INL/EXT-15-36774 Rev. 1

September 2015

Release 1.0 of MARMOT: A Mesoscale Fuel Performance Code

D. Schwen
P. Chakraborty
M. R. Tonks
B. Fromm
Y. Zhang
X. Bai
D. A. Andersson

INL/EXT-15-36774 Rev. 1

Release 1.0 of MARMOT: A Mesoscale Fuel Performance Code

D. Schwen
P. Chakraborty
M. R. Tonks
B. Fromm
Y. Zhang
X. Bai
*D. A. Andersson**

September 2015

Idaho National Laboratory
Fuel Modeling and Simulation Department
Idaho Falls, Idaho 83415

***From MST-8, Los Alamos National Laboratory, Los Alamos, NM**

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

Release 1.0 of MARMOT: A Mesoscale Fuel Performance Code

D. Schwen, P. Chakraborty, M. R. Tonks
B. Fromm, Y. Zhang, X. Bai

Computational Materials Science Group
Fuels Modeling & Simulation Department
Idaho National Laboratory
Idaho Falls, ID

D. A. Andersson
MST-8
Los Alamos National Laboratory
Los Alamos, NM

November 2015

Abstract

MARMOT is the mesoscale fuel performance code under development as part of the US DOE Nuclear Energy Advanced Modeling and Simulation Program. With this report, we release a new version of MARMOT, summarize its implemented capabilities, and document the ongoing development of new capabilities planned for the next release. The purpose of MARMOT is to predict the coevolution of microstructure and material properties of nuclear fuel and cladding due to stress, temperature, and irradiation damage. It accomplishes this using the phase field method coupled to solid mechanics and heat conduction. MARMOT is based on the Multiphysics Object-Oriented Simulation Environment (MOOSE), and much of its basic capability in the areas of the phase field method, mechanics, and heat conduction come directly from MOOSE modules.

Contents

1 Introduction

With this report we release the version 1.0 of the MARMOT microstructural simulation code. We briefly enumerate the capabilities developed for and implemented into MARMOT. Future reports of this kind will be of incremental nature and will reference the previous reports. The development of MARMOT is managed using the git source control system. Development of MARMOT is continuous and ongoing. Rigorous automatic testing procedures ensure that every incremental version of MARMOT is a usable product on its own. For this report we used the git *tag* "v1.0" to mark a specific point in the development history as our first official release (Fig. 1.1).

For a more comprehensive discussion of simulation results produced with MARMOT and for a discussion of code verification and validation we refer to the MARMOT Assessment Report [1] published in April 2015.

1.1 MARMOT Development

NEAMS FPL developed the MARMOT code as a robust numerical tool for mesoscale modeling of fuel performance to predict the coevolution of microstructure and properties in fuel and cladding materials, providing a new capability at length scales above the first principles density functional theory and MD domains. MARMOT accomplishes this using the phase field method coupled with finite strain mechanics and heat conduction. MARMOT is based on the open source Multiphysics Object-Oriented Simulation Environment (MOOSE) [2] and solves the coupled partial differential equations defining the physics using the finite element method [3]. MARMOT is being developed in order to facilitate the development of improved materials models for fuel performance, but it is also being developed as a powerful tool in and of itself for the simulation of mesoscale fuel performance.

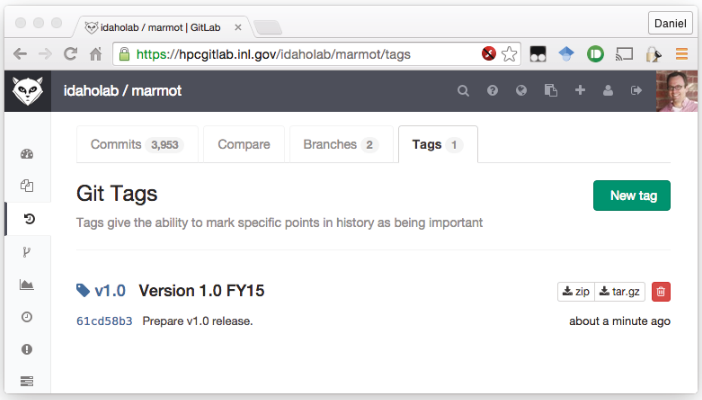
While the goal of the MARMOT tool is focused on investigating fuel and cladding materials, many of the capabilities employed by the code could also be applied to other materials and applications. Therefore, the general capabilities for the phase field method, solid mechanics, and heat conduction are contained in physics modules that are distributed with the MOOSE framework. Any user that downloads MOOSE can instantly use these tools to rapidly develop multiphysics mesoscale simulation tools for a wide range of different applications. These are the `phase_field`, `tensor_mechanics`, and `heat_conduction` modules. MARMOT builds on these modules and adds specific materials and models for fuel and cladding materials.

The splitting of MARMOT components into export controlled and open sourced parts allows us to make as many capabilities as possible available to the general research community. This fosters the proliferation of MOOSE based research codes, helps MOOSE/MARMOT get established as a standard research code, and generates code contributions from external users. We connect with these external users through the MOOSE users mailing list and work on developing lasting collaborations though joint proposals and by utilizing the INL student internship program.

1.2 External Use of MARMOT

For the access controlled MARMOT code we have active users and contributors at the following institutions:

- University of Arkansas
- University of South Carolina



Tags give the ability to mark specific points in history as being important

Prepare v1.0 release.

Figure 1.1: Screen shot of the git source control web frontend at https://hpcgitlab.inl.gov showing the tagged v1.0 release of MARMOT.

- Purdue University
- Penn State University
- BYU Idaho
- University of Idaho
- Oregon State University
- Missouri University of Science and Technology
- Massachusetts Institute of Technology
- Los Alamos National Laboratory
- Washington State University
- University of Wisconsin

The use of the open source phase field and tensor mechanics components that are being developed for MARMOT is more difficult to track. An analysis of MOOSE user mailing list contributions shows active contributions from the following institutions.

- Max-Planck-Institut für Eisenforschung, Germany
- Idaho State University
- Shanghai Jiao Tong University, Shanghai
- Texas A&M University
- University of Michigan
- Commonwealth Scientific and Industrial Research Organisation, Australia

The mailing list currently has over 400 subscribers, with a majority being silent. We expect that with an approximate fraction of 20% of all mailing list posts dealing with phase field and tensor mechanics issues the number of unreported phase field / tensor mechanics users is substantial. We plan to address this issue with a MOOSE/MARMOT user census in the upcoming fiscal year. This will ensure that we can reach out to current and potential users

1.3 Mechanistic Material Model Development

Fuel and cladding materials undergo significant microstructure evolution during reactor operation. This evolution also changes the fuel properties, directly impacting fuel performance and safety. Traditional fuel performance codes account for these changes in properties using materials models that are empirical fits to experimental data and are correlated to burn-up and temperature. However, these models can only be interpolated within conditions where the tests were conducted and cannot be trusted when the irradiation conditions change, because burn-up is not a unique measure of the history of the fuel material. The Fuels Product Line (FPL) in the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program is developing new materials models that are mechanistic and are based on microstructure rather than burn-up.

