An Overview of the MOOSE Framework and Physics Modules

John W. Peterson
jw.peterson@inl.gov

Idaho National Laboratory
SIAM Conference Computational Science and Engineering (CSE 2017)

February 27, 2017

- MOOSE is a collaborative project; its success is due to the over 125 talented scientists and engineers who have contributed to it over the years.

Cody Permann, Derek Gaston, John W. Peterson, David Andrs, Andrew E. Slaughter, Daniel Schwen, Jason D. Hales, Andy Wilkins, Benjamin Spencer, Jason M. Miller, Michael Tonks, Yaqi Wang, Philip Jagielski, Fande Kong, Larry Aagesen, Chris Green, Sudipta Biswas, Brian Alger, Ryosuke Park, Pritam Chakraborty, Dmitry Karpeev, Alex Lindsay, Robert Carlsen, Sebastian Schunert, Hailong Chen, Ziyu Zhang, Sterling Harper, Marie Backman, Andrea Jokisaari, Wen Jiang, Stephen Novascone, Roy H. Stogner, Stephanie Pitts, Swetha Veeraraghavan, Karthikeyan Chockalingam, John Hutchins, Bradley Fromm, Shane Stafford, Jesse Carter, Richard Williamson, Al Casagranda, Scott A. Schoen, Aaron Butterfield, Ben Spencer, Avery Bingham, Malachi Tolman, Katherine Smith, Jacob Bair, Yidong Xia, Kyle Gamble, Alex McCaskey, Paul Talbot, Mathias Winkel, Jed Brown, Vanessa Gertman, Ryan Norman, Joshua E. Hansel, Alexandra Gertman, Karim Ahmed, John-Michael Bradley, Yongfeng Zhang, Luanjing Guo, Heather Sheldon, Christopher Walton, Wenfeng Liu, Shiyuan Gu, Roger Pawlowski, Rachel Waxman, Michael Short, Micah Johnson, John Mangeri, Jarin French, Jacob L. Bair, Chandana Jayasundara, Bertrand Lagree, William Hoffman, Steve Prescott, Liangzhe Zhang, Ian Greenquist, Hao Wang, Frederick Gleicher, Floyd Hilty, Chris Newman, Bob Kinoshita, Axel Seoane, Andrew Parnell, William Taitano, Tami Grimmett, Shuaifang Zhang, Shane Stimpson, Robert Nourgaliev, Robert A. Kinoshita, Mike Rose, Michael Pernice, Matt Ellis, Lauren Winterholler, Justin Coleman, Joshua J. Cogliati, Danielle Perez, Daniel Ruprecht, Damien Lebrun-Grandie, Cormac Garvey, Antonio Cervone, Adam Cahill, Hardik Kothari, Yang Xia, Weixiong Zheng, Thomas Poulet, Thomas Hannah, Stephen Thomas, Srivatsan Hulikal, Samet Kadioglu, Robert Podgorney, Richard Martineau, Paul Millet, Nick Thompson, Martin Lesueur, Lei Zhao, Julian Andrej, Jacob Peterson, Hai Huang, Giovanni Pastore, Derek Stucki, Cristian Rabiti, Chuan Lu, Bulent Biner

- MOOSE = Multiphysics Object Oriented Simulation Environment.
- Originated at Idaho National Laboratory, circa 2008.
- Open source (LGPL 2.1), C++.
- `https://github.com/idaholab/moose`
- Output of CLOC from `framework` directory:

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                            638          14158          14628          64391
C/C++ Header                   671          11084          25272          23903
Python                          78           1764           3617           6630
-------------------------------------------------------------------------------
```

idaholab / moose

Watch 51 | Unstar 192 | Fork 322

Code | Issues 624 | Pull requests 33 | Projects 0 | Wiki | Pulse | Graphs | Settings

Multiphysics Object Oriented Simulation Environment http://www.mooseframework.org Edit

multiphysics simulation fem finite-elements object-oriented parallel amr Manage topics

15,082 commits | **3** branches | **1** release | **84** contributors | LGPL-2.1

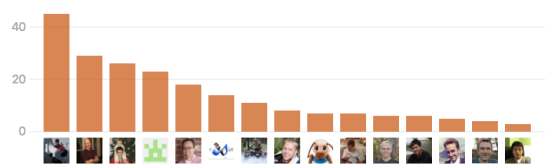
January 25, 2017 – February 25, 2017 Period: 1 month

Overview

136 Active Pull Requests | **207** Active Issues

113 Merged Pull Requests	**23** Proposed Pull Requests	**165** Closed Issues	**42** New Issues

Excluding merges, **36 authors** have pushed **239 commits** to devel and **239 commits** to all branches. On devel, **2,172 files** have changed and there have been **314,042 additions** and **33,831 deletions**.



113 Pull requests merged by **24** people

- MOOSE makes extensive use of libMesh capabilities:
  - Adaptive mesh refinement.
  - Error estimators.
  - Finite element families.
  - Software configuration and build system.
  - Mesh partitioners.
  - Generic multithreading APIs.
  - Object-oriented MPI wrappers.
  - Numerical linear algebra interfaces to PETSc, SLEPc, Trilinos.
  - Quadrature rules.
  - `ReplicatedMesh` and `DistributedMesh`.
  - Mesh file I/O.
  - ...

- The framework code does not implement any particular physics.
- The physics modules are responsible for this aspect.
- Complete list of modules under active development: `heat_conduction`, `phase_field`, `rdg`, `stochastic_tools`, `chemical_reactions`, `contact`, `fluid_properties`, `level_set`, `navier_stokes`, `porous_flow`, `tensor_mechanics`, `xfem`.
