cutting plane, and fragments are created for each continuous region within the cut element. Fragments are identified by the set of edges defining their boundaries. Finally, a new child element is created corresponding to each fragment, with new temporary nodes to replace all nodes that are not connected to a fragment. Neighboring fragments are merged if they share a common edge.

Prior to this work, the EFA had been successfully applied to solve problems with 2D stationary, growing, and branching cracks, as well as 3D stationary cracks. However, this algorithm had never been fully developed to handle propagating 3D cracks. When applied to 3D propagating cracks, the EFA code gave incorrect numbers of element fragments, which led to errors in all subsequent calculations. For example, Figure 2.3 shows a simple test that was used to evaluate the EFA code. Element 9 was supposed to be split into two fragments but the code gave only one in the case when the element to the left of it was split in a previous time step. The root causes of these issues, which were related to the way that previously-cut elements were handled when new cuts were added, were identified and addressed. The code now gives the correct set of element fragments for growing cracks in 3D.

2.3 Testing 3D Element Fragment Algorithm

Wherever possible, unit tests are employed in the MOOSE framework to test the various components that make up a capability as directly as possible. The EFA code has had a number of unit tests since its original development, which are contained in the `ElementFragmentAlgorithmTest` class. These tests directly run the EFA and compare the sets of permanent nodes and embedded nodes generated by the algorithm with accepted results. Over 20 new test cases have been created in this work to fully test the EFA in 3D, including cases with propagating cracks. Figure 2.4 shows the mesh and crack for each of these tests. These test examples represent a wide range of possible cutting scenarios. Most of them are based on a simple $2 \times 2 \times 2$ mesh. The last two tests are based on the example shown in Figure 2.3. The EFA code has passed all of these tests.

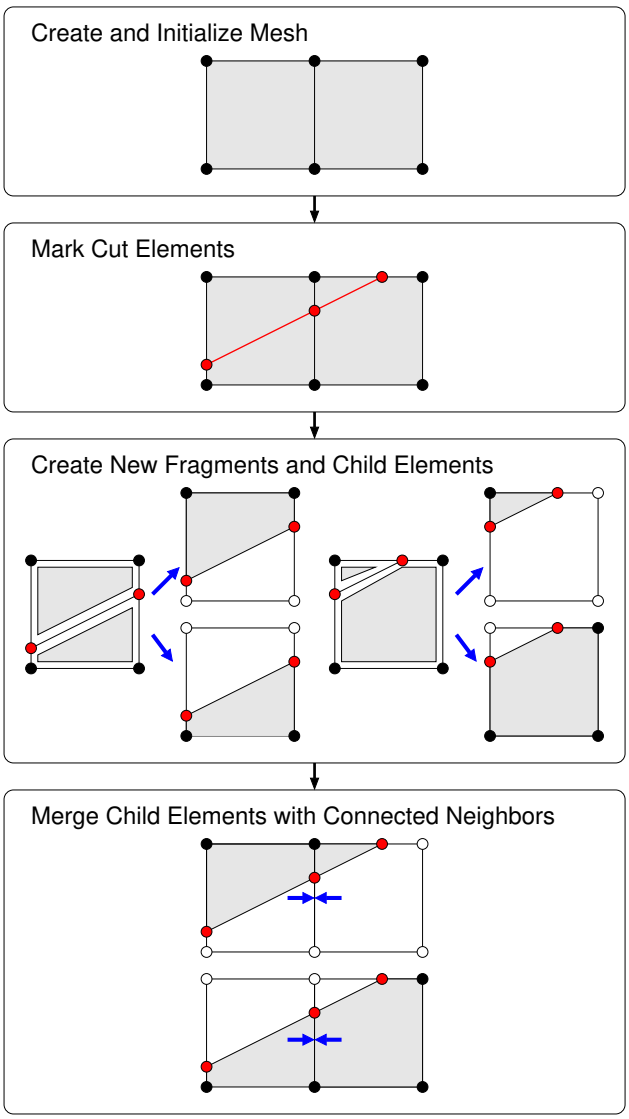


Figure 2.2: Flow chart of the process of the element fragment algorithm for element splitting and reconnection used in MOOSE's phantom-node-based XFEM

2.4 Refactoring Fracture Integral Computation Code

Fracture integrals are evaluated at discrete points that are sampled to represent the crack front. These are widely used in empirical fracture mechanics criteria to determine whether or not a crack propagates, as well as the speed of propagation for subcritical cracks. The original implementation of fracture integrals in MOOSE uses a set of `PostProcessor` objects to compute these integrals along the fronts of 3D cracks. The crack front is discretized into a set of points, and a `PostProcessor` object is created corresponding to each of these points for computing the fracture integrals.

The problem with this approach is that it only works for a fixed number of crack front points. As a crack grows, the size of the crack front will tend to increase, and it is often desirable to increase the number of points used to discretize the crack front. The `VectorPostprocessor` system in MOOSE (which was developed after the original implementation of fracture integrals) is much better suited to this scenario, because a vector of values can be computed, each corresponding to a different point on the crack front, and

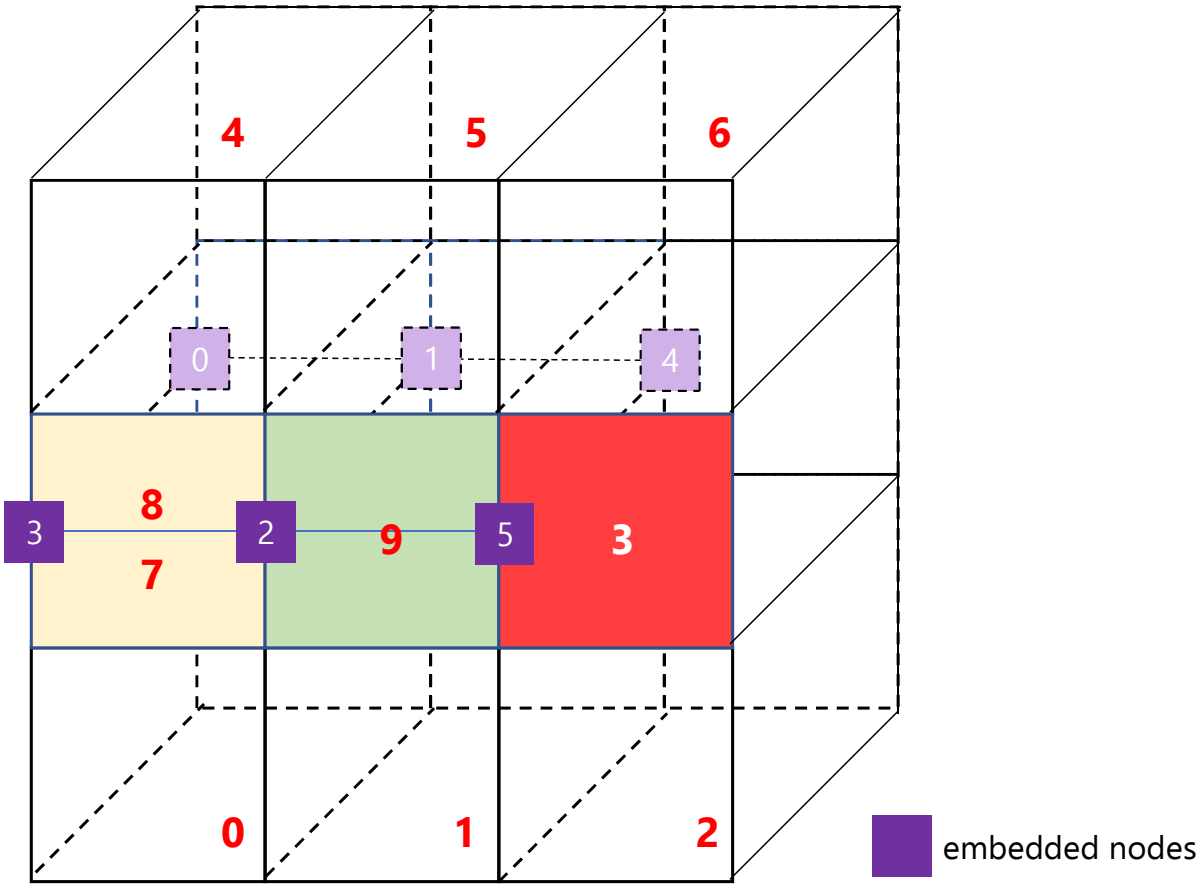


Figure 2.3: A simple example with a $3 \times 3 \times 1$ mesh for evaluating the 3D EFA code. A crack initially splits the yellow element, and then grows to split the green element at the second time step. The yellow element is fully cut and split into two new elements. The green element is cut with the crack tip on its edge. The red element is not cut but has the crack tip on its edge. Purple squares show embedded nodes.

the size of the vector can change over time to reflect an evolving set of crack front points.

An extensive refactoring of the fracture integral system, which consists of the `Postprocessor` objects, some other utility classes, and an `Action` (`DomainIntegralAction`) that sets them up, has been performed. All of the `Postprocessor` classes have been converted to equivalent `VectorPostprocessor` classes, and `DomainIntegralAction` has been revised to set up these new objects. It also creates a set of `Postprocessor` objects that report the components of the integral at the various crack front points, with the intent to provide the same set of outputs to the user for backward compatibility. The same suite of test problems that ran previously runs correctly with the refactored code.

One of the simple mesh cutter objects has been set up to obtain fracture integral results from `VectorPostprocessors` to allow for it to physically drive crack propagation using various fracture mechanics criteria.

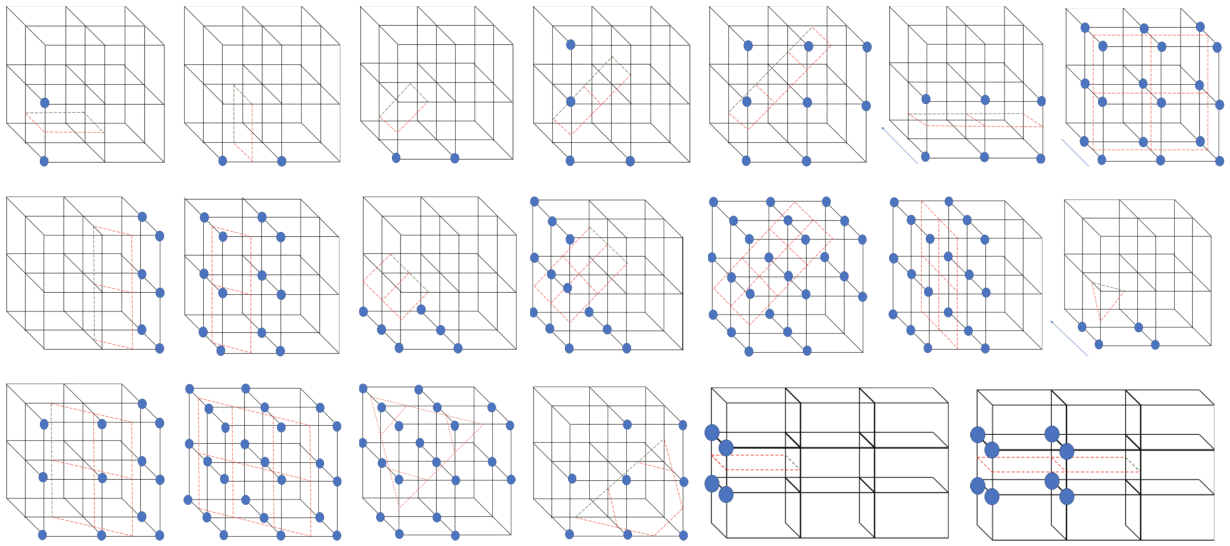


Figure 2.4: 3D test cases of the element fragment algorithm. Dotted lines form the crack surface, where green dotted lines show the crack front. Blue circles show the locations of newly created permanent nodes.

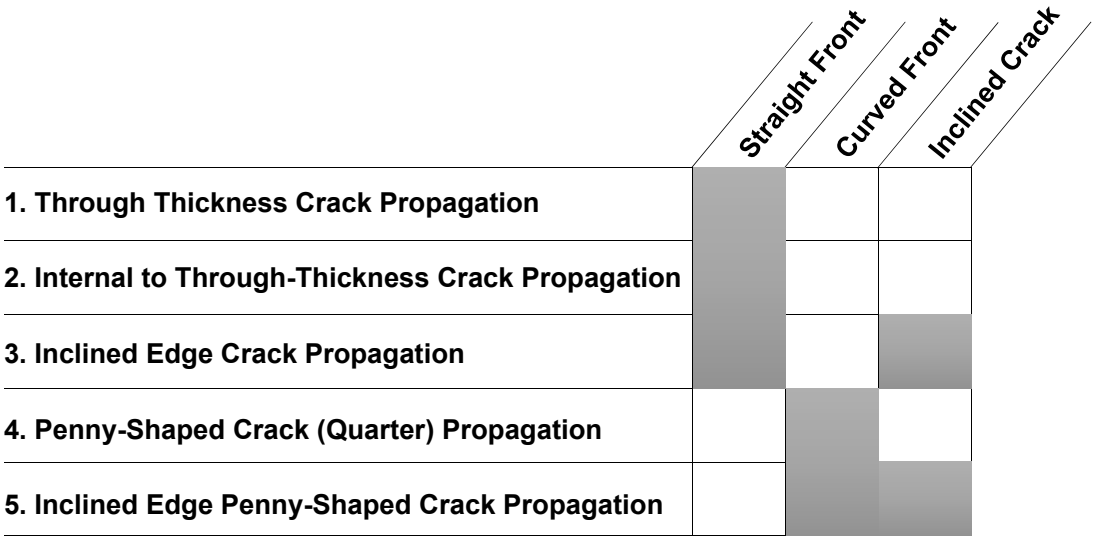


Figure 2.5: Summary of numerical examples for testing the 3D crack propagation code.

2.5 Testing 3D Crack Propagation with Numerical Examples

Five numerical examples have been created to test the 3D crack propagation code as illustrated in Figure 2.5. They challenge the code with distinct crack propagation and cutting scenarios:

- Test 1 has a through-thickness edge crack that propagates uniformly along one direction. All cracked elements are cut in halves (Figure 2.6a).
- Test 2 has an internal edge crack that features a multilinear crack front. The crack expands in all directions. The top and bottom segments of the crack front grow to hit boundaries of the solid body,

causing the transition into a through-thickness crack (Figure 2.6b).

- Test 3 is similar to Test 1 but features an inclined crack. This challenges the EFA code a bit more as some elements may be cut into triangular and pentagonal prisms (Figure 2.6c).
- Test 4 has a penny shaped crack with a curved crack front (Figure 2.6d).
- Test 5 is the most complicated test out of the five, featuring an inclined crack with a curved crack front. The EFA code is challenged the most here as the crack propagation may lead to a variety of cutting scenarios (Figure 2.6e).

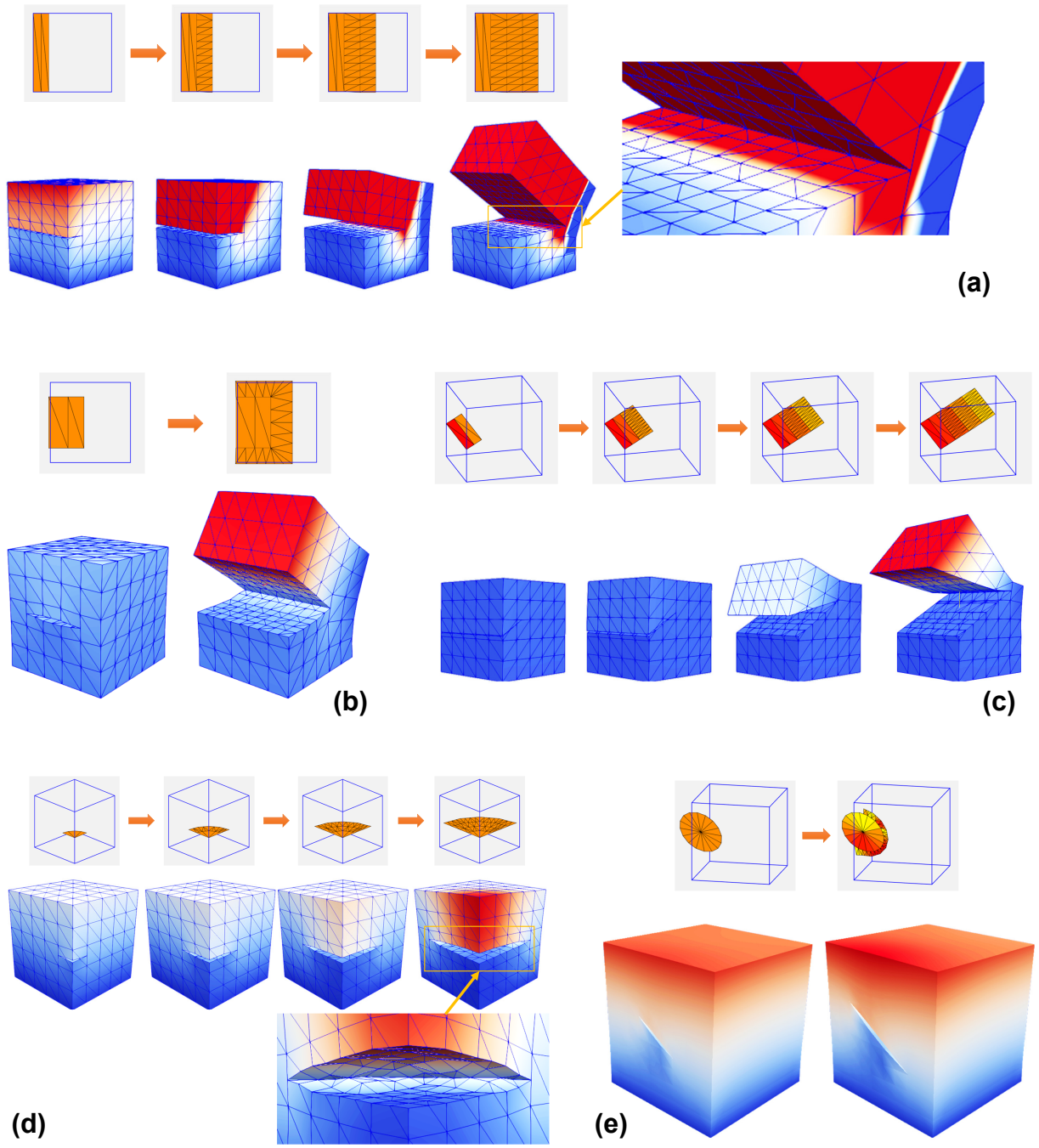


Figure 2.6: Five numerical examples: (a) through-thickness crack propagation; (b) internal to through-thickness crack propagation; (c) inclined edge crack propagation; (d) penny-shaped crack (quarter) propagation; and (e) inclined penny-shaped crack (half) propagation. In all of these examples, the crack is embedded in a cubic solid body and forced to propagate at a constant speed. The bottom surface of the solid is fully constrained, while distributed tensile force is applied on the top surface. The top of each subfigure shows growth of the cutter mesh (or crack propagation). The bottom of each subfigure shows the deformed solid structure as the crack propagates. The color indicates the magnitude of displacement with red being the largest and blue being the smallest.

The code behaves as expected for all cases in Figure 2.6. The deformation of the structures agrees completely with physical expectations. In addition to the global deformed structures, Figure 2.6a and Figure 2.6b further provide zoomed-in views of the deformation near the crack tip. Both plots show well connected elements with reasonable deformation, demonstrating effectiveness of the EFA in these 3D simulations. In addition to these simple tests, another example is made based on Test 4. The difference is that the penny shaped crack in the new example propagates with a time-dependent speed defined by $3.5 - 0.5 \times t$. Results are similar to that shown in Figure 2.6d, with increasingly slower crack propagation which can be seen in the cutter mesh evolution (Figure 2.7).

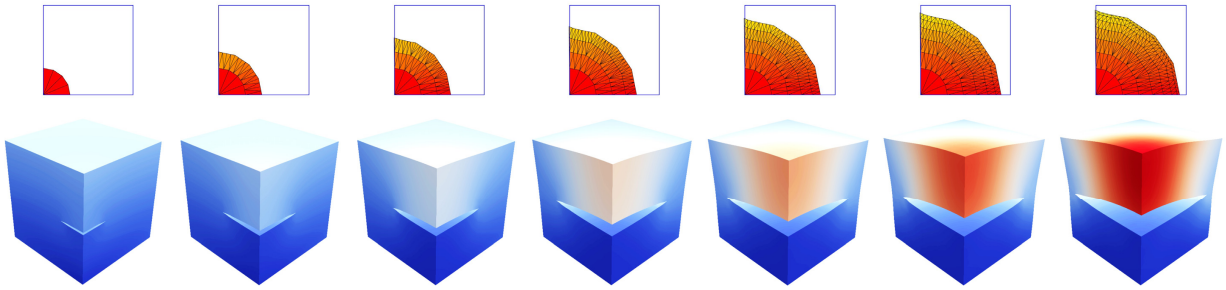


Figure 2.7: Time-dependent propagation of a self-similar penny shaped crack.

2.6 Application to Model Cladding Burst Behavior

To demonstrate this capability on a more realistic problem, the newly developed 3D crack propagation capability is then applied to model propagation of a crack during burst of a section of cladding representative of what would be seen in nuclear fuel. Figure 2.8a shows an experimental image of a cladding tube that has failed during a burst test. A vertical through-thickness crack is apparent in addition to the ballooning deformation of the tube as a result of the internal loading.

To simulate the cracking process, a finite element model has been created with an inner radius of 0.418 cm, an outer radius of 0.474 cm, and a height of 2 cm (Figure 2.8b). The bottom and top surfaces are constrained fully, while pressure is applied on the inner surface. XFEM is used to model crack propagation, although it is important to note that crack propagation is fully prescribed, and is not based on physical behavior of the system.

Initially, a through-thickness crack of 0.1 cm is placed in the middle of the tube aligned with the length direction. The crack is set to propagate at both ends by one element every time step. Simulation results are shown in Figure 2.8c in the form of deformed structures at selected time steps. Both the ballooning deformation and crack propagation as seen in the experiment are captured in these simulation results.

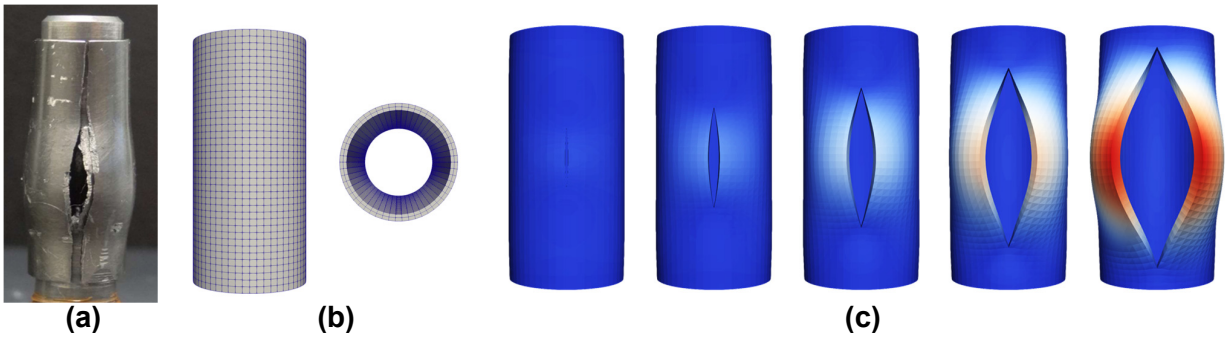


Figure 2.8: A simple model of cladding burst behavior with crack growth represented by a prescribed growing cutting plane with XFEM in MOOSE: (a) cladding fracture revealed in a burst experiment from the literature; (b) side and top views of the finite element model; and (c) crack evolution visualized in the deformed cladding with color indicating the magnitude of displacement.

2.7 Limitations of Current Capability and Future Work

The examples presented here have demonstrated that the XFEM capability now has the foundational components needed to model crack propagation in 3D, and this has been demonstrated on a variety of problems with prescribed growth of simple cutting planes. The mesh-based geometric cutter has also been demonstrated to be able to handle complex patterns of prescribed incremental growth.

Several major obstacles to a completely general 3D capability have been addressed by this work, but there are still some limitations that need to be addressed:

- Fracture integrals currently work for planar cracks only.
- The mesh cutter object is limited to planar initial cracks but has the capability to grow into non-planar configurations.
- The mesh cutter object is not fully integrated with fracture integral vector postprocessors.
- Crack initiation (i.e. insertion of a mesh cutter when a criterion is met) has not been implemented.
- Fracture integral VectorPostprocessors need more work to allow for changing the number of crack front points.

A completely general capability for non-planar cracking simulation will require additional work to address the limitations listed above. For example, more work needs to be done on the fracture integrals to allow them to work for non-planar cracks, and the mesh cutter object needs some modifications to allow non-planar initial cracks. Full integration of the mesh cutter and the fracture integral VPPs will require the number of crack front points to be adaptive as the crack grows. Simulation of subcritical crack growth will require the implementation of various fracture mechanics criteria using the fracture integral results as the input. In general, there are no foreseen major technical obstacles to completing this work, as the existing capabilities provide a very good foundation for these future developments.

3 Incorporation of High Temperature Metal Inelasticity Models in Grizzly

During Fiscal Year 2019, NEAMS funding supported the integration of two fundamentally very different types of material model for the inelastic response of metals under high temperature conditions into Grizzly. Because the temperatures at which advanced reactors are expected to operate is much higher than the normal operating range of light water reactors, new constitutive models for the inelastic material behavior of structural metals are required. In this chapter we discuss two different approaches that can be used to address this need for improved high temperature constitutive models. This effort allows Grizzly to leverage constitutive model development effort performed at two other national laboratories.

We describe first the integration of a reduced order model (ROM) based on a mesoscale model within Grizzly as a J2 constitutive model. The integrated ROM tracks a subset of physical microstructure components and, through the use of Legendre polynomials, uses these microstructure components as well as the stress state and temperature to calculate a strain measure. This ROM was developed during Fiscal Year 2019 by Los Alamos National Laboratory (LANL) and is representative of an emerging modeling capability.

Second we describe the integration of a mature constitutive model library with Grizzly. The Nuclear Engineering material Model Library (NEML) is a set of modular engineering scale constitutive models developed by Argonne National Laboratory (ANL). The integration of NEML enables the immediate modeling of several engineering scale materials with Grizzly.

In this chapter we describe the integration of each of these constitutive model approaches within Grizzly. To demonstrate the successful integration of the constitutive models we present a set of demonstration problems before concluding with some remarks on proposed future work.

3.1 Integration of a Reduced Order Model for Metals in Grizzly

A criticism often raised of the traditional and empirical models used on the engineering scale is that these models do not adequately account for the microstructure evolution which governs macroscale engineering material performance. Conversely, physically based microstructure models, which track populations of different microstructure features are rightly faulted for being too computationally expensive for use in practical engineering simulations. The need for a robust and reliable method to bridge these two modeling scales has long been acknowledged.

Reduced order models (ROMs) offer a reliable method of transforming computationally expensive microstructure informed models into a condensed and efficient model. As such, ROMs are capable of predicting the material performance of an engineering scale geometry in a reasonable amount of simulation time. We have adapted a ROM, based on a Visco-Plastic Self Consistent (VPSC) model [8] and developed by Los Alamos National Laboratory (LANL), for integration with Grizzly through the MOOSE framework.

3.1.1 Background

VPSC models describe a polycrystalline geometry as a collection of grains, each of which is embedded within a homogeneous medium composed of all the other crystals in the metal sample, and solve the stress and strain state within the crystal through the application of the Eshelby tensor. Similar to crystal plasticity models [9] VPSC models track the evolution of microstructure features such as dislocation densities and defects. The dislocation densities often include mobile or glissile dislocations, which are free to move under a sufficient applied driving force, and immobile or locked dislocations, which are trapped on other dislocations or defects [10].

The ROM we have integrated into Grizzly over this year was developed through the use of many VPSC simulations for high temperature and high pressure operating conditions. This ROM is a function of five or six physical parameters: the mobile dislocation density ($1/m^2$), the immobile dislocation density ($1/m^2$), a stress measure (MPa), a strain measure, the temperature (K), and, optionally, the radiation dose received by the material (dpa). Because of the foundation in the physically-based microstructure VPSC model, this ROM represents a successful balance between the computationally efficiency required for large scale simulations and the retention of microstructure evolution tracking needed to predict material behavior in complex environments.

We have integrated two ROMs into Grizzly through the MOOSE framework:

1. a preliminary ROM for HT9 within a thin-walled pressure vessel stress assumption, and
2. a finalized ROM for stainless steel 316H within the J2 yield surface assumption.

Since the integration for both of these ROMs was similar, we focus our discussion in the report below on the stainless steel 316H ROM integration.

3.1.2 Implementation of ROM as a Constitutive Model

A signification consideration in our development of the stainless steel 316H ROM integration was to maintain consistency between two different use cases. The first use case mirrors the code structure used during the ROM development: a standalone code which can be run simply and quickly for a single time step at a single representative material point. The second use case aligns with the integration of the ROM within the MOOSE framework: a set of classes containing a constitutive model that can be called by the MOOSE framework to evaluate the material response at a large number of element quadrature points for many time steps.

To meet this design requirements, we implemented a set of independent classes to contain the constitutive ROM code, shown in Figure 3.1. These independent classes consist of a base class and an inheriting child class. The base class contains the equations for the Legendre polynomials and the multiplication of these polynomials by the ROM coefficients. These operation are common to all ROMS of this type. The child class contains all of the material-specific information for each ROM including input parameter normalization constants and the material specific ROM Legendre polynomial coefficients.

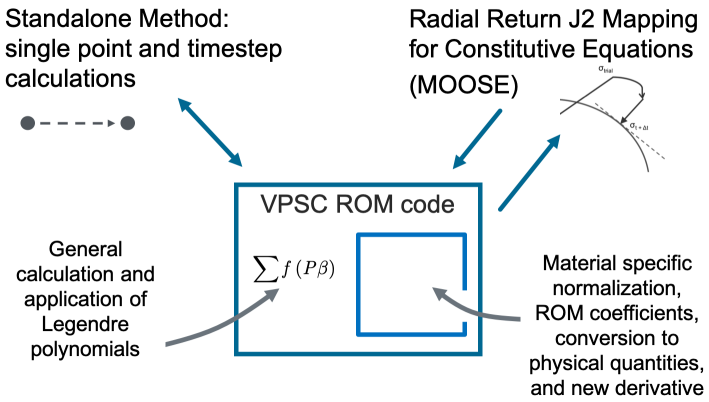


Figure 3.1: Schematic of the integrated ROM code to enable consistent use by both a standalone code and within the MOOSE framework J2 radial return mapping algorithm as a constitutive model.

We connect the ROM constitutive classes to the MOOSE framework through the previously developed radial return mapping algorithm. This was originally developed to return a numerical trial elastic stress to the isotropic J2 yield surface in plasticity models or to iteratively update creep strains until achieving convergence in creep models. In the case of plasticity, this algorithm iterates until the numerical trial stress is returned to the yield surface with an effective inelastic strain [11].

Used with the ROM, this algorithm computes the change inelastic strain as a function of a constitutive flow rule divided by the derivative of that flow rule with respect to the effective inelastic strain. For the stainless steel 316H ROM, the change in the effective inelastic strain is calculated as:

$$d\Delta p = \frac{\phi\left(\rho_m, \rho_i, T, \sigma_{vm}^{trial}, \Delta p\right)}{\frac{\partial \phi}{\partial \Delta p}} \tag{3.1}$$

where ρ_m is the mobile dislocation density, ρ_i is the immobile dislocation density, T is the temperature, σ_{vm}^{trial} is the effective trial stress provided by the J2 radial return mapping algorithm, and Δp is the effective inelastic strain. The values of the dislocation densities are taken from the previous timestep while the effective inelastic strain is the value from the previous iteration of the radial return algorithm. The effective trial stress is calculated by the radial return algorithm immediately prior to the ROM flow rule call, as a function of the effective inelastic strain.

$$\sigma_{vm}^{trial} = \sigma_{vm} - 3G\Delta p \tag{3.2}$$

where σ_{vm}^{trial} is the effective stress resulting from the FEM boundary condition application, G is the material shear modulus, and Δp is the effective inelastic strain from the previous radial return algorithm iteration.

In the case of the stainless steel 316H ROM, there are 5 input parameters, 45 input parameter normalization constants, and 3,072 Legendre polynomial coefficients for 3 output parameters. The input parameters

are listed in Equation 3.1, and the ROM output parameters are the effective inelastic strain, and the mobile and immobile dislocation densities.

The design decision to develop the ROM as a function of the effective stress and effective inelastic strain significantly improves the computational efficiency of the constitutive model. If the ROM used the full stress tensor and full inelastic strain tensors as inputs, the number of associated constants and coefficients would greatly increase. For example, the number of ROM inputs with the full stress and strain tensors would increase to 21 and the number of normalizing constants would grow to 693 to accommodate the larger number of 11 ROM outputs. By leveraging the material symmetry inherent in the cubic crystal lattice structure of steel, the J2 isotropic assumption mitigates the computational requirements of the ROM while retaining the physical microstructure basis provided by the dislocation density inputs.

The derivative of the constitutive flow rule in Equation 3.1 is a requirement for the robust integration of the ROM within the MOOSE framework. Because only the effective trial stress is a function of the effective inelastic strain, we were able to reduce the flow rule derivative derivation from a 5-to-3 mapping (the numerator of Equation 3.1) to the more numerically efficient 1-to-1 mapping (the effective trial stress to the effective inelastic strain). The flow rule derivation we have implemented utilizes the chain rule and thus includes derivations of the input parameter normalization, Legendre polynomial products, and output parameter conversion calculations. The implementation of this flow rule derivative enabled the use of the stainless steel 316H ROM as a radial return mapping constitutive model within Grizzly.

To ensure that our Grizzly implementation of the stainless steel 316H ROM is reliable, we designed a two sets of tests. The first set of tests is focused on verifying the ROM predictions throughout the allowable input parameter space while the second test set is intended to demonstrate the Grizzly implementation of the ROM on an engineering scale problem. We discuss first the verification tests in Section 3.1.3 and the engineering scale demonstration in Section 3.1.4.

3.1.3 Verification Testing throughout the Input Range Space

Our design of the verification tests was intended to survey the entire allowable ROM input parameter space. Our goals in developing these verification tests were two-fold: to ensure the Grizzly-integrated ROM was numerically reliable and to evaluate the ROM prediction trends. To achieve these goals, we evenly divided the allowable ROM input parameter space into six sections per input parameter, as shown in Table 3.1. From the input parameter space divisions, we kept the initial mobile and immobile dislocation densities paired, such that a mobile dislocation density of 9.0e12 ($1/m^2$) corresponded only to an immobile dislocation density of 8.6e11 ($1/m^2$). For each of the initial dislocation density pairs, we individually varied the temperature and effective deviatoric pressure values for a total of 216 verification simulations.