In this microstructure based approach, the current state of the microstructure is defined by a set of microstructure variables, e.g. average grain size, grain boundary coverage, and intragranular gas bubble porosity. These variables are evolved with time using mechanistic equations defining the physics of the phenomena. In turn, the material properties are functions of these variables as defined by other mechanistic equations. Thus, this set of variables and mechanistic models describes the interplay of the various microstructure changes that take place within the fuel during irradiation and predicts the resultant degradation in material performance.

1.4 Report Overview

In this report we list the capabilities of version 1.0 of the MARMOT microstructural simulation tool and give an outlook on the current and planned developments. The report is structured into five sections. The introduction section is followed by sections for the phase field 2 and tensor mechanics 3 physics modules, which are being developed for and used by MARMOT but are distributed as part of the opensource MOOSE framework. MARMOT specific material models are listed in section 4. Concluding remarks are given in section 5.

2 Phase Field Capabilities

The `phase_field` module in MOOSE contains the necessary tools to solve the partial differential equations for the phase field method that define the microstructure variable evolution to minimize the overall free energy. The evolution of non-conserved order parameters η_i (representing phase regions and grains) is governed by the Allen-Cahn equation (2.1) and conserved order parameters c_i (representing concentrations) are evolved using the Cahn-Hilliard equation (2.2).

2.1 Modular free energies

$$\frac{\partial \eta_j}{\partial t} = -L_j \frac{\delta F}{\delta \eta_j} \tag{2.1}$$

$$\frac{\partial c_i}{\partial t} = \nabla \cdot M_i \nabla \frac{\delta F}{\delta c_i} \tag{2.2}$$

F is the total free energy of the modeled system as a function of the phase field variables, which can be formulated as a volume integral

$$F = \int_\Omega \left[f_\text{loc}(\vec{c},\vec{\eta}) + f_\text{gr}(\nabla\vec{c},\nabla\vec{\eta}) + E_\text{d} \right] dV, \tag{2.3}$$

over multiple free energy density contributions, where Ω is the simulation domain, f_loc is the local free energy density, f_gr is the gradient energy contribution, and E_d is the contribution of other sources of energy. The $\vec{c},\vec{\eta}$ and $\nabla\vec{c},\nabla\vec{\eta}$ indicate a functional dependence on all conserved and non-conserved order parameters in the domain and their gradients, respectively. Executing the variational derivatives in (2.1) and (2.2) yields terms containing the derivatives of the local free energy density f_loc with respect to all order parameters.

The thermodynamic properties of the modeled system are determined by the thermodynamic potential in f_loc. The gradient contribution f_gr is the reason the phase field model represents interfaces with a diffuse width, and contributes to the interfacial energy. f_loc is therefore the primary input needed to formulate a new phase field material model. In the `phase_field` module, the residuals for the generic phase field equations are provided as kernels, while the free energy and its derivatives are supplied by material objects. We use a special material interface to provide material properties for all necessary derivatives of the free energy. In general, users use the provided kernels without modification, and only create material objects defining different free energies.

This material interface is now also used to provide the mobilities M_i and L_j, which allows them to be functions of arbitrary non-linear variables. All derivatives of these mobilities arising in either the residual or Jacobian equations are provided through derivative material properties. Furthermore we generalized the mobility expressions to allow for tensorial forms which accommodate anisotropy. We were able to implement this without code duplication by utilizing C++ templates, where the type of the mobility value (and its derivatives) is a template parameter that can be set to scalar or tensorial types.

The standard MOOSE solver uses the preconditioned Jacobian-free Newton Krylov method (PJFNK), provided by the PETSc library [4]. To improve the convergence of the solve, the chosen preconditioning matrix should be as close as possible to the actual Jacobian of the problem. Computing the Jacobian matrix entries effectively means providing the derivatives of the residual vector with respect to all non-linear variables of the problem, thus requiring additional derivatives (including cross-derivatives) of the free energy functional. Especially for quaternary two phase free energy this amounts to a large number of derivatives to be evaluated.

2.1.1 Automatic differentiation

To create a material object that defines the free energy for a phase field model, code must be written that defines the thermodynamic free energy expression, but also its derivatives. For non-conserved order parameters, the 2nd derivatives are needed and for conserved order parameters, up to the 3rd derivatives could be required. This is complicated even more when a free energy is a function of multiple variables, because all cross derivatives are also required. To avoid having to take and implement all the derivatives, we have implemented automatic symbolic differentiation.

MOOSE uses the Function Parser library that is included as a third-party plugin in the underlying libMesh finite element library [5]. The Function Parser Library accepts a mathematical function definition given as a plain text string. This string is lexically parsed into an intermediate tree representation and then transformed into a stack machine bytecode. This bytecode can then be executed by the function parser bytecode interpreter module as often as necessary without further transformations.

This intermediate tree representation of the function parser expressions lends itself to algorithmic transformations, such as an automatic differentiation procedure. In this tree structure, leaf nodes can correspond to constants or variables, and internal nodes correspond to mathematical operators and functions with the arguments contained in the respective child nodes or subtrees. The derivative of the leaf nodes yields 0 for all nodes that do not represent the variable the derivative is taken with respect to, and 1 for all nodes that do represent the variable. The derivatives of the internal nodes are constructed recursively according to a set of elementary derivative rules.

The function parser library provides a comprehensive algebraic optimizer that groups, reorders, and transforms the function expression into an equivalent but faster to evaluate form. This optimization stage delivers a speedup of a factor of two, on average. The algebraic simplifications are essential to remove the trivial leaf node derivatives which may lead to evaluation errors such as divisions by zero, that can be avoided by simple term cancellations. In the above example, the simplifications reduce the derivative expression to $2x(y+5)$.

2.1.2 Just-in-time compilation

To further improve the performance of the parsed and runtime interpreted functions, we have developed a just-in-time (JIT) compilation module. At runtime, the generated bytecode sequences are automatically transformed into small C source code files. A compiler is dispatched to compile each function file into a dynamically linkable library, which then is loaded on the fly using the `dlopen` POSIX system call. If at any stage the JIT compilation fails the function evaluation falls back on the bytecode interpreter, otherwise the generated machine code is called. The time overhead of the additional compilation step is on average of the order of 0.1s per function expression or below, depending on the system the simulation is executed on. This is further mitigated by a caching system. A unique hash is computed from the function bytecode and the compiled functions are stored permanently using the hash as a filename. Recompilation will only occur if the bytecode, and thus the function expression, changes. Trivial function changes, namely the modification of constants, will in most cases not trigger a recompilation. Through this automatic differentiation system we achieve a significant reduction in developer time and remove a source of developer errors that are difficult to track down and debug. The resulting models offer optimal convergence properties due to the complete implementation of the full Jacobian matrix.

