

Bison Documentation Update, NEAMS Fuels Product Line Review Meeting

Stephanie A Pitts

September 2018



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

Bison Documentation Update, NEAMS Fuels Product Line Review Meeting

Stephanie A Pitts

September 2018

**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

Nuclear Energy

Bison Documentation Update

Stephanie Pitts

NEAMS Fuels Product Line Review Meeting
Idaho Falls, ID
September 19, 2018

Bison Documentation Objectives

- **Flexible, unified, and comprehensive documentation system developed in parallel with source code**
 - Generated from source code and theory files written in Markdown
 - Built with the MooseDocs system
 - Merge requests are required to include documentation files
- **Recent development **Focus on New Users****
 - Step-by-step setup instructions and tutorials
- **Theory and user **Unified Documentation** implementation**
 - Detailed unified information for each class and action
- **Rapid documentation access for **Experienced Users****
 - Class index pages with integration of relevant MOOSE documentation



Bison Documentation Objectives

■ Flexible, unified, and comprehensive documentation system

Bison

Getting Started ▾ Examples & Tutorials ▾ Theory ▾ Documentation ▾ Help ▾

Bison

A Finite Element-Based Nuclear Fuel Performance Code

Bison is a finite element-based nuclear fuel performance code applicable to a variety of fuel forms including light water reactor fuel rods, TRISO fuel, and metallic rod and plate fuel. It is a multiphysics fuel analysis tool that solves fully-coupled thermomechanical problems.

↓

Getting Started

Bison models complex and multiphysics problems integral to Nuclear Fuel Analysis in many different geometries and dimensions, from simplified and fast 1D simulations to complete and specific 3D analyses.

Learn how to run Bison on your own machine with our [Git Instructions](#) and [Example Problem Tutorials](#).

⚙️

Code Documentation

Bison fuel models are developed to describe temperature and burnup dependent thermal properties, fission product swelling, densification, thermal and irradiation creep, fracture, and fission gas production and release.

Explore the overview in the [Theory Manual](#) and read about the individual code elements in the detailed [Code Documentation](#).

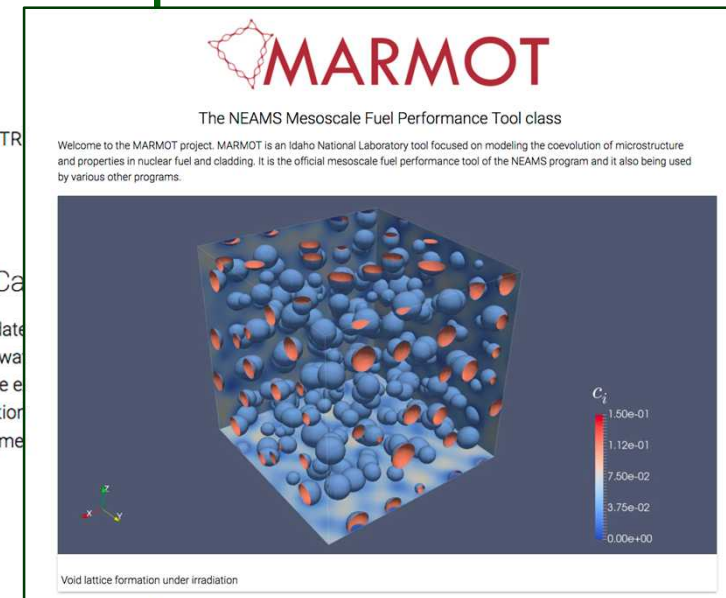
📄

Assessment Case Studies

Bison is currently being validated against a wide variety of integral light water reactor fuel rod experiments. See the descriptions and the simulation results in our over 70 different assessment cases in our [Assessment Report](#).

Bison is built on MOOSE

Bison is based on [MOOSE](#) and can efficiently solve problems using standard workstations or very large high-performance computers in a variety of different dimensions, including full 3D, 2D-RZ axisymmetric, layered axisymmetric 1D, and spherically symmetric 1D systems.





U.S. DEPARTMENT OF
ENERGY

Nuclear Energy

Bison Documentation Website

<https://hpcsc.inl.gov/ssl/BISON/site/> (Requires a Bison license,
Built daily from Bison's master branch)

<https://bison.inl.gov/Documentation/index.aspx> (External)

With a bison-opt executable, build the documentation locally

```
cd ~/projects/bison/docs  
./moosedocs.py build --serve
```

New User Focus: Git and Bison Use Instructions

Bison

🔍


Getting Started ▾

Examples & Tutorials ▾

Theory ