Overview of the MOOSE Framework and Applications to Materials Science

Larry Aagesen, Yongfeng Zhang, Daniel Schwen, Pritam Chakraborty, Bulent Biner, Jianguo Yu, Chao Jiang, Ben Beeler, Wen Jiang, Andrea Jokisaari, Yipeng Gao, Sudipta Biswas

Michael Tonks

Xianming Bai

Paul Millett

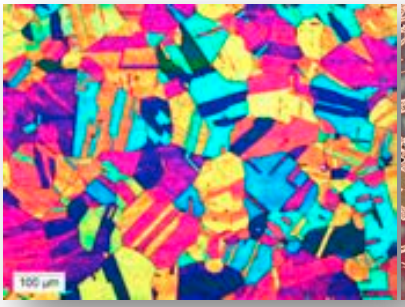
Karim Ahmed

Overview

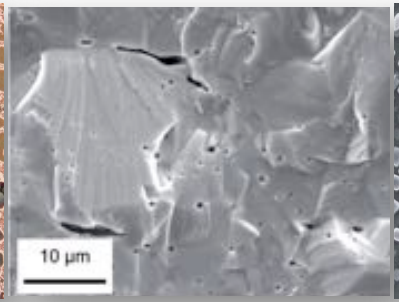
- General overview of the MOOSE framework
- MOOSE tools for meso-scale modeling
- Phase-field modeling
- Examples and applications

Material Behavior

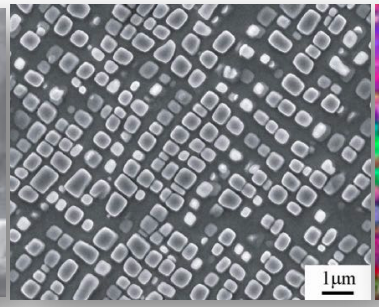
- A key objective of materials science is to understand the impact of microstructure on macroscale material behavior.


Annealed Stainless Steel

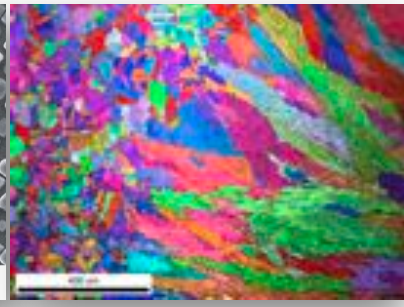

Cast Bronze



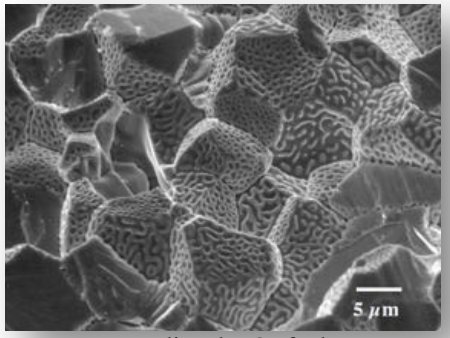
Sintered UO2



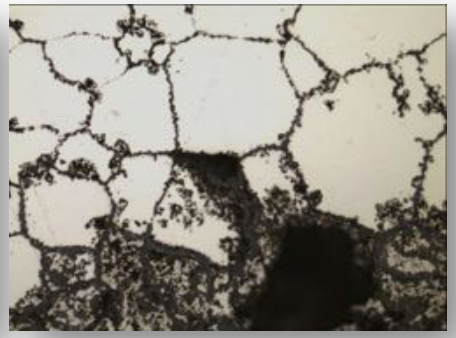
Co-Al-W Superalloy

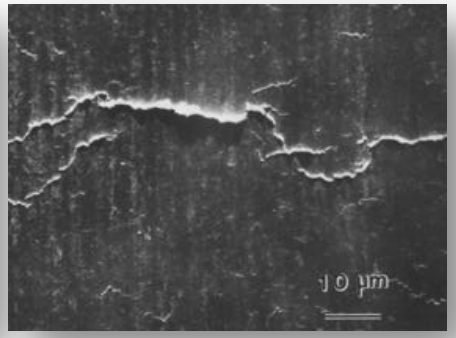

Friction stir welded stainless steel

- An essential part of that is predicting the impact of microstructure evolution.



Irradiated UO_2 fuel


Corrosion in stainless steel



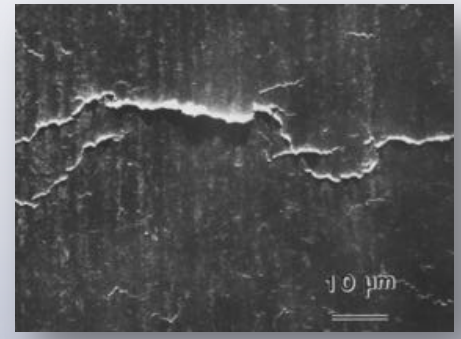
Micro-cracking in steel

Hydride in Zircaloy

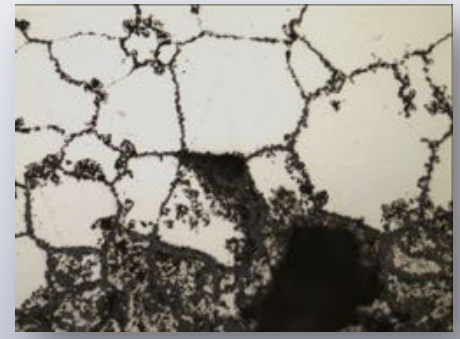
Material Behavior is Multiphysics

- Material behavior is influenced by many different physics, for example:

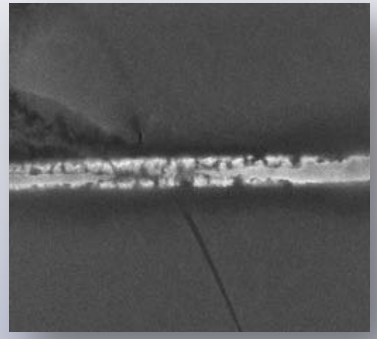
Mechanics
- Dislocations
- Cracking
- Stress-driven Diffusion



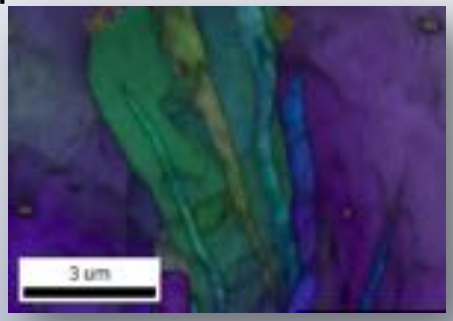
Chemistry
- Corrosion
- Oxidation
- Reactive transport



Electricity/Magnetism
- Electromigration
- Ferroelectricity
- Ferromagnetism

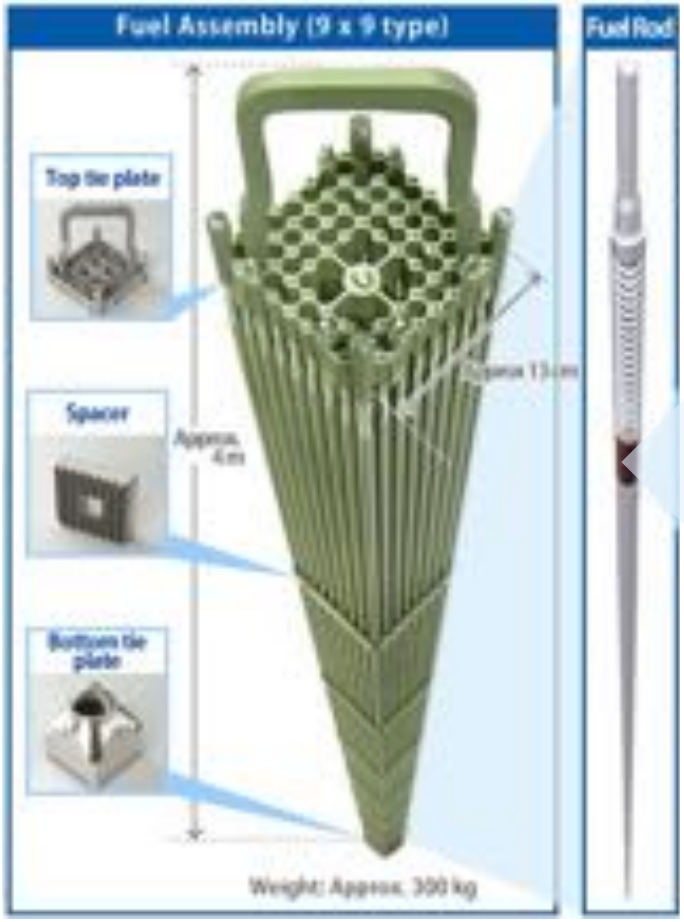


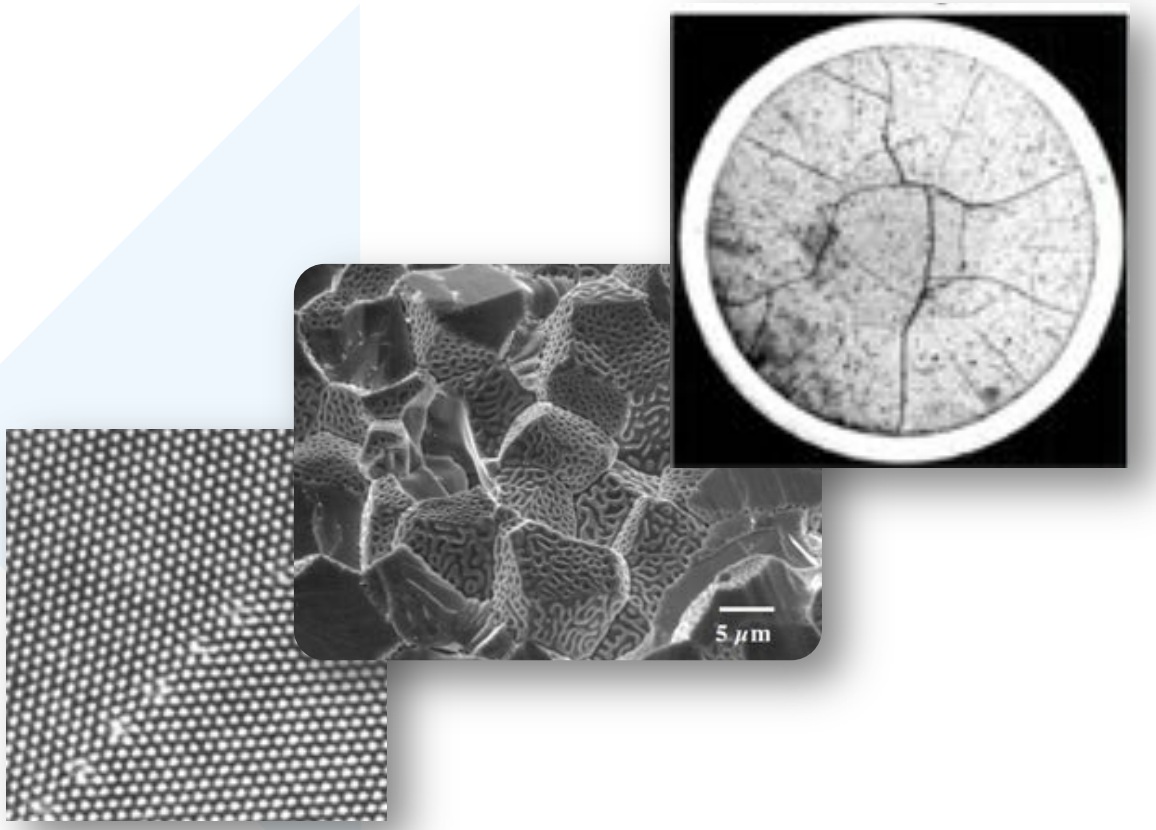
Heat Conduction
- Species transport
- Melting
- Precipitation



Material Behavior is Multiscale

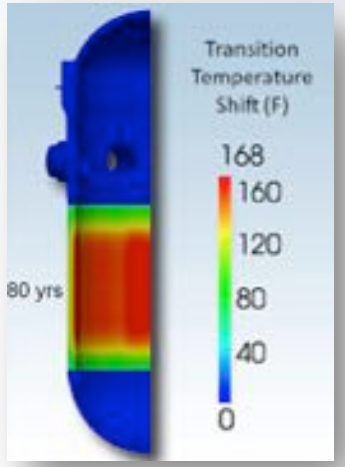
- Material behavior at the atomistic and microscales drives macroscale response.