Table 3.1: Survey of temperatures, applied effective stress, and mobile dislocations varied in testing of the ROM

Temperature (K)	Pressure (MPa)	Mobile Dislocations (m^{-2})	Immobile Dislocations (m^{-2})
950	50.0	1.0e13	1.0e12
910	40.9	9.0e12	8.6e11
870	30.0	8.0e12	7.2e11
830	20.0	7.0e12	5.8e11
790	10.0	6.0e12	4.4e11
750	2.0	5.0e12	3.0e11

These verification simulations were performed on a single 8-noded hexahedron element for a simulation time of 100s. Throughout the simulations, the temperature and effective deviatoric pressure were held

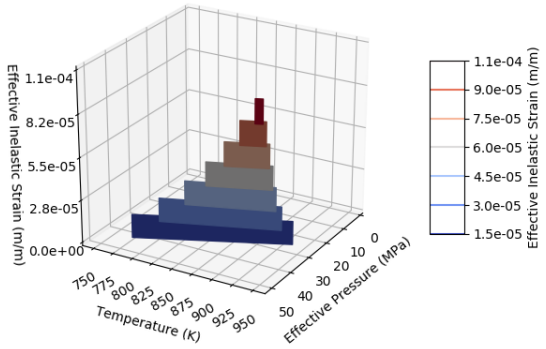
constant while the mobile and immobile dislocation densities and the effective inelastic strain were allowed to evolve in accordance with the ROM. The results of these verification simulations are shown in Figures 3.2 and 3.3.

These verification survey results demonstrate that the Grizzly-integrated ROM is reliable throughout the entire allowable input parameter space. Furthermore the integrated ROM predictions are reasonable and in line with the physical microstructure evolution of metals at high temperature.

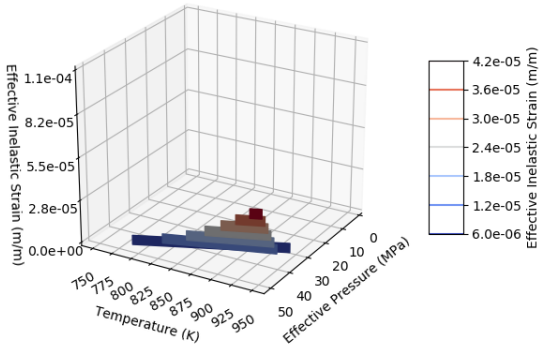
3.1.4 Simplified Cladding Tube Application

Our second demonstration test is representative of a cladding tube, and is modeled as a 2D axisymmetric geometry. In this simulation we quickly ramp the internal pressure, and then hold it constant. The temperature on the tube is increased more slowly over the simulation, with the highest temperature at each time step occurring in the center of the tube. The results of this demonstration simulation are shown in Figure 3.4.

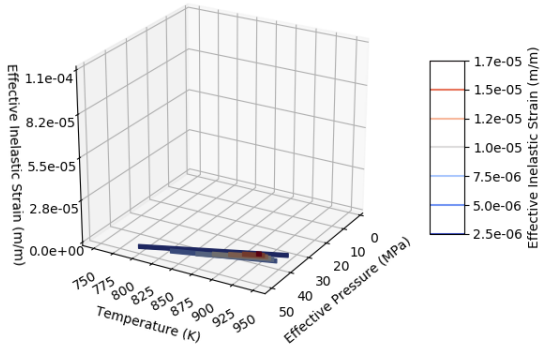
While neither the effective inelastic strain nor the mobile dislocation density appeared to evolve at the start of the simulation, a clear change in both quantities is observed under the high temperatures at the simulation end. As is expected from the verification test results, Section 3.1.3, at the high temperatures we see an increase in the effective inelastic strain and a decrease in the mobile dislocation density. Both quantities exhibit the greatest change at the center of the tube where the temperature is the highest.



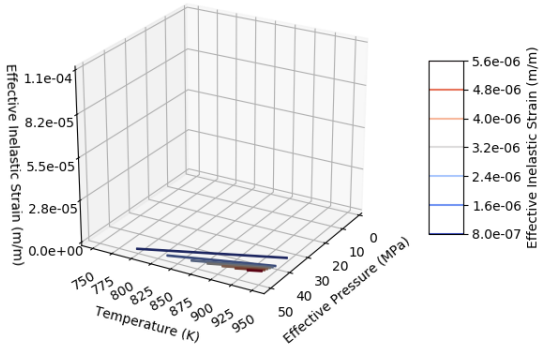
(a) Initial dislocation densities: ρ_m = 1.0e13 m^{-2} and ρ_i = 1.0e12 m^{-2}



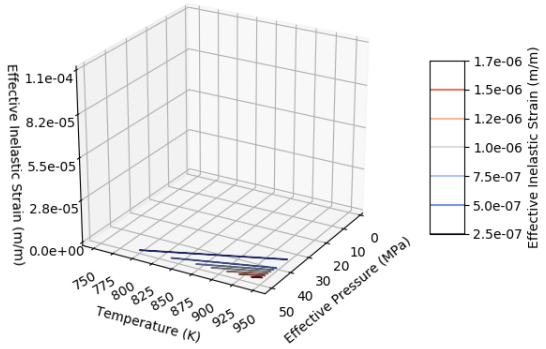
(b) Initial dislocation densities: ρ_m = 9.0e12 m^{-2} and ρ_i = 8.6e11 m^{-2}



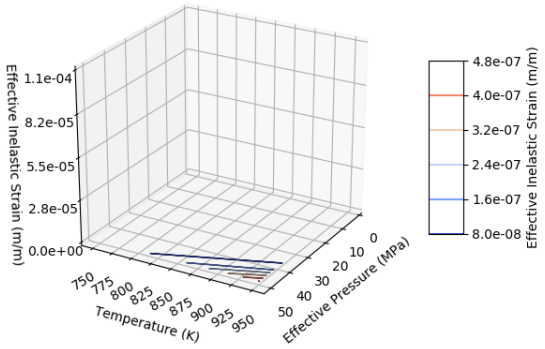
(c) Initial dislocation densities: ρ_m = 8.0e12 m^{-2} and ρ_i = 7.2e11 m^{-2}



(d) Initial dislocation densities: ρ_m = 7.0e12 m^{-2} and ρ_i = 5.8e11 m^{-2}

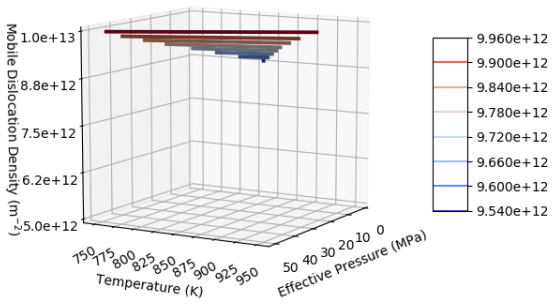


(e) Initial dislocation densities: ρ_m = 6.0e12 m^{-2} and ρ_i = 4.4e11 m^{-2}

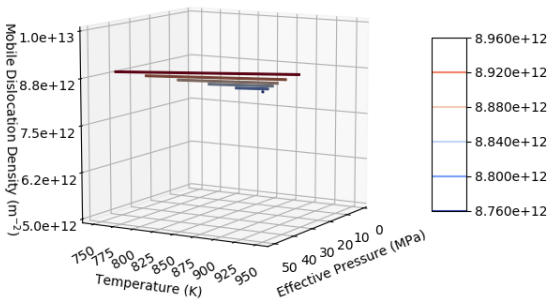


(f) Initial dislocation densities: ρ_m = 5.0e12 m^{-2} and ρ_i = 3.0e11 m^{-2}

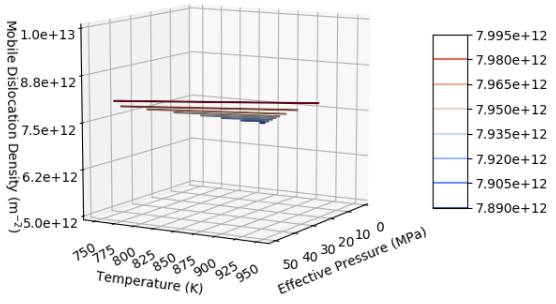
Figure 3.2: Survey of the effective inelastic strain response as a function of static temperature, applied effective stress, and mobile and immobile dislocation densities demonstrates that the effective inelastic strain is quite sensitive to the initial dislocation densities. For each initial dislocation density pair, the highest effective inelastic strain occurs at the highest temperature and effective pressure.



