INL/MIS-20-57334-Revision-0

BISON TRISO Training Material

Jason D Hales

March 2020

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/MIS-20-57334-Revision-0

BISON TRISO Training Material

Jason D Hales

March 2020

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy

Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

U.S. DEPARTMENT OF ENERGY | Office of NUCLEAR ENERGY

Clean. Reliable. Nuclear.

TRISO in BISON

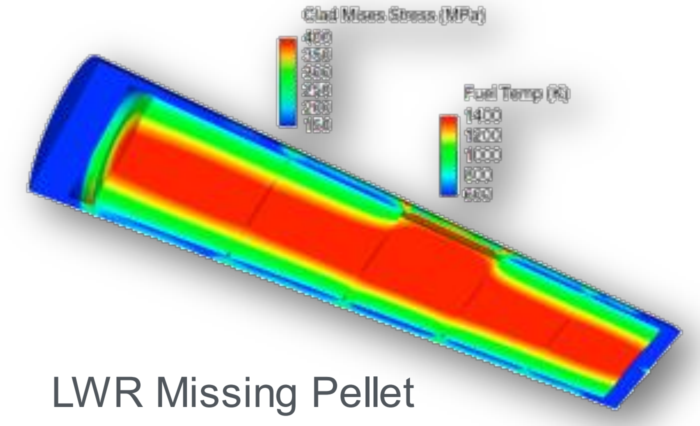
Computational Mechanics and Materials
Idaho National Laboratory

History of BISON

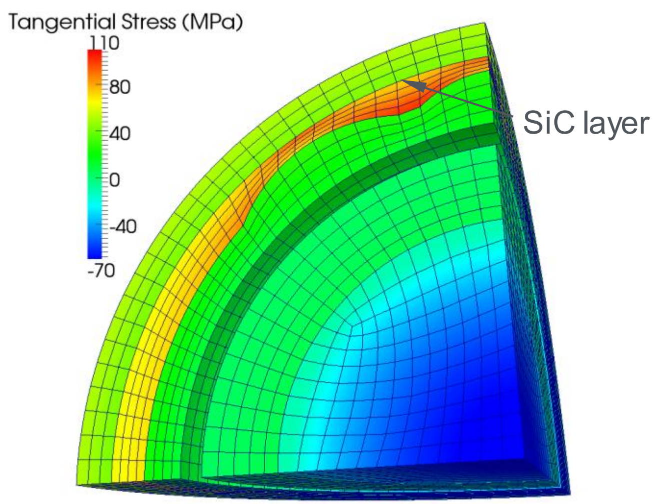
NEAMS/CASL Fuels Programs

Historical Overview

- Overarching objective to deliver an integrated set of **predictive** computational tools for nuclear fuel performance analysis and design
- A multiscale approach has been adopted in which engineering-scale simulations are informed by mesoscale simulations of microstructure evolution, which are enabled by parameters obtained from atomistic simulations
- Primary products are **BISON** for engineering-scale analysis and **Marmot** for mesoscale analysis, both built upon the **MOOSE** computational framework



LWR Missing Pellet Surface Analysis



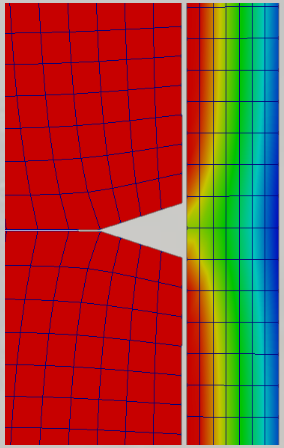
Defective TRISO Particle

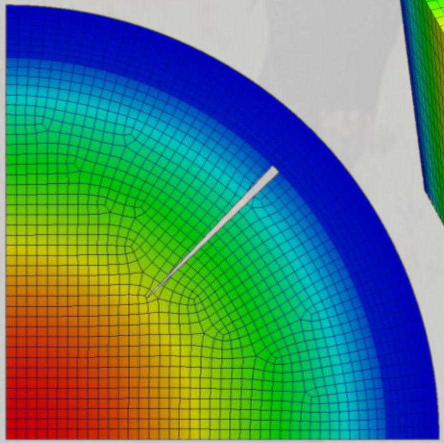
Bison: What is it?

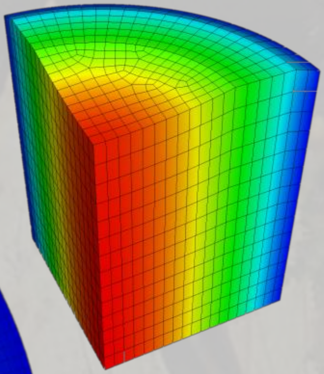
- Bison once numbered in the tens of millions and ranged over much of North America.
- Bison can weigh 1000+ kg and stand 1.8+ m high at the shoulder.
- Bison can jump 1.8 m vertically.
- Bison can run 60+ km/h.

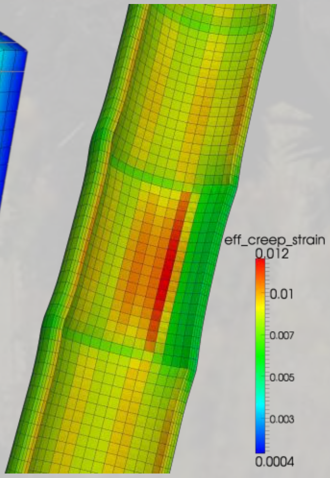
BISON: What is it?

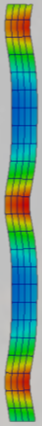
- A finite element, thermo-mechanics code with material models and other customizations to analyze nuclear fuel
  - Accepts user-defined meshes/geometries
    - 1D, 2D, or 3D
  - Runs on one processor or many
  - Analyzes a variety of fuel types
  - Couples to other analysis codes

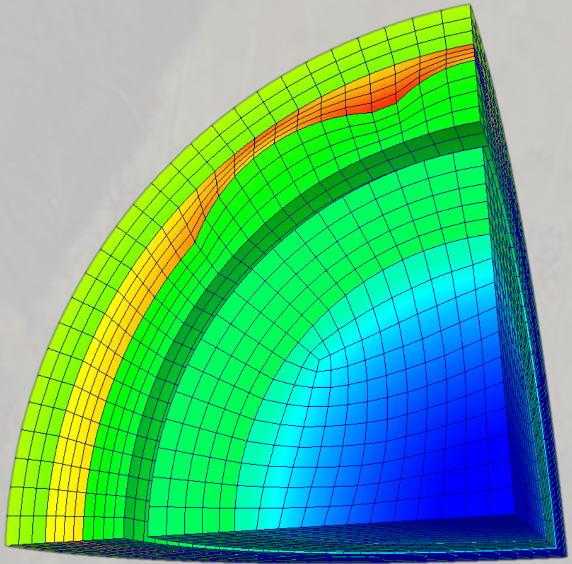






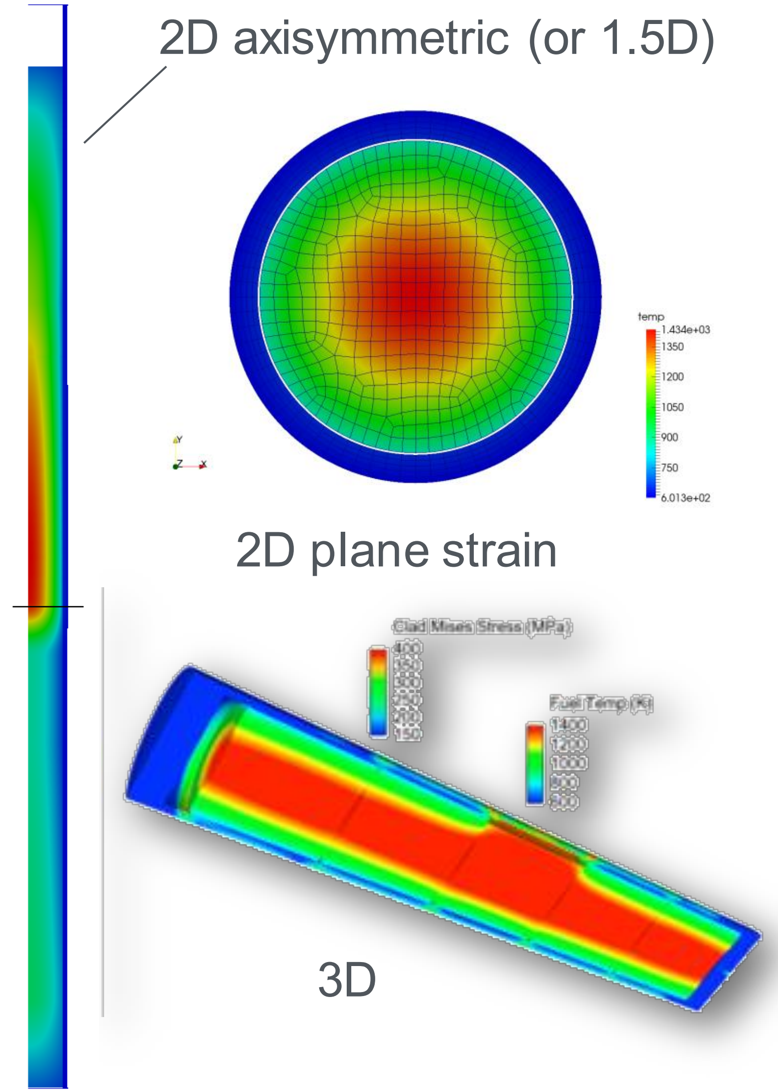






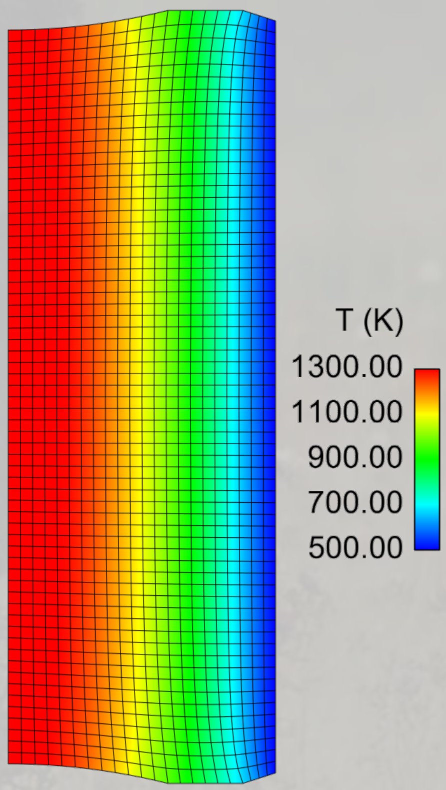
BISON Fuel Performance Code

- Finite element-based engineering scale fuel performance code
- Solves the fully-coupled thermomechanics and species diffusion equations in 1D, 1.5D, 2D axisymmetric or plane-strain, or full 3D
- Used for LWR, ATF, TRISO, and metallic fuels
- Applicable to both steady and transient operations and includes LOCA and RIA capability for LWR fuel
- Readily coupled to lower length scale material models
- Designed for efficient use on parallel computers
- Development follows NQA-1 process



BISON Requirements and Limitations

- BISON requires:
  - An input file that describes thermal and mechanical material models, boundary conditions, initial conditions, power history
  - A mesh provided either directly in the input file or through a separate mesh file
- BISON cannot currently model:
  - Very high strain rate analyses (e.g., car crashes)
  - Structural elements (membranes, shells, beams)
  - Melting or flowing material
- BISON is not:
  - A thermal-hydraulics or CFD code
  - A neutronics code



BISON

BISON Governing Equations

- Energy conservation (transient heat conduction with fission source)

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + E_f \dot{F}$$

Fission density rate = f(**x**, t)

- Species conservation (transient oxygen or fission product diffusion with radioactive decay)

Fickian diffusion

Soret diffusion

$$\frac{\partial C}{\partial t} = \nabla \cdot D\left(\nabla C - \frac{CQ^*}{FRT^2}\nabla T\right) - \lambda C + S$$

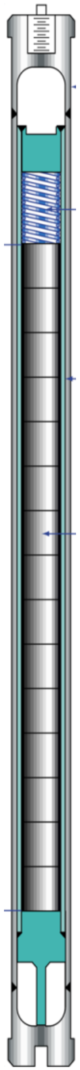
Radioactive decay

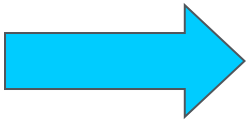
- Momentum conservation (Cauchy's equation of equilibrium)

$$\nabla \cdot \mathbf{T} + \rho \mathbf{f} = 0$$

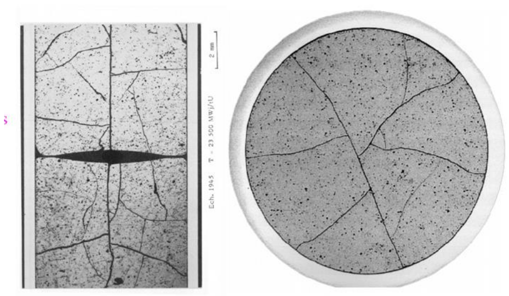
Fuel Behavior During Irradiation

At beginning of life, a fuel element is quite simple . . .