- Output of CLOC from `modules` directory:

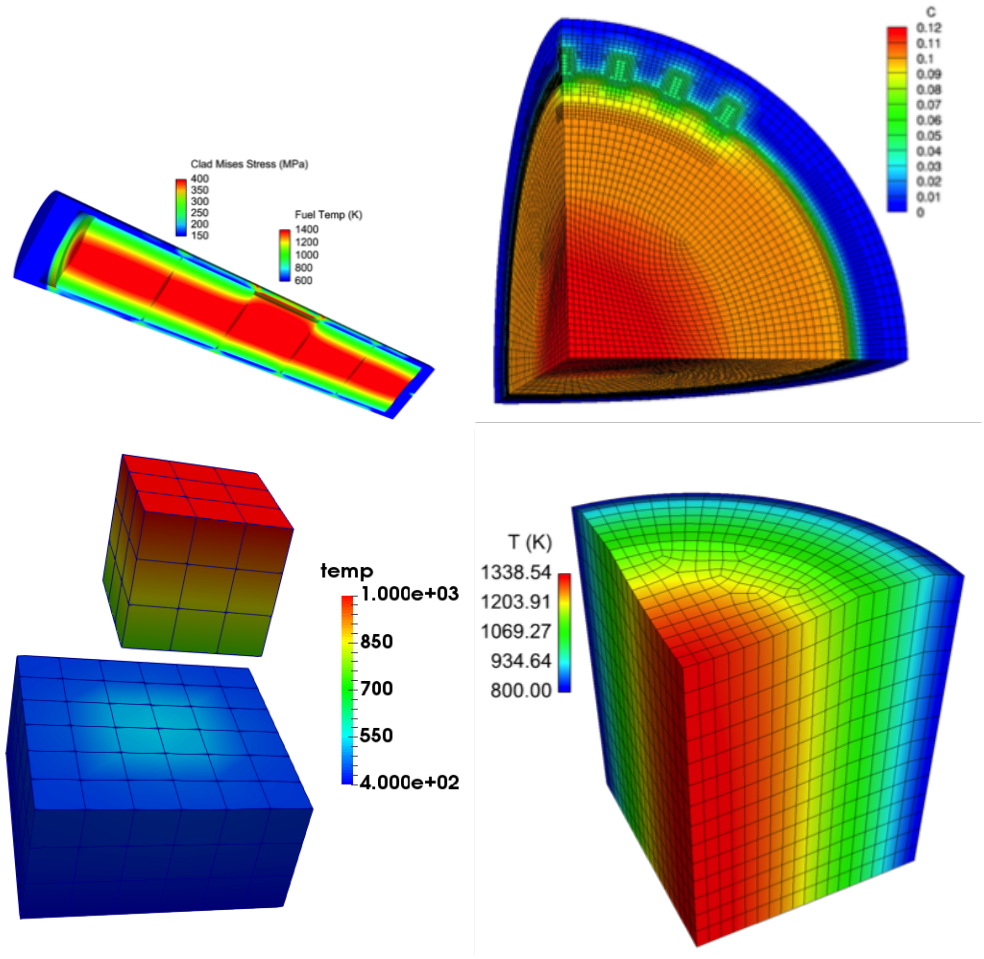
```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                            985          17568          13986          88023
C/C++ Header                  1008          12552          20443          27285
-------------------------------------------------------------------------------
```

- Most MOOSE users write C++ applications for their specific physics.
- `https://github.com/idaholab/stork`
- These applications (which we encourage to be named after animals) can depend on MOOSE and one or more of the physics modules, in addition to implementing their own physics.

- All PRs into MOOSE must pass an extensive regression testing suite.
- `https://www.moosebuild.org`

- libMesh PRs are also tested by this system.

- Upcoming *free* MOOSE training, March 28–30, 2017:
  - Penn State University
  - University of Florida
- Register at `http://www.mooseframework.org`.



- Primary developers: Jason Hales, Derek Gaston, David Andrš, Benjamin Spencer, Stephen Novascone, Michael Tonks, Sudipta Biswas.

- The `heat_conduction` module solves the transient solid heat conduction equation:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot k \nabla T + f \tag{1}$$

- Temperature: T
- Density: ρ
- Specific heat: c_p
- Thermal conductivity: k
- Volumetric heat source: f

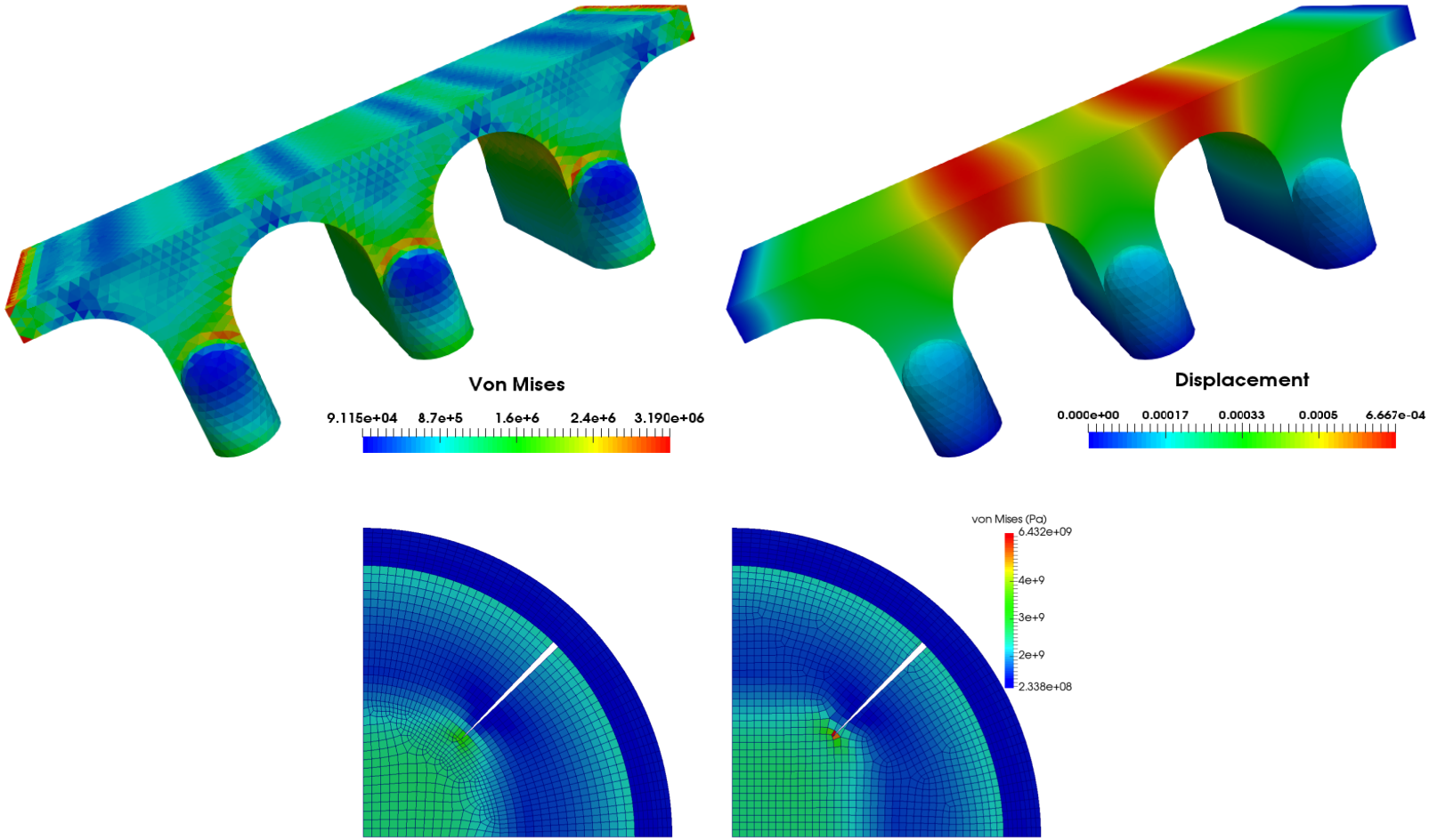
- An important application in nuclear fuels performance involves accurately computing the amount of heat transfer across small gaps.