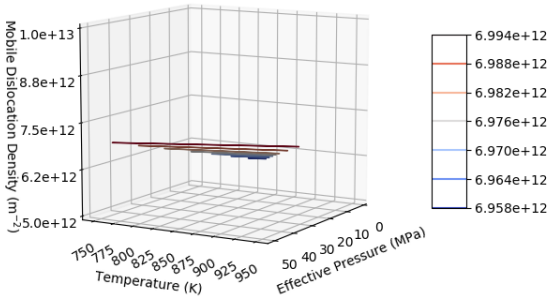
(a) Initial dislocation densities: ρ_m = 1.0e13 m^{-2} and ρ_i = 1.0e12 m^{-2}



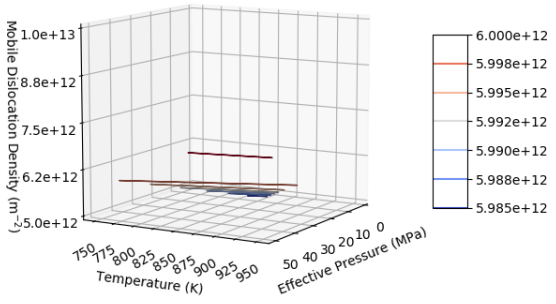
(b) Initial dislocation densities: ρ_m = 9.0e12 m^{-2} and ρ_i = 8.6e11 m^{-2}



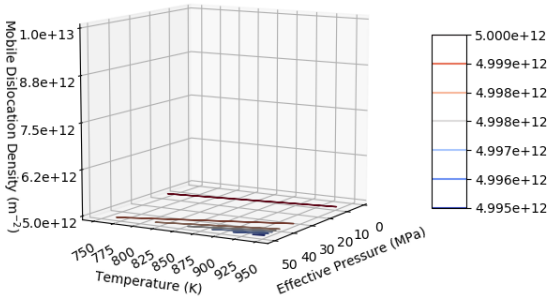
(c) Initial dislocation densities: ρ_m = 8.0e12 m^{-2} and ρ_i = 7.2e11 m^{-2}



(d) Initial dislocation densities: ρ_m = 7.0e12 m^{-2} and ρ_i = 5.8e11 m^{-2}



(e) Initial dislocation densities: ρ_m = 6.0e12 m^{-2} and ρ_i = 4.4e11 m^{-2}



(f) Initial dislocation densities: ρ_m = 5.0e12 m^{-2} and ρ_i = 3.0e11 m^{-2}

Figure 3.3: Survey of mobile dislocation density over different static effective deviatoric pressures and temperatures demonstrates the preservation of mobile dislocation densities at the lower temperatures and pressures. This trend is expected because the high pressures and temperatures both increase the driving force on dislocation motion which leads to dislocation annihilation and absorption.

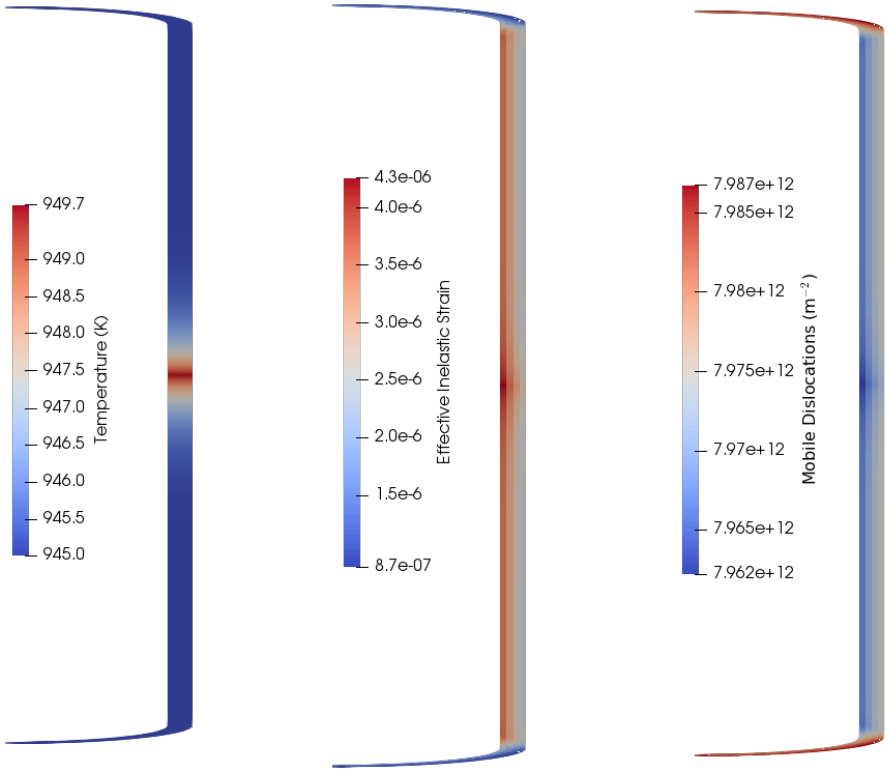


Figure 3.4: Evolution of effective inelastic strain and mobile dislocation density follow the pattern of increasing temperature at the center of the tube, with the highest effective inelastic strain and lowest mobile dislocation density occurring in the tube center. These simulation meshes have been scaled in the radial direction by a factor of 10 to improve visualization ease.

3.1.5 Future Work

The primary focus of the work documented here was to integrate the ROM with Grizzly for a single material. Basic tests were performed to assess whether this model gives the desired behavior.

Extensive additional verification and validation of this ROM is required to gain confidence in this approach, and will be a major focus of additional work in this area. Work is planned in collaboration with both LANL and Argonne National Laboratory (ANL) to identify experimental data sets which can be used for these validation of this model at the component level to test its applicability to engineering problems of interest. This effort will include quantifying the effects of uncertainty in the material response to the component-level response.

In addition, future work is planned to integrate similar models for additional structural metals with Grizzly. This work will involve close collaboration with LANL. In conjunction with LANL, we are exploring revisions to the code structure to maximize flexibility such that the same general ROM class structure can be used for multiple metals. Beyond integrating additional J2 based ROMs into Grizzly, exploring the use of other parameterizations of the inelastic material response is of interest for materials for which the inelastic response is not well characterized only by the von Mises stress. While the J2 isotropic treatment is acceptable for cubic crystalline metals such as steel, a generalization of this approach will likely be essential for property modeling materials with non cubicle crystalline structures, such as hexagonal close packed (HCP) crystal structures.

3.2 Develop, Commit, and Test Models for High Temperature Creep and Plasticity in Metals

The Nuclear Engineering Material model Library (NEML) is a modular library to support constitutive model development and usage in finite element software. NEML was developed at Argonne National Laboratory (ANL), and while targeted primarily at modeling high-temperature response of metals for nuclear applications, is applicable for modeling a variety of materials. The modular structure of NEML code allows for simulations to easily include models that are a composite of multiple models. Definitions of material models specifically of interest for high-temperature advanced reactor applications have been developed in NEML, such as one for 316H stainless steel [12].

The NEML code provides C, C++ and Fortran interfaces for integrating the library into other software packages. NEML supports two methods for creating and interfacing with the NEML material models: a Python binding method and an XML format which NEML reads natively. NEML is distributed under an open-source license, made publicly available through the GitHub website and has extensive online documentation [13].

NEML clearly offers important capabilities to expand the capabilities of Grizzly to allow it to be used for advanced reactor applications. NEML development has been motivated by the needs of the Advanced Reactor Technologies (ART) program, and NEML embodies significant knowledge in that area. The goal of 2019 Grizzly development in this area was to provide fully tested models for high temperature creep and plasticity, and this has been met by fully integrating NEML into Grizzly.

3.2.1 Integration of of NEML with BlackBear and Grizzly

Prior to the present work, interfaces to MOOSE-based codes had already been developed at ANL. However, these were not integrated into Grizzly or regularly tested by the automated testing system used by MOOSE. The following developments were made to fully integrate NEML into Grizzly:

- NEML was added as a git submodule to BlackBear.
- Material models to interface with NEML were added to BlackBear.
- Issues with using the MOOSE build system with NEML were resolved.
- Regression tests were added to BlackBear to ensure that the NEML interface remains functional.
- A demonstration pressure vessel model was run to ensure that Grizzly and BlackBear can robustly converge on a realistic problem using a NEML model for 316H stainless steel.

Details of this work are described below.

3.2.1.1 NEML as a submodule to BlackBear

The git version control system is used to manage all MOOSE-based codes managed by INL. Dependencies on external codes can be managed through a mechanism known as submodules, which allow a git repository for a code to bring in a specific version of another code that is also managed with git. Submodules are already used within the MOOSE environment to manage such dependencies. For example, MOOSE manages the version of the libMesh library that it links to through a submodule, and application codes such as Grizzly use submodules to link to a specific version of MOOSE. The version of the code linked to through the submodule can be updated as needed, and this is regularly done in the MOOSE codes. Every time a submodule is updated, thorough testing is done to ensure that updating that version of the external code does not adversely affect the codes that depend on it.