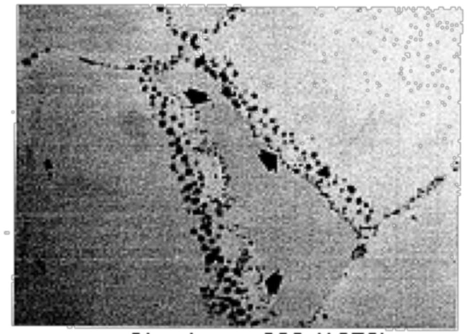


but irradiation brings about substantial complexity



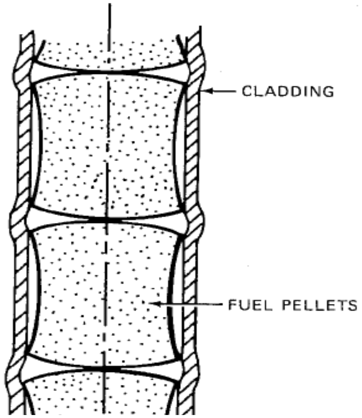
Michel et al, Eng Frac Mech, **75**, 3581 (2008)

Fuel Fracture



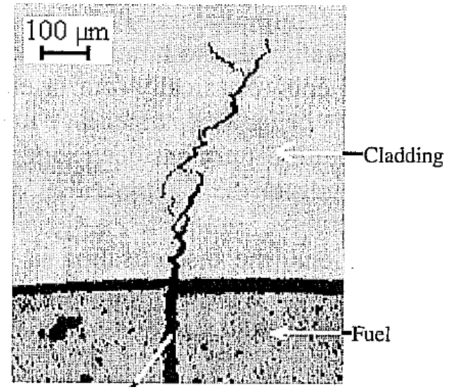
Olander, p. 323 (1978)

Fission gas



Olander, p. 584 (1978)

Multidimensional contact and deformation

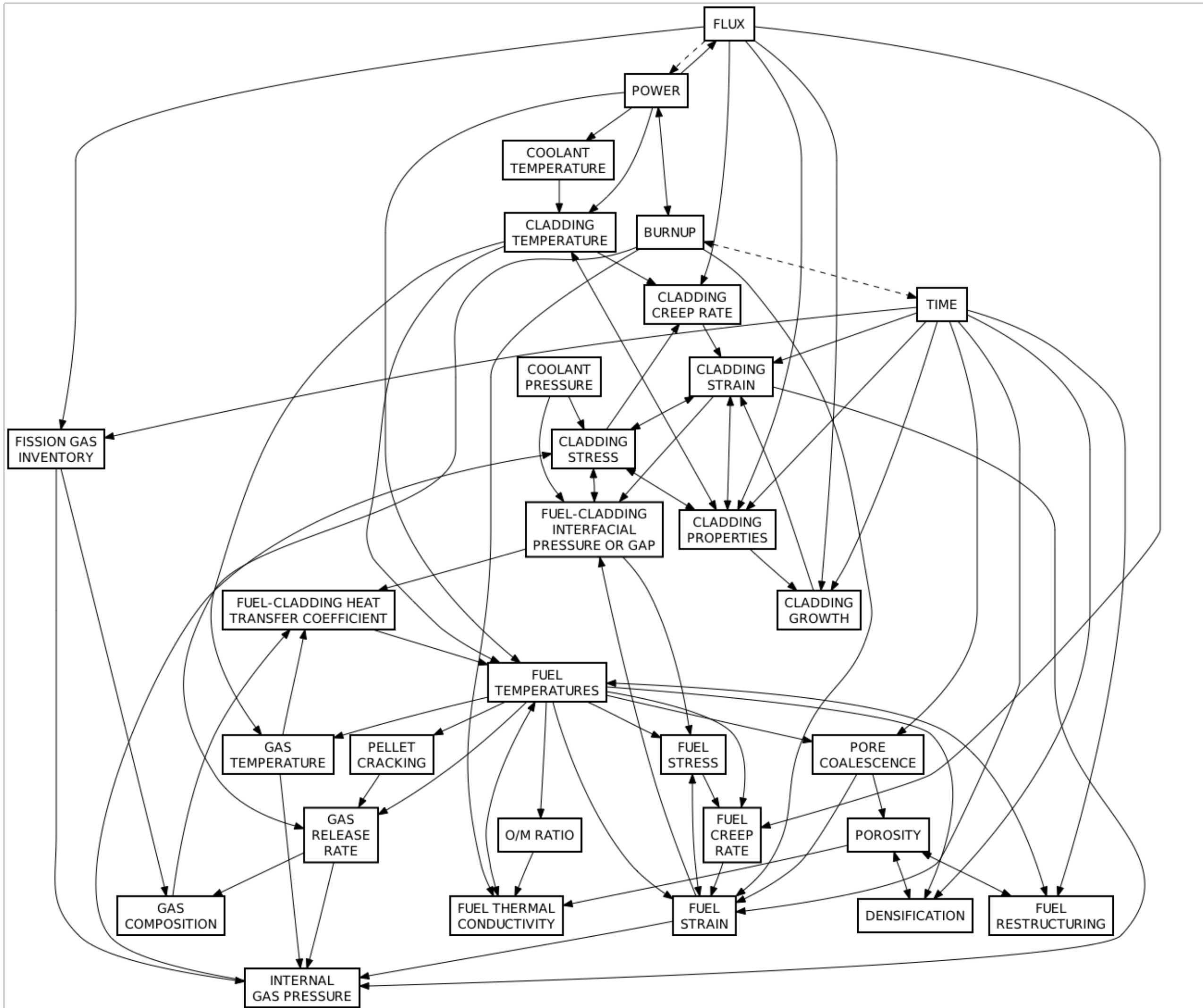


Bentejac et al, PCI Seminar (2004)

Stress Corrosion Cracking Cladding Failure

Fuel Behavior Modeling: Coupled Multiphysics and Multiscale

- **Multiphysics**
  - ➢ **Fully-coupled nonlinear thermomechanics**
  - ➢ **Mass transport**
  - ➢ **Chemistry**
  - ➢ **Neutronics**
  - ➢ **Thermal-hydraulics**
- **Multi-space scale**
  - ➢ **Important physics operate at level of microstructure**
  - ➢ **Need real predictions at engineering scale**
- **Multi-time scale**
  - ➢ **Long, steady operation**
  - ➢ **Short power ramps**
  - ➢ **Rapid transients**



Reproduced from Beyer et al., BNWL-1898, PNPL (1975)

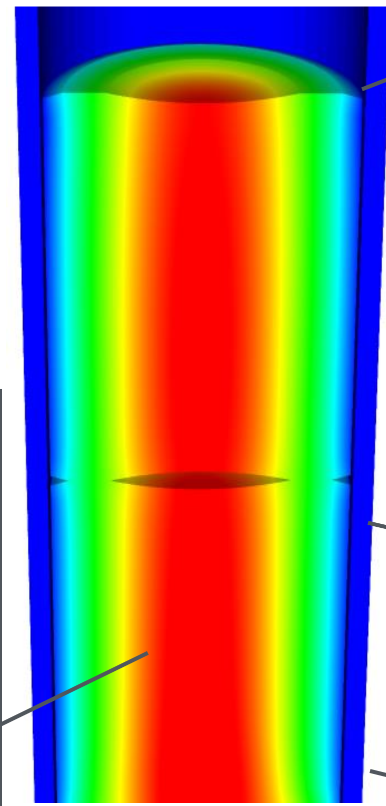
BISON Application: LWR Fuel

General Capabilities

- Finite element based 1D spherical, 2D-RZ and 3D fully-coupled thermo-mechanics with species diffusion
- Linear or quadratic elements with large deformation mechanics
- Steady and transient operation
- Massively parallel computation
- Meso-scale informed material models

Oxide Fuel Behavior

- Temperature/burnup dependent conductivity
- Heat generation with radial and axial profiles
- Thermal expansion
- Solid and gaseous fission product swelling
- Densification
- Thermal and irradiation creep
- Fracture via relocation or smeared cracking
- Fission gas release (two stage physics)
  - transient (ramp) release
  - grain growth and grain boundary sweeping
  - athermal release



Temperature

Gap/Plenum Behavior

- Gap heat transfer with $k_g = f(T, n)$
- Mechanical contact (master/slave)
- Plenum pressure as a function of:
  - evolving gas volume (from mechanics)
  - gas mixture (from FGR model)
  - gas temperature approximation

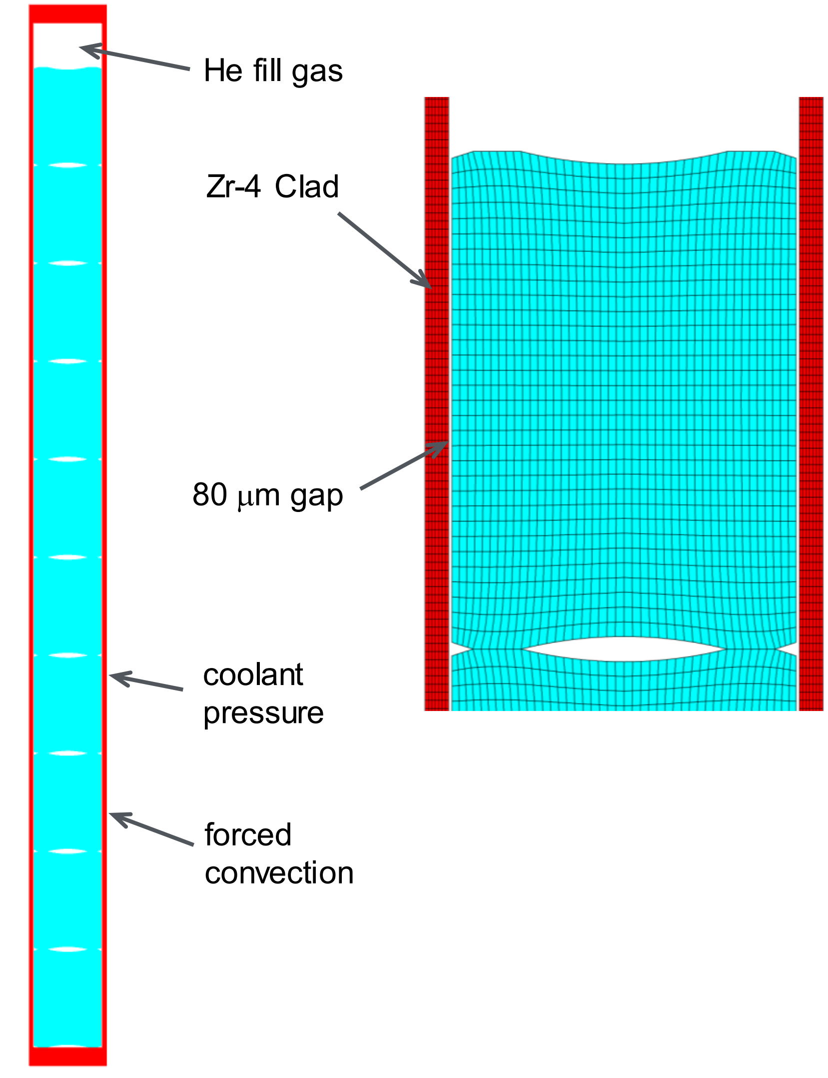
Cladding Behavior

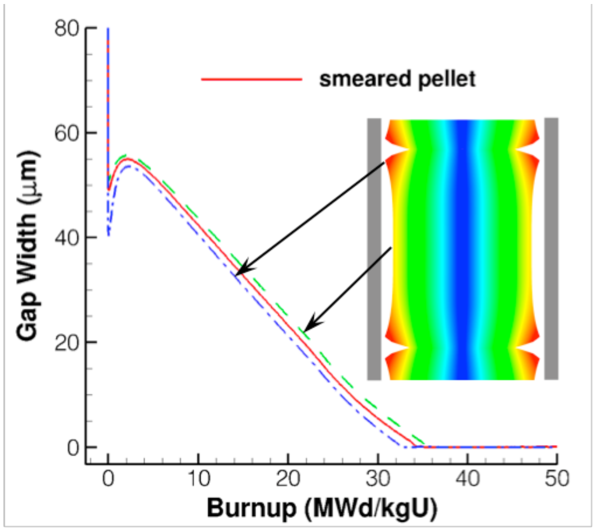
- Thermal expansion
- Thermal and irradiation creep
- Irradiation growth
- Gamma heating
- Combined creep and plasticity