- The `heat_conduction` module has a Lagrange multiplier-inspired approach for this.
- In the simplest case, we consider the steady state weak formulation of (1) applied to two bodies $\Omega^{(1)}$, $\Omega^{(2)}$ which are nearly in contact along a segement of the boundary denoted by $\Gamma_C^{(1)}$.

$$\sum_{m=1}^{2}\left[\int_{\Omega^{(m)}}\left(k^{(m)}\nabla T^{(m)}\cdot\nabla v^{(m)}-f^{(m)}v^{(m)}\right)\mathrm{d}x+\int_{\Gamma_N^{(m)}}q_N^{(m)}v^{(m)}\,\mathrm{d}s\right]$$
$$+\int_{\Gamma_C^{(1)}}\lambda\left(v^{(1)}-Pv^{(2)}\right)\mathrm{d}s=0 \tag{2}$$

$$\int_{\Gamma_C^{(1)}}\left[\lambda-h\left(T^{(1)}-PT^{(2)}\right)\right]\mu\,\mathrm{d}s=0 \tag{3}$$

- λ represents the heat flux into the gap.
- h is a heat transfer coefficient $\propto \frac{k}{\ell}$, where k is the thermal conductivity of the gap, and ℓ is the distance between the bodies.
- P is the smoothed outward normal projection operator from body (2) to (1).

- S. Falletta and B. P. Lamichhane, "Mortar finite elements for a heat transfer problem on sliding meshes," *Calcolo*, vol. 46, pp. 131–148, June 2009, 10.1007/s10092-009-0001-1.
- S. Hüeber and B. I. Wohlmuth, "Thermo-mechanical contact problems on non-matching meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, pp. 1338–1350, Mar. 2009, 10.1016/j.cma.2008.11.022.
- J. D. Hales, D. R. Gaston, and D. Andrš, "Algorithms for thermal and mechanical contact in nuclear fuel performance analysis," in *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)*, (Sun Valley, ID), May 5–9, 2013.
- S. R. Novascone, B. W. Spencer, J. D. Hales, and R. L. Williamson, "Evaluation of coupling approaches for thermomechanical simulations," *Nuclear Engineering and Design*, vol. 295, pp. 910–921, Dec. 2015, 10.1016/j.nucengdes.2015.07.005.



- Primary developers: Daniel Schwen, Andy Wilkins, Michael Tonks, Pritam Chakraborty, Stephanie Pitts, Andrea Jokisaari, Jason Hales, Benjamin Spencer, Swetha Veeraraghavan, Derek Gaston, Wen Jiang, Sudipta Biswas, Larry Aagesen.

- The `tensor_mechanics` module solves the equations of solid mechanics

$$\nabla \cdot (\boldsymbol{\sigma} + \boldsymbol{\sigma}_0) + \vec{b} = \vec{0} \tag{4}$$

plus associated boundary conditions, where

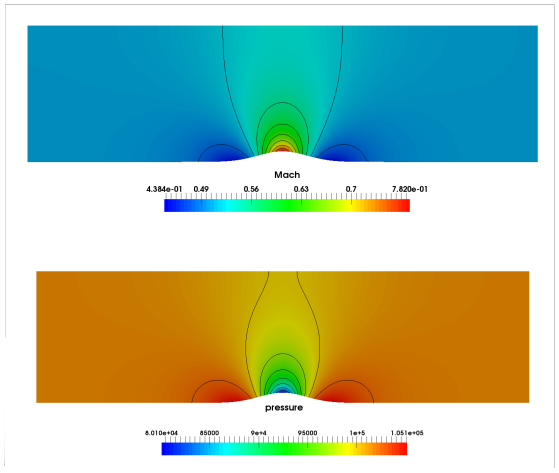
- $\boldsymbol{\sigma}$ is the Cauchy stress tensor.
- $\boldsymbol{\sigma}_0$ is an "extra stress" (due to coupling with other physics).
- $\vec{b}$ is a body force.