2.1.3 Expression Builder

To generate functional expressions for free energies and mobilities that can be passed to the automatic differentiation algorithm we have developed the ExpressionBuilder system. ExpressionBuilder is a C++ class that gets added to MOOSE objects such as Materials through inheritance and makes a set of new nested classes available to hold terms and functions. A term is defined as a symbol, a number or a mathematical operator and its parameters. A function is defined as a term with a substitution rule that substitutes the function arguments into the underlying term. In addition ExpressionBuilder provides overloaded operators that accept the new term and function types as parameters and returns augmented terms. Terms are internally stored as tree structures. Nodes in these trees can represent operators or mathematical functions (such as the logarithm or trigonometric functions) or symbols and numbers. Operators and function nodes have child nodes representing their arguments (e.g. left hand side and right hand side). The overloaded operators assemble a tree representing an entire expression step by step. Named terms and functions can be combined to form new terms and functions, allowing the construction of complex expressions. The syntax of expression builder is designed to naturally match mathematical expressions as they can be found in scientific publications.

2.2 Multiphase models

Multiphase model development requires the construction of a global free energy functional spanning the entire phase space of the system. One common approach is utilizing a linear combination of the free energy densities $f_{loc,j}$ of each phase in the system.

$$f_{loc}(\vec{c},\vec{\eta}) = \left[\sum_j h(\eta_j) f_{loc,j}(\vec{c})\right] + Wg(\vec{\eta}) \tag{2.4}$$

A switching function $h(\eta)$ smoothly changes from 0 to 1 as η goes from 0 to 1. The total weight of all phase free energy contributions at each point in the simulation volume is exactly unity, which translates to the need to enforce the constraint $k(\vec{\eta}) = 0$ for

$$k(\vec{\eta}) = \left[\sum_j h(\eta_j)\right] - 1. \tag{2.5}$$

For the gradient interface term f_{gr} multiple models are implemented as MOOSE kernels. The most comprehensive model accepts individual gradient energy factors $\kappa_i j$ for interfaces between arbitrary pairs of order parameters η_i, η_j and is defined as

$$f_{gr} = \sum_{\substack{a,b\\ b\neq a}} \frac{1}{2}\kappa_{ab} \left|\eta_a \nabla\eta_b - \eta_b \nabla\eta_a\right|^2. \tag{2.6}$$

2.2.1 Constraint enforcement

Two phase systems can easily be modeled using a single order parameter η_1 and the explicit constraint $\eta_2 = 1-\eta_1$, which, for a symmetric switching function with $h(\eta) = 1-h(1-\eta)$, satisfies the constraint k. For n-phase systems with $n > 2$ it becomes advantageous to use n order parameters. In this case the constraint k is not automatically satisfied and needs to be enforced by other means. In the MOOSE phase field module we offer two methods to enforce the switching function sum constraint, a hard constraint utilizing the Lagrange multiplier technique and a soft constraint through a penalty term added to the free energy.

The hard constraint is applied by introducing a Lagrange multiplier λ as a field variable. With $a_j(\vec{\eta},\vec{c},v)$ being the weak form (Allen-Cahn) residual for the jth non-conserved order parameter, we need to find $(\vec{\eta},\lambda)$ satisfying the boundary conditions such that

$$a_j(\vec{\eta},\vec{c},v)+\int_\Omega \lambda\frac{\partial k}{\partial \eta_j}v\,dx = 0 \tag{2.7}$$

$$\int_\Omega q\frac{\partial(\lambda k)}{\partial \lambda}dx = 0 \tag{2.8}$$

holds for every test function v and q. We note that these equations alter the character of the Jacobian matrix of the non-linear problem substantially by introducing a zero block on the Jacobian diagonal. This can complicate the solve substantially. By replacing the constraint k with a modified constraint

$$\bar{k}(\vec{\eta},\lambda)=k(\vec{\eta})-\frac{\varepsilon}{2}\lambda, \tag{2.9}$$

the Jacobian fill term $\frac{\varepsilon}{2}\lambda$ introduces a small λ dependence in the constraint through an ε (which defaults to 10^{-9}). This results in an on-diagonal Jacobian value of $-\varepsilon$ in the kernel of Eq. (2.8), while it drops out in the residual of Eq. (2.7). This is necessary to force a Jacobian matrix with full rank, avoiding *Zero pivot* PETSc-Errors, and greatly improves convergence. This approach results in a violation of the constraint by about ε, though this violation can be kept small by using an ε as small as possible.

As an alternative we implemented a soft constraint by constructing a penalty contribution f_p to the free energy as

$$f_p=\chi\left[1-\sum_j h(\eta_j)\right]^2, \tag{2.10}$$

where χ is a configurable penalty factor.

2.3 Nucleation model

To deal with precipitate nucleation, as it is for example occurring during the aging process of the RPV materials, the phase field method needs to be augmented. Phase field is intrinsically fluctuation free and strictly minimizes the free energy in absence of external driving forces. Nucleation processes depend on thermal fluctuation for nuclei to overcome the Gibbs barrier which results from energy penalty of a newly forming interface between nucleus and matrix. Two classes of approaches for implementing nucleation phenomena in phase field can be found in the literature. One approach is adding thermal fluctuations to the order parameter fields (which is implemented in the MOOSE phase field module through the conserved noise classes). The main drawback of this approach is the timestep reduction incurred by adding the short timescale fluctuations to potentially long timescale diffusion processes. This drawback is avoided by the second type of nucleation model in which nuclei above the critical size are directly inserted into the simulation cell, bypassing the nucleus formation step. When a stable nucleus is inserted into the simulation cell all conserved order parameters must retain their total value. In practice that means a depletion zone must be constructed around the nuclei. It is not straight forward how that zone is supposed to look like in heterogeneous microstructures.