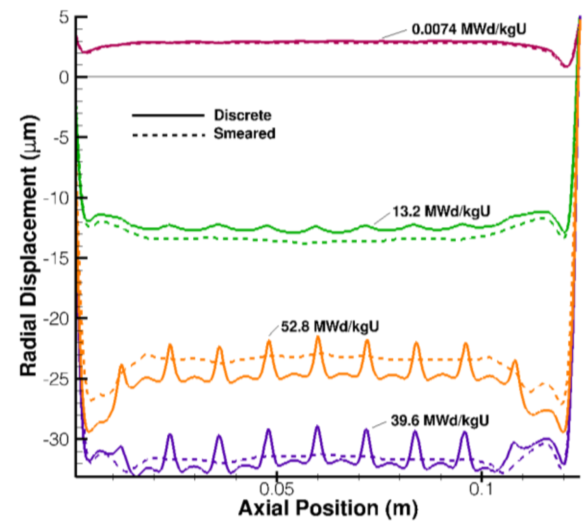
Coolant Channel

- Closed channel thermal hydraulics with heat transfer coefficients

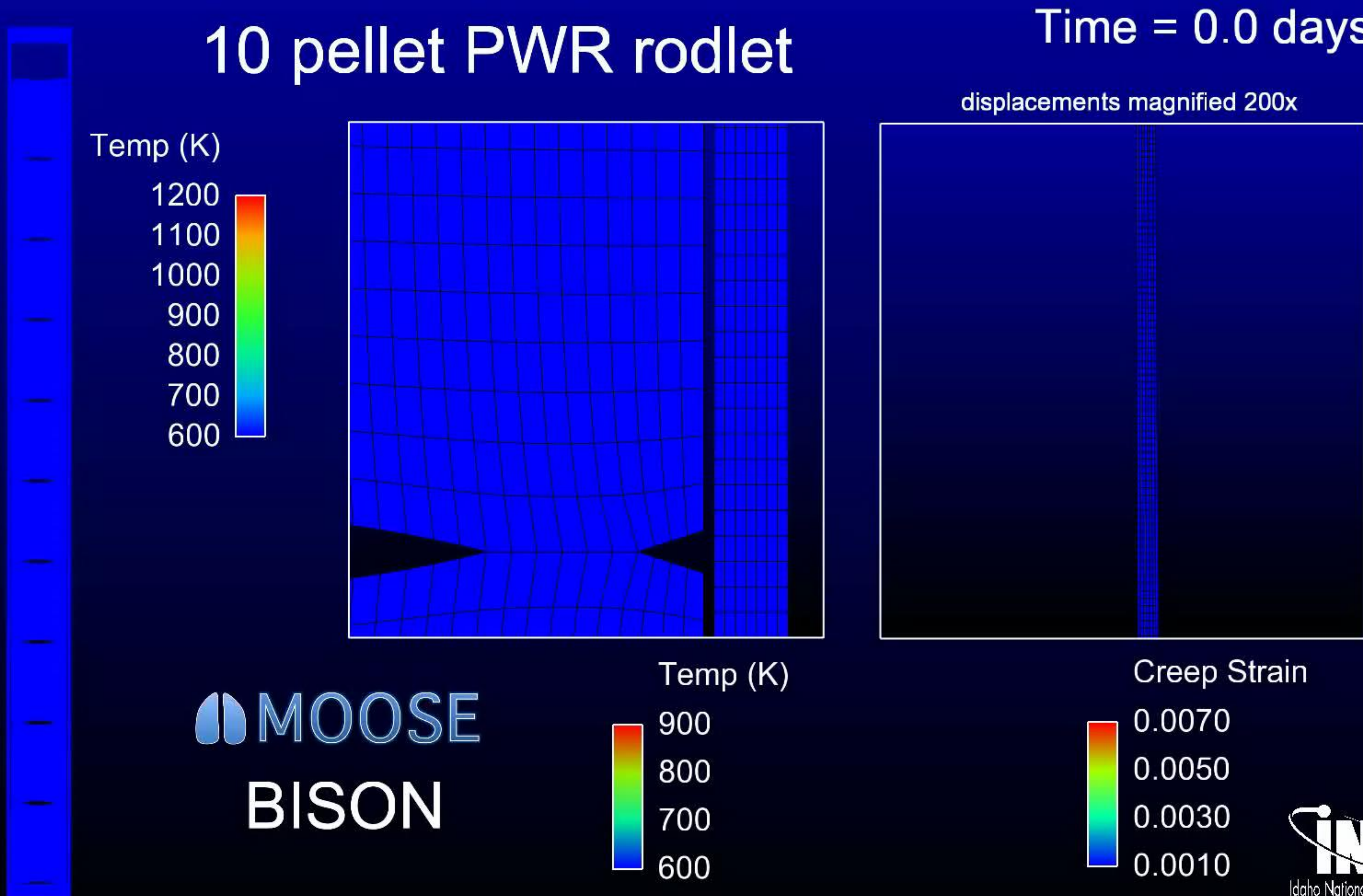
Discrete Pellet LWR Rodlet (2D-RZ multiphysics)







10 pellet PWR rodlet

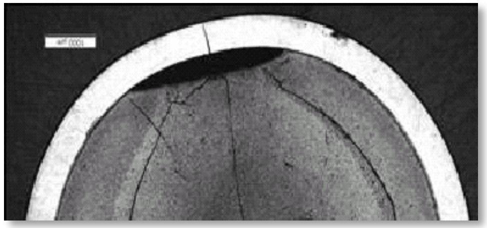


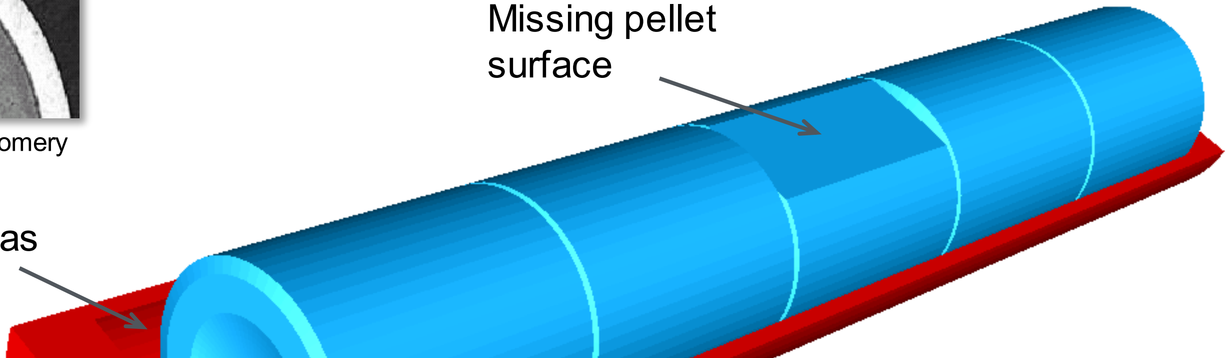
MOOSE

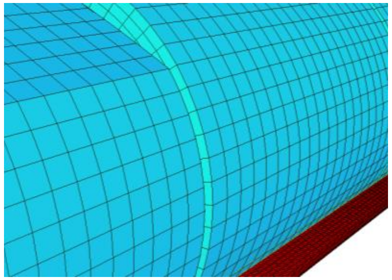
BISON

Idaho National Laboratory

PCMI - Missing Pellet Surface Analysis

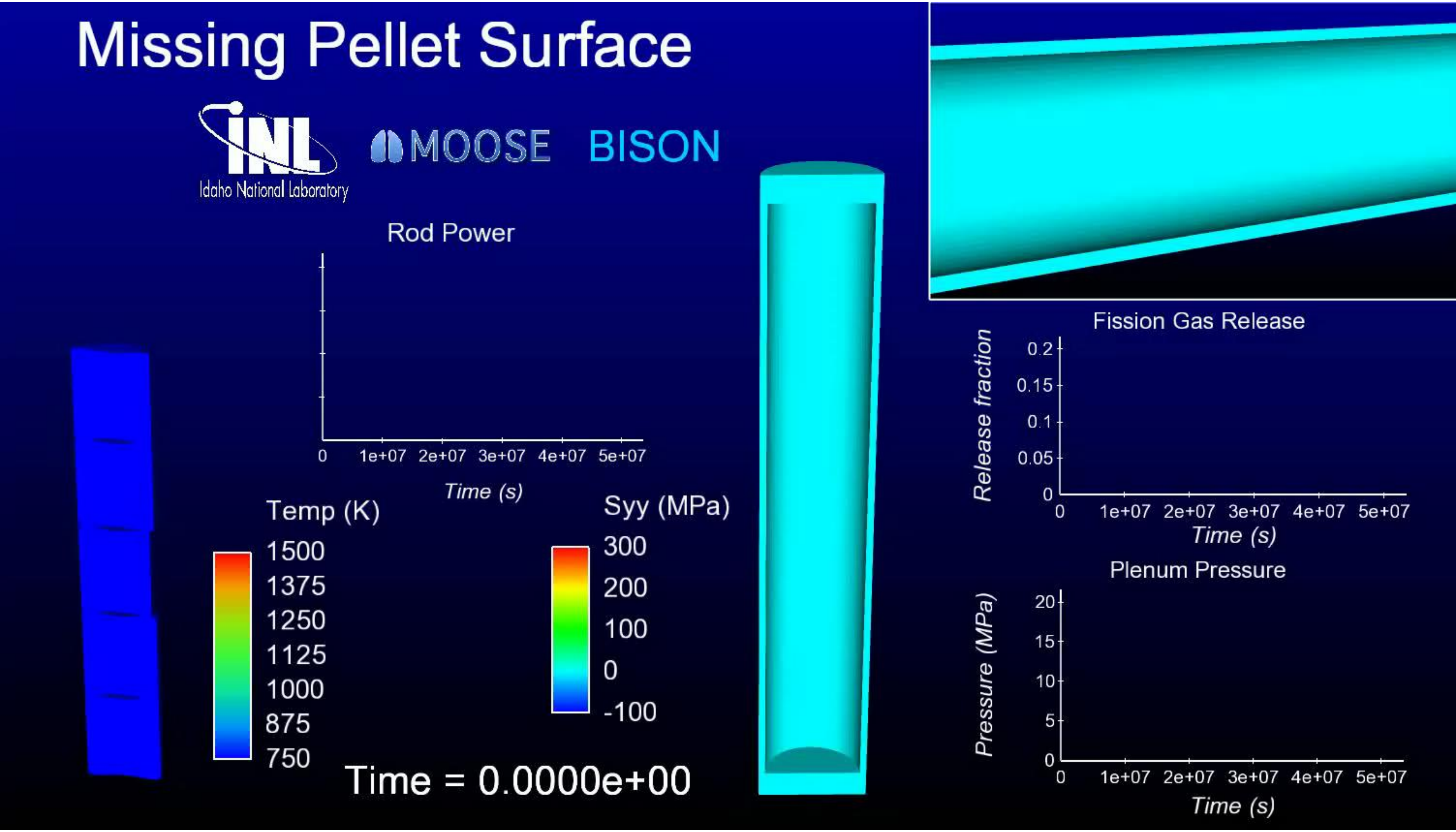

Micrograph from Robert Montgomery





- High resolution 3D calculation (250,000 elements, 1.1×10^6 dof) run on 120 processors
- Simulation from fresh fuel state with a typical power history, followed by a late-life power ramp

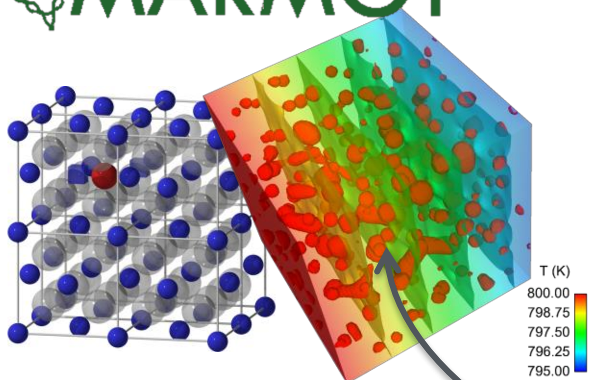
Animation – Missing Pellet Surface with Power Ramp



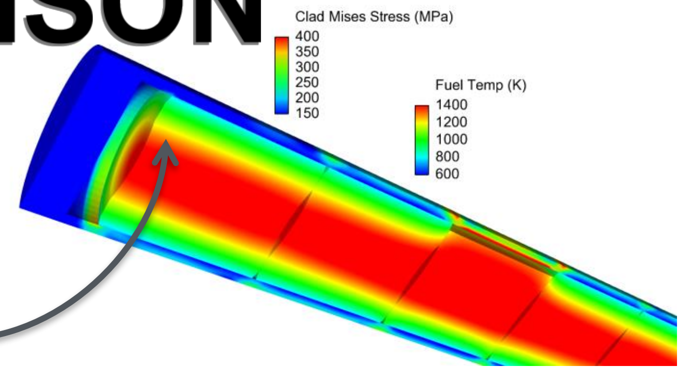
MOOSE-**BISON**-Marmot (MBM)

- The MOOSE-BISON-Marmot (MBM) codes provide an advanced multidimensional, multiphysics, multiscale fuel performance capability

MARMOT



BISON



Atomistic/Mesoscale Material Model Development
- Predicts microstructure evolution in fuel and cladding
- Used with atomistic methods to develop multiscale materials models

MOOSE

Multiphysics Object-Oriented Simulation Environment

- Simulation framework allowing rapid development of FEM-based applications

Advanced Multidimensional Fuel Performance Code
- Models a wide variety of fuel types and geometries at an engineering scale
- Applicable for steady, transient and accident conditions

Nuclear Energy Modeling and Simulation Program

Fuels Technology Area

- New program (FY20+) combines Fuels scope from both NEAMS and CASL
  - Advanced Reactors (Metallic, TRISO, UC/UN)
  - Light Water Reactors (ATF, Increased Enrichment and Extended Burnup, Accident Analysis)
  - Code framework advancements (Robustness, Speed, Ease-of-Use, Documentation, Software Quality)
- **Early stage R&D performed must be relevant to industry and coordinated with the NRC**
- FY20 Work Packages: Managers
  - Engineering Scale Advanced Reactor Fuel Performance: Steve Novascone
  - Engineering Scale LWR Fuel Performance: Giovanni Pastore
  - Advanced Numerical Model Development and Usability Improvements: Daniel Schwen
  - Lower Length Scale Model Development: Larry Aageson
  - Lower Length Scale Model Development – LANL: David Andersson
  - Metallic Fuel Development and Validation – ANL: Latif Yacout
  - Thermochemistry Model Development – ORNL: Srdjan Simunovic

Questions?