- A constitutive relation is required to relate the stress to the strain $\boldsymbol{\epsilon}$, for example

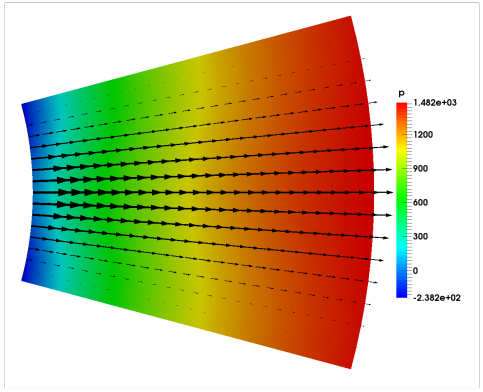
$$\boldsymbol{\sigma} = \boldsymbol{C}\,(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_0) \tag{5}$$

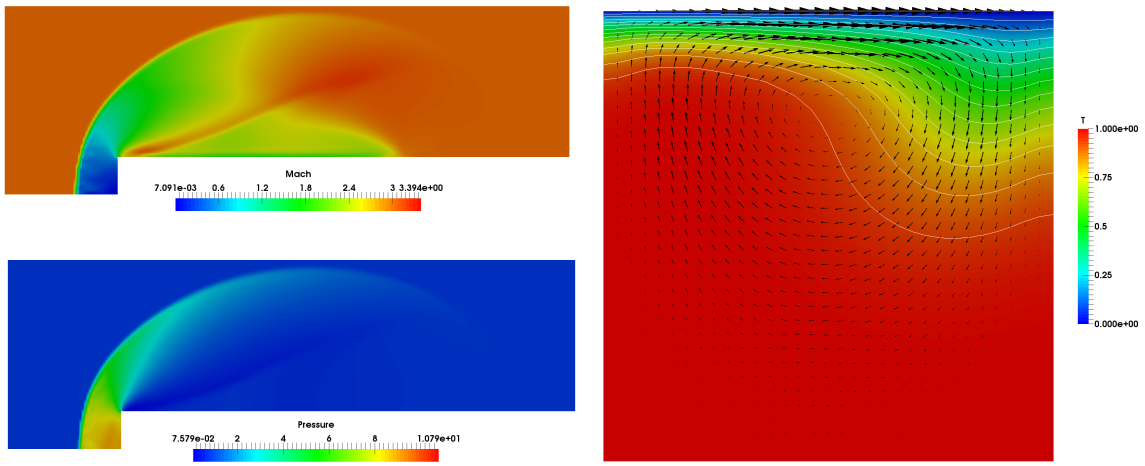
which is only valid for small strains.

- The `tensor_mechanics` module supports both linear elasticity and finite strain mechanics, including elasticity and plasticity.
- Since the form of the governing equations is independent of the material being simulated, the user needs to define (or use existing versions of) the tensors $\boldsymbol{C}$, $\boldsymbol{\epsilon}$, and $\boldsymbol{\sigma}$ in order to perform a simulation.
- Symmetric special cases including generalized plane strain, axisymmetric, axially-revolved, and spherically-symmetric problems are also supported.
- Many MOOSE-based applications, as well as the `phase_field` and `porous_flow` modules, depend on the `tensor_mechanics` module to couple the mechanical response of materials to other physics.

- D. P. Adhikary, C. T. Jayasundara, R. K. Podgorney, and A. H. Wilkins, "A robust return-map algorithm for general multisurface plasticity," *International Journal for Numerical Methods in Engineering*, vol. 109, pp. 218–234, Jan. 2017, 10.1002/nme.5284.
- K. Chockalingam, M. R. Tonks, J. D. Hales, D. R. Gaston, P. C. Millett, and L. Zhang, "Crystal plasticity with Jacobian-Free Newton-Krylov," *Computational Mechanics*, vol. 51, pp. 617–627, May 2013, 10.1007/s00466-012-0741-7.
- J. D. Hales, S. R. Novascone, R. L. Williamson, D. R. Gaston, and M. R. Tonks, "Solving nonlinear solid mechanics problems with the Jacobian-free Newton Krylov Method," *Computer Modeling in Engineering & Sciences*, vol. 84, no. 2, pp. 123–152, 2012, `http://tinyurl.com/heohfr5`.
- M. M. Rashid, "Incremental kinematics for finite element applications," *International Journal for Numerical Methods in Engineering*, vol. 36, no. 23, pp. 3937–3956, 1993, 10.1002/nme.1620362302.







- Primary developers: John Peterson, Alex Lindsay, Fande Kong, Adam Cahill, Yidong Xia, Martin Lesueur.

- The `navier_stokes` module solves both the compressible and incompressible Navier–Stokes equations.
- Compressible:
  - Continuous and discontinuous Galerkin (finite volume) conserved variable formulations are available.
  - Continuous formulation is stabilized by Streamline-Upwind/Petrov-Galerkin (SU/PG) terms.
- Incompressible:
  - Fully-coupled LBB-stable "Taylor–Hood" and pressure projection (Chorin) formulations are available.
  - Can optionally couple mass and momentum conservation equations to a convected temperature field.
- Uses `fluid_properties` module interfaces for ideal gas, sound speed, etc.

- The equations governing the flow of a compressible fluid are:

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}_i}{\partial x_i} - \frac{\partial \vec{G}_i}{\partial x_i} = \vec{0} \tag{6}$$

where, in three dimensions,

$$\vec{U} \equiv \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{bmatrix} \qquad \vec{F}_i = \begin{bmatrix} \rho u_i \\ \rho u_i u_1 + \delta_{i1} p \\ \rho u_i u_2 + \delta_{i2} p \\ \rho u_i u_3 + \delta_{i3} p \\ \rho u_i H \end{bmatrix} \qquad \vec{G}_i = \begin{bmatrix} 0 \\ \tau_{i1} \\ \tau_{i2} \\ \tau_{i3} \\ k \frac{\partial T}{\partial x_i} + \tau_{i\ell} u_\ell \end{bmatrix}$$

- Summation over the repeated indices i, ℓ is implied.