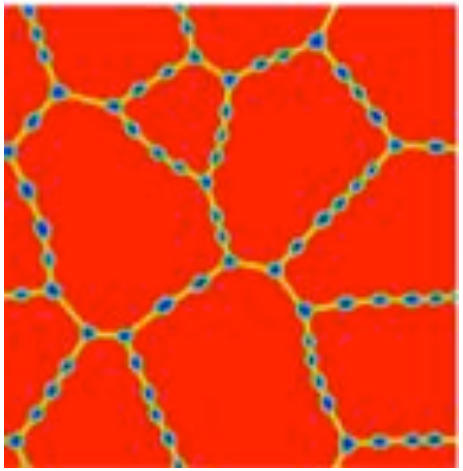
Multiscale Modeling Approach

- Simulations at smaller scales inform the models at increasing length scales



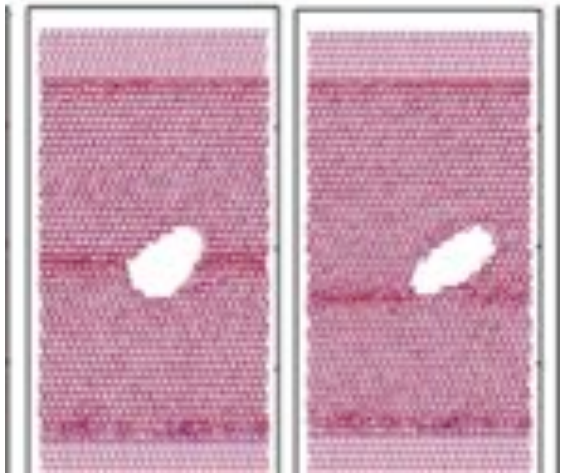
Engineering scale simulation

- Predictive modeling at the engineering scale



Mesoscale models

- Predict and define microstructure evolution
- Determine effect of evolution on material properties

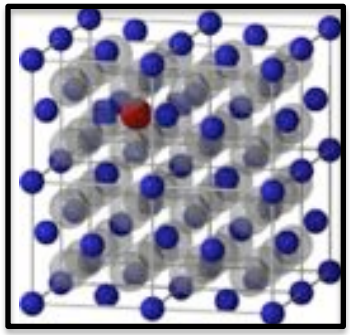


Atomic scale microstructure MD

- Investigate role of idealized interfaces
- Determine interfacial properties

Atomic scale bulk DFT + MD

- Identify important bulk mechanisms
- Determine bulk material parameters

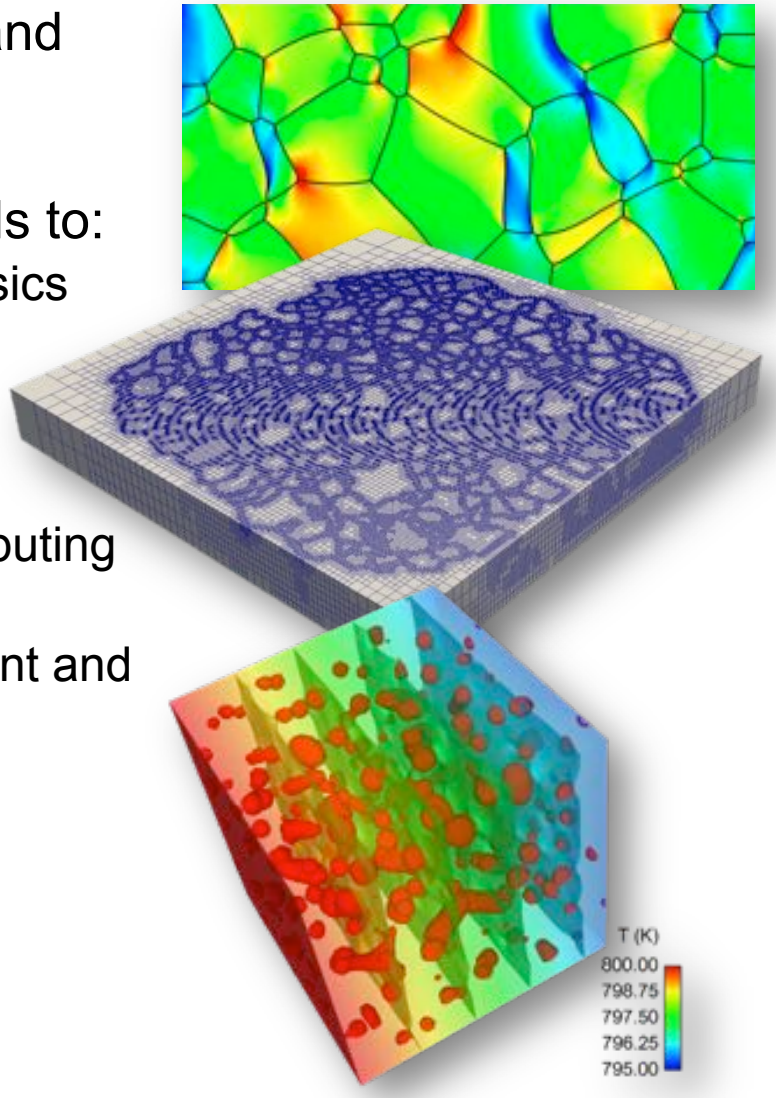


nm μm mm

Lengthscale

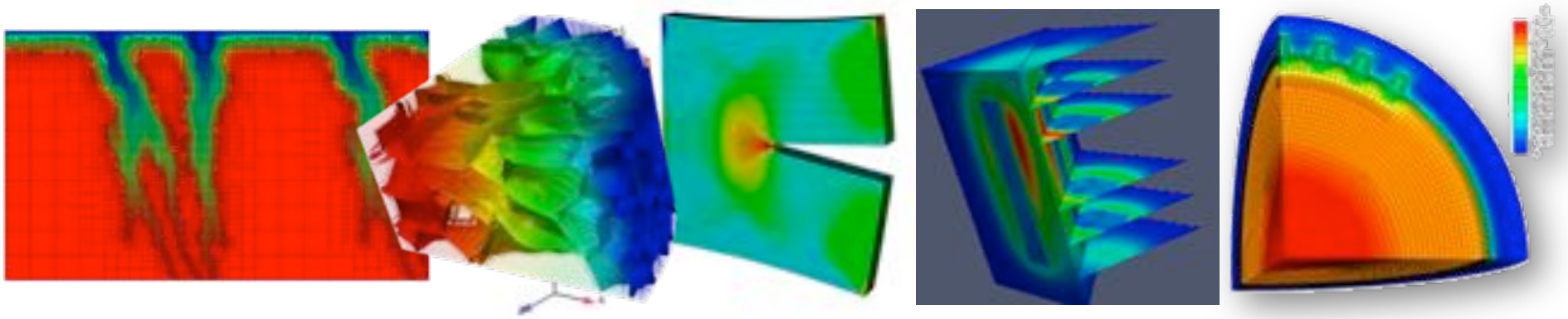
Materials Modeling Requirements

- To model material behavior at the meso- and macroscales requires that we deal with its inherent complexity.
- A tool for modeling material behavior needs to:
  - Easily handle multiple, tightly coupled physics
  - Have tools for multiscale modeling
- It would also be nice if it
  - Were simple to use and develop
  - Took advantage of high performance computing
  - Were free and open source
  - Had a team of full time staff for development and support
  - Had a strong user community



Multiphysics Object Oriented Simulation Environment

- MOOSE is a finite-element, multiphysics framework that simplifies the development of advanced numerical applications.
- It provides a high-level interface to sophisticated nonlinear solvers and massively parallel computational capability.

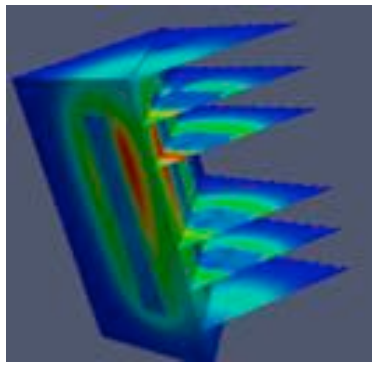


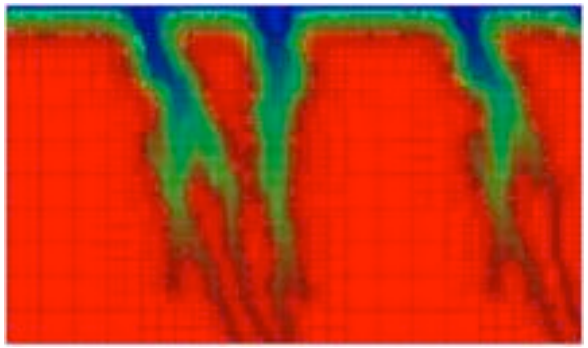
- MOOSE has been used to model thermomechanics, neutronics, geomechanics, reactive transport, microstructure modeling, computational fluid dynamics, and more every day!
- It is open source and freely available at mooseframework.org

MOOSE

- Tool for develop simulation tools that solve PDEs using FEM

- Spatial discretization with finite elements, where each variable can use a different element type, i.e. different shape functions
- Easy to couple multiple PDE
- Implicit or explicit time integration is available
- Dimension agnostic, same code can be used in 1- to 3-D
- Inherently parallel, solved with one to >10000 processors
- Provides access to mesh and time step adaptivity
- Easy simulation tool development
- Can read and write various mesh formats
- Strong user community
- Newton or Jacobian free solvers.