We have developed a free energy based discrete nucleation model for the phase field method to address the issue of conserving concentrations and establishing physical depletion zones. At a nucleation site the local free energy density of the system is modified with an additional term that is computed from a list of order parameter (concentration or phase) values and their desired target values for the phase that is supposed to nucleate. As a simple first approach the sum of squares of the differences of an order parameter value c_i and its target values $\bar{c}_i$ is multiplied with a prefactor γ and added to the free energy density. This free energy modification coerces the system to form stable precipitates.

$$f_{\text{nuc}}=\gamma\sum_i(c_i-\bar{c}_i)^2 \tag{2.11}$$

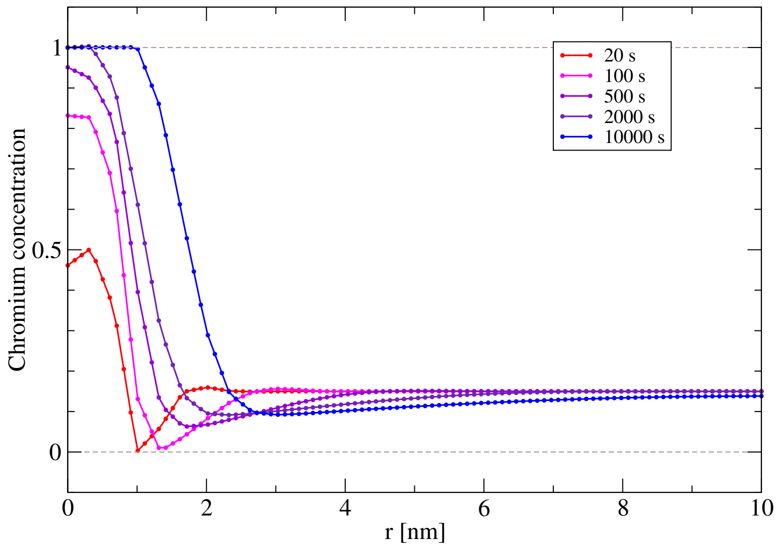


Figure 2.1: Concentration profiles along a spherical chromium rich α' particle in an Fe-15%Cr alloy at different times during the nucleation and growth process.

Nucleus positions are randomly chosen according to a nucleation probability density which can be calculated using classical nucleation theory. The nucleus list is updated at each timestep, adding new nucleation sites (if nucleation events were determined to happen) and removing old nucleation sites once they expire. Each nucleation site is kept in the list for a set hold time. The hold time is chosen sufficiently long for the precipitate to form on the diffusive timescale of the system. As the precipitate forms at the target composition the free energy density f_{nuc} goes down to zero at the nucleation site.

The nucleation algorithm is implemented in MOOSE using two user objects and a material class. One user object manages the global nucleus list, which is synchronized between MPI processes. The second user object uses this list to create a map of all quadrature points in the system that are covered by a nucleation site. A customizable radius can be assigned to the nucleation sites to stabilize precipitates of a given minimum size. The material class accesses the map user object to decide whether the free energy should be modified for a given quadrature point. This approach allows to update the map only if the nucleus list has changed or the mesh was adapted, ensuring the best possible performance.

Data from an example nucleation simulation in an Fe-15%Cr alloy is shown in figure 2.1. The chromium concentration profiles of a developing nucleus as a function of distance r from the nucleus center are plotted as a function of time. At early times the nucleus forms by absorbing solute from the immediate surroundings of the nucleations site. A small concentration depression can be seen in the nucleus center. This is an artifact of the model and can be adjusted by modifying the nucleation energy penalty profile. At later times solute has diffused over longer length scales widening the depletion region and forming a plateaued concentration profile in the precipitate.

Figure 2.2 shows the maximum chromium concentration observed at any point in the simulation cell and total nucleation free energy penalty as functions of time. The maximum concentration saturates at about unity, which is the expected α' concentration in the precipitate. At this point an α' nucleus has fully formed, but it may be below the critical size. The nucleation energy penalty zone stabilizes this nucleus and drives further growth until the volume of the nucleus completely covers the penalty volume. At that point the nucleation free energy penalty reaches zero. The size of the penalty zone must be chosen so that a nucleus of stable size above the critical radius formed. Growth of the particle can then proceed in a natural way, unperturbed by the nucleation algorithm. Any applied nucleation penalty is switched off after a chosen time to allow the precipitate to evolve naturally, which could include vanishing due to coarsening (e.g. Ostwald ripening) processes. The switch off time for the nucleation penalty should be chosen to approximately match the the vanishing of the nucleation free energy.

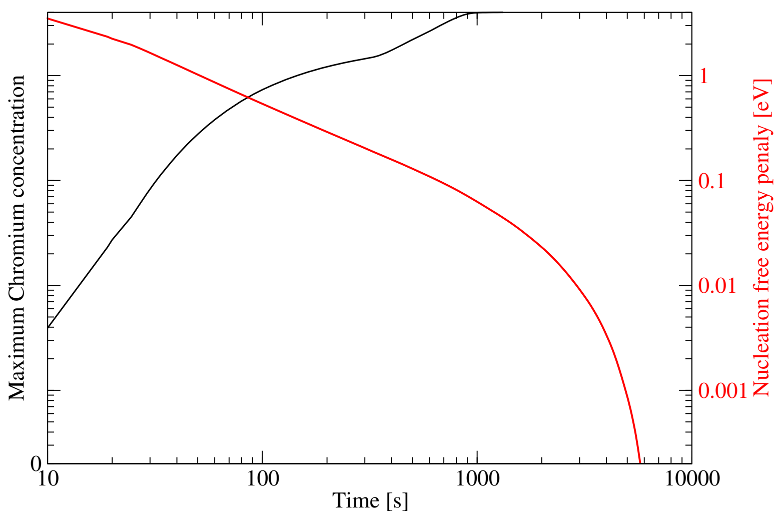


Figure 2.2: Maximum chromium concentration observed at any point in the simulation cell and total nucleation free energy penalty as functions of time. The maximum concentration saturates at about unity, which is the expected α' concentration in the precipitate. The nucleation free energy penalty reaches zero as soon as a sufficiently large α' particle is formed. At that point the stable size of the particle is reached and the growth regime takes over.

2.4 Mechanics coupling

The modular free energy approach allows for a straight forward coupling of the solid mechanics to the phase field equations. A material class is provided to compute the local elastic free energy density using the stress and strain states of the sample as a function of position. This `ElasticEnergyMaterial` automatically provides the necessary derivatives with respect to the coupled variables in the underlying mechanics properties. A coupling can emerge for example if a composition dependent Eigenstrain class was added to the problem. This Eigenstrain class will automatically set the needed derivatives of strain with respect to the composition variables. These derivatives are picked up by the elastic energy material to provide composition derivatives of the elastic energy. The elastic energy is combined with the chemical free energy using a `DerivativeSumMaterial` object.