The Grizzly code base is split into two separate codes: Grizzly and BlackBear. Grizzly is an export-controlled code, but many of the models it uses do not have such restrictions, so these are made publicly available through BlackBear, which is an open-source code. Grizzly uses the git submodule mechanism to bring in all of the models in BlackBear. The interface to NEML, including the git submodule that is used to bring in that library was added to BlackBear, so that it is available to both Grizzly and BlackBear.

3.2.1.2 Material Models to Interface with NEML

Two material models that interface with NEML were contributed to BlackBear by developers at ANL:

- `NEMLStress`
- `NEMLThermalExpansionEigenstrain`

These models work within the mechanics model system provided by MOOSE's TensorMechanics module, providing stress and eigenstrain calculations, respectively, which are computed by calling NEML. These models behave like standard Material objects in MOOSE. Each model has an input block where parameters are defined, although very little data for these models is actually provided in the MOOSE input file. NEML uses XML files to define databases of material properties, and the user simply provides the location of the XML material database and the name of the model in the MOOSE input block for these materials.

INL developers worked together with ANL developers to ensure that these NEML linkage models were fully documented and tested in accordance with Grizzly development standards.

3.2.1.3 Integration with MOOSE build system

One obstacle encountered with integration of NEML with Grizzly was that NEML depended on an external XML parsing library (for reading its material database) that was difficult to ensure was installed on every computing platform that MOOSE supports. This made testing of the NEML models difficult within Grizzly because all aspects of the code need to be automatically tested on all platforms.

To address this issue, INL developers modified the NEML code to instead use the RapidXml library [14] for XML parsing. RapidXml is a C++ header-only library, which allows it to be easily integrated into NEML by simply bundling it with NEML. Once this change was made to NEML, BlackBear and Grizzly were able to be fully tested including the NEML functionality on all MOOSE-supported platforms.

The only remaining complexity in the NEML build process is that NEML depends on functionality provided by components of the boost library that are not used in the standard MOOSE build. A user has to specifically request that the full boost library is linked with libMesh when they are installing libMesh before they compile BlackBear of Grizzly. This is done by adding the `--with-boost` option to the `update_and_rebuild_libmesh.` script that all users run to compile libMesh as part of the standard compilation process. This is a minor inconvenience, and could be addressed by making further modifications to NEML to remove that dependency.

3.2.1.4 NEML Model Regression Testing

A set of regression tests were created using a simple linear strain hardening and power law creep material (both of which have native MOOSE TensorMechanics equivalent implementations) to benchmark test the performance and accuracy of those models. The regression tests are single element models with simple boundary conditions. The linear strain hardening material is loaded uniaxially until yielding and then allowed to harden up to a strain of $\sim 0.5\%$. The power law creep material is tested using a NAFEMS benchmark biaxial loading creep test. The model is run to a specific time and the strains in each direction are compared to a reference solution. A summary of the results of the benchmarking study are provided below.

3.2.1.5 Linear Strain Hardening Material

The stress-strain results for both the native MOOSE and NEML linear strain hardening material models agree very well (Figure 3.5). The performance of the two models is summarized in Table 3.2. The number of elements was changed by increasing the number of elements in the x and y directions using the internal mesh generation capability from 1×1 to 10×10 and 20×20. As can be seen, the NEML model is somewhat slower than the native MOOSE implementation, although it takes slightly fewer linear iterations.

The performance of this type of problem is highly dependent on the behavior of the return mapping iterations, which has been the topic of significant optimization effort in the native MOOSE models. The internal behavior of the NEML models has not been investigated, and these slower run times are not a major concern, as it is expected that they could be similarly optimized.

	Native MOOSE			NEML		
# Elements	**Run Time(s)**	**Lin It**	**NL It**	**Run Time(s)**	**Lin It**	**NL It**
1	0.236	35	35	0.255	35	35
100	1.05	148	35	5.66	134	34
400	3.43	146	34	22.1	143	34

Table 3.2: Performance comparison between native MOOSE and NEML linear strain hardening materials, showing run times, as well as linear and nonlinear iteration counts used in the preconditioned Jacobian-free Newton Krylov solver.

Figure 3.5: Stress strain behavior for linear strain hardening test for the native MOOSE model (BlackBear) and the NEML model.

3.2.1.6 Power Law Creep Material

The strain vs. time result for both the native MOOSE and NEML power law creep material models again agree fairly well, although the NEML model failed due to an internal error after a certain point in the simulation, as shown in Figure 3.6. It is suspected that the failure to converge is due to an issue with the return mapping procedure within NEML, although that has not been investigated.

The performance of the two models up to the point where the NEML model was terminated is summarized in Table 3.3. The number of elements was changed in the same way as was done for the linear strain hardening model. Again, the NEML library becomes slower than the native MOOSE implementation as the model size is increased, which needs further investigation. These performance and convergence issues are not a major concern, and simply need a more thorough investigation.

	BlackBear			NEML		
# Elements	**Run Time(s)**	**Lin It**	**NL It**	**Run Time(s)**	**Lin It**	**NL It**
1	0.292	73	36	0.279	72	36
100	1.94	88	34	3.80	138	48
400	6.76	90	34	14.44	144	48

Table 3.3: Performance comparison between native MOOSE and NEML power law creep materials, showing run times, as well as linear and nonlinear iteration counts used in the preconditioned Jacobian-free Newton Krylov solver.

Figure 3.6: Strain vs. time behavior for power law creep test for the native MOOSE model (BlackBear) and the NEML model.

3.2.2 Simplified Pressure Vessel Application

To test NEML on a more representative problem of engineering interest, a model of a simple pressure vessel consisting of a cylindrical section with hemispherical caps was constructed and tested with the parameters for a 316H stainless material model described in [12].

Two versions of this pressure vessel model were constructed: a 2D axisymmetric model containing 250 4-noded quadrilateral elements and 306 nodes, and a 3D model containing 4500 8-noded hexahedral elements and 5766 nodes. The boundary conditions, solution tolerances and time stepping for both models was identical. The side and bottom of the simplified pressure vessel had symmetry boundary conditions (i.e., fixed x-displacements and fixed y-displacements, respectively). The temperature was ramped to 450 K over one hour and then held at that value for a day. Similarly, an internal pressure of 15 MPa was applied to the vessel using the same ramp times as the temperature.

Figures 3.7 and 3.9 show the meshes for each simulation. In addition, Figures 3.8 and 3.10 show the hoop stresses and strains for the 2D and 3D simulations at the end of the hold time. As should be expected, the results are essentially identical. Table 3.4 lists the run times for the 2D and 3D models for different numbers of CPUs and the resulting speedups, which indicate good parallel scalability.

2D			3D		
# CPUs	**Run Time (s)**	**Speedup**	**# CPUs**	**Run Time (s)**	**Speedup**
1	567.1	—	9	2592	—
2	223.8	2.5	18	1348	1.9
4	146.3	3.9	36	694	3.7
8	62.3	9.1	72	395	6.6

Table 3.4: Parallel performance for 2D and 3D simple pressure vessel models.

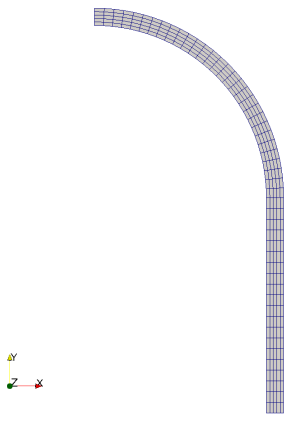
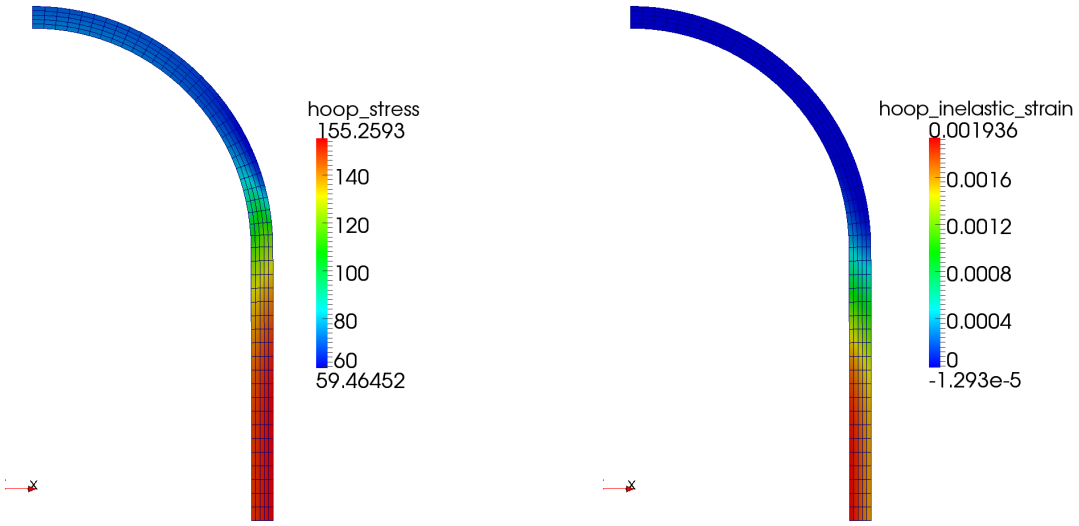


Figure 3.7: 2D axisymmetric mesh for simplified pressure vessel model.