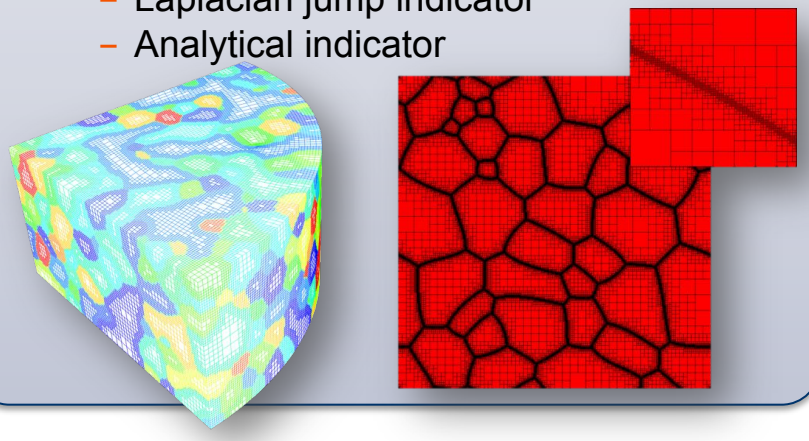


Mesh and Time Step Adaptivity

- Any model implemented with MOOSE has access to mesh and time step adaptivity

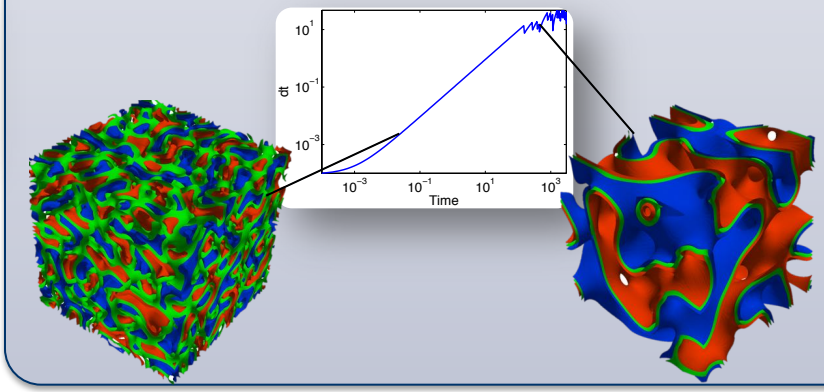
Mesh Adaptivity

- Requires no code development
- Refinement or coarsening is defined by a marker that be related to
  - An error estimator
  - Variable values
  - Stipulated by some other model
- Error indicators include the
  - Gradient jump indicator
  - Flux jump indicator
  - Laplacian jump indicator
  - Analytical indicator



Transient Time Step Adaptivity

- The time step in transient simulations can change with time
- Various time steppers exist to define dt:
  - Defined by a function
  - Adapts to maintain consistent solution behavior
  - Adapts to maintain consistent solution time
- Users can write new time steppers



Mesoscale Modeling with the MOOSE framework

- All of the code required to easily create your own phase field application is in the open source MOOSE modules (MOOSE-PF).

MOOSE

Framework

All of the code that forms the basis of the MOOSE framework

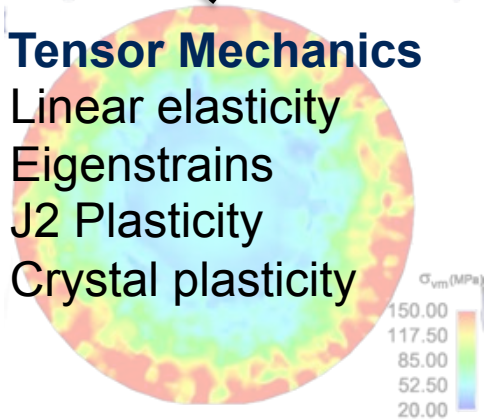
Modules

Phase Field

- Cahn-Hilliard and Allen-Cahn equations
- Free energy based development

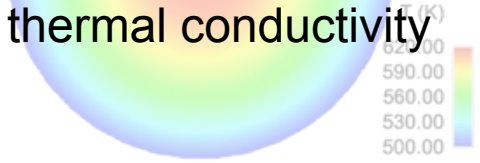
Tensor Mechanics

- Linear elasticity
- Eigenstrains
- J2 Plasticity
- Crystal plasticity



Heat Conduction

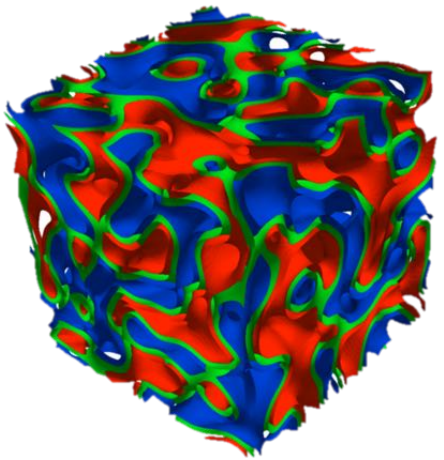
- Steady state
- Transient
- Locally varying thermal conductivity

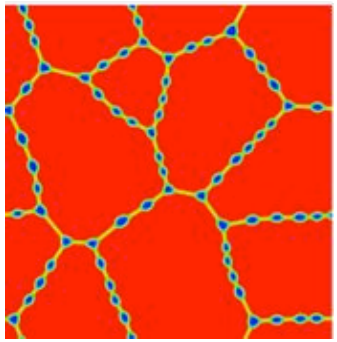


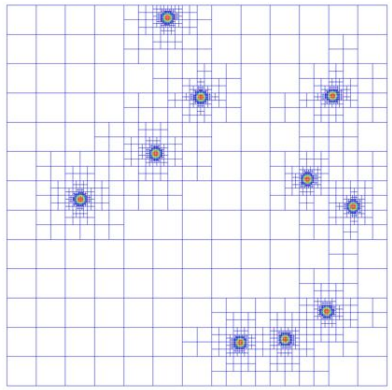
MOOSE-PF *Generic Phase Field Library*

- Provides the tools necessary to develop phase field models using FEM.

- Base classes for solving Cahn Hilliard equations
  - Direct solution
  - Split solution
- Base classes for Allen-Cahn equations
- Grain growth model
- Grain remapping algorithm for improved efficiency
- Initial conditions
- Postprocessors for characterizing microstructure



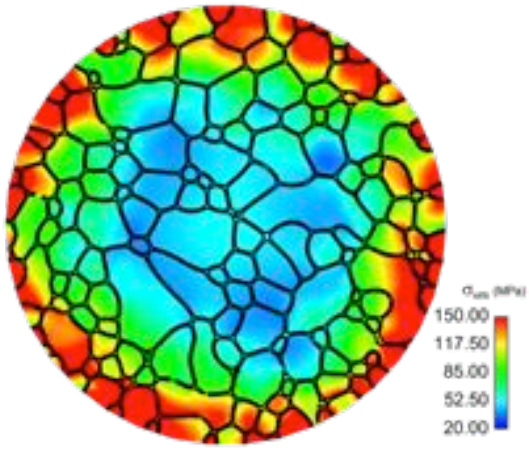


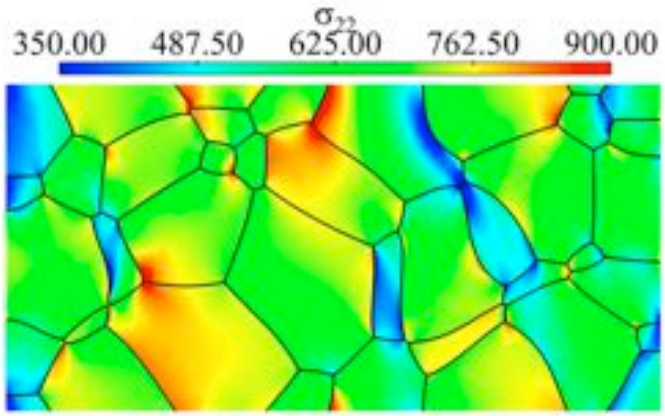


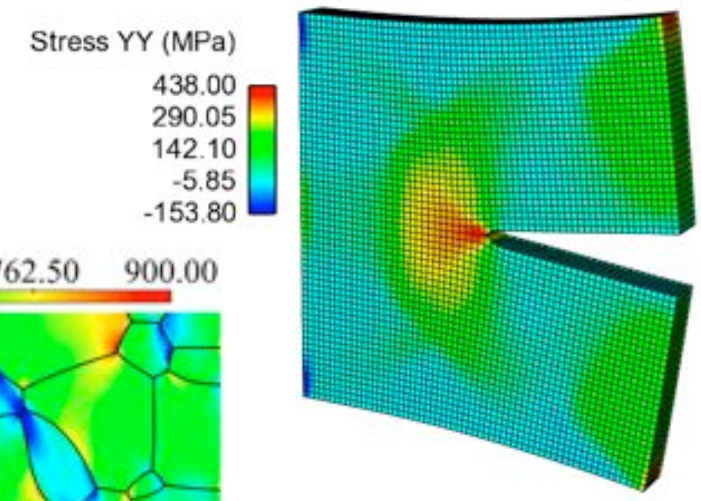
MOOSE-Tensor Mechanics

- Provides the tools necessary for modeling mechanical deformation and stress at the mesoscale.

- Anisotropic elasticity tensors that can change spatially
- Linear elasticity
- Eigen strains
- Finite strain mechanics
  - J2 plasticity
  - Crystal plasticity



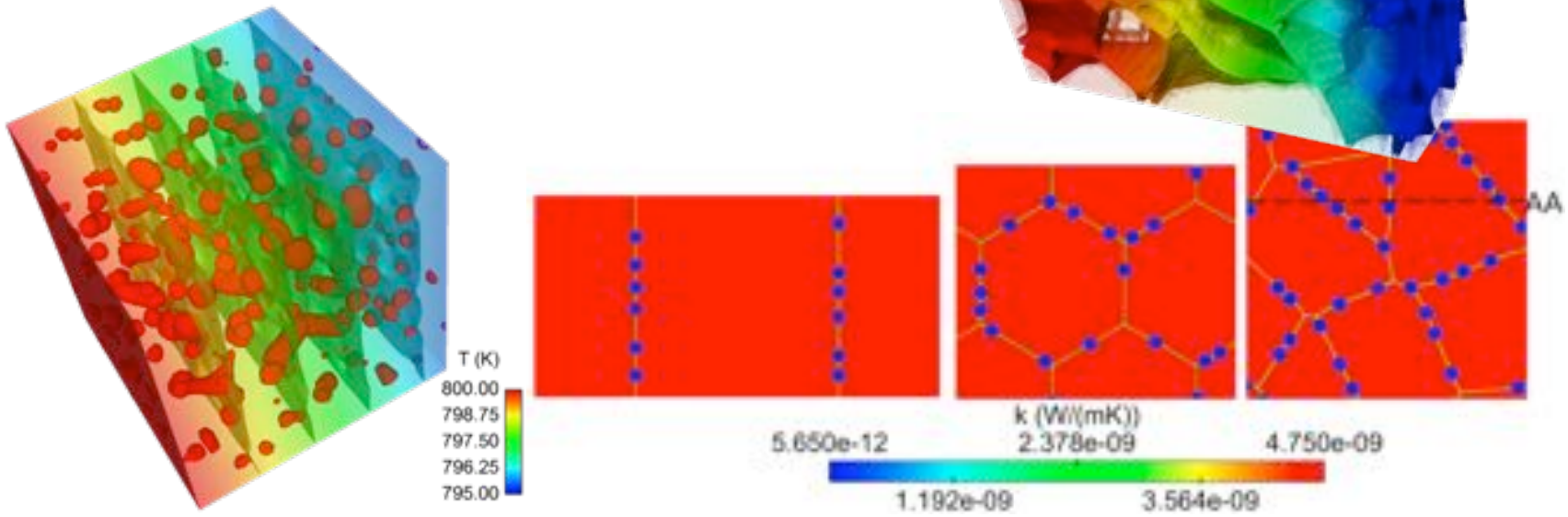




MOOSE-Heat Conduction

- Provides the tools necessary for modeling heat conduction and temperature gradients at the mesoscale.