2.4.1 Rigid body motion

The simulation of sintering processes, which is of interest in the fabrication of fuels, requires the modeling of rigid body motion of individual powder grains. We have implemented a rigid body motion system [6] within the phase field framework based on a system of integro-differential equations that accumulate forces and torques on individual particles, which are defined by their unique phase order parameters. The body forces are applied to the particles as advection terms that move the particles as rigid bodies.

2.5 Fracture model

Brittle fracture at the microstructure scale, involving complicated crack paths, can be modeled using the phase-field approach. Damage evolution and material degradation is achieved by solving the coupled system of equations

$$\nabla . \left(\left[(1-c)^2 + k \right] \underline{\underline{\sigma_0^+}} - \underline{\underline{\sigma_0^-}} \right) = 0 \tag{2.12a}$$

$$l \Delta c - \beta = 0 \tag{2.12b}$$

$$\dot{c} - \frac{1}{\eta} \left\langle \beta + 2(1-c) \frac{\psi_0^+}{g_c} - \frac{c}{l} \right\rangle_+ = 0 \tag{2.12c}$$

where c represents the damage variable that evolves from 0 to 1, with 0 and 1 representing the undamaged and completely damaged states respectively. In equation 2.12c, η is a viscosity parameter that can be utilized to approach the rate independent limit ($\eta \to 0$) and g_c is an energy release type parameter. ψ_0^+ is the positive component of strain energy driving damage growth and defined as

$$\psi_0^+ = \lambda \langle \varepsilon_1 + \varepsilon_2 + \varepsilon_3 \rangle_+^2 / 2 + \mu \left(\langle \varepsilon_1 \rangle_+^2 + \langle \varepsilon_2 \rangle_+^2 + \langle \varepsilon_3 \rangle_+^2 \right) \tag{2.13}$$

where ε_1, ε_2 and ε_3 are the principal strains, λ is the Lamé constant and μ is the shear modulus. The utilization of the positive component of strain energy as the driver for damage evolution prevents crack growth under compressive strain states. The signed stresses in equation 2.12a follows

$$\underline{\underline{\sigma_0^\pm}} = \sum_{a=1}^{3} \left[\lambda \langle \varepsilon_1 + \varepsilon_2 + \varepsilon_3 \rangle_\pm + 2\mu \langle \varepsilon_a \rangle_\pm \right] \underline{n_a} \otimes \underline{n_a} \tag{2.14}$$

where the operators $\langle \rangle_\pm$ are defined as $\langle x \rangle_\pm = \frac{1}{2}(x \pm |x|)$ and $\underline{n_a}$ are the directions of the principal strain components. Such a representation of the damaged stress emulates contact of cracked faces. The applicability of the present model is currently limited to isotropic small-strain elastic materials. This model has been utilized to study the intergranular fracture behavior in UO_2 fuels [7] and quasi-brittle fracture in nuclear-grade graphites. A typical failed configuration of a representative volume and the stress-strain evolution as obtained from phase-field fracture simulations of these materials are shown in Figures 2.3 and 2.4.

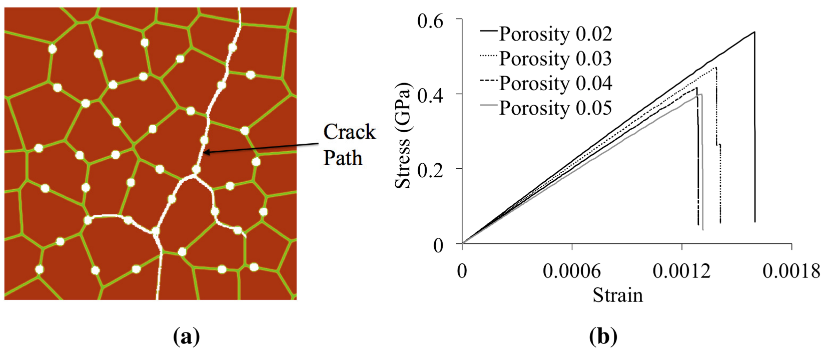


Figure 2.3: Modeling of microstructure scale intergranular fracture in UO_2 using phase-field fracture. (a) Crack propagation in one of the microstructures pulled on the top and right faces with a displacement ratio of 0.7. Crack arrest at triple points and branching thereof can be observed. (b) Stress-strain evolution under uniaxial straining of microstructures with different porosity.

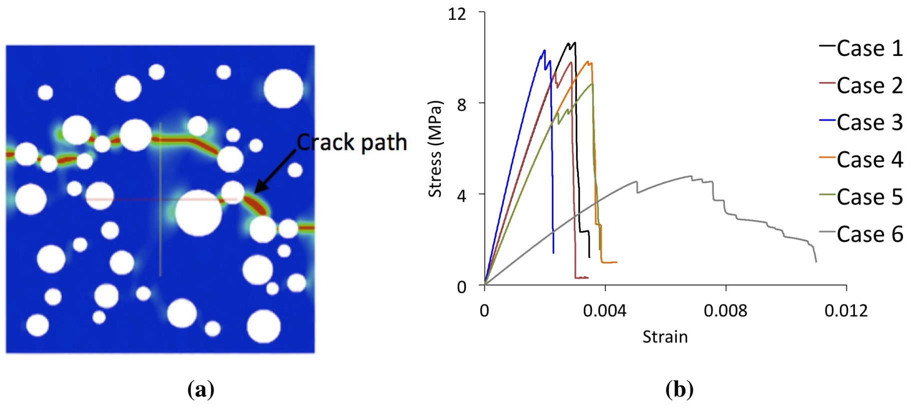


Figure 2.4: Modeling of microstructural scale quasi-brittle fracture in nuclear-grade graphite H-451. A log-normal distribution of pores with an area fraction of 0.2 has been considered. (a) Crack propagation in one of the microstructures pulled on the top face. The right face is kept traction-free; (b) Stress-strain evolution in the microstructure for different loading cases: Case-1 and 2 are for uniaxial loading, Case-3 for biaxial tensile loading, Case-4 and 5 for biaxial tension-compression loading with the same loading ratio and Case-6 for biaxial tension-compression with a loading ratio of 0.5.

2.6 Phonon transport

We have started initial investigations on coupling the Boltzmann transport equations of the MOOSE based Rattlesnake application to MARMOT microstructure simulations for calculating the phonon transport in microstructure samples. Phonons, as charge neutral quasi-particles, can be simulated with equations very similar to the equations that govern neutron transport. By mapping the corresponding mean free paths and group velocities to the phonon transport properties we were able to simulate the phonon transport in simple microstructures that result from phase field simulations or are reconstructed from experimental data. Future developments of this capability will allow us to determine the thermal conductivity of nano structured fuels and structural materials with precipitates and defect clusters.