(a) Hoop Stress

(b) Hoop Strain

Figure 3.8: Hoop stress and strain contours for the 2D simplified pressure vessel model.

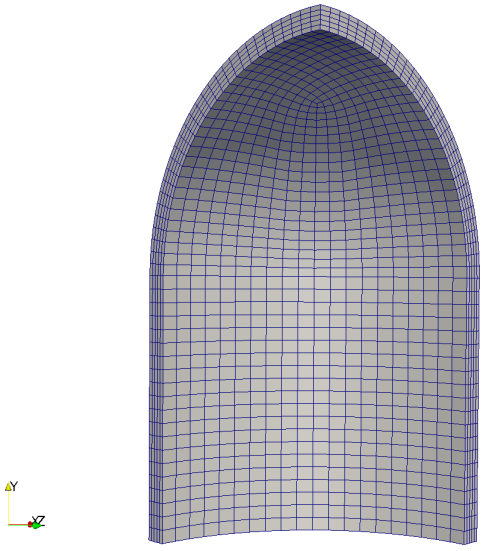
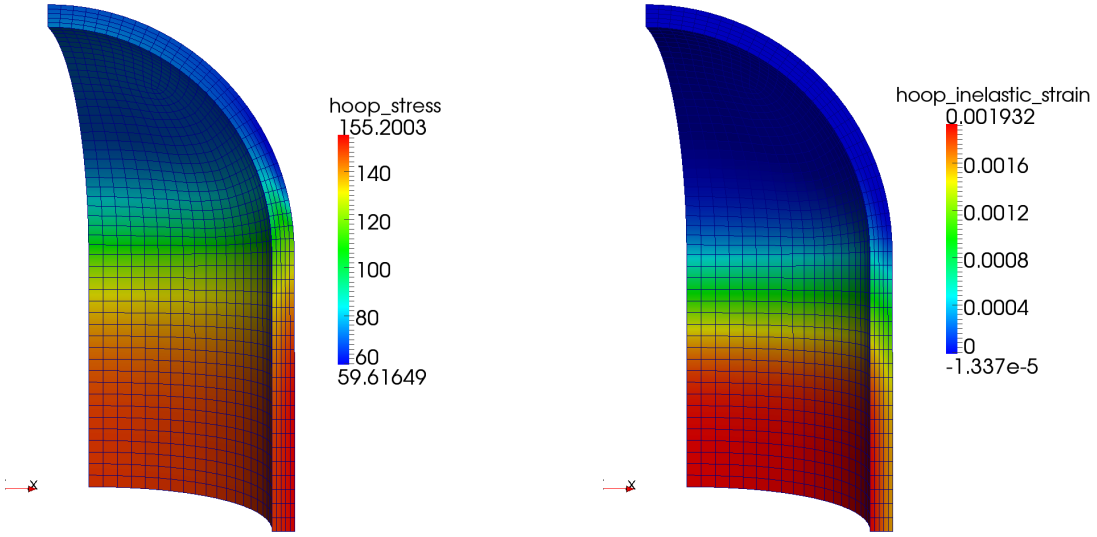


Figure 3.9: 3D mesh for simplified pressure vessel model.

3.2.3 Summary Future Work

The tests shown here demonstrate that NEML has been successfully integrated into Grizzly, and that Grizzly can run multiple types of material response modeled within NEML, including a proposed model for 316H stainless steel.



(a) Hoop Stress

(b) Hoop Strain

Figure 3.10: Hoop stress and strain contours for the 3D simplified pressure vessel model.

The simple creep and plasticity model test cases indicate that there are still some performance and robustness issues that should be addressed. It is unclear whether these only affect these basic models, or also more complex models such as that for 316H.

The major follow-on task to this work is to test the NEML models with Grizzly on analyses of component tests to validate this approach against experimental data.

4 Summary and Future Work

This report summarizes development in three important areas to enable Grizzly to address structural materials issues in advanced reactors:

- The XFEM capability has been further developed to permit representation of propagating cracks in 3D. This is an important foundation for a general capability that will ultimately allow for realistic fracture modeling in a variety of structural materials for advanced reactors, as well as LWRs and in nuclear fuel.
- The code structure for integrating models for high temperature inelastic response of metals based on a reduced order model representation of a mesoscale model into MOOSE-based codes has been developed. Based on this structure, a model for 316H stainless steel has been implemented, and models for other materials are currently in development.
- An interface to the NEML library, which provides access to engineering constitutive models developed to support analysis of metals for high temperature advanced reactor applications for the DOE ART program, has been fully integrated into the Grizzly and BlackBear codes. This has been tested with a NEML model of 316H stainless steel on a representative pressure vessel model run with Grizzly.

Details of logical follow-on work have been provided in the individual sections of this report where they were discussed. These tasks are summarized at a high level here:

- The 3D XFEM capability should be further generalized to permit the use of fracture integrals to define propagation of fully general crack geometries, and to model subcritical crack growth such as that due to fatigue or stress corrosion cracking.
- Incorporation of mesoscale-based reduced order models for high temperature inelastic response for other materials, including accounting for departure from J2 behavior.
- Validation testing of both the mesoscale-based models and NEML engineering models, comparing predicted component-level response with experimental data.

Acknowledgments

The material model development work described here builds heavily on work performed by Mark Messner at Argonne National Laboratory, and the team at Los Alamos National Laboratory that includes Laurent Capolungo, Arul Mariyappan, Aaron Tallman, and Christopher Matthews.

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract DE-AC07-05ID14517. Accordingly, the U.S. Government retains a non-exclusive, royalty free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

Bibliography

[1] Ziyu Zhang, Wen Jiang, John E. Dolbow, and Benjamin W. Spencer. A modified moment-fitted integration scheme for X-FEM applications with history-dependent material data. *Computational Mechanics*, 62(2):233–252, August 2018.

[2] Benjamin Spencer, Hai Huang, John Dolbow, and Jason Hales. Discrete modeling of early-life thermal fracture in ceramic nuclear fuel. In *Proceedings of WRFPM 2014*, Paper No. 100061, Sendai, Japan, September 2014.

[3] Benjamin W. Spencer, Wen Jiang, John E. Dolbow, and Christian Peco. Pellet cladding mechanical interaction modeling using the extended finite element method. In *Proceedings of Top Fuel 2016*, Boise, ID, September 2016.

[4] John Dolbow, Ziyu Zhang, Benjamin Spencer, and Wen Jiang. Fracture capabilities in Grizzly with the eXtended Finite Element Method (X-FEM). Technical Report INL/EXT-15-36752, Idaho National Laboratory, Idaho Falls, ID, September 2015.

[5] B.W. Spencer, W.M. Hoffman, and M.A. Backman. Modular system for probabilistic fracture mechanics analysis of embrittled reactor pressure vessels in the Grizzly code. *Nuclear Engineering and Design*, 341:25–37, January 2019.

[6] A. Hansbo and P. Hansbo. A finite element method for the simulation of strong and weak discontinuities in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 193(33-35):3523–3540, Aug 2004.

[7] P.M.A. Areias and T. Belytschko. A comment on the article "A finite element method for simulation of strong and weak discontinuities in solid mechanics" by A. Hansbo and P. Hansbo Comput. Methods Appl. Mech. Engrg. 193 (2004) 3523-3540. *Computer Methods in Applied Mechanics and Engineering*, 195(9-12):1275–1276, 2006.

[8] R A Lebensohn and C N Tomé. A self-consistent anisotropic approach for the simulation of plastic deformation and texture development of polycrystals: Application to zirconium alloys. *Acta Metallurgica et Materialia*, 41(9):2611–2624, September 1993.

[9] R J Asaro. Crystal Plasticity. *Journal of Applied Mechanics*, 50(4b):921–934, December 1983.

[10] Derek Hull and David J Bacon. *Introduction to Dislocations*, volume 37. Elsevier, 2011.

[11] Fionn Dunne and Nik Petrinic. *Introduction to Computational Plasticity*. Oxford University Press on Demand, 2005.

[12] T.-T. Phan, M. C. Messner, and T.-L. Sham. A unified engineering inelastic model for 316H stainless steel. In *Proceedings of ASME 2019 Pressure Vessels and Piping Conference*, Paper No. 93641, San Antonio, TX, July 2019.

[13] NEML — Nuclear Engineering Material model Library. `http://neml.readthedocs.io/en/stable/index.html`, 2019.

[14] Marcin Kalicinski. RAPIDXML manual. http://rapidxml.sourceforge.net/manual.html, 2019.