- Steady state heat conduction
- Transient term
- Effective thermal conductivity calculation
- Spatially varying thermal conductivity



MARMOT

- Models the coevolution of microstructure and properties in reactor materials

No physics

Applicable to all materials

Specifically for reactor materials

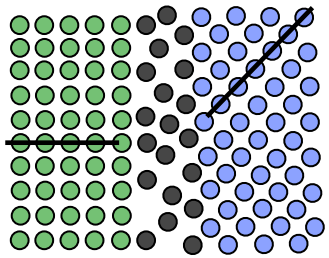
- MARMOT is in use by researchers at laboratories and universities:

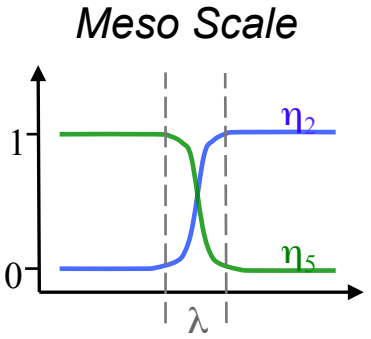
Argonne NATIONAL LABORATORY
Los Alamos NATIONAL LABORATORY EST. 1943
Washington State University
Pacific Northwest NATIONAL LABORATORY
The University of Wisconsin Madison
BYU Idaho
MIT Massachusetts Institute of Technology
Ohio State
Michigan
University of Oxford

The Phase Field Method

- Microstructure described by a set of continuous variables…
  - Non-Conserved Order Parameters

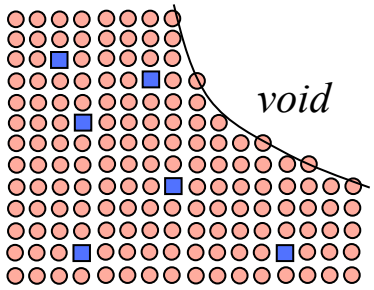
$$\frac{\partial \eta_j}{\partial t} = -L\frac{\delta F}{\delta \eta_j}$$

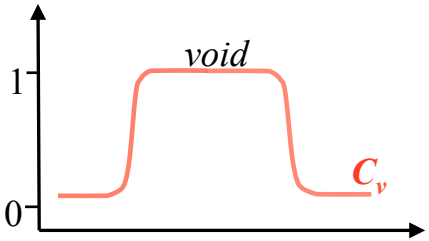




  - Conserved Concentrations

$$\frac{\partial c_i}{\partial t} = \nabla \cdot \left(M(c_i) \nabla \frac{\delta F}{\delta c_i} \right)$$

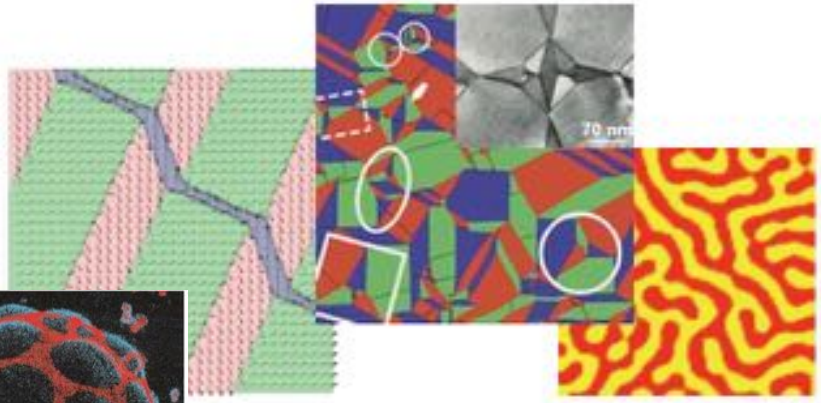




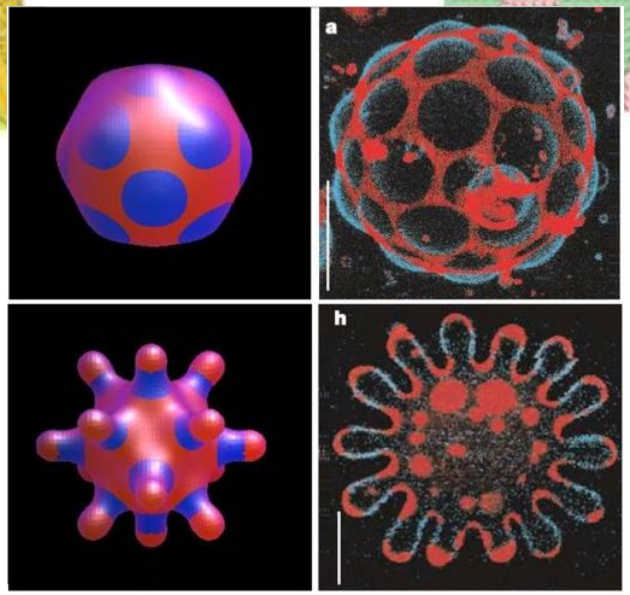
- The variables evolve to minimize a functional defining the free energy

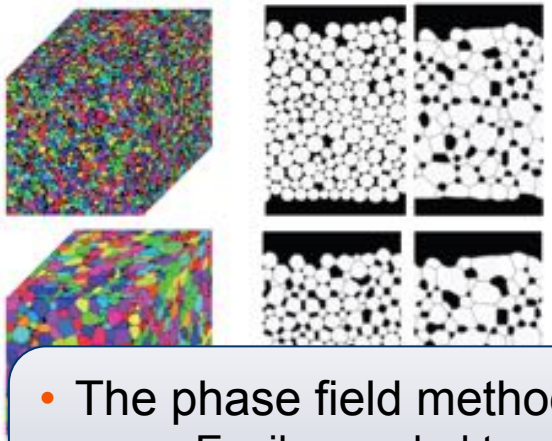
Phase Field Has Been Used in Many Areas

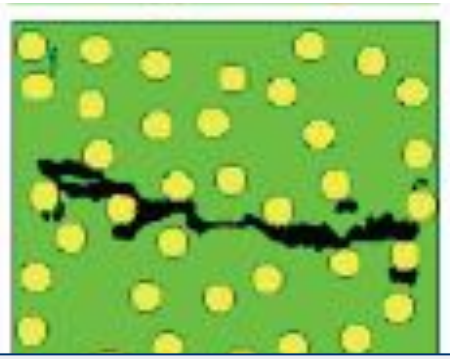
solidification (dendrite growth)



phase transformations



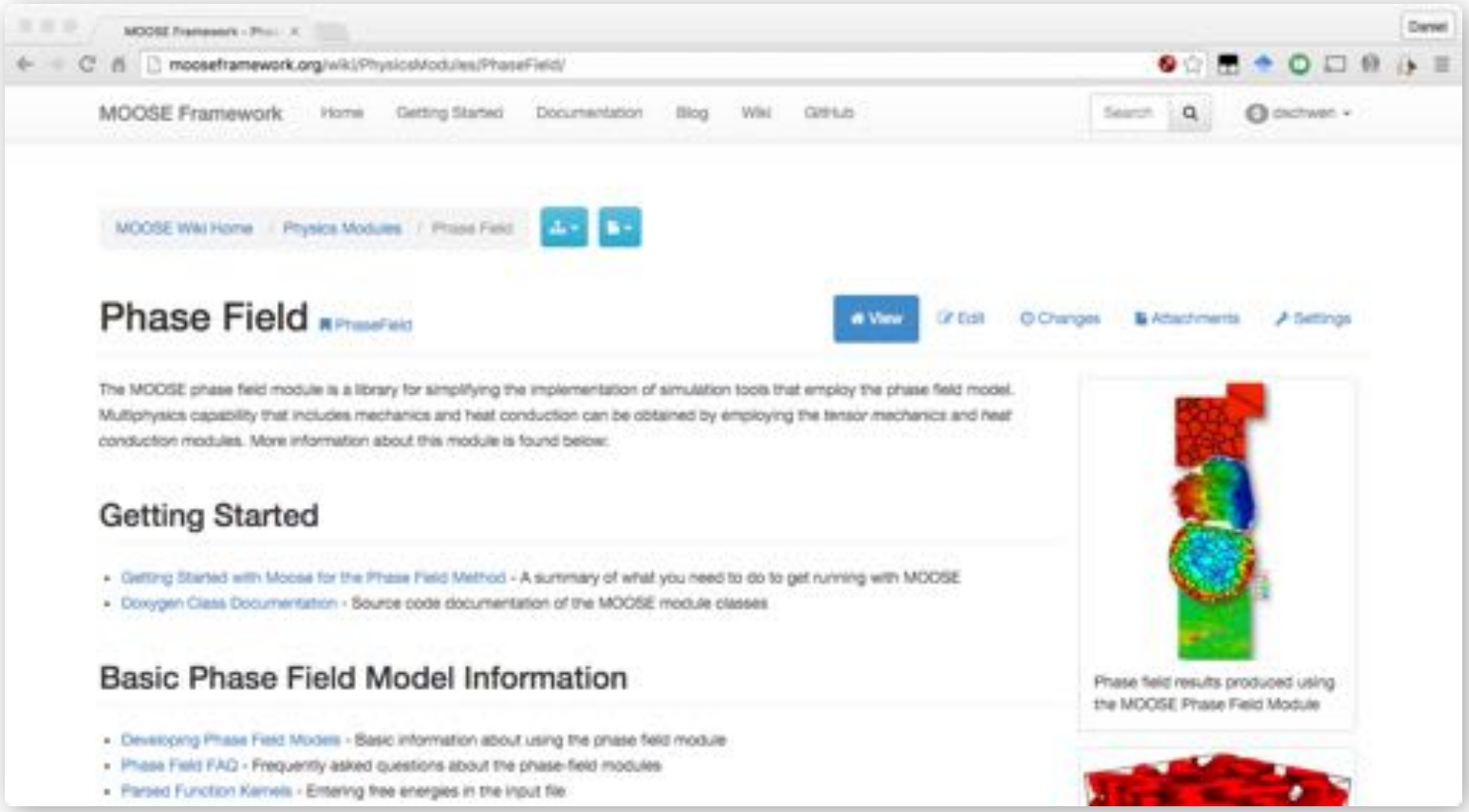




- The phase field method is our method of choice because it can be:
  - Easily coupled to other physics such as mechanics or heat conduction
  - Quantitative and can represent real materials

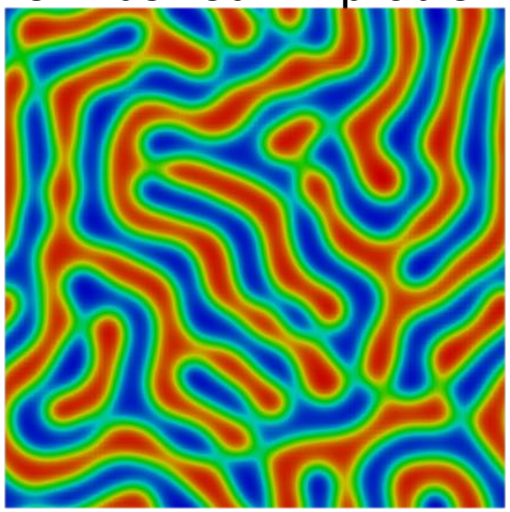
Phase Field Documentation

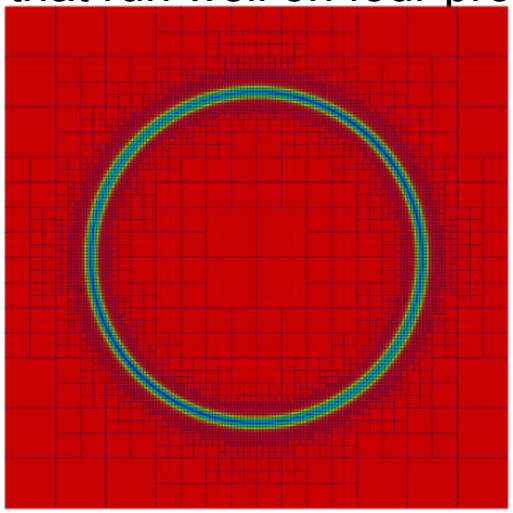
- Documentation for the phase field module is found on the mooseframework.org wiki:
  - http://mooseframework.org/wiki/PhysicsModules/PhaseField/



Examples

- Example input files for MOOSE-PF can be found in the examples directory: `moose/modules/phase_field/examples/`
  - These are midsized 2D problems that run well on four processors





- The tests can serve as additional examples
  - There are many tests for the various components of MOOSE
  - Each test runs in less then 2 seconds on one processor

The Phase Field Equations

- Non-conserved variables (phases, grains, etc.) are evolved using an **Allen-Cahn** (aka Ginzburg-Landau) type equation:

$$\frac{\partial \eta_j}{\partial t} = -L \frac{\delta F}{\delta \eta_j}$$

- Conserved variables are evolved using a **Cahn-Hilliard** type equation:

$$\frac{\partial c_i}{\partial t} = \nabla \cdot \left(M(c_i) \nabla \frac{\delta F}{\delta c_i} \right)$$

- Both equations are functions of *variational derivatives* of a functional defining the free energy of the system in terms of the variables

$$F = \int_V \left(f_{loc}(c_i, \eta_j, ..., T) + E_d + \sum_i \frac{\kappa_i}{2} (\nabla c_i)^2 + \sum_j \frac{\kappa_j}{2} (\nabla \eta_j)^2 \right) dV$$

Local energy

Gradient energy

Variational Derivative

The functional derivative (or variational derivative) relates a change in a functional to a change in a function that the functional depends on.