3 Tensor Mechanics Capabilities

The `tensor_mechanics` module contains the MOOSE components implementing a comprehensive set of solid mechanics models. It covers small strain linear elasticity, finite strain elasticity, and various plasticity models. In this chapter we give an overview over the latest developments in the mechanics capabilities of MARMOT.

3.1 Modular Mechanics

During FY15 the tensor mechanics module was redesigned using a more modular design. The computation of basic mechanics properties such as stress, strain, and elasticity tensors has been split into separate MOOSE material classes, allowing for the combination of different modules into a variety of mechanics models.

3.1.1 Strain

In the MOOSE finite element formulation the variables being solved for are the mesh displacements. The strain is calculated using one of multiple material classes. For the small strain approximation the `ComputeSmallStrain` material computes the strain as $\varepsilon = \frac{1}{2}((\nabla d) + (\nabla d)^T)$, where (∇d) is the tensor with the displacement gradients as column vectors. For incremental finite strain formulations the `ComputeFiniteStrain` material provides a strain tensor utilizing stateful material properties. Both formulations exist for 2D strains, plane strains, and axisymmetric RZ strains. The stress free strain (or Eigenstrain) is added using classes derived from `ComputeStressFreeStrainBase`. Materials for constant Eigenstrains and variable (e.g. composition and temperature) dependent Eigenstrains are provided.

3.1.2 Stress

The stress response of the material can be computed using either linear elasticity (`ComputeLinear-ElasticStress`) or plasticity (`ComputeMultiplasticityStress`).

3.1.3 Elasticity tensor

The elasticity tensor is provided by materials derived from `ComputeElasticityTensorBase`. Constant and variable dependent material classes exist. Convenience classes for specifying isotropic material properties using combinations of the bulk modulus, Lamé's constant, shear modulus, Young's modulus, and Poisson's ratio as well as options to provide rotation angles for anisotropic elasticity tensors are provided.

3.2 Crystal Plasticity

Crystal plasticity models can capture grain-level anisotropy in plastic deformation behavior by considering dislocation glide on individual slip systems. The total deformation gradient (F) is multiplicatively split into an elastic (F^e) and plastic (F^p) part as

$$\underline{\underline{F}} = \underline{\underline{F}}^e \underline{\underline{F}}^p \tag{3.1}$$

where F^p evolves in the intermediate configuration following

$$\dot{\underline{\underline{F}}}^p \underline{\underline{F}}^{p-1} = \sum_{\alpha} \dot{\gamma}^{\alpha} \underline{m}_0^{\alpha} \otimes \underline{n}_0^{\alpha} \tag{3.2}$$

In equation 3.2, $\dot{\gamma}^{\alpha}$ is the glide rate on an individual slip system, α, and, $\underline{m}_0^{\alpha}$ is the direction of glide and $\underline{n}_0^{\alpha}$ is the glide plane normal in the intermediate configuration. The glide rate is based on the power law

$$\dot{\gamma}^{\alpha} = \dot{\gamma}_0 \left| \frac{\tau^{\alpha}}{s^{\alpha}} \right|^m sgn\left(\tau^{\alpha}\right) \tag{3.3}$$

where $\dot{\gamma}_0$ is a reference strain rate, m is the power-law exponent, τ^{α} are the resolved shear stresses and s^{α} are the slip system resistances on individual slip systems. The resolved shear stress on a slip system can be obtained from

$$\tau^{\alpha} = \underline{\underline{T}}^* : \underline{m}_0^{\alpha} \otimes \underline{n}_0^{\alpha} \tag{3.4}$$

where $\underline{\underline{T}}^*$ is the 2nd Piola-Kirchhoff stress in the intermediate configuration. The evolution of the slip system resistances follows

$$\dot{g}^{\alpha} = \sum_{\beta} q^{\alpha\beta} h^{\beta} \left| \dot{\gamma}^{\beta} \right| \tag{3.5a}$$

$$h^{\alpha} = h_0 \left| 1 - \frac{s^{\alpha}}{\tau_0} \right|^a sgn\left(1 - \frac{s^{\alpha}}{\tau_0}\right) \tag{3.5b}$$

and is based on [8]. In equation 3.5, $q^{\alpha\beta}$, h_0, τ_0 and a are parameters governing the self and latent hardening behavior. The implicit integration algorithm proposed in [9] has been employed to integrate the equations. A sub-stepping algorithm has also been implemented to enable efficient integration of the model. The distribution of one of the Euler angles, normal component of F^p along the loading direction in a Voronoi tessellated microstructure and the stress-strain evolution obtained from this model is shown in Figure 3.1.

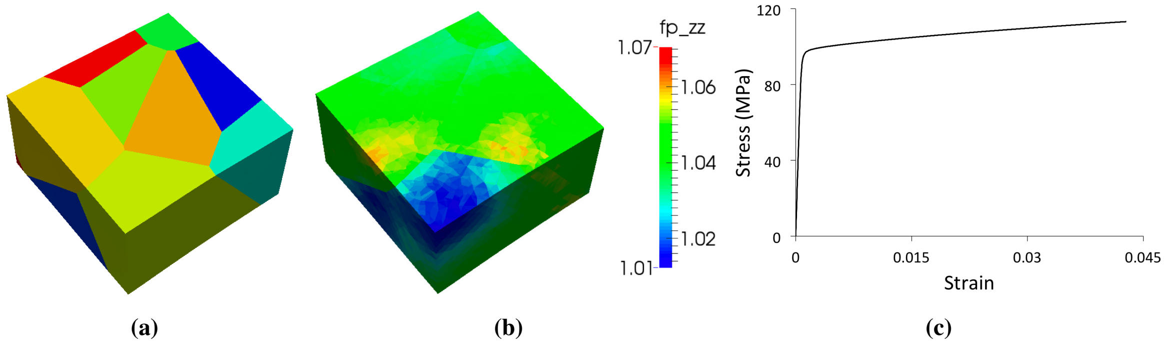


Figure 3.1: Demonstration of crystal plasticity model applied to a Voronoi tessellated polycrystalline microstructure. (a) One of the component of Euler angles on a plane. (b) Distribution of normal component of F^p along the loading direction on that plane. (c) Stress-strain evolution.

This model has been extended in Grizzly for ferritic/ferritic-martensitic steels to capture the effect of irradiation damage on localized plasticity and softening at high dose-levels.