Clean. **Reliable.** **Nuclear.**

U.S. DEPARTMENT OF ENERGY | Office of NUCLEAR ENERGY

Clean. Reliable. Nuclear.

MOOSE

TRISO in BISON

Computational Mechanics and Materials
Idaho National Laboratory

MOOSE

- MOOSE website
  - https://mooseframework.org/
- Input file syntax definitions and class documentation
  - https://mooseframework.org/syntax/index.html
- MOOSE physics modules
  - https://mooseframework.org/modules/index.html
- MOOSE workshop
  - https://mooseframework.org/workshop/#/

Questions?

Clean. **Reliable. Nuclear.**

U.S. DEPARTMENT OF ENERGY | Office of NUCLEAR ENERGY

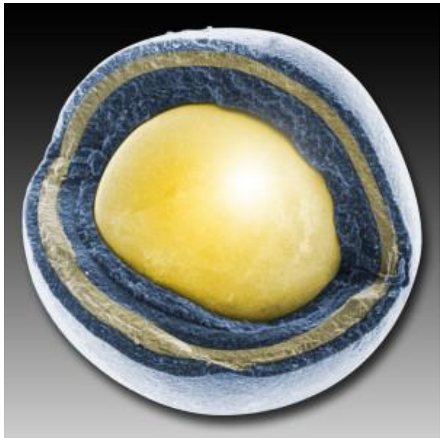
Clean. Reliable. Nuclear.

TRISO in BISON

Computational Mechanics and Materials
Idaho National Laboratory

TRISO in BISON circa 2013 and Today

TRISO in BISON circa 2013



- Rich Williamson started BISON with INL LDRD funds over ten years ago.
- In 2012, the first major BISON paper was published.
  - http://dx.doi.org/10.1016/j.jnucmat.2012.01.012
  - Paper focused on LWR fuel but also had a TRISO example
  - TRISO analysis featured heat conduction and species diffusion
- At about that time, the BISON team decided to add a baseline TRISO capability, including thermal, mass diffusion, and mechanical models.

BISON TRISO Paper, JNM 2013

Journal of Nuclear Materials 443 (2013) 531–543

Contents lists available at ScienceDirect

Journal of Nuclear Materials

journal homepage: www.elsevier.com/locate/jnucmat

Multidimensional multiphysics simulation of TRISO particle fuel

J.D. Hales, R.L. Williamson *, S.R. Novascone, D.M. Perez, B.W. Spencer, G. Pastore

Fuel Modeling and Simulation, Idaho National Laboratory, P.O. Box 1625, Idaho Falls, ID 83415-3840, United States

A R T I C L E I N F O

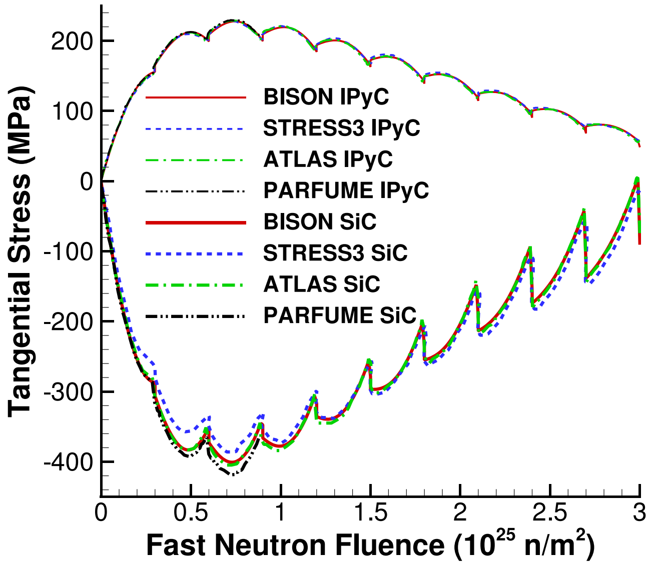
Article history:
Received 12 May 2013
Accepted 30 July 2013
Available online 8 August 2013

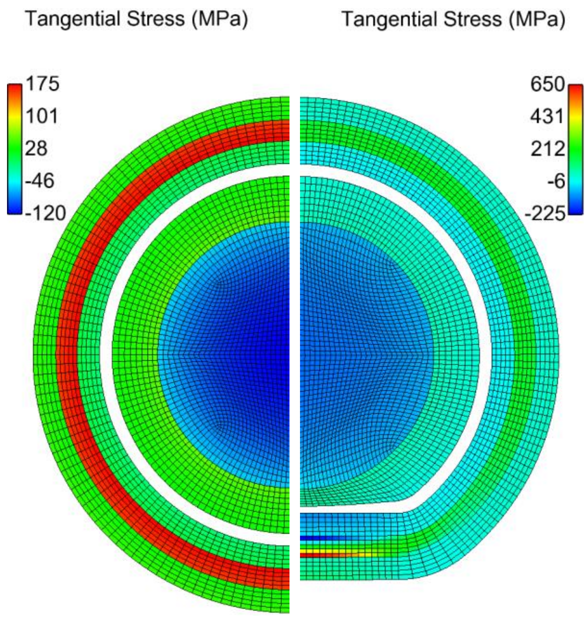
A B S T R A C T

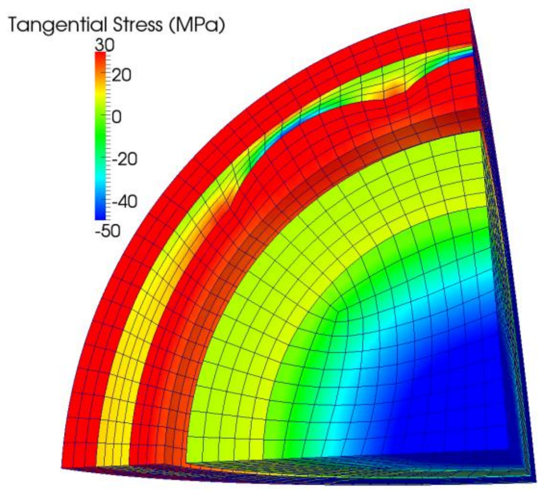
Multidimensional multiphysics analysis of TRISO-coated particle fuel using the BISON finite element nuclear fuels code is described. The governing equations and material models applicable to particle fuel and implemented in BISON are outlined. Code verification based on a recent IAEA benchmarking exercise is described, and excellent comparisons are reported. Multiple TRISO-coated particles of increasing geometric complexity are considered. The code's ability to use the same algorithms and models to solve problems of varying dimensionality from 1D through 3D is demonstrated. The code provides rapid solutions of 1D spherically symmetric and 2D axially symmetric models, and its scalable parallel processing capability allows for solutions of large, complex 3D models. Additionally, the flexibility to easily include new physical and material models and straightforward ability to couple to lower length scale simulations makes BISON a powerful tool for simulation of coated-particle fuel. Future code development activities and potential applications are identified.

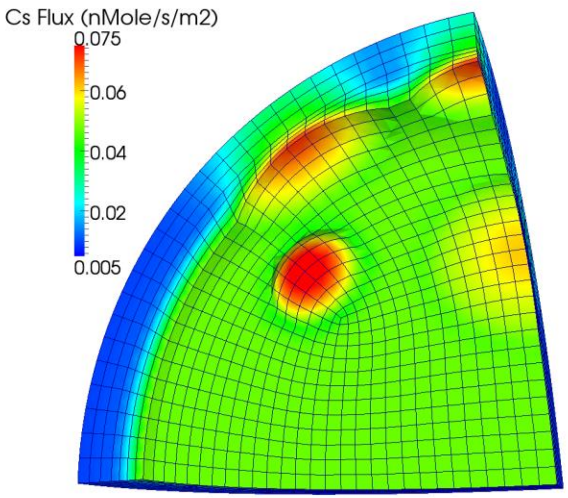
http://dx.doi.org/10.1016/j.jnucmat.2013.07.070

BISON TRISO Paper, JNM 2013









TRISO Stagnation

- After the 2013 paper, we talked with potential stakeholders about our capability.
- But, very little interest.
- No interest → no funding → no development.
- Added an internal 1D TRISO mesh generation capability in 2018.
- Things started to pick up in FY19.

Recent Interest in TRISO in BISON

Kairos Power

U.S.NRC

NASA

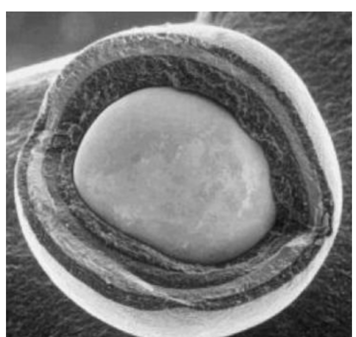
GENERAL ATOMICS

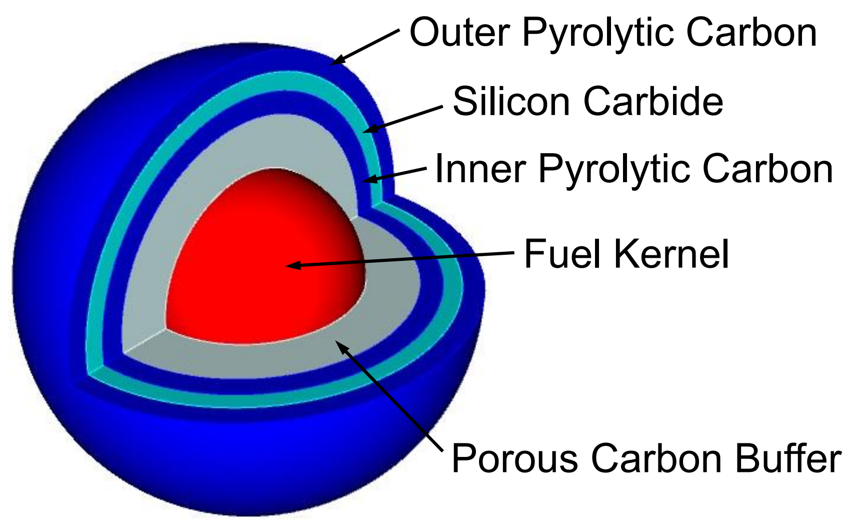
DEPARTMENT OF DEFENSE UNITED STATES OF AMERICA

Westinghouse

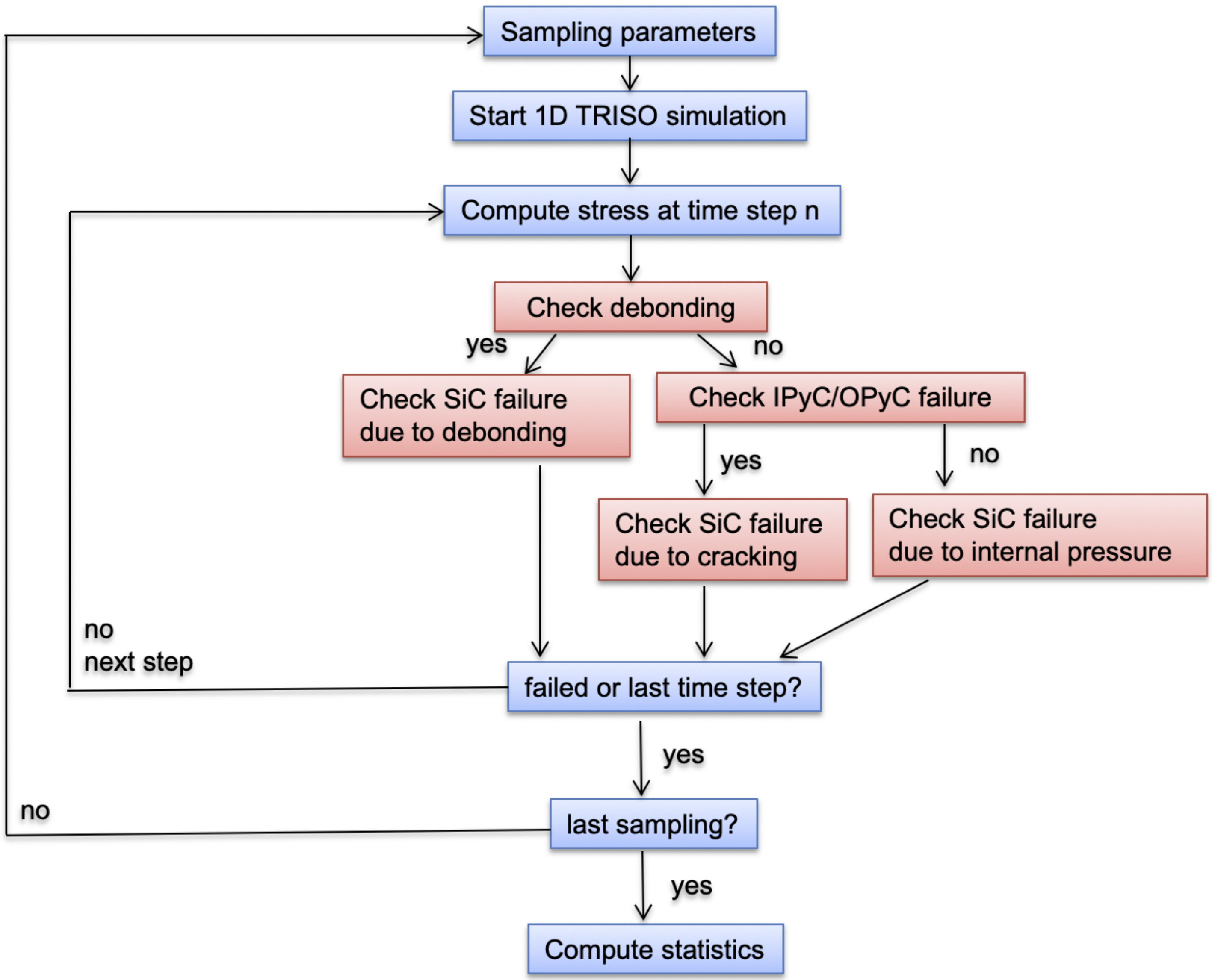
Current TRISO Development

- Adding more complete set of models from PARFUME
  - Elastic properties
  - Thermal properties
  - Mass diffusion properties
  - Kernel
    - Swelling
    - Fission gas release
  - Buffer
    - Creep
    - Irradiation strain
  - PyC
    - Creep
    - Irradiation strain
  - SiC
    - Palladium penetration
  - Matrix





Current TRISO Development Continued: Monte Carlo Scheme for Predicting Failure



Questions?