Wikipedia, "Functional derivative"

$$F = \int f(r, c, \nabla c)\, dV$$

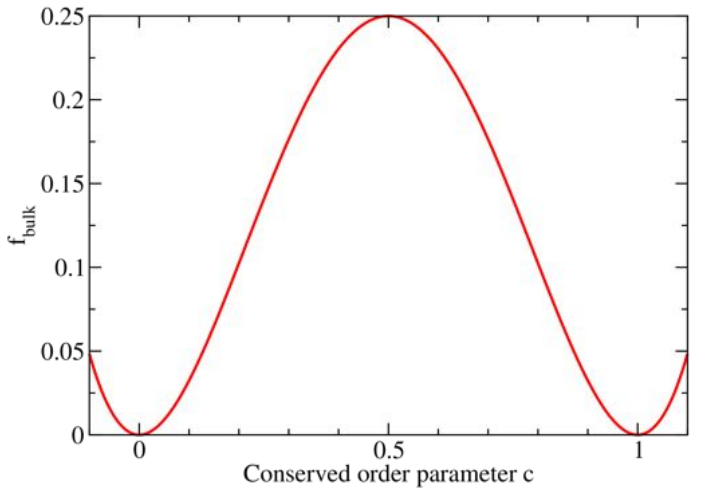
$$\frac{\delta F}{\delta c} = \frac{\partial f}{\partial c} - \nabla \cdot \frac{\partial f}{\partial \nabla c}$$

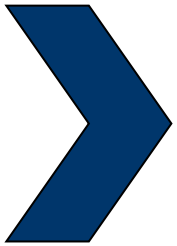
F	total free energy
f	free energy **density**

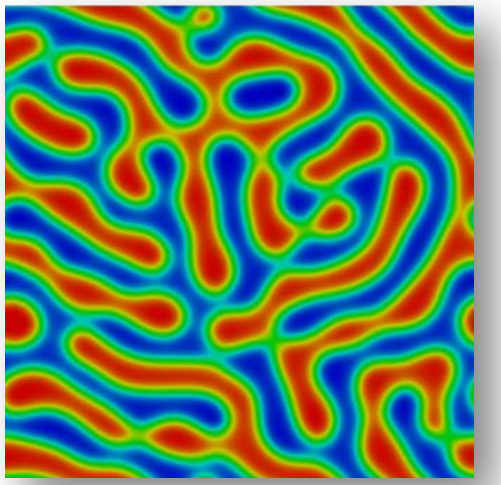
- Derivative with respect to the **gradient!**
- Gradient energy term in phase field (very few functional forms)
- Bulk free energy (contains the thermodynamics of the system)
  - Simple partial derivative

Phase Field Implementation in MOOSE

- The kernels required to solve the phase field equations have been implemented in the phase field module
- In general, a developer will not need to change the kernels but simply use the kernels that have already been implemented
- New models are implemented by defining the free energy and mobility with their derivatives in *material* objects.







Derivative Function Materials

- Each MOOSE Material class can provide **multiple Material Properties**
- A Derivative Function Material is a MOOSE Material class that provides a well defined set of Material Properties
  - A function value, stored in the material property `F` (the `f_name` of the Material)
  - All derivatives of `F` up to a given order with respect to the non-linear variables `F` depends on
- The derivatives are regular Material Properties with an enforced naming convention
  - Example `F`, `dF/dc`, `d^2F/dc^2`, `dF/deta`, `d^2F/dcdeta` ...
  - You don't need to know the property names besides `F`, unless you want to look at them in the output!
- Recap:
  Each Derivative Function Material provides **one** Function **together** with its derivatives!
- That function can be a *Free Energy Density*, a *Mobility*, or whatever you may need.

Solving the Allen-Cahn Equation

- After taking the variational derivative, the strong form of the Allen-Cahn residual equation is

$$\frac{\partial \eta_j}{\partial t} = -L\left(\frac{\partial F}{\partial \eta_j} + \frac{\partial E_d}{\partial \eta_j} - \kappa_j \nabla^2 \eta_j\right)$$

- Each piece of the weak form of the residual equation has been implemented in a kernel:

$$\mathcal{R}_{\eta_j} = \left(\frac{\partial \eta_j}{\partial t}, \psi_m\right) + \left(L_j \kappa_j \nabla \eta_j, \nabla \psi_m\right) + L_j \left(\frac{\partial f_{loc}}{\partial \eta_j} + \frac{\partial E_d}{\partial \eta_j}, \psi_m\right)$$

`TimeDerivative` `ACInterface` `AllenCahn`

- Parameters must be defined in a *material* object
- The free energy density and its derivatives are defined in a Derivative Function Material

Solving the Cahn-Hilliard Equation

- Due to the fourth-order derivative, solving the Cahn-Hilliard equation can be hard. In MOOSE there are two available approaches
  - Residual: $$\mathcal{R}_{c_i} = \frac{\partial c_i}{\partial t} - \nabla \cdot M(c_i) \left(\nabla \frac{\partial f_{loc}}{\partial c_i} + \nabla \frac{\partial E_d}{\partial c_i} \right) + \nabla \cdot M(c_i) \nabla \left(\kappa_i \nabla^2 c_i \right)$$
  - We can put this in weak form:

$$\left(\frac{\partial c_i}{\partial t}, \psi_m \right) = -(\kappa_i \nabla^2 c_i, \nabla \cdot (M_i \nabla \psi_m)) - \left(M_i \nabla \left(\frac{\partial f_{loc}}{\partial c_i} + \frac{\partial E_d}{\partial c_i} \right), \nabla \psi_m \right)$$

  - But, solving this residual requires **higher order elements**

- Another option is to split the equation into two:

Strong Form

$$\frac{\partial c_i}{\partial t} = \nabla \cdot (M_i \nabla \mu_i)$$

$$\mu_i = \frac{\partial f_{loc}}{\partial c_i} - \kappa_i \nabla^2 c_i + \frac{\partial E_d}{\partial c_i}$$

Weak Form

$$\left(\frac{\partial c_i}{\partial t}, \psi_m \right) = -\left(M_i \nabla \mu_i, \nabla \psi_m \right)$$

$$\left(\mu_i, \psi_m \right) = \left(\frac{\partial f_{loc}}{\partial c_i}, \psi_m \right) + \left(\kappa_i \nabla c_i, \nabla \psi_m \right) + \left(\frac{\partial E_d}{\partial c_i}, \psi_m \right)$$

  - The split form can be solved with **first-order elements**.

The Direct Solution of the Cahn-Hilliard Equation

- Each piece of the weak form of the Cahn-Hilliard residual equation has been implemented in a kernel

$$\mathcal{R}_{c_i} = \underbrace{\left(\frac{\partial c_i}{\partial t}, \psi_m\right)}_{\texttt{TimeDerivative}} + \underbrace{\left(\kappa_i \nabla^2 c_i, \nabla \cdot (M_i \nabla \psi_m)\right)}_{\texttt{CHInterface}} + \underbrace{\left(M_i \nabla \left(\frac{\partial f_{loc}}{\partial c_i} + \frac{\partial E_d}{\partial c_i}\right), \nabla \psi_m\right)}_{\texttt{CahnHilliard}}$$

- Parameters must be defined in a material object
- The free energy density and its derivatives are defined in an energy material object (e.g. `DerivativeParsedMaterial`)
- Mobilities can also depend on non-linear variables M(c) and can be supplied through Derivative Function Materials
- Due to the second order derivative, third order Hermite elements must be used to discretize the variables

The Split Solution of the Cahn-Hilliard Equation

- The weak form of the split Cahn-Hilliard residual equation has also been implemented in kernels:

$$\mathcal{R}_{\mu_i} = \left(\frac{\partial c_i}{\partial t}, \psi_m\right) + (M_i \nabla \mu_i, \nabla \psi_m)$$

`CoupledTimeDerivative` `SplitCHWRes`

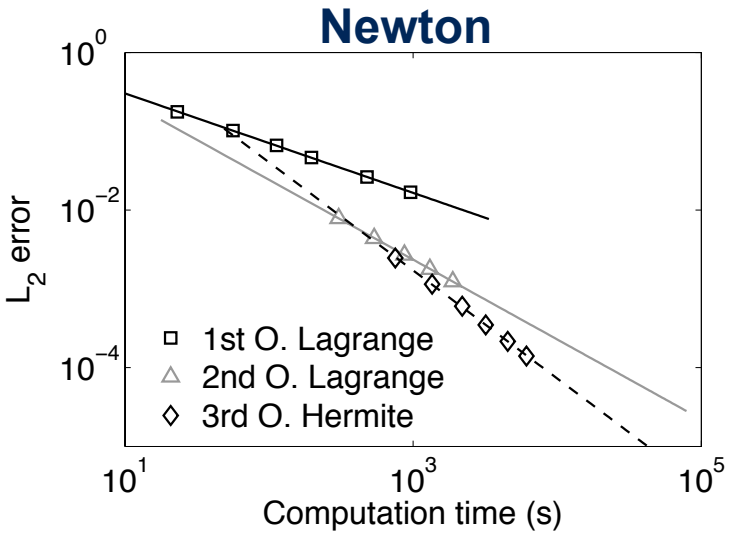
$$\mathcal{R}_{c_i} = (\kappa_i \nabla c_i, \nabla \psi_m) + \left(\left(\frac{\partial f_{loc}}{\partial c_i} + \frac{\partial E_d}{\partial c_i} - \mu_i\right), \psi_m\right)$$

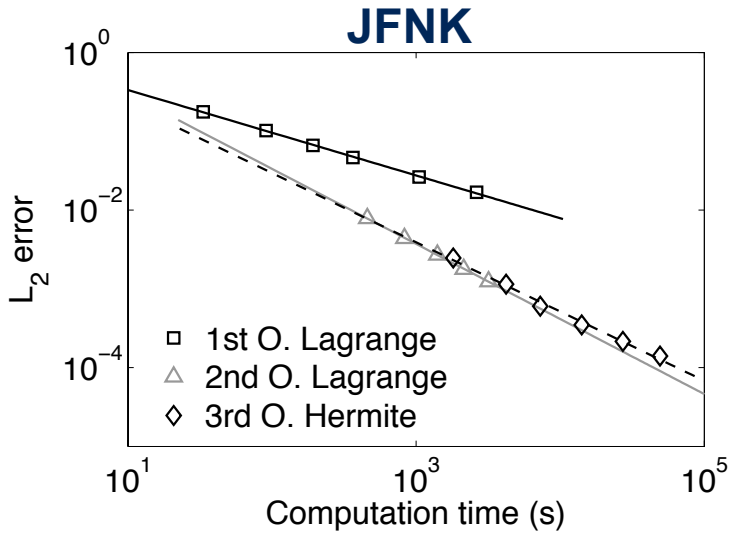
`SplitCHParsed`

- Parameters must be defined in a material object
- The free energy density and its derivatives are defined in an energy material object (as with the direct solve, making it easy to switch between the two)
- Residuals are reversed to improve convergence (`CoupledTimeDerivative`)

Cahn-Hilliard Solution

- We have done a quantitative comparison between the direct and the split solutions of the Cahn-Hilliard equation.
  - The split with 1st order elements is the most efficient.
  - The direct solution has the least error.