- The SU/PG-stabilized weak formulation of (6) is:

$$\int_\Omega \left(\frac{\partial \vec{U}}{\partial t} \cdot \vec{W} + \left(\vec{G}_i - \vec{F}_i \right) \cdot \frac{\partial \vec{W}}{\partial x_i} \right) \mathrm{d}\Omega + \int_\Gamma \hat{n}_i \left(\vec{F}_i - \vec{G}_i \right) \cdot \vec{W} \, \mathrm{d}\Gamma$$
$$+ \sum_K \int_{\Omega_K} \boldsymbol{A}_i^T \frac{\partial \vec{W}}{\partial x_i} \cdot \boldsymbol{\tau}_{\mathrm{SUPG}} \vec{R}(\vec{U}) \, \mathrm{d}\Omega_K = 0 \tag{7}$$

where $\vec{W}$ is a "system" test function, $\vec{R}$ is the strong form of the residual in (6), and

$$\boldsymbol{A}_i \equiv \frac{\partial \vec{F}_i}{\partial \vec{U}} \tag{8}$$

are the flux Jacobian matrices.

- Obtains correct convergence rates (in entropy norm) for isentropic flows.
- Ideal gas equation of state (via `fluid_properties` module) only.
- Shock capturing terms are implemented but additional tuning is required.

- The finite volume formulation of (6) proceeds by applying the divergence theorem on each element Ω_K to obtain

$$|\Omega_K|\frac{\partial \vec{U}_K}{\partial t} + \sum_j \int_{S_j} \hat{n}_i(\vec{F}_i - \vec{G}_i)\,\mathrm{d}S_j = 0 \tag{9}$$

  where

$$\frac{\partial \vec{U}_K}{\partial t} \equiv \frac{1}{|\Omega_K|}\int_{\Omega_K} \vec{U}\,\mathrm{d}\Omega_K \tag{10}$$

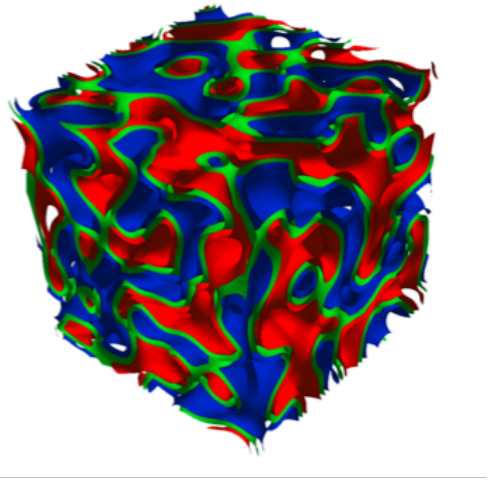
  is the cell-averaged value of $\vec{U}$.
- The $\sum_j$ goes over all faces S_j (internal and boundary) in the mesh.

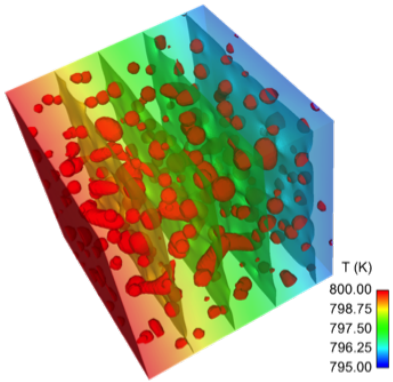
- Since the fluxes $\vec{F}$, $\vec{G}$ are discontinuous across the faces S_j, a numerical approximation to the flux must be introduced.
- This approximation must be consistent with the physics, and generally depends on the jump and average values of the fluxes on either side of the face.
- The `navier_stokes` module implements the approximate Riemann solver-based "VHLLC" numerical flux and a few others.
- Various slope reconstruction methods, limiters, and approaches for implementing boundary conditions (Riemann invariant, characteristic) are also supported.

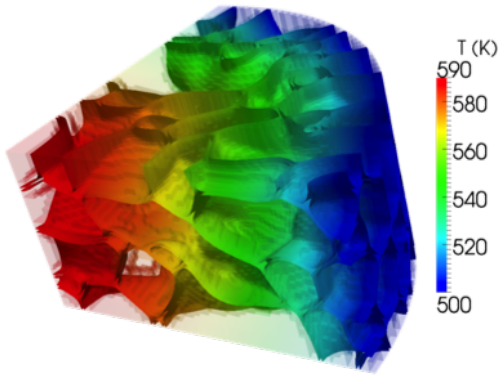
- R. A. Berry, J. W. Peterson, H. Zhang, R. C. Martineau, H. Zhao, L. Zou, and D. Andrš, "RELAP-7 Theory Manual," Tech. Rep. INL/EXT-14-31366, Idaho National Laboratory, Feb. 2014, `http://tinyurl.com/ojygjzw`.
- B. S. Kirk, *Adaptive Finite Element Simulation of Flow and Transport Applications on Parallel Computers*. PhD thesis, ASE-EM dept., The University of Texas at Austin, May 2007, `http://tinyurl.com/j7erk2u`.
- X. Liu, Y. Xia, H. Luo, and L. Xuan, "A class of Rosenbrock methods for a reconstructed discontinuous Galerkin method for the unsteady compressible flows," in *22nd AIAA Computational Fluid Dynamics Conference*, (Dallas, Texas), June 22–26, 2015, 10.2514/6.2015-2448.
- Y. Xia, X. Liu, and H. Luo, "A finite volume method based on WENO reconstruction for compressible flows on hybrid grids," in *52nd AIAA Aerospace Sciences Meeting*, (National Harbor, Maryland), Jan. 13–17, 2014, 10.2514/6.2014-0939.

- The Navier–Stokes module also depends on the newly-created `rdg` (Reconstructed Discontinuous Galerkin) module for some of its slope reconstruction algorithms.
- RDG methods improve the spatial accuracy of underlying DG methods without significantly increasing storage and computation cost.
- The RDG capabilities are in a separate module since they are not specific to CFD simulations.

- Y. Xia, X. Liu, H. Luo, and R. Nourgaliev, "A third-order implicit discontinuous Galerkin method based on a Hermite WENO reconstruction for time-accurate solution of the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 79, no. 8, pp. 416–435, 2015, 10.1002/fld.4057.
- Y. Xia, H. Luo, M. Frisbey, and R. Nourgaliev, "A set of parallel, implicit methods for a reconstructed discontinuous Galerkin method for compressible flows on 3D hybrid grids," *Computers & Fluids*, vol. 98, pp. 134–151, 2014, 10.1016/j.compfluid.2014.01.023.