Clean. **Reliable. Nuclear.**

U.S. DEPARTMENT OF ENERGY | Office of NUCLEAR ENERGY

Clean. Reliable. Nuclear.

TRISO in BISON

Computational Mechanics and Materials
Idaho National Laboratory

Organization of BISON Code, Tests, and Examples

BISON Repository Layout

bison
Nuclear Fuel Performance Code

Unstar 15 | Fork 175 | SSH git@hpcgitlab.hpc.inl.gov:idah | Global

Files (1.7 GB) Commits (10,881) Branches (11) Tags (7) Readme LICENSE Add Changelog Add Contribution guide Set up CI

devel bison / History Find file

Updating stable MOOSE submodule refs #000
Moose Test Account (system account) committed about 15 hours ago
bbe16cc9

Name	Last commit	Last update
.gitlab/issue_templates	Add Feature Request tag. #946	7 months ago
assessment	Specialized versions of several postprocessors for La...	2 days ago
benchmarks	assessment folder reorganized, loca and ria subfolde...	
doc	Merge branch 'kairos_models' into 'devel'	
examples	Specialized versions of several postprocessors for La...	2 days ago
gui	Remove include guards in favor of pragma once. Clos...	8 months ago
iapws95 @ afe14835		
include	Merge branch 'kairos_models' into 'devel'	a day ago
moose @ c64a8e6d		
scripts	Merge branch 'cleanupValidationScripts' into 'devel'	a month ago
src	Merge branch 'kairos_models' into 'devel'	a day ago
test	Merge branch 'kairos_models' into 'devel'	a day ago
tools	converted scripts to python3 #1057	a month ago
unit	Fortran to C++ for caneal MATPRO (#900)	7 months ago
verification	Remove include guards in favor of pragma once. Clos...	8 months ago

This is the view from
https://hpcgitlab.inl.gov/idaholab/bison

BISON src Directory

idaholab > bison > **Repository**

devel | bison / src / | History | Find file

Merge branch 'kairos_models' into 'devel' ...
Gamble, Kyle committed a day ago — dc9f3ca0

Name	Last commit	Last update
..		
actions	Addition of LayeredPlenumTemperaturePostprocessor...	a month ago
auxkernels	Add FissionRateMOX and IntraGranularFissionGas ch...	2 months ago
base	Bison to BISON. #653	2 months ago
bcs	Optional Meyer Hardness Model (Temperature Depen...	a month ago
cxxmatpro	Fortran to C++ for caneal MATPRO (#900)	7 months ago
functions	Fix compile time warnings Ref #706	a month ago
ics	Remove include guards in favor of pragma once. Clos...	
kernels	Add MOX Oxygen Diffusion	
materials	Merge branch 'kairos_models' into 'devel'	
matpro	Another round toward combined creep and plasticity	
mesh	Add class descriptions to SmearedPelletMesh/Genera...	a month ago
meshgenerators	Plate mesh capability. #1073	a week ago
parser	Addition of LayeredPlenumTemperaturePostprocessor...	a month ago
postprocessors	Specialized versions of several postprocessors for La...	2 days ago
userobject	PARFUME models	a day ago
utils	PARFUME models	a day ago
vectorpostprocessors	const Function (MOOSE #13460)	8 months ago
main.C	Bison to BISON. #653	2 months ago

The layout of the include directory is very similar. Source files (.C files) will have corresponding include files (.h files).

BISON test/tests Directory

```
/Users/halejd/gitProjects/bison
inl604289|devel> ls test/tests
2d_arrhenius/
ADMetallicFuelWastage/
Al2O3/
GrainRadiusPorosity_test/
GraphiteMatrixElasticityTensor/
GraphiteMatrixSpecificHeat/
GraphiteMatrixThermalConductivity/
MetallicFuelWastage/
OxideEnergyDeposition/
ThermalFuel_error_messages/
VSwellingU3Si2/
ad_arrhenius_material_property/
ad_ht9_thermal/
ad_upuzr_burnup/
ad_upuzr_fast_neutron_flux/
ad_upuzr_fission_gas_release/
ad_upuzr_fission_rate/
ad_upuzr_thermal/
anisotropic_swelling/
arrhenius_diffusion_coef/
arrhenius_material_property/
average_axial_position/
average_burnup/
axial_relocation/
burnup_action/
carbon_monoxide_production/
check_error/
circular_cross_section_mesh/
constitutive_heat_conduction/
convective_heat_transfer/
coolant_channel_model/
creep_HT9/
creep_SiC/
creep_U10Mo/
creep_mox/
creep_uo2/
creep_upuzr/
cumulative_damage_index/
decay_heating/
diffusion_limited_reaction/
dislocation_density/
dryCask/
effective_burnup_aux/
element_integral_power/
example_problem_test/
failure_cladding_zr/
failurecladHT9/
fast_neutron_flux/
fcci_ht9/
fecral/
fecral_oxidation/
fgr_fraction/
fgr_percent/
fgr_upuzr/
fill_gas_thermal_conductivity/
fission_gas_1d/
fission_gas_behavior_sifgrs/
fission_gas_behavior_u3si2/
fission_gas_release_formas/
fission_rate_LWR/
fission_rate_MOX/
fission_rate_axial/
fission_rate_from_power_density/
fission_rate_heat_source/
fission_to_thermal_power/
fuelrodlinevaluesampler/
gamma_heating/
gap_heat_transfer/
gap_heat_transfer_fission/
gap_heat_transfer_htonly/
gap_heat_transfer_mixedgas/
gap_heat_transfer_radiation/
gap_jump_distance/
gap_perfect_transfer/
generic_material_failure/
grain_radius_aux/
hotpressing_uo2/
hydride/
hydrogen/
ifba_he_production/
increment_limited_time_step/
irradiation_growth/
irradiation_growth_Zr4/
layered2D/
layered_1D/
mamox/
mechHT9/
mechTests/
mechZry/
mechanical_uo2/
meso_thcond_test/
mox_oxygen_to_metal_ratio/
mox_pore_velocity/
oxidation_cladding/
oxygen_aux/
oxygen_transport/
partial_sum_heat_flux/
percolation/
performance_outputs_action/
phase_transition_zircaloy/
phase_upuzr/
plate_mesh/
plenum_pressure/
plenum_temp/
power_peaking_function/
radial_avg_fuel_enthalpy/
radial_crack_counter/
radial_power_factor/
radioactive_decay/
radius_aux/
relocation_UO2/
relocation_recovery_UO2/
side_ave_incr_tensor_component/
side_int_var_incr_postprocess/
side_integral_mass_flux/
smeared_pellet_mesh/
smeared_pellet_mesh_generator/
solid_mechanics_deprecated/
stan_neumann/
standard_lwr_outputs_action/
submodel_end_bc/
swelling/
temperature_jump_distance/
tensor_mechanics/
thermalChromium/
thermalCompositeSiC/
thermalD9/
thermalFastMOX/
thermalFeCrAl/
thermalFuel_HaldenMOX/
thermalFuel_HaldenUO2/
thermalFuel_NFIR/
thermalFuel_NFImod/
thermalHT9/
thermalMAMOX/
thermalMOX/
thermalMonolithicSiC/
thermalNa/
thermalSilicideFuel/
thermalTests/
thermalUO2/
thermal_expansion_zry/
thermal_irradiation_creep/
thermal_irradiation_creep_plas/
thermirrad_creep_zr42/
thermo_mech_oxygen/
triso/
triso_failure/
un_swelling/
upuzr_burnup/
upuzr_dictra/
upuzr_diffusivity/
upuzr_fast_neutron_flux/
upuzr_fission_rate/
upuzr_phase_lookup/
vswelling_upuzr/
zirconium_diffusion/
zrdiffusivity_upuzr/
zrh_formation/
zry_plasticity/
```

BISON has about 1800 regression tests.

BISON examples Directory

```
/Users/halejd/gitProjects/bison
inl612500|devel> ll examples/
1.5D_restart/           2D_plane_strain_rod/    accident_tolerant_fuel/ metal_fuel/             non-cylindrical_fuel/   restart/
1.5D_rodlet_10pellets/  3D_rodlet_3pellets/     axial_relocation/       mox_fuel/               percolation/            spent_fuel/
2D-RZ_rodlet_10pellets/ TRISO/                  fast_mox_sifgrs/        multiapp/               pore_migration/         temperature_tables/

/Users/halejd/gitProjects/bison
inl612500|devel> ll examples/TRISO/
total 0
drwxr-xr-x   6 halejd  INEL-NT\Domain Users   192B Feb 21 14:47 ./
drwxr-xr-x  20 halejd  INEL-NT\Domain Users   640B Feb 21 14:47 ../
drwxr-xr-x   9 halejd  INEL-NT\Domain Users   288B Feb 21 14:47 accident_simulation/
drwxr-xr-x   4 halejd  INEL-NT\Domain Users   128B Feb 21 14:47 full_particle/
drwxr-xr-x   4 halejd  INEL-NT\Domain Users   128B Feb 21 14:47 one_layer/
drwxr-xr-x   4 halejd  INEL-NT\Domain Users   128B Feb 21 14:47 two_layers/

/Users/halejd/gitProjects/bison
inl612500|devel> ls examples
1.5D_restart/           2D_plane_strain_rod/    accident_tolerant_fuel/ metal_fuel/             non-cylindrical_fuel/   restart/
1.5D_rodlet_10pellets/  3D_rodlet_3pellets/     axial_relocation/       mox_fuel/               percolation/            spent_fuel/
2D-RZ_rodlet_10pellets/ TRISO/                  fast_mox_sifgrs/        multiapp/               pore_migration/         temperature_tables/

/Users/halejd/gitProjects/bison
inl612500|devel> ls examples/TRISO/
accident_simulation/ full_particle/       one_layer/           two_layers/

/Users/halejd/gitProjects/bison
inl612500|devel> ls examples/TRISO/full_particle/
1D/ 2D/

/Users/halejd/gitProjects/bison
inl612500|devel> ls examples/TRISO/full_particle/1D
examples            full_particle_1D.i  gold/               tests
```

Example BISON Source File

```
/****************************************************************/
/*            DO NOT MODIFY THIS HEADER                         */
/*                                                              */
/*                          BISON                               */
/*                                                              */
/*    (c) 2015 Battelle Energy Alliance, LLC                    */
/*              ALL RIGHTS RESERVED                             */
/*                                                              */
/*   Prepared by Battelle Energy Alliance, LLC                  */
/*     Under Contract No. DE-AC07-05ID14517                     */
/*     With the U. S. Department of Energy                      */
/*                                                              */
/*     See COPYRIGHT for full restrictions                      */
/****************************************************************/

#include "UCOVolumetricSwellingEigenstrain.h"

registerMooseObject("BisonApp", UCOVolumetricSwellingEigenstrain);

template <>
InputParameters
validParams<UCOVolumetricSwellingEigenstrain>()
{
  InputParameters params = validParams<ComputeEigenstrainBase>();
  params.addClassDescription("Fission-induced swelling (percent per percent FIMA) for UCO");
  params.addParam<Real>("swelling_scale_factor", 1.0, "Multiplier for UCO swelling");
  return params;
}

UCOVolumetricSwellingEigenstrain::UCOVolumetricSwellingEigenstrain(
    const InputParameters & parameters)
  : ComputeEigenstrainBase(parameters),
    _swelling_scale_factor(getParam<Real>("swelling_scale_factor")),
    _burnup(getMaterialProperty<Real>("burnup")),
    _swelling(declareProperty<Real>("swelling"))
{
}

void
UCOVolumetricSwellingEigenstrain::initQpStatefulProperties()
{
  _swelling[_qp] = 0;
  ComputeEigenstrainBase::initQpStatefulProperties();
}

void
UCOVolumetricSwellingEigenstrain::computeQpEigenstrain()
{
  Real volumetric_swelling_strain = _swelling_scale_factor * 0.8 * _burnup[_qp];
  Real strain_component = computeVolumetricStrainComponent(volumetric_swelling_strain);
  _swelling[_qp] = volumetric_swelling_strain;

  _eigenstrain[_qp].zero();
  _eigenstrain[_qp].addIa(strain_component);
}
```

src/materials/tensor_mechanics/UCOVolumetricSwellingEigenstrain.C

Example BISON Test

```
/Users/halejd/gitProjects/bison
inl604289|devel> ls test/tests/triso/UCOVolumetricSwellingEigenstrain/
UCOVolumetricSwellingEigenstrain.i          gold/
UCOVolumetricSwellingEigenstrain_out.csv    tests
```

```
/Users/halejd/gitProjects/bison
inl604289|devel> cat test/tests/triso/UCOVolumetricSwellingEigenstrain/tests
[Tests]
  [./UCOVolumetricSwellingEigenstrain]
    type = 'CSVDiff'
    input = 'UCOVolumetricSwellingEigenstrain.i'
    csvdiff = 'UCOVolumetricSwellingEigenstrain_out.csv'
    requirement = "BISON shall calculate volumetric swelling of UCO."
    design = 'UCOVolumetricSwellingEigenstrain.md'
    issues = '#1074'
  [../]
[]
```

Example BISON Test Continued

```
# UCO fission-induced swelling
# The geometry is a unit cube made of UCO subject to burnup-induced swelling.
#
# The swelling is simply 0.8 * burnup.  Burnup is ramped from 0 to 0.125.
# Thus, swelling increases to 0.1.  The final volume is 1.1.

[GlobalParams]
  # Set initial fuel density, other global parameters
  displacements = 'disp_x disp_y disp_z'
  volumetric_locking_correction = true
  order = FIRST
  family = LAGRANGE
[]

[Mesh]
  [mesh]
    type = GeneratedMeshGenerator
    dim = 3
    xmax = 1.0
    ymax = 1.0
    zmax = 1.0
  []
[]

[Problem]
  coord_type = XYZ
[]
```

View remainder of input file in editor.