- However, practically speaking the split is often the best choice, since our simulations can be computationally expensive.

Simple Phase Field Model Development

- As stated above, the microstructure evolves to minimize the free energy
- Thus, the free energy functional is the major piece of the model
- Phase field model development is modular, with all development focused around the free energy

Free energy: $$F = \int_V \left(f_{loc}(c_i, \eta_j, ..., T) + E_d + \sum_i \frac{\kappa_i}{2}(\nabla c_i)^2 + \sum_j \frac{\kappa_j}{2}(\nabla \eta_j)^2 \right) dV$$

Differential equations: $$\left(M_i \nabla \left(\frac{\partial f_{loc}}{\partial c_i} + \frac{\partial E_d}{\partial c_i} \right), \nabla \psi_m \right) \quad (\kappa_i \nabla c_i, \nabla \psi_m) + \left(\left(\frac{\partial f_{loc}}{\partial c_i} + \frac{\partial E_d}{\partial c_i} - \mu_i \right), \psi_m \right)$$

`CahnHilliard` `SplitCHParsed`

Free Energy Density Material

```
fbulk = 1/4*(1 + c)^2*(1 - c)^2
dfbulk/dc     = c^3 - c
d^2fbulk/dc^2 = 3*c^2 - 1
d^3fbulk/dc^3 = 6*c
```

Reminder:

$$\nabla f(c,\eta) = \nabla c \frac{\partial f}{\partial c} + \nabla \eta \frac{\partial f}{\partial \eta}$$

Phase field models that are not based on a free energy can be implemented using normal MOOSE syntax

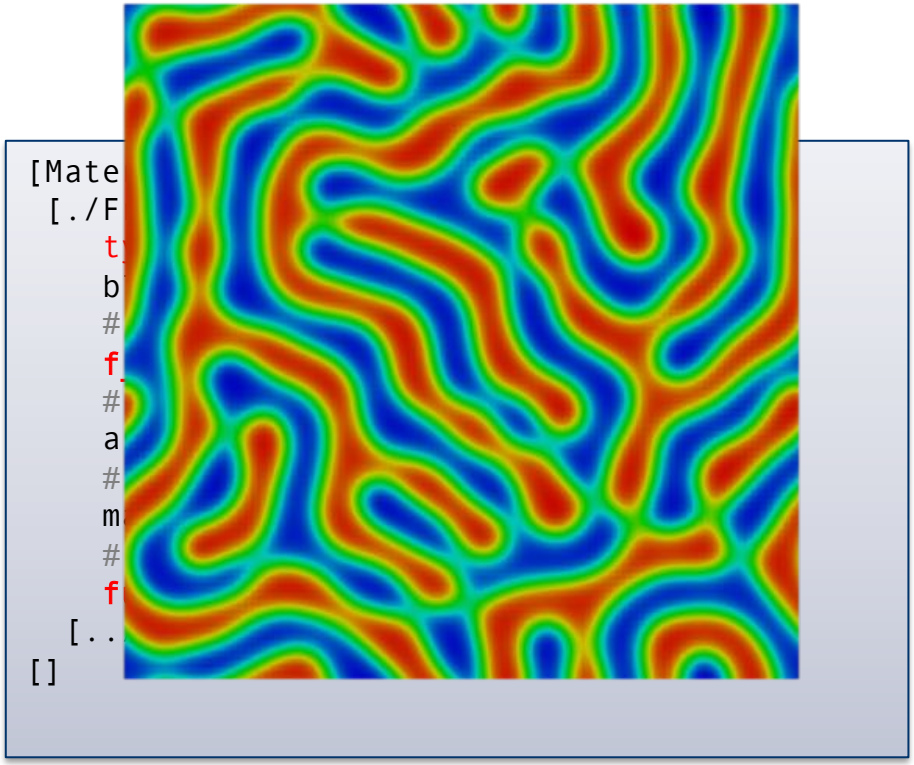
Derivative Function Materials

- The free energy and its derivatives can be defined in materials classes in four different ways:
  - The derivatives can be defined directly by the user, by inheriting from `DerivativeFunctionMaterialBase`
  - The derivatives can be calculated automatically, with the free energy defined in the input file using `DerivativeParsedMaterial`
  - The derivatives can be calculated automatically, with the free energy hard coded in a material object (`ExpressionBuilder`)
  - CALPHAD free energies (only for simple models now)

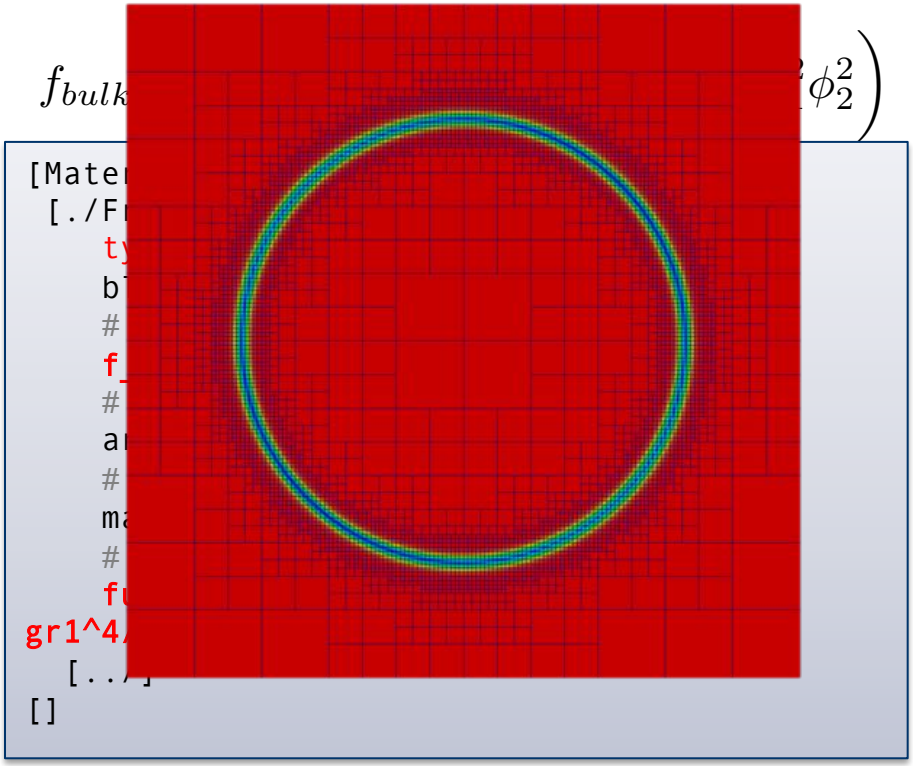
- A derivative material has an `f_name` (the function name)
- Property names of the derivatives are constructed automatically
  (using the value of `f_name` according to fixed rules set in the `DerivativeMaterialPropertyNameInterface` class)
- Add Derivative Function Materials using the `DerivativeSumMaterial`
  (sums function values and derivatives)

Automatic Free Energy Differentiation

- To simplify development even more, you can only enter the free energy functional and all derivatives are automatically evaluated analytically



+ Cahn-Hilliard

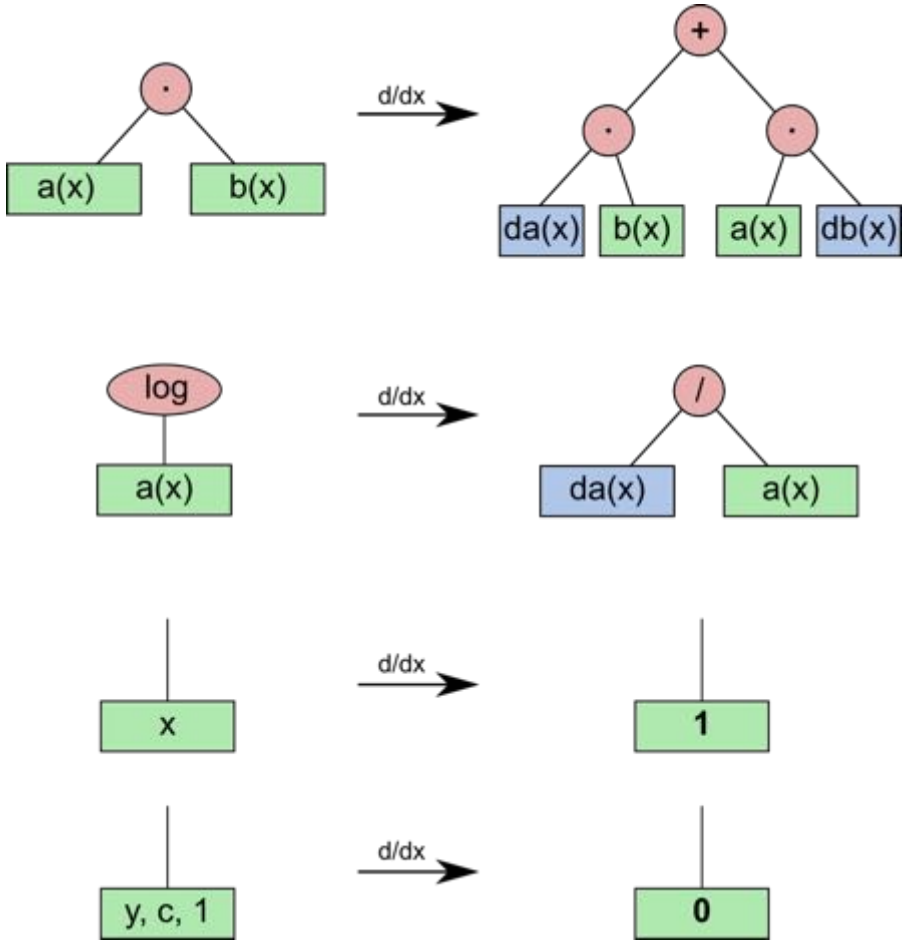


+ Allen-Cahn

Automatic Differentiation

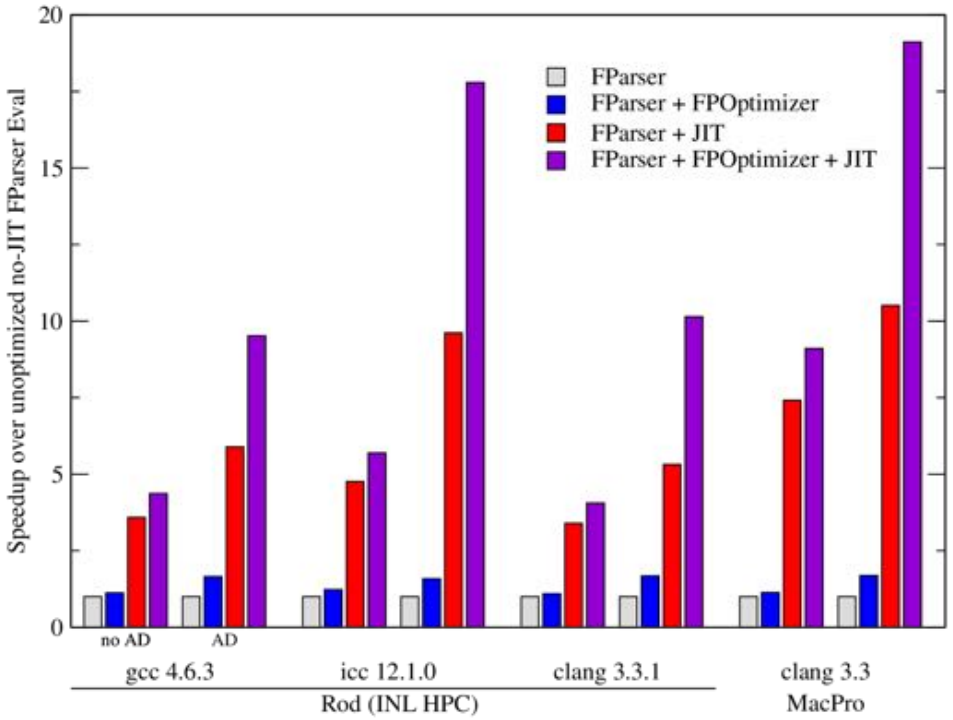
Symbolic differentiation of free energy expressions

- Based on FunctionParser
  http://warp.povusers.org/FunctionParser/
  to allow **runtime** specification of mathematical expressions
- Mathematical expressions
  ➔ Tree data structures
- Recursively apply differentiation rules starting at the root of the tree
- Eliminate source of human error
- Conserve developer time



Performance considerations

- Aren't interpreted functions slower than natively compiled functions?
- Just In Time (JIT) compilation for FParser functions
- Parsed functions (automatic differentiation) now as fast as hand coded functions
- Makes the rapid Phase Field model development more attractive
- ~80ms compile time per function. Results cached.