3.3 Hyperelastic Viscoplastic model

The hyperelastic viscoplastic model solves the general rate dependent multi-plasticity equations in the intermediate configuration. Similar to equation 3.1, the total deformation gradient is multiplicatively split and the evolution of the plastic component of deformation gradient is represented by

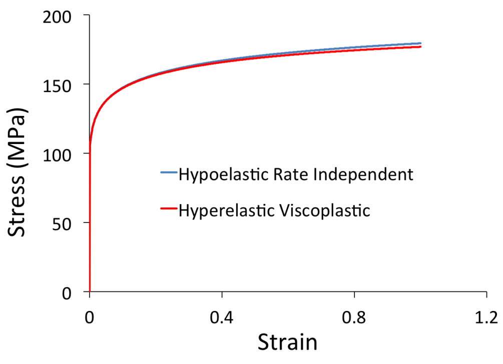
$$\dot{\underline{\underline{F}}}^p \underline{\underline{F}}^{p-1} = \sum_{i=1}^{N} \dot{\lambda}^i \underline{\underline{r}}_i \tag{3.6}$$

where N is the number of plastic flow models, $\dot{\lambda}^i$ is the flow rate for any flow model and $\underline{\underline{r}}_i$ is the associated flow direction. In essence, this model can concurrently handle multiple flow rules and potentials. Presently the flow potential for J_2 plasticity has been implemented and the flow rate ($\dot{\lambda}^i$) is defined using the power law as

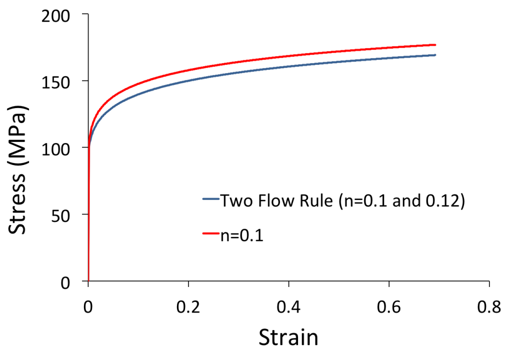
$$\dot{\lambda} = \lambda_0 \left(\frac{\sigma_{eq}}{\sigma_{YS}(\varepsilon^p_{eq})} \right)^m \tag{3.7}$$

where σ_{eq} is the equivalent stress, λ_0 is the reference flow rate, σ_{YS} is the yield stress and ε^p_{eq} is the equivalent plastic strain. Any other flow potential and flow rate model can be implemented within this framework by inheriting from the J_2 plasticity user object. The implicit backward Euler scheme is used to integrate the rate equations and the residual equation is constructed for the $\dot{\lambda}^i$ variables. The internal variables and the 2^{nd} Piola-Kirchhoff (PK2) stress is updated at every Newton-Raphson iteration for $\dot{\lambda}^i$. The calculation of the PK2 stress in this model is performed in an user object. This also allows the incorporation of damage models in a coupled manner as opposed to more standard staggard algorithms.

Initial comparison of this model has been performed with the hypoelastic rate independent J_2 plasticity model. Near rate independency in this model has been achieved by selecting a large value for m (= 50). The comparison of the stress-strain evolution is shown in Fig. 3.2(a). A comparison of the stress-strain as obtained from single and 2 flow model is shown in Fig. 3.2(b). Both the models use J_2 plasticity with Ramberg-Osgood hardening. However, the hardening exponent (n) is different for the two models.



(a)



(b)

Figure 3.2: (a) Comparison of stress-strain evolution between hyperelastic viscoplastic and hypoelastic rate independent plastic models. (b) Comparison of stress-strain evolution between single and 2 flow models. A Ramberg-Osgood hardening model with hardening exponent n = 0.1 and 0.12 has been used in the different models.

4 Material Models

MARMOT contains specific material models for nuclear applications. We briefly list the existing capabilities in this chapter.

4.1 General material models

The parsed material system with the automatic differentiation support allows us to move a large portion of the material model development from the source code into input files. This allows rapid turnaround times during model development and allows external contributors to develop and share material models without programming expertise.

4.1.1 Iron chromium

We have implemented the iron chromium model by [10] (as shown in 2.3) which provides the free energy surface of an iron chromium binary alloy as a function of chromium concentration and temperature. We entered the full free energy expression from this publication into a MOOSE input file using a parsed function material with automatic differentiation. A running phase field model was obtained in a matter of minutes. In agreement with the published free energy surface and resulting phase diagram we observe practically no solubility of iron in the chromium precipitates, while the chromium solubility in the iron matrix is at around 6.7% at the simulation temperature of 500 K.

4.1.2 CALPHAD

The thermodynamic database effort by the CALPHAD project has generated a wealth of free energy and mobility data that could be of use for constructing future multi-phase and multi-component phase field models. We have developed a set of classes that implement a subset of the existing CALPHAD functional forms. This allows users to specify parameterizations for CALPHAD type free energies found in the literature. This has been tested on the zirconiumhydride and uranium-zirconium system. These classes utilize the ExpressionBuilder system (2.1.3) internally.

4.1.3 Polycrystalline materials

An isotropic grain boundary energy model for polycrystalline samples with multiple components (e.g. gas bubbles) is available in MARMOT and has been used to model polycrystalline metals and UO_2 fuels. The polycrystalline material system supports individual grain properties such as varying orientations and anisotropic mechanical properties. A preliminary version of an anisotropic grain boundary energy model that is fit to atomistic grain boundary energy data as function of relative grain orientation and grain boundary plane is included in the 1.0 release of MARMOT. This so-called *five degrees of freedom* model is in active development and is based on a paper by Bulatov *et al.* [11]. We are working on generating a grain boundary energy function fit for the UO_2 system.

4.2 Specific material models

4.2.1 Uranium dioxide xenon

We have implemented a UO_2Xe model using the Kim-Kim-Suzuki (KKS) multiphase model [12]. KKS addresses the issue of systems with large phase free energy differences in the interfacial regions. The Xenon gas solubility in the solid UO_2 matrix is very low, with large free energy penalties for large gas concentrations. In the bubble phase the equilibrium gas concentration is determined by the vacancy to gas atom ratio of a gas. In the bubble matrix interfacial region both the order parameter as well as the concentration change from the bubble equilibrium values to the matrix equilibrium values over a finite distance due to the soft interface approximation. In that interfacial region the phase free energy of both phases is computed for the intermediate concentration range, which results in large free energy densities from the solid phase contribution. This effectively increases the interfacial free energy of the bubbles significantly to an unphysical value.

The KKS model solves this by introducing the concept of phase concentrations, which are effectively the fractions of the total concentration held in a given phase. In this model the gas concentration is largely shifted to the gas phase to avoid the free energy penalty. Solving for these added variables requires additional differential equations. In the KKS model these are given by mass conservation equations and a constraint that requires the chemical potentials of each component to be pointwise equal across all phases.

4.2.2 Uranium zirconium

We implemented the uranium zirconium free energy and mobility model from Chevalier *et al.* [13] using the ExpressionBuilder system (2.1.3). Details of this U-Zr model can be found in the AFC report [14]. This effort is ongoing and will be continued in the future.