Example BISON Example

```
/Users/halejd/gitProjects/bison
inl612500|devel> ls examples/TRISO/full_particle/1D
examples              full_particle_1D.i  gold/              tests

/Users/halejd/gitProjects/bison
inl612500|devel> cat examples/TRISO/full_particle/1D/tests
[Tests]
  [./full_particle]
    type = RunApp
    input = 'full_particle_1D.i'
    check_input = True
    method = opt
  [../]
[]
```

Example BISON Example Continued

```
[GlobalParams]
  density = 10810.0
  order = SECOND
  family = LAGRANGE
  displacements = 'disp_x'
[]

[Mesh]
  type = TRISO1DMesh
  elem_type = EDGE3
  coordinates = '0 2.485e-4 3.425e-4 3.425e-4 3.835e-4 4.195e-4 4.595e-4'
  mesh_density = '6 6 0 6 8 6'
  block_names = 'fuel buffer IPyC SiC OPyC'
[]

[Problem]
  coord_type = RSPHERICAL
[]

[Variables]
  [./temperature]
    initial_condition = 1346.0
  [../]
[]

[AuxVariables]
  [./fluence]
  [../]
  [./fast_neutron_flux]
  [../]
  [./fission_rate]
    block = fuel
  [../]
  [./burnup]
    block = fuel
  [../]
  [./grain_radius]
    initial_condition = 5.0e-6
  [../]
[]

[Modules/TensorMechanics/Master]
  [./fuel]
    block = fuel
    add_variables = true
    strain = FINITE
```

View remainder of input file in editor.

Questions?

Clean. **Reliable. Nuclear.**

TRISO in BISON

Computational Mechanics and Materials
Idaho National Laboratory

Running and Evaluating a Model in BISON

Running and Evaluating a Model in BISON

- Let's look at test/tests/triso/UCOElasticityTensor/UCOElasticityTensor.i.
- We will
  1. Walk through the input file step by step.
  2. Run the file and review the information printed to the terminal.
  3. Examine the CSV and Exodus output.

1. Review Input File

- In the terminal or an editor, view test/tests/triso/UCOElasticityTensor/UCOElasticityTensor.i

```
# Elastic Properties of UCO
# The geometry is a unit cube made of UCO material (initial density = 11.25 g/cm^3)
#   subject to elastic strain.
# Displacement boundary conditions are applied to induce a strain in the x-axis
#   only such that the density becomes 8 g/cm^3.
# The temperature is varied from 673.15 to 2073.15 K.
#
# The analytic solution for stress xx is compared to the BISON result in the xlsx file.
#
# Sample from the xlsx file:
#
# Temp (C) | stress_xx | BISON stress_xx
# ----------+-----------+----------------
#  400.00  | 0.00E+00  | 0.00E+00
#  470.00  | 6.24E+09  | 6.24E+09
#  540.00  | 1.15E+10  | 1.15E+10
#  610.00  | 1.59E+10  | 1.59E+10
#  680.00  | 1.96E+10  | 1.96E+10

[GlobalParams]
  # Set initial fuel density, other global parameters
  displacements = 'disp_x disp_y disp_z'
  order = FIRST
  family = LAGRANGE
  initial_enrichment = 0.15 #[wt-]
  O_U = 1.5
  C_U = 0.4
[]

[Mesh]
  [mesh]
    type = GeneratedMeshGenerator
    dim = 3
    xmax = 1.0
    ymax = 1.0
    zmax = 1.0
  []
[]
```

2. Run Input File

- First, compile bison-opt.

```
> make
```

- Next, go to the directory containing the input file.

```
> cd test/tests/triso/UCOElasticityTensor
```

- Finally, run the input file.

```
> ../../../../bison-opt -i UCOElasticityTensor.i
```

- Information to the terminal: BISON header and version, problem information, solve progress, and Postprocessor values.

3. Examine Results

- First, view the CSV file.

```
> cat UCOElasticityTensor_out.csv
```

- Next, view the Exodus output file.
- There isn’t one!
- Use the command line to add the Exodus output option.

```
> ../../../../bison-opt -i UCOElasticityTensor.i Outputs/exodus=true
```

- Load the Exodus output file in Paraview.

```
> paraview UCOElasticityTensor_out.e
```

Questions?

Clean. **Reliable. Nuclear.**

TRISO in BISON

Computational Mechanics and Materials
Idaho National Laboratory

IAEA Benchmark Cases

[1] Hales, et al., *JNM*, 443, 2013.

[2] Advances in high temperature gas cooled reactor fuel technology. Technical Report IAEA-TECDOC-1674, International Atomic Energy Agency, 2012.

IAEA Benchmark Cases

- IAEA cases studied in 2013 BISON TRISO paper:

Case	Geometry	Description
1	SiC layer	Elastic only
2	IPyC layer	Elastic only
3	IPyC/SiC	Elastic with no fluence
4a	IPyC/SiC	Swelling and no creep
4b	IPyC/SiC	Creep and no swelling
4c	IPyC/SiC	Creep and swelling
4d	IPyC/SiC	Creep- and fluence-dependent swelling
5	TRISO	350 μm kernel, real conditions
6	TRISO	500 μm kernel, real conditions
7	TRISO	Same as 6 with high BAF PyC
8	TRISO	Same as 6 with cyclic temperature
10	HFR-K3	10% FIMA, 5.3×10^{-25} n/m^2 fluence
11	HFR-P4	14% FIMA, 7.2×10^{-25} n/m^2 fluence

Simple cases with analytical solutions (cases 1–3)

More complex material behavior, but fully prescribed. (cases 4a–4d)

Single particle with realistic conditions. (cases 5–8)

Internal pressure set by FGR, CO production. (cases 10–11)

IAEA Benchmarks from BISON TRISO JNM Paper

the normal direction; and either the penetration distance or the contact force must be zero at all times.

In BISON, these contact constraints are enforced through the use of node/face constraints. Specifically, the nodes on one side of the interface are prevented from penetrating faces on the other side of the interface. This is accomplished in a manner similar to that detailed by Heinstein and Laursen [37]. First, a geometric search determines which nodes have penetrated faces. For those nodes, the internal force computed by the divergence of stress is moved to the appropriate face at the point of contact. Those forces are distributed to the nodes attached to the face by employing finite element shape functions.

The tangential relationship between the contacting surfaces is modeled as free slip or frictionless.

2.4. Particle internal pressure

The pressure in the buffer and gap regions of the particle is computed based on the ideal gas law,

$$P = \frac{nRT}{V} \tag{28}$$

where P is the pressure, n is the moles of gas, R is the ideal gas constant, T is the temperature, and V is the gas-filled volume between the fuel and the IPyC layer. This pressure is applied to the surface of the buffer and to the inner surface of the IPyC layer.

The moles of gas, the temperature, and the cavity volume in this equation are free to change with time. The moles of gas n at any time is the original amount of gas (computed based on original pressure, temperature, and volume) plus the amount in the cavity due to fission gas released and carbon monoxide produced. The fission gas released is divided into 0.153 parts krypton and 0.847 parts xenon. The carbon monoxide production is based on oxygen production as described in Eq. (10).

The temperature T is taken as the average temperature of the buffer layer exterior and IPyC inner surfaces, though any other measure of temperature could be used. It is assumed that initial porosity of the buffer layer is open and available to accommodate fission product gases and carbon monoxide. This initial volume prior to gap opening is computed by taking the volume of the buffer layer minus the fraction of that layer presumed to be pyrolytic carbon. This fraction was assumed to be the ratio of the buffer density to the IPyC density. The cavity volume V is recomputed as needed based on the evolving geometry.

The ideal gas law is used here for simplicity. It would be straightforward to implement a more complex law suitable for higher temperatures.

Table 1
IAEA CRP-6 benchmark cases considered in the BISON coated-particle verification exercise. HFR-K3 and HFR-P4 are German pebble and fuel element experiments, respectively.

Case	Geometry	Description
1	SiC layer	Elastic only
2	IPyC layer	Elastic only
3	IPyC/SiC	Elastic with no fluence
4a	IPyC/SiC	Swelling and no creep
4b	IPyC/SiC	Creep and no swelling
4c	IPyC/SiC	Creep and swelling
4d	IPyC/SiC	Creep- and fluence-dependent swelling
5	TRISO	350 μm kernel, real conditions
6	TRISO	500 μm kernel, real conditions
7	TRISO	Same as 6 with high BAF PyC
8	TRISO	Same as 6 with cyclic temperature
10	HFR-K3	10% FIMA, 5.3×10^{-25} n/m^2 fluence
11	HFR-P4	14% FIMA, 7.2×10^{-25} n/m^2 fluence

3. Code verification

As part of an International Atomic Energy Agency (IAEA) Coordinated Research Program (CRP-6) on HTGR reactor fuel technology, a set of benchmarking activities were developed to compare fuel performance codes under normal operation and operational transients [18]. Sixteen benchmark cases were identified, ranging in complexity from a simple fuel kernel having a single elastic coating layer, to realistic TRISO-coated particles under a variety of irradiation conditions. In each case, the particle geometry, constitutive relations, material properties and operating conditions were carefully prescribed to minimize differences between the various code predictions; details are given in [18]. As an early code verification exercise, BISON has been applied to 13 of the 16 benchmark cases as summarized in Table 1.

In the present study, the models for all benchmark cases use eight quadratic finite elements across the width of each coating layer. For cases 1 and 2, numerical solutions were also obtained with twelve elements across the coating layer to determine whether the mesh was sufficiently refined. Maximum tangential stresses obtained from the eight- and twelve-element models differed at most by 0.1%, demonstrating adequate mesh convergence with eight elements.

Cases 1–3 were limited to single and double coating layers and tested simple elastic thermomechanical behavior against analytical solutions. A comparison of the analytical and BISON numerical solutions for the maximum tangential stress, which occurs at the inner surface of the various layers is shown in Table 2. Comparisons are excellent.

Cases 4a–4d included both IPyC and SiC layers and investigated pyrolytic carbon layer behavior under a variety of conditions. Cases 5–8 considered a single TRISO particle with more complexity added with each subsequent case. For cases 1–4d, the internal gas pressure was fixed at 25 MPa while cases 5–8 included a linear pressure ramp. The particle temperature was held uniform at 1273 K for cases 1–7, but for case 8 was cycled ten times between 873 and 1273 K, characteristic of fuel in a pebble bed reactor. For cases 4–7, Table 3 compares BISON computed solutions to the range of solutions from eight coated-particle fuel codes included in the CRP-6 exercise (see [18]). Comparisons are of the tangential stress at the inner surface of both the IPyC and SiC layers, at the end of irradiation. The BISON solutions are always within the range of values computed by the other codes. Note that tabulated values defining the ranges were extracted from plots in [18] and are thus not precise.

Although code comparisons in Table 3 are provided only at the end of irradiation, comparisons were made at various intermediate times during the irradiation period. The BISON solutions were always within the range of solutions produced by the CRP-6 codes.

Fig. 3 compares solutions for case 8, which involved a cyclic particle temperature, during the full irradiation history. In this figure, BISON solutions of the tangential stress at the inner wall of the IPyC and SiC layers are compared to solutions from three codes from the CRP-6 exercise, namely PARFUME [11], ATLAS [15] and STRESS3 [38]. As above, data for the code comparisons were extracted from plots in [18]. For the IPyC layer, the four solutions essentially overlay each other during the entire irradiation period.

Table 2
Comparison of the BISON computed maximum tangential stress (MPa) to the analytical solution for Cases 1–3.

Case	Layer	Analytical	BISON	Error (%)
1	SiC	125.19	125.23	0.032
2	IPyC	50.200	50.287	0.173
3	IPyC/SiC	8.8/104.4	8.7/104.5	1.14/0.10

Table 3
Comparison of the BISON computed tangential stress (MPa) to the range of values computed by the codes included in the CRP-6 exercise. Comparisons are at the inner surface of each layer and at the end of irradiation.