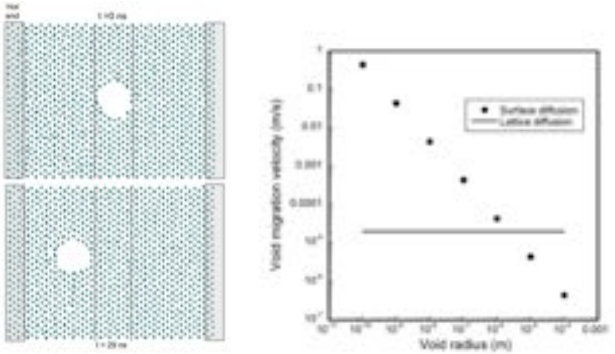
Examples and Applications

MARMOT Example: Void Migration

- Multiscale investigation of void migration in a temperature gradient (Soret effect):

Atomistic

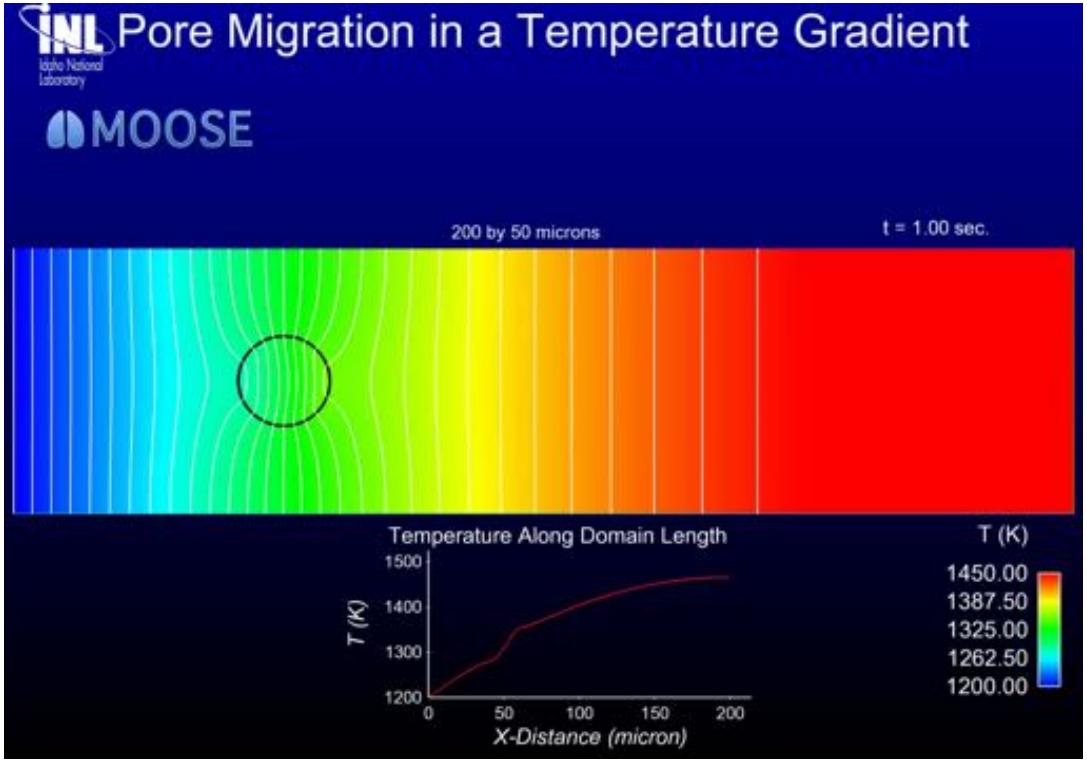
- MD studies identify the diffusion mechanisms active in the migration of nanovoids



From Desai (2009)

Mesoscale

- The migration of larger voids is modeled with MARMOT with surface and lattice diffusion



Zhang et al., Computational Materials Science, 56 (2012) 161-5

Particle and Pore Pinning

- Defects such as pores or precipitates on GBs impede the GB migration by applying an opposing force.
- To account for the interaction of GBs with a particle defined by the variable c, we add a term to the free energy

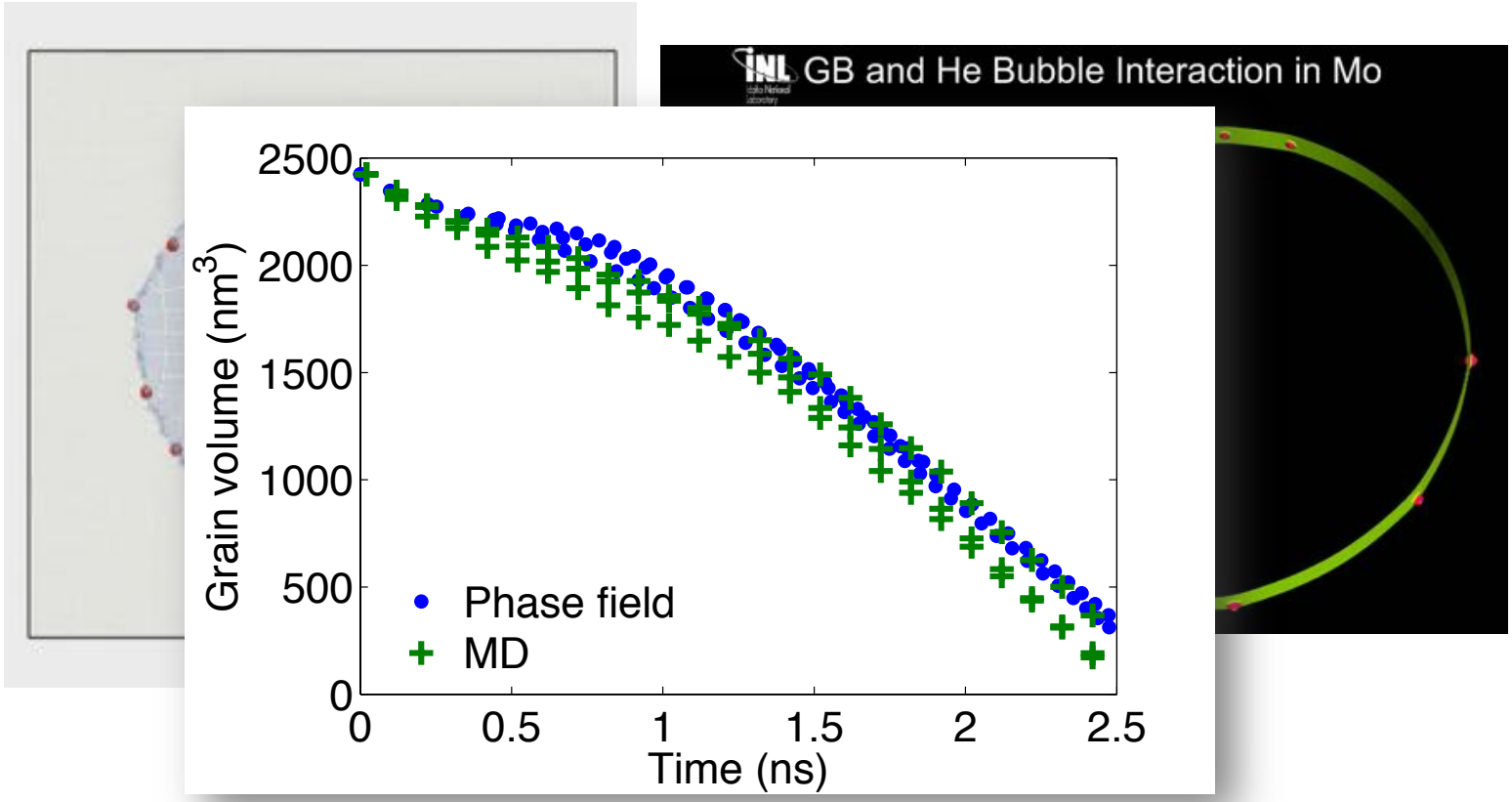
$$f(c,\eta_i) = \sum_i \left(\frac{\eta_i^4}{4} - \frac{\eta_i^2}{2}\right) + \left(\frac{c^4}{4} - \frac{c^2}{2}\right) + a_{GB}\sum_i\sum_{j>i}\eta_i^2\eta_j^2 + a_s\sum_i c^2\eta_i^2$$

- The term is implemented in the kernel `ACGBPoly`
- It is activated using the simplified grain growth syntax by adding a coupled variable `c`

```
[Kernels]
  [./PolycrystalKernel]
    c = c
  [../]
[]
```

Particle and Pore Pinning

- We verified this model by simulating an identical system using MD simulation and the phase field model
  - 10 He bubbles (r = 6 nm) in Mo bicrystal (R = 20 nm) at 2700 K.