- Primary developers: Daniel Schwen, Cody Permann, Michael Tonks, Larry Aagesen, Sudipta Biswas, Bradley Fromm.

- The `phase_field` module solves the Allen–Cahn and Cahn–Hilliard equations:

$$\frac{\partial \eta_j}{\partial t} = -L_j \frac{\delta F}{\delta \eta_j} \tag{11}$$

$$\frac{\partial c_i}{\partial t} = \nabla \cdot \left(M_i \nabla \frac{\delta F}{\delta c_i} \right) \tag{12}$$

- Non-conserved order parameters: η_j, $j = 1, \ldots, N_\eta$
- Conserved order parameters: c_i, $i = 1, \ldots, N_c$
- Total free energy: F
- Mobilities: L_j, M_i

- The Cahn–Hilliard equation (12) involves terms proportional to $\nabla^4 c_i$, i.e. fourth-order spatial derivatives.
- The "direct" finite element formulation of (12) is obtained after multiplying by a test function and applying the divergence theorem: once for second-order terms, twice for fourth-order terms.
- This formulation requires C^1 elements, in libMesh one uses

```
FEType(THIRD, HERMITE);
```

- The "split" formulation represents the free energy derivatives $\frac{\delta F}{\delta c_i}$ as independent variables μ_i

$$\frac{\partial c_i}{\partial t} = \nabla \cdot (M_i \nabla \mu_i) \tag{13}$$

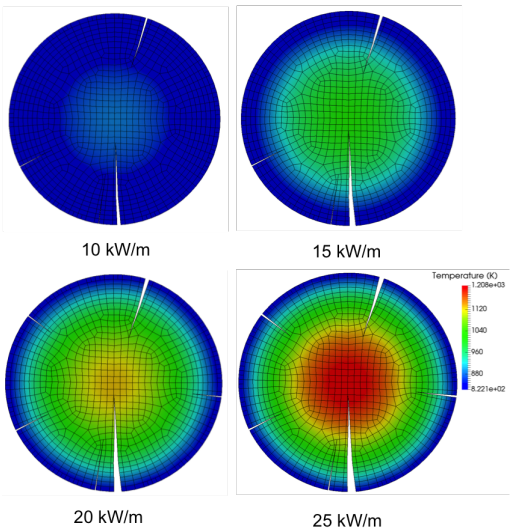
$$\mu_i = \frac{\delta F}{\delta c_i} \tag{14}$$

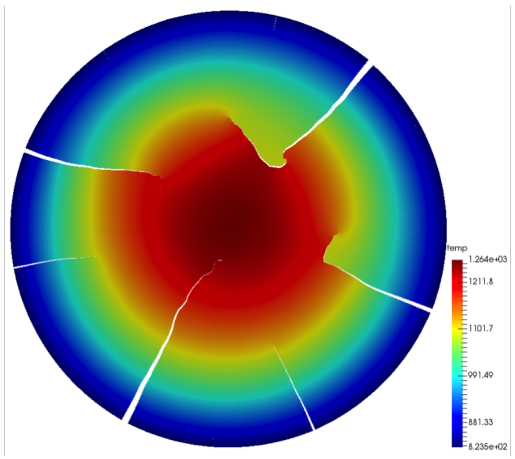
- In this formulation, only C^0 elements are required, e.g. in libMesh

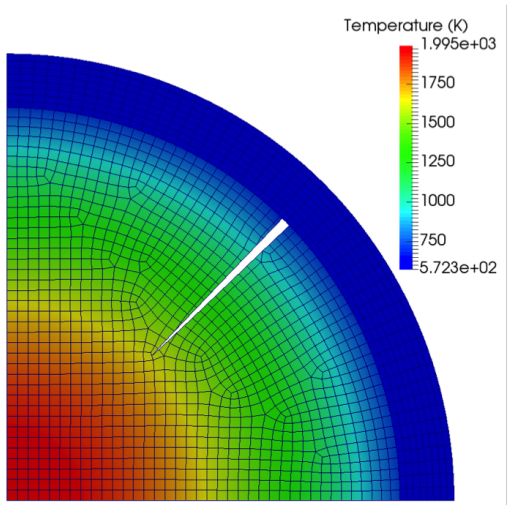
```
FEType(FIRST, LAGRANGE);
```

- The `phase_field` module employs the so-called "modular free energy" approach.
- Basic idea: the structure of the equations is the same regardless of the phase field model.
- Only the free energy functional, F, and the various mobility coefficients vary.
- In the `phase_field` module, users implement `MOOSE Material` objects representing the free energy and its derivatives.
- A robust symbolic automatic differentiation capability with optimized bytecode and JIT compilation assists in these endeavors.

- M. R. Tonks, D. R. Gaston, P. C. Millett, D. Andrs, and P. Talbot, "An object-oriented finite element framework for multiphysics phase field simulations," *Computational Materials Science*, vol. 51, pp. 20–29, Jan. 2012. 10.1016/j.commatsci.2011.07.028.
- L. Zhang, M. R. Tonks, D. Gaston, J. W. Peterson, D. Andrs, P. C. Millett, and B. S. Biner, "A quantitative comparison between C^0 and C^1 elements for solving the Cahn–Hilliard equation," *Journal of Computational Physics*, vol. 236, pp. 74–80, Mar. 2013, 10.1016/j.jcp.2012.12.001.
- C. J. Permann, M. R. Tonks, B. Fromm, and D. R. Gaston, "Order parameter re-mapping algorithm for 3D phase field model of grain growth using FEM," *Computational Materials Science*, vol. 115, pp. 18–25, Apr. 2016. 10.1016/j.commatsci.2015.12.042.
- D. Schwen, L. K. Aagesen, J. W. Peterson, and M. R. Tonks, "Rapid multiphase-field model development using a modular free energy based approach with automatic differentiation in MOOSE/MARMOT," *To appear: Computational Materials Science*, 2017, arXiv e-print 1702.06450.