Case	Layer	CRP-6 codes [range]	BISON
4a	IPyC/SiC	[925,970]/[−775,−850]	928/−819
4b	IPyC/SiC	[−25,−25]/[138,142]	−25.0/139
4c	IPyC/SiC	[25,27]/[83,92]	26.0/89.4
4d	IPyC/SiC	[25,35]/[71,88]	27.8/87.0
5	IPyC/SiC	[40,58]/[−56,−28]	41.9/−32.2
6	IPyC/SiC	[27,38]/[28,48]	29.2/44.9
7	IPyC/SiC	[37,50]/[10,25]	38.0/24.6

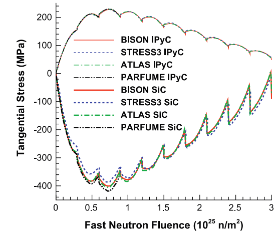


Fig. 3. Code comparison for case 8, which included a ten cycle temperature history. Plotted is the tangential stress at the inner wall of the IPyC and SiC layers.

In the SiC layer, the four solutions are quite similar but some differences are evident, particularly for the first four temperature cycles. The BISON solution falls roughly midway between the PARFUME and STRESS3 solutions and is essentially identical to the ATLAS solution.

Cases 9–13 in CRP-6 were more complicated benchmarks based on past or planned experiments with TRISO-coated particles. The two cases considered here (10 and 11) were based on German fuel from pebble and fuel element experiments. Again, details are provided in [18]. Although material properties and constitutive relations were prescribed for these cases, they differed from cases 1–8 in two ways: (1) the internal pressure was not fixed but instead determined by fission gas release and CO production and (2) the particle size was prescribed as a population (mean value and standard deviation) rather than a single value. BISON solutions were based on the gas release and CO production models described above; however, for simplicity, only a single particle size was considered based on the mean particle diameter.

Fig. 4 provides code comparisons of the total gas pressure (Fig. 4a) and tangential stress at the inner wall of the SiC layer (Fig. 4b) for benchmark cases 10 and 11. Again, BISON is compared to three codes from the CRP-6 exercise. Substantial differences exist in these solutions, particularly for the gas pressure. The BISON solution histories, however, compare well to the range of solutions given by the three well-established codes chosen for comparison.

As stated in [18], the differences between various code predictions shown in Figs. 4a and 4b can be largely attributed to the models used to calculate fission gas release and CO production in the kernel. A detailed description of these models is not available in [18], limiting more detailed investigation. One obvious and significant difference is that both BISON and ATLAS employ the simple Proksch et al. [34] empirical model for CO production while PARFUME [11] uses a detailed thermochemical model.

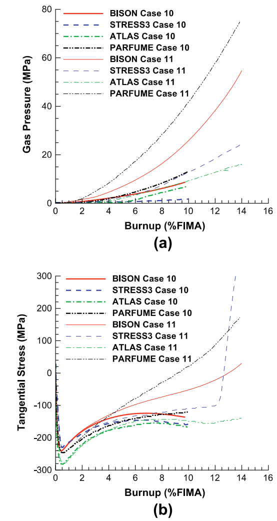


Fig. 4. Code comparisons of the total gas pressure (a) and tangential stress at the inner wall of the SiC layer (b) for benchmark cases 10 and 11.

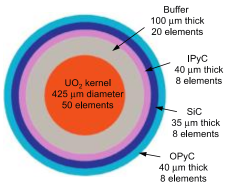


Fig. 5. Schematic showing the particle geometry assumed in the model. For a 1D mesh, the number of finite elements used in each layer is indicated.

4. Demonstration problems

Three problems are considered to demonstrate BISON applicability to coated-particle fuel. The first is a 1D spherically symmetric TRISO-coated particle evaluated during periods of normal irradiation, storage, and accident testing. The second problem considers

Explore the Benchmark Cases

Questions?

Clean. **Reliable. Nuclear.**

TRISO in BISON

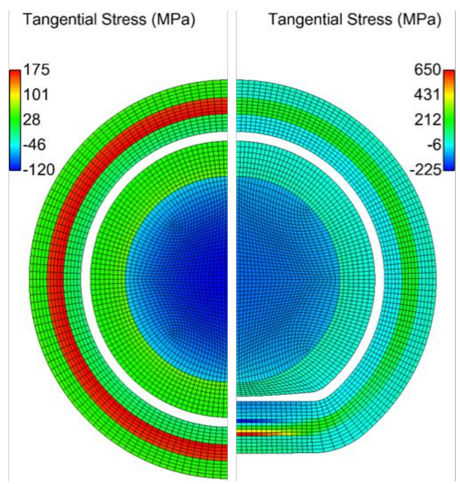
Computational Mechanics and Materials
Idaho National Laboratory

Meshing

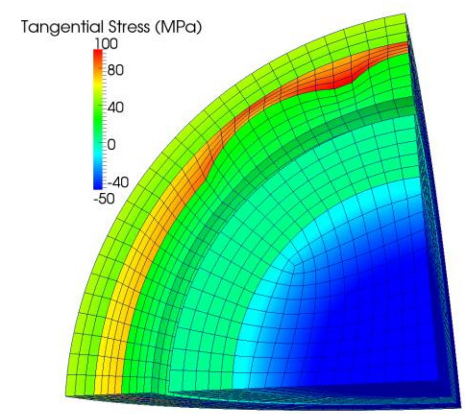
Mesh Generation

- Clearly mesh generation depends on the type of analysis to be run.
  - 1D (fast, spherically symmetric)
  - 2D (medium, axisymmetric)
  - 3D (slow, symmetry across planes or not symmetric)

[BISON]



[BISON/CUBIT]



[CUBIT]

Example BISON Documentation File

TRISO 1D Mesh

Creates a 1D mesh for use with TRISO analysis

Description

Creates a 1D mesh appropriate for use in TRISO analysis. The user supplies radial coordinates that mark the boundaries of mesh blocks. A list of numbers of elements per block is also supplied. Sidesets are created at each mesh boundary.

Example Input Syntax

```
[Mesh]
  type = TRISO1DMesh
  elem_type = EDGE2
  coordinates = '0 .1 .1 .2'
  mesh_density = '1 0 2'
  block_names = 'fred wilma'
[]
```

(test/tests/triso/mesh/mesh_with_coincident_nodes.i)

Screenshot from https://mooseframework.org/bison/source/mesh/TRISO1DMesh.html

Input Parameters

Required Parameters

coordinates Radial coordinates of mesh block boundaries.

mesh_density A list giving the number of elements in each interval (could be zero for a gap).

Optional Parameters

allow_renumbering (True) If allow_renumbering=false, node and element numbers are kept fixed until ...

block_names A list of names to be assigned to the mesh blocks.

elem_type (EDGE3) The type of element from libMesh to generate

ghosting_patch_size The number of nearest neighbors considered for ghosting purposes when 'iterati...

max_leaf_size (10) The maximum number of points in each leaf of the KDTree used in the nearest neigh...

parallel_type (DEFAULT) DEFAULT: Use libMesh::ReplicatedMesh unless --distributed-mesh is specifi...

1D Mesh Generation in BISON

- Mesh generation for 1D TRISO in BISON is done using TRISO1DMesh.
- TRISO1DMesh supports an arbitrary number of layers.

```
[Mesh]
  type = TRISO1DMesh
  elem_type = EDGE3
  coordinates = '0 2.485e-4 3.425e-4 3.425e-4
3.835e-4 4.195e-4 4.595e-4'
  mesh_density = '6 6 0 6 8 6'
  block_names = 'fuel buffer IPyC SiC OPyC'
[]
```

(See assessment/TRISO/benchmark/IAEA_CRP-6/case_11/case_11_1D.i)

1D Mesh Generation in BISON

- Run Case 11.

```
> cd assessment/TRISO/benchmark/IAEA_CRP-6/case_11
> ../../../../../bison-opt -i case_11_1D.i
```

1D Mesh Generation in BISON

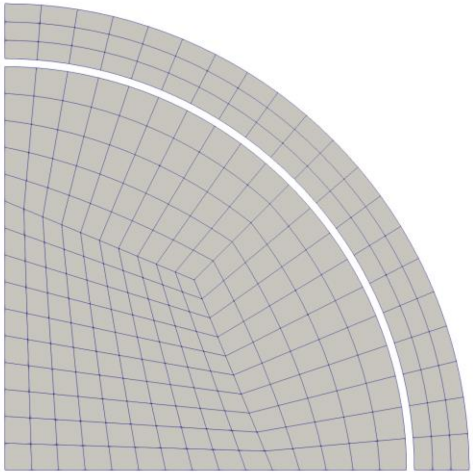
- View mesh for Case 11.

```
> paraview case_11_1D_out.e
```

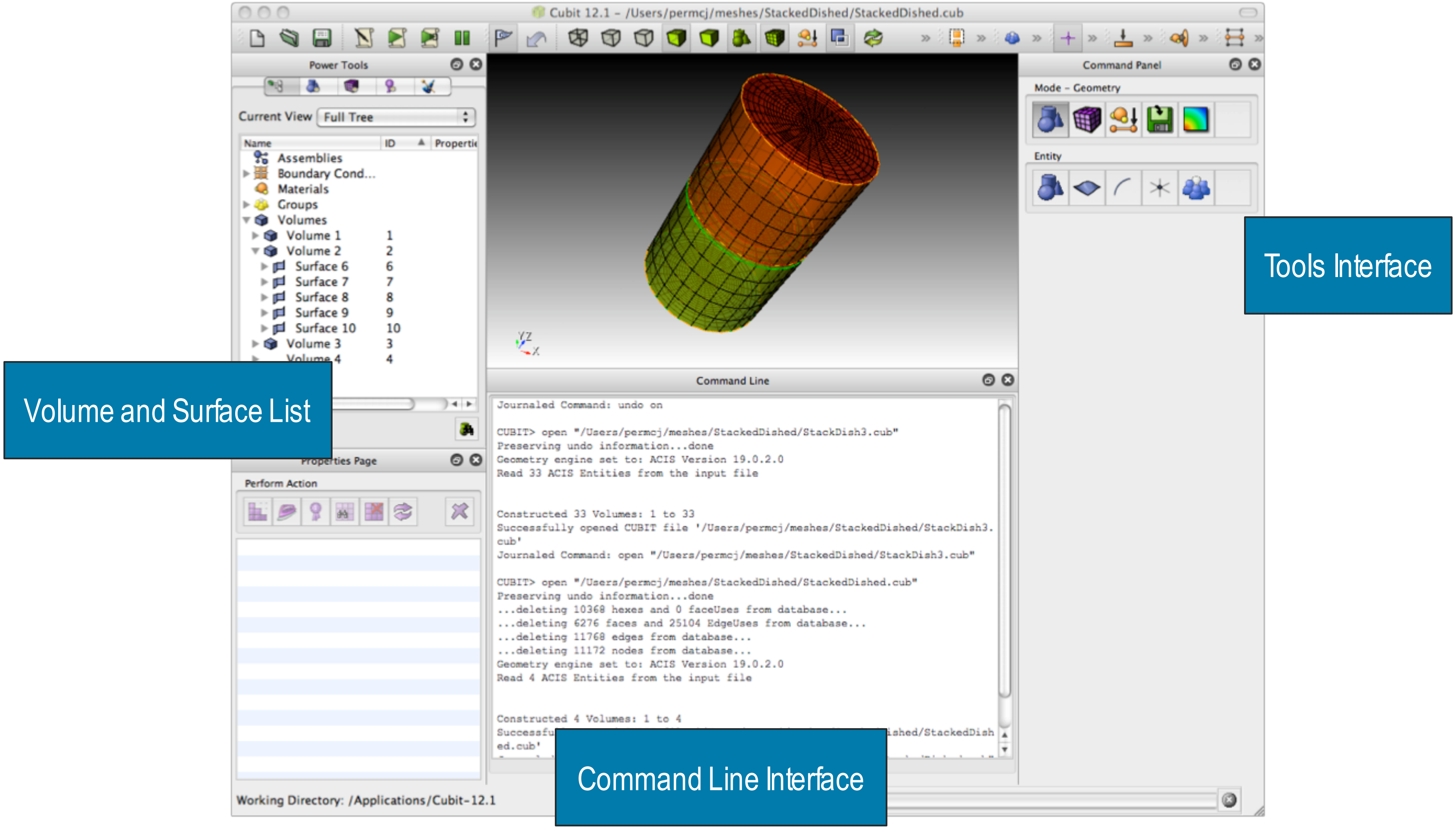
- Orient –Z.
- Change line width.
- Note block names and sideset names.

2D Mesh Generation in BISON

- CircularCrossSectionMesh will create quarter- or half-circle meshes for axisymmetric analysis.
- This tool was built with LWR fuel in mind but can be used for TRISO meshes.
- This tool is a bit more involved. See the documentation pages and tests for more information.



Mesh Generation with CUBIT



Questions?

Clean. **Reliable. Nuclear.**

U.S. DEPARTMENT OF ENERGY | Office of NUCLEAR ENERGY

Clean. Reliable. Nuclear.

TRISO in BISON

Computational Mechanics and Materials
Idaho National Laboratory

TRISO Thermal Models

TRISO Thermal Models

- Thermal modeling for TRISO fuel follows the same pattern as for any other fuel:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + E_f \dot{F}$$

- That is, we need to
  1. Define density, specific heat, and thermal conductivity, which may be functions of temperature or other parameters.
  2. Invoke the heat conduction and heat conduction time derivative kernels.