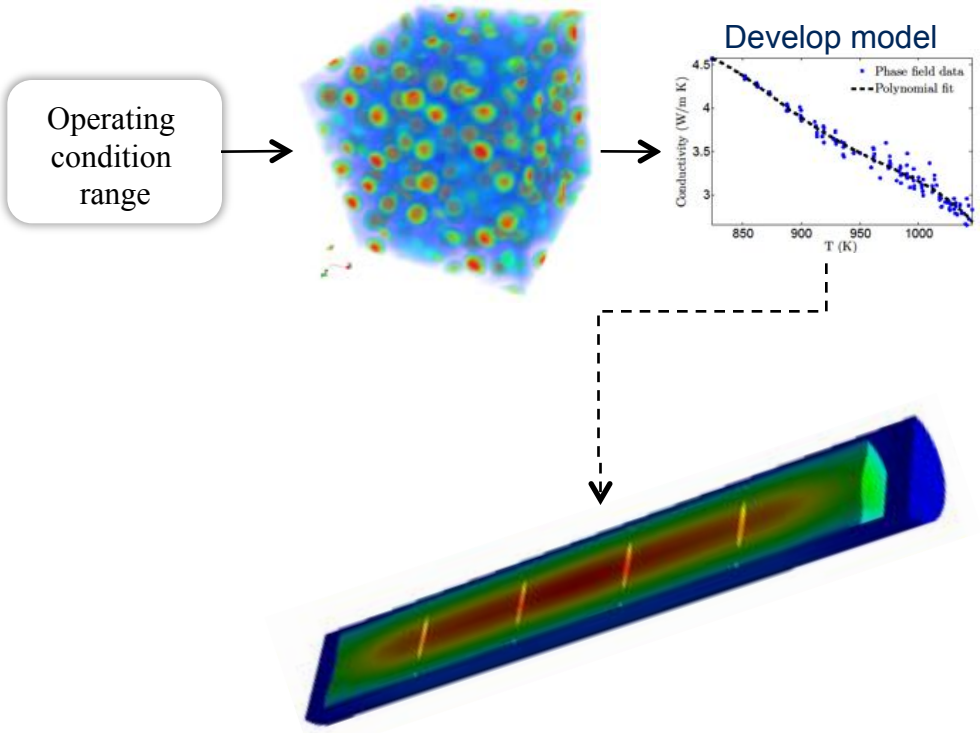


Coupling to Larger Length-Scales

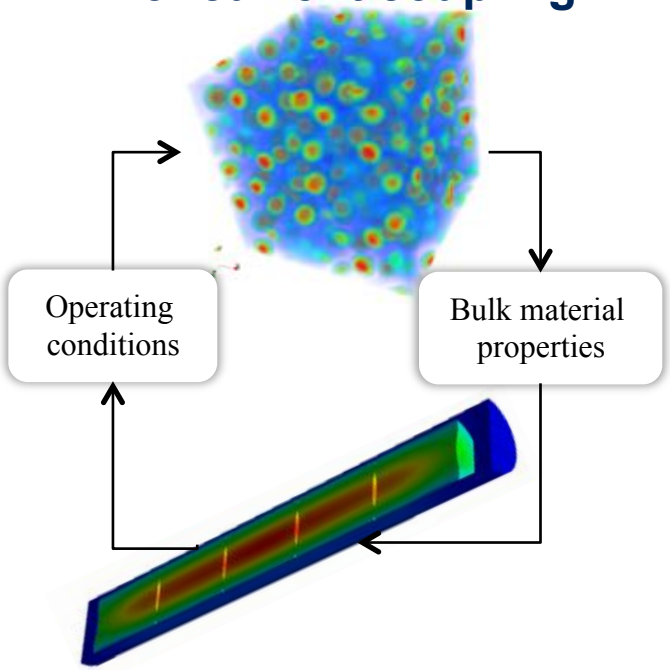
- MARMOT can be used in both hierarchical and concurrent coupling

Hierarchical coupling

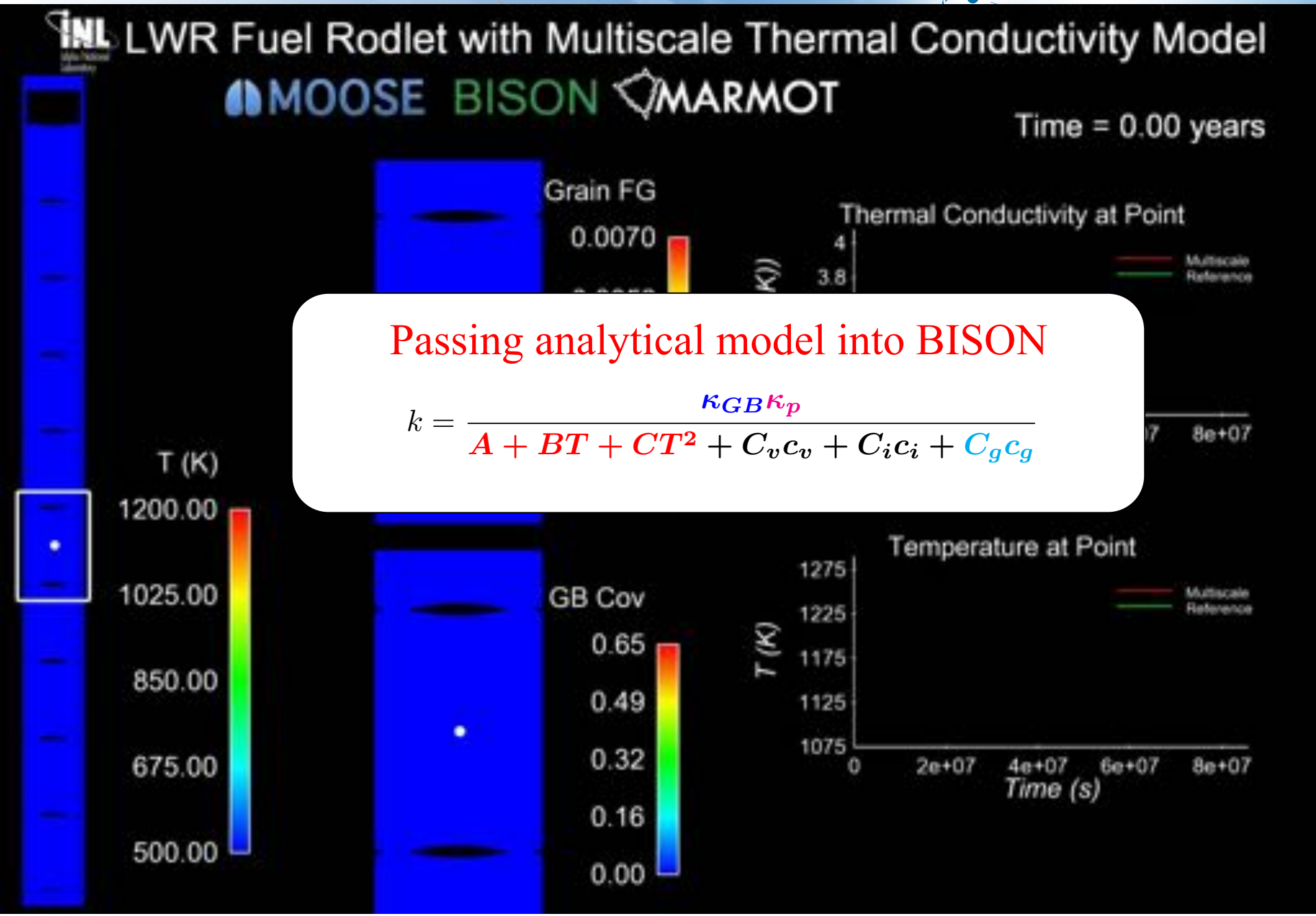
- Lower length-scale models are run separately to construct materials models.
- Macroscale simulations are efficient.



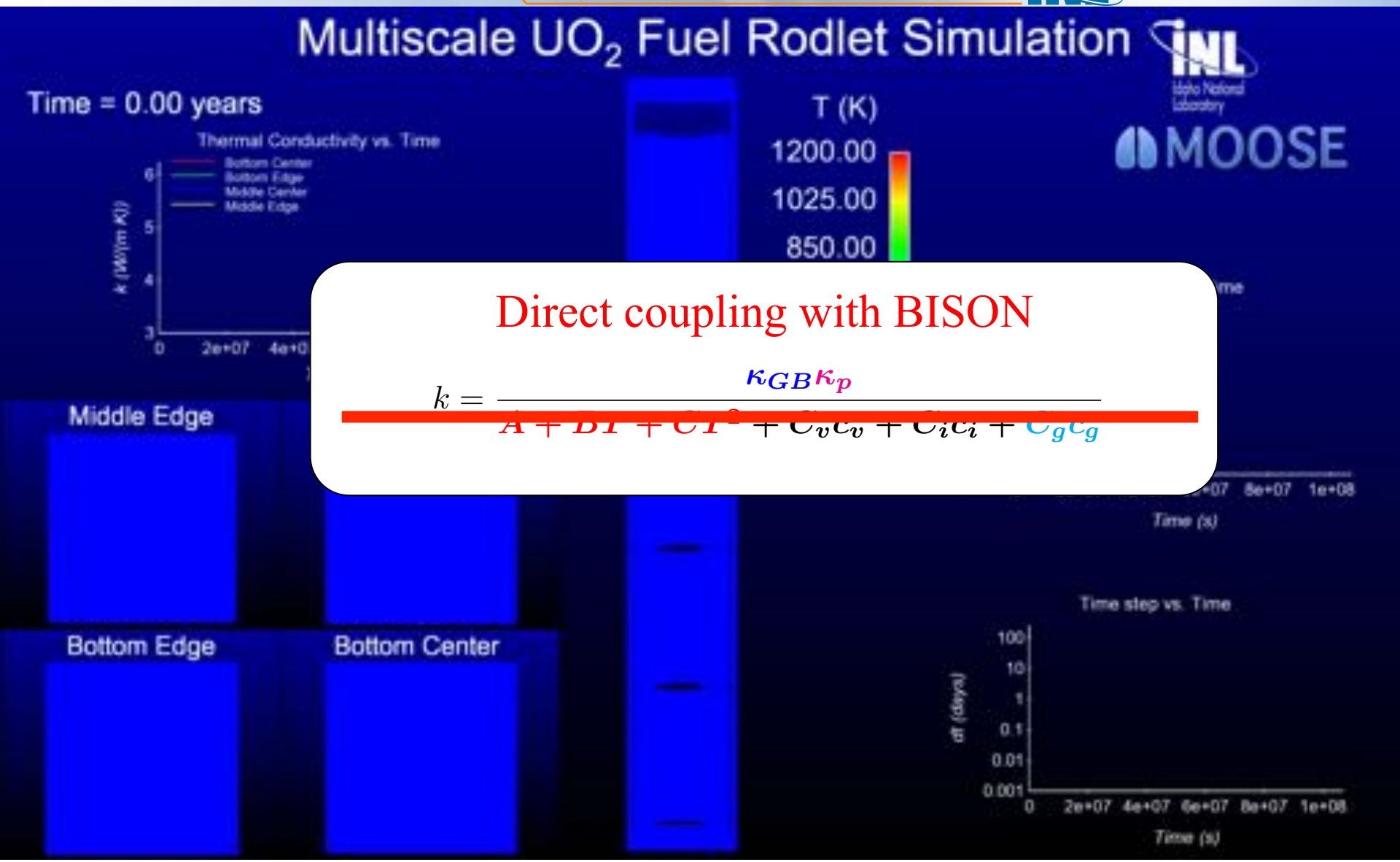
Concurrent coupling



- Codes are run simultaneously and information is passed back and forth.
- Captures interaction between the scales
- Can locate important coupled behaviors
- More computationally expensive


LWR Fuel Rodlet with Multiscale Thermal Conductivity Model
MOOSE BISON MARMOT
Time = 0.00 years
Grain FG
0.0070
Thermal Conductivity at Point
4
3.8
Multiscale
Reference
8e+07
Passing analytical model into BISON
$k = \frac{\kappa_{GB}\kappa_p}{A + BT + CT^2 + C_v c_v + C_i c_i + C_g c_g}$
T (K)
1200.00
1025.00
850.00
675.00
500.00
GB Cov
0.65
0.49
0.32
0.16
0.00
Temperature at Point
1275
1225
1175
1125
1075
T (K)
0
2e+07
4e+07
6e+07
8e+07
Time (s)
Multiscale
Reference

Multiscale UO_2 Fuel Rodlet Simulation



Direct coupling with BISON

$$k = \frac{\kappa_{GB}\kappa_p}{A + BT + CT^2 + C_v c_v + C_i c_i + C_g c_g}$$

Thank you!

- For more information, please see http://mooseframework.org
- Github repository: https://github.com/idaholab/moose
- 2-3 day training workshops at INL and other locations (keep an eye on the website for dates and locations)
- Mailing list: to subscribe, send an email to
  moose-users+subscribe@googlegroups.com
  or see
  http://mooseframework.org/getting-started/