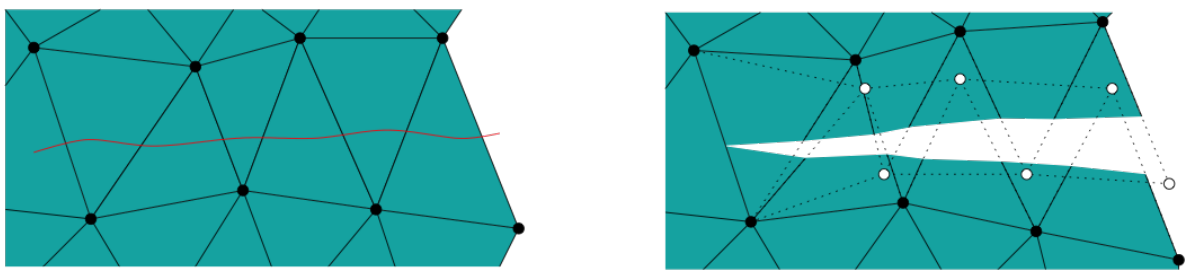




- Primary developers: Benjamin Spencer, Wen Jiang.

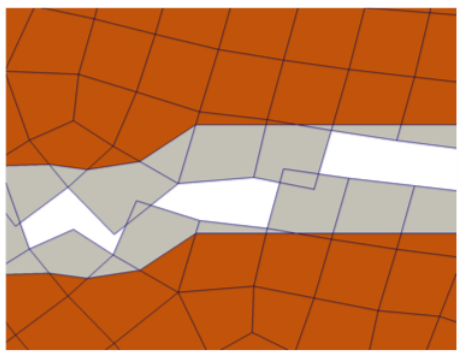
- The eXtended Finite Element Method (XFEM) is a modification to the standard FEM that introduces discontinuities into otherwise continuous solutions.
  - Cracks and fracture.
  - Interfaces between materials or phases.
  - Free surfaces.
- In fracture problems, the crack evolves as the physics dictates rather than along element boundaries.
- Fracture affects stress fields, fission gas release, thermal conductivity, and cladding integrity in nuclear fuel performance calculations.

- The `xfem` module employs the "phantom node" method of Hansbo.
- In the phantom node method, cracked elements are duplicated and assigned physical and non-physical parts.

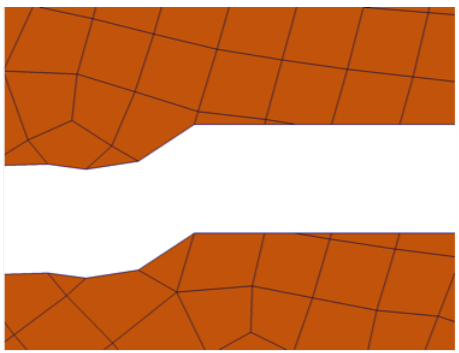


- Requires mesh cutting algorithms and specialized quadrature rules for accurately integrating the discontinuous functions.

- The `xfem` module developers have also created an open source Paraview plugin which assists in visualizing cracked elements.

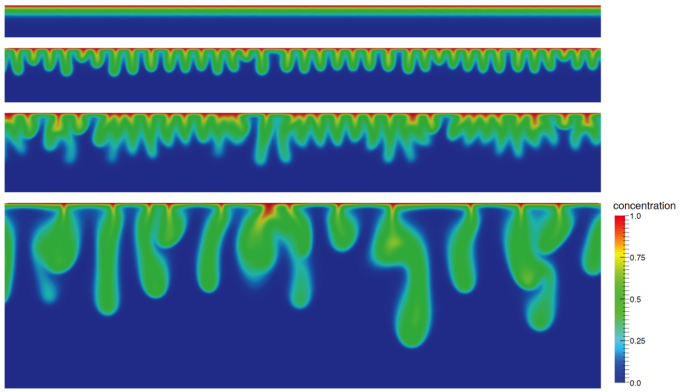


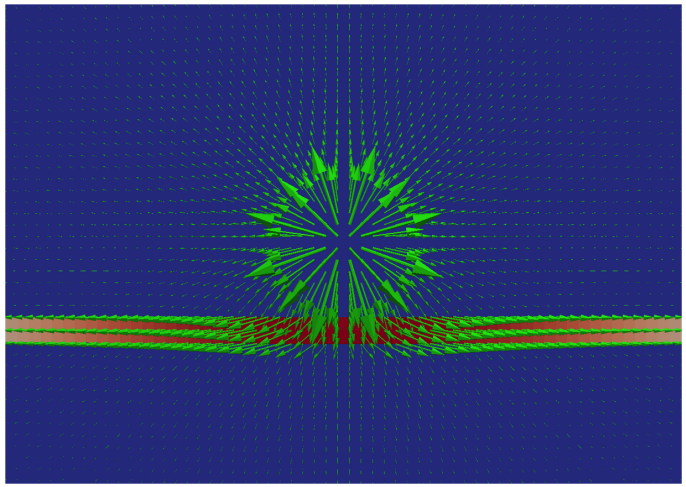
Before
Overlapping elements.
Non-physical parts shown in grey.