Thermal Material Properties

- Often, thermal properties are specified as constants.

```
[Materials]
  [./SiC_elasticity_tensor]
    type = ComputeIsotropicElasticityTensor
    block = SiC
    youngs_modulus = 3.7e11
    poissons_ratio = 0.13
  [../]
  [./SiC_elastic_stress]
    type = ComputeFiniteStrainElasticStress
    block = SiC
  [../]
  [./SiC_temp]
    type = HeatConductionMaterial
    block = SiC
    thermal_conductivity = 13.9           # J/m-s-K
    specific_heat = 620.0                 # J/kg-K
  [../]
  [./SiC_den]
    type = Density
    density = 3180.0                      # kg/m^3
    block = SiC
  [../]
[]
```

Thermal Material Properties Continued

- Thermal properties may also be computed in a Material object.

```
[./fuel_thermal]
  type = ThermalFuel
  thermal_conductivity_model = FINK_LUCUTA
  block = fuel
  temp = temp
  burnup = burnup
  initial_porosity = 0.0
[../]
```

Thermal Kernels

```
[Kernels]
  [./heat_ie]
    type = HeatConductionTimeDerivative
    variable = temp
  [../]
  [./heat]
    type = HeatConduction
    variable = temp
  [../]
[]
```

Questions?

Clean. **Reliable. Nuclear.**

U.S. DEPARTMENT OF ENERGY | Office of NUCLEAR ENERGY

Clean. Reliable. Nuclear.

TRISO in BISON

Computational Mechanics and Materials
Idaho National Laboratory

TRISO Mechanical Models

TRISO Mechanical Models

- Mechanical modeling for TRISO fuel follows the same pattern as for any other fuel:

$$\nabla \cdot \mathbf{T} + \rho \mathbf{f} = 0$$

- This is more involved than thermal modeling.
    1. Define constitutive response (define, e.g., an elasticity tensor and an object to convert strain to stress).
    2. Define so-called eigenstrains (thermal strain, irradiation strain)
- The code provides input shortcuts to invoke the computation of strain and the divergence of stress. This shortcut may or may not be used.

Simple Elasticity

```
[Materials]
  [./SiC_elasticity_tensor]
    type = ComputeIsotropicElasticityTensor
    block = SiC
    youngs_modulus = 3.7e11
    poissons_ratio = 0.13
  [../]
  [./SiC_elastic_stress]
    type = ComputeFiniteStrainElasticStress
    block = SiC
  [../]
  [./SiC_temp]
    type = HeatConductionMaterial
    block = SiC
    thermal_conductivity = 13.9          # J/m-s-K
    specific_heat = 620.0                # J/kg-K
  [../]
  [./SiC_den]
    type = Density
    density = 3180.0                     # kg/m^3
    block = SiC
  [../]
[]
```

Creep

```
[Materials]
  [./IPyC_elasticity_tensor]
    type = ComputeIsotropicElasticityTensor
    block = IPyC
    youngs_modulus = 3.96e10
    poissons_ratio = 0.33
  [../]
  [./IPyC_stress]
    type = PyCCreep
    block = IPyC
    flux = fast_neutron_flux
    temperature = temp
    density = 1900.0                     # kg/m^3
  [../]
  [./IPyC_temp]
    type = HeatConductionMaterial
    block = IPyC
    thermal_conductivity = 4.0          # J/m-s-K
    specific_heat = 720.0                # J/kg-K
  [../]
  [./IPyC_den]
    type = Density
    density = 1900.0                     # kg/m^3
    block = IPyC
  [../]
```

Creep + Irradiation Strain

```
[Materials]
  [./IPyC_elasticity_tensor]
    type = ComputeIsotropicElasticityTensor
    block = IPyC
    youngs_modulus = 3.96e10
    poissons_ratio = 0.33
  [../]
  [./IPyC_stress]
    type = PyCCreep
    block = IPyC
    flux = fast_neutron_flux
    temperature = temp
    density = 1900.0                    # kg/m^3
  [../]
  [./IPyC_temp]
    type = HeatConductionMaterial
    block = IPyC
    thermal_conductivity = 4.0         # J/m-s-K
    specific_heat = 720.0               # J/kg-K
  [../]
  [./IPyC_den]
    type = Density
    density = 1900.0                    # kg/m^3
    block = IPyC
  [../]
  [./IPyC_densification]
    type = PyCIrradiationEigenstrain
    block = IPyC
    fluence = fluence
    pyc_type = dense
    eigenstrain_name = IPyC_eigenstrain
  [../]
```

Shortcut for Strain and Divergence of Stress

```
[Modules/TensorMechanics/Master]
  use_displaced_mesh = true
  generate_output = 'stress_xx stress_yy stress_zz stress_xy stress_yz stress_zx hydrostatic_stress'
  strain = FINITE
  incremental = true
  add_variables = false
  [./default]
    block = 'fuel buffer IPyC OPyC'
    eigenstrain_names = 'thermal_strain swelling_strain'
  [../]
  [./SiC]
    block = 'SiC'
    eigenstrain_names = 'thermal_strain'
  [../]
[]
```

Screenshot from
examples/TRISO/accident_simulation/triso2D_accident.i

With No Shortcut for Strain and Divergence of Stress

```
[Kernels]
  [./TensorMechanics]
    use_displaced_mesh = true
  [../]
```

```
[Materials]

  [./fission_gas_release]              # Sifgr fission gas release mode
    type = Sifgrs
    block = fuel
    temp = temp
    fission_rate = fission_rate        # coupling to fission_rate aux variable
    grain_radius_const = 5.0e-6
  [../]

  [./stress]
    type = ComputeLinearElasticStress
    block = 'fuel buffer SiC'
  [../]
  [./strain]
    type = ComputeRSphericalSmallStrain
    block = 'fuel buffer SiC'
    eigenstrain_names = thermal_strain
  [../]
```

TRISO Mechanical Models in BISON

- UCO
  - UCOElasticityTensor
  - UCOVolumetricSwellingEigenstrain
- PyC/Buffer
  - PyCElasticityTensor
  - PyCIrradiationEigenstrain
  - PyCCreep

- If the material requires only linear elasticity and thermal expansion, no specialized models are required.

Questions?

Clean. **Reliable. Nuclear.**

U.S. DEPARTMENT OF ENERGY | Office of NUCLEAR ENERGY

Clean. Reliable. Nuclear.

TRISO in BISON

Computational Mechanics and Materials
Idaho National Laboratory

Fission Gas and Internal Pressure

Fission Gas Production and Release for TRISO Fuel

- BISON provides two models for fission gas production and release:
  1. Sifgrs
     - For use with UO_2.
     - Used in all our LWR cases.
  2. UCOFGR
     - For use with UCO.
     - New model developed in cooperation with Kairos Power.

https://mooseframework.org/bison/source/materials/Sifgrs.html

Too new but will be at
https://mooseframework.org/bison/source/materials/UCOFGR.html
Can build our own…

Sifgrs

```
[Materials]

  [./fission_gas_release]                # Sifgr fission gas release mode
    type = Sifgrs
    block = fuel
    temp = temp
    fission_rate = fission_rate          # coupling to fission_rate aux variable
    grain_radius_const = 5.0e-6
  [../]

  [./stress]
    type = ComputeLinearElasticStress
    block = 'fuel buffer SiC'
  [../]
  [./strain]
    type = ComputeRSphericalSmallStrain
    block = 'fuel buffer SiC'
    eigenstrain_names = thermal_strain
  [../]
```

```
[Postprocessors]
  [./fis_gas_released]                # fission gas released to plenum (moles)
    type = ElementIntegralFisGasReleasedSifgrs
    block = fuel
    execute_on = 'initial linear nonlinear timestep_begin timestep_end'
  [../]
```

UCOFGR

```
[Materials]
  [./burnup]
    type = GenericFunctionMaterial
    prop_names = burnup
    prop_values = burnup
  [../]

  [./thermal]
    type = HeatConductionMaterial
    block = '1'
    thermal_conductivity = 1.0
    specific_heat = 1.0
    temp = temp
  [../]

  [./fission_gas_release]
    type = UCOFGR
    block = '1'
    average_grain_radius = 10e-6
    triso_geometry = particle_geometry
    temperature = temp
    fission_rate = fission_rate
  [../]
```

Can use ElementIntegralMaterialProperty Postprocessor to compute fission gas produced (fis_gas_produced) and fission gas released (fis_gas_released).

Internal Pressure

- BISON uses the ideal gas law to compute internal pressure.
- The PlenumPressure object was built for use with LWRs, but it works just as well for TRISO.
- It is a boundary condition and appears in the BCs area of the input file.
- We must supply:
  - Volume (a Postprocessor value)
  - Gas temperature (a Postprocessor value)
  - Initial pressure (a raw number; used to compute initial moles of gas)
  - Added moles of gas over time (one or more Postprocessor values)

PlenumPressure

```
[./PlenumPressure] #  apply plenum pressure on clad inner walls and pellet surfaces
  [./plenumPressure]
    boundary = BufferGapVol
    initial_pressure = 0
    startup_time = 1.0e4
    R = 8.3143
    output_initial_moles = initial_moles                # coupling to post processor to get initial fill gas mass
    temperature = ave_temp_interior                     # coupling to post processor to get gas temperature approximation
    volume = volumeGas                                  # coupling to post processor to get gas volume
    material_input = 'fis_gas_released co_production'   # coupling to post processor to get fission gas added, co added
    output = plenum_pressure                            # coupling to post processor to output plenum/gap pressure
  [../]
[../]
```

InternalVolume

- InternalVolume is a Postprocessor

```
[./volumeGas]
  type = InternalVolume
  boundary = BufferGapVol
  # ro = 3.125e-4
  # ri = 2.125e-4
  # vb = 4/3*pi*(ro^3-ri^3) = 8.76e-11
  # buffer density = 1000
  # PyC density     = 1900
  # fill ratio = 10/19
  # vb*10/19 = 4.6e-11
  # Must remove 4.6e-11 m^3 from the volume
  addition = -4.6e-11
  execute_on = 'initial linear nonlinear timestep_begin timestep_end'
[../]
```

Gas Temperature

- For the gas temperature, we use SideAverageValue

```
[./ave_temp_interior]
  type = SideAverageValue
  boundary = BufferGapVol
  variable = temp
  execute_on = 'initial linear nonlinear timestep_begin timestep_end'
[../]
```

Added Moles of Gas

- If using Sifgrs:

```
[./fis_gas_released]          # fission gas released to plenum (moles)
  type = ElementIntegralFisGasReleasedSifgrs
  block = fuel
  execute_on = 'initial linear nonlinear timestep_begin timestep_end'
[../]
```

- For CO production with UO_2 fuel:

```
[./co_production]
  type = CarbonMonoxideProduction
  total_fissions = total_fissions
  time_int_surf_temp = time_int_surf_temp
  execute_on = 'initial linear nonlinear timestep_begin timestep_end'
[../]
```

Questions?

Clean. **Reliable. Nuclear.**

TRISO in BISON

Fission Product Diffusion

Computational Mechanics and Materials
Idaho National Laboratory

TRISO Fission Product Diffusion Models

- Fission product diffusion modeling for TRISO fuel follows the same pattern as for any other fuel:

$$\frac{\partial C}{\partial t} = \nabla \cdot D\left(\nabla C - \frac{CQ^*}{FRT^2}\nabla T\right) - \lambda C + S$$

- That is, we need to
  1. Define diffusion coefficient D.
  2. Define Soret coefficients (if used).
  3. Define decay (C) and source (S) terms.
  4. Invoke the diffusion kernels.

Screenshots that follow are from examples/TRISO/accident_simulation/triso2D_accident.i

Define Diffusion Coefficient Material Properties

```
[./fuel_conc]
  type = ArrheniusDiffusionCoef
  block = fuel
  d1 = 5.6e-8                    # m^2/s
  q1 = 209.0e+3                  # J/mol
  d2 = 5.2e-4                    # m^2/s
  q2 = 362.0e+3                  # J/mol
  gas_constant = 8.3143          # J/K-mol
  temp = temp
[../]
```

```
[./SiC_conc]
  type = ArrheniusDiffusionCoef
  block = SiC
  d1 = 5.5e-14                         # m^2/s
  d1_function = d1_function
  d1_function_variable = fluence
  q1 = 125.0e+3                        # J/mol
  d2 = 1.6e-2                          # m^2/s
  q2 = 514.0e+3                        # J/mol
  gas_constant = 8.3143                # J/K-mol
  temp = temp
[../]
```

Be sure to define coefficient for each material.

Invoke Decay, Source, and Diffusion Kernels

```
[Kernels]
  [./heat_ie]
    type = HeatConductionTimeDerivative
    variable = temp
  [../]
  [./heat]
    type = HeatConduction
    variable = temp
  [../]
  [./heat_source]
     type = NeutronHeatSource
     variable = temp
     block = fuel
     energy_per_fission = 3.2e-11         # units of J/fission
     fission_rate = fission_rate
  [../]
  [./mass_ie]
    type = TimeDerivative
    variable = conc
  [../]
  [./mass]
    type = ArrheniusDiffusion
    variable = conc
  [../]
  [./mass_source]
    type = BodyForce
    variable = conc
    function = power_history
    value = 1.22e-5      # units of moles/m**3-s
    block = fuel
  [../]
  [./mass_decay]
    type = Decay
    variable = conc
    radioactive_decay_constant = 7.297e-10   # units:(1/sec)  The constant for Cesium
  [../]
[]
```

Questions?

Clean. **Reliable. Nuclear.**