4.3 Initial conditions

MARMOT contains a library of initial condition classes that are specifically designed to setup commonly used microstructures at the beginning of a simulation. These microstructures can be used as starting points for simulating the microstructural evolution of the material systems of interest. Initial condition data can also be use to perform virtual experiments on fixed microstructures, such as mechanical property or thermal conductivity measurements.

We supply a series of initial conditions to set up patterns of precipitates or gas bubbles, polycrystalline samples (geometric or randomized Voronoi tessellations), and perturbed interface structures. A capability specifically developed for MARMOT is the EBSD reader system which allows the construction of simulation samples from experimentally characterized microstructures from EBSD sectioning data or X-ray tomography data.

4.4 Fission Gas Release

Apart from the capability for microstructure based simulation of the diffusion and precipitation of fission gas in polycrystalline samples MARMOT contains an implementation of the analytical fission gas release model by Pastore *et al.* [15]. The model implements a 1D spherical grain approximation that can be coupled to a BISON simulation using the MOOSE MultiApps capability. Multiple MARMOT child simulations are managed by a BISON master simulation. BISON computes the global temperature and grain boundary fission gas distribution within a fuel pellet, while MARMOT computes the grain boundary coverages and the fractions of trapped and dissolved fission gas.

5 Conclusions

The MARMOT tool being developed by the NEAMS FPL predicts the coevolution of microstructure and material properties in fuel and cladding materials due to temperature, stress, and irradiation damage. MARMOT is built using the capabilities available in the `phase_field`, `tensor_mechanics`, and `heat_conduction` modules in the open source MOOSE framework. Though MARMOT is being developed as a mesoscale fuel performance tool, it is also being used to inform engineering scale fuel performance codes to develop microstructure rather than burn-up based material models.

MARMOT has built-in capabilities for constructing multi-phase multi-component phase field models that are fully coupled to a solid mechanics solve. The mechanical properties can be phase and composition dependent. Our modular free energy system with automatic differentiation allows for incremental model development where pieces fit together like LEGO bricks. We believe this makes MARMOT a powerful research platform beyond its immediate nuclear applications, providing a flexibility and extensibility that will ensure its usefulness in future NEAMS applications.

MARMOT is developed by a community of users and the code base is managed with the Git version control system. It is hosted on an internal INL Gitlab repository which has automated tools for issue tracking, automated testing, and collaborative code review. The MARMOT code base undergoes rigorous quality assurance and verification throughout its development, and the materials models are also being validated when data is available. The code is verified using the method of manufactured solution, comparison to analytical models, and comparison to molecular dynamics simulations.

MARMOT has been funded by the NEAMS program to develop models for UO_2. Up to now, this development has been focused on models of fission gas behavior, grain growth, cracking, and effective material property calculation. Starting in fiscal year 2015, MARMOT is also being funded to develop models of U-Si fuel, though this efforts are waiting for the development of a U-Si inter-atomic potential. In FY 2015 the Advanced Fuel Campaign under the Fuel Cycle Research & Development (FCRD) program and the U.S. High-Performance Research Reactors (USHPRR) program are also funding the development of metal fuels models in MARMOT.

Future work on MARMOT will include the development of new materials models. A basis for such developments will be an improved implementation of the KKS model to support multi-phase systems with arbitrary numbers of phases. We will be working on novel radiation damage models by coupling MARMOT to Monte Carlo methods for the simulation of the primary damage phase. Enhancements are planned for the brittle fracture models and model for coupling crystal plasticity with phase field is planned. Work will continue on supporting anisotropic interface properties, coupling to thermodynamic databases, and supporting CALPHAD free energy models.

Bibliography

[1] M. R. Tonks, D. Schwen, Y. Zhang, P. Chakraborty, X. Bai, B. Fromm, J. Yu, M. C. Teague, and D. A. Andersson. Assessment of MARMOT: A mesoscale fuel performance code. Technical Report INL/EXT-15-35108, Rev. 1, Idaho National Laboratory, 2015.

[2] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandié. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. *Nucl. Eng. Design*, 239:1768–1778, 2009.

[3] M.R. Tonks, D. Gaston, P.C. Millett, D. Andrs, and P. Talbot. An object-oriented finite element framework for multiphysics phase field simulations. *Comp. Mat. Sci.*, 51(1):20–29, 2012.

[4] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[5] Benjamin S. Kirk, John W. Peterson, Roy H. Stogner, and Graham F. Carey. libMesh: A C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3-4):237–254, January 2006.

[6] Yu U. Wang. Computer modeling and simulation of solid-state sintering: A phase field approach. *Acta Materialia*, 54(4):953–961, February 2006.

[7] P Chakraborty, Y Zhang, and M R Tonks. Multi-scale modeling of microstructure dependent intergranular brittle fracture using a quantitative phase-field based method. *Computational Materials Science*, Submitted, 2015.

[8] S.R. Kalidindi, C.A. Bronkhorst, and L. Anand. Crystallographic texture evolution in bulk deformation processing of fcc metals. *Journal of the Mechanics and Physics of Solids*, 40(3):537 – 569, 1992.

[9] S. Balasubramanian. *Polycrystalline Plasticity: Application to Deformation Processing of Lightweight Metals*. PhD thesis, Massachusetts Institute of Technology, 1998.

[10] D Schwen, E Martinez, and A Caro. On the analytic calculation of critical size for alpha prime precipitation in fecr. *Journal of Nuclear Materials*, 439(1):180–184, 2013.

[11] Vasily V. Bulatov, Bryan W. Reed, and Mukul Kumar. Grain boundary energy function for fcc metals. *Acta Materialia*, 65:161 – 175, 2014.

[12] Seong Gyoon Kim, Won Tae Kim, and Toshio Suzuki. Phase-field model for binary alloys. *Phys. Rev. E*, 60:7186–7197, Dec 1999.

[13] Pierre-Yves Chevalier, Evelyne Fischer, and Bertrand Cheynet. Progress in the thermodynamic modelling of the OUZr ternary system. *Calphad*, 28(1):15–40, March 2004.

[14] K. A. Gamble, R. L. Williamson, D. Schwen, Y. Zhang, S. R. Novascone, and P. Medvedev. BISON and MARMOT development for modeling fast reactor fuel performance. Technical Report INL/EXT-15-36440, Rev. 1, Idaho National Laboratory, 2015.

[15] G. Pastore, L. Luzzi, V. Di Marcello, and P. Van Uffelen. Physics-based modelling of fission gas swelling and release in UO_2 applied to integral fuel rod analysis. *Nucl. Engrg. Design*, 256:75–86, 2013.