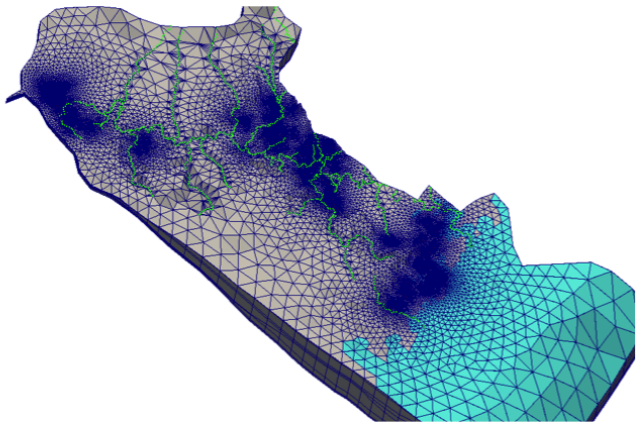


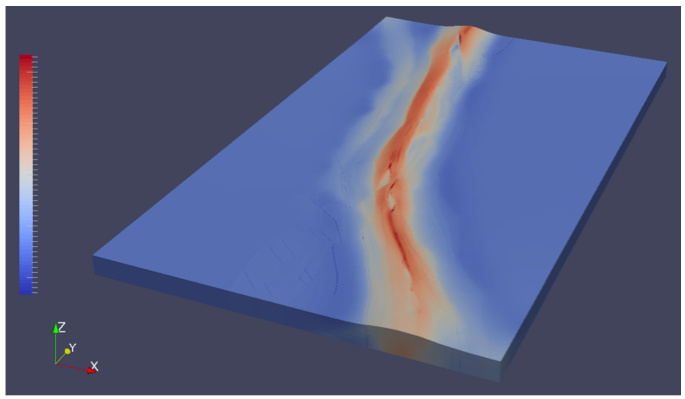
After
Clipping is applied to remove non-physical parts.

- N. Moës, J. Dolbow, and T. Belytschko, "A finite element method for crack growth without remeshing," *International Journal for Numerical Methods in Engineering*, vol. 46, no. 1, pp. 131–150, 1999, `http://tinyurl.com/jqnosn8`.
- T. Belytschko and T. Black, "Elastic crack growth in finite elements with minimal remeshing," *International Journal for Numerical Methods in Engineering*, vol. 45, no. 5, pp. 601–620, `http://tinyurl.com/h2h4z5o`.
- 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*, vol. 193, no. 33-35, pp. 3523–3540, 2004, 10.1016/j.cma.2003.12.041.
- B. W. Spencer, W. Jiang, J. E. Dolbow, and C. Peco, "Pellet cladding mechanical interaction modeling using the extended finite element method," in *Proceedings of Top Fuel*, (Boise, ID), Sept. 2016, `http://tinyurl.com/h6kfkva`.









- Primary developers: Andy Wilkins, Chris Green, Heather Sheldon.

- The `porous_flow` module solves for multiphase/multispecies flow, heat transfer, and solid mechanics in porous media.
- The mass density conservation equation for species K is:

$$\frac{\partial M^K}{\partial t} + M^K \left(\nabla \cdot \vec{v}_s\right) + \nabla \cdot \vec{F}^K - q^K = 0 \tag{15}$$

- The mass density depends on the saturation, density, and mass fraction of each phase, β, of species K, as well as the porosity, φ:

$$M^K \equiv \varphi \sum_\beta S_\beta \rho_\beta \omega_\beta^K + (1 - \varphi)\, C^K \tag{16}$$

- The flux consists of both advective and diffusive/dispersive contributions:

$$\vec{F}^K \equiv \sum_\beta \omega_\beta^K \vec{F}_{adv}^\beta + \vec{F}_{diff}^K \tag{17}$$

where

$$\vec{F}_{adv}^\beta \equiv -\rho_\beta \frac{\boldsymbol{k}\boldsymbol{k}_{rel}^\beta}{\mu^\beta} \left(\nabla P_\beta - \rho_\beta \vec{g}\right) \tag{18}$$

$$\vec{F}_{diff}^K \equiv -\sum_\beta \rho_\beta D_\beta^K \nabla \omega_\beta^K \tag{19}$$

and $\boldsymbol{k}$ is the permeability tensor, P_β is the pore pressure, $\vec{g}$ is the gravitational acceleration, and D_β^K is the dispersion tensor.

- In equation (15), $\vec{v}_s$ is the velocity of the porous solid skeleton. This term couples the porous flow equations to the tensor mechanics equations.
- The porosity, φ, is itself a time-dependent variable which depends on pore pressure, temperature, and the volumetric expansion of the solid.
- Several standard constitutive relations, such as Kozeny-Carman, Langmuir adsorption, van Genuchten permeability, etc. are also included in the module.
- Uses `fluid_properties` module interfaces for some equations of state.

- The conservation of rock and fluid energy density is governed by:

$$\frac{\partial E}{\partial t} + E\left(\nabla \cdot \vec{v}_s\right) + \nabla \cdot \vec{F}^T - q^T = 0 \tag{20}$$

- The energy density depends on the porosity, the density ρ_R, and specific heat C_R of the rock, and the internal energy E^β of each phase, according to:

$$E \equiv (1-\varphi)\rho_R C_R T + \varphi \sum_\beta S_\beta \rho_\beta E^\beta \tag{21}$$

- The heat flux again consists of conductive and advective components:

$$\vec{F}^T \equiv -\vartheta \nabla T + \sum_\beta h_\beta \vec{F}^\beta_{adv} \tag{22}$$

  where ϑ is the thermal conductivity tensor and h_β is the enthalpy of phase β.
- The thermal conductivity and enthalpy in general depend on the phase saturations, temperatures, and other unknowns through provided equations of state.

- The velocity of the solid porous skeleton is governed by:

$$\rho_{mat} \frac{\partial \vec{v}_s}{\partial t} = \nabla \cdot \boldsymbol{\sigma} + \rho_{mat} \vec{b} \tag{23}$$

where

- ρ_{mat} is the undrained density.
- $\boldsymbol{\sigma}$ is the total stress tensor.
- $\vec{b}$ is the external force density.
- The stress depends on the effective pore pressure,

$$P_f \equiv \sum_{\beta} S_\beta P_\beta \tag{24}$$

- The `porous_flow` module supports different choices of primary variables, and allows variable switching.
- All Jacobian contributions are available, so the nonlinear solver can employ either the Jacobian-free or standard Newton-Krylov algorithm.
- Mass lumping and full upwinding are used to ensure stability near sharp fronts and robustness in the event of phase disappearance.
- Excellent (96%) line coverage in the regression test suite!
- Not currently optimized for user friendliness or simple cases, although these improvements are planned.

- A. H. Wilkins, "MOOSE's PorousFlow module." Online theory manual, Feb. 2017. `http://tinyurl.com/jashrzw`.

Thank you for your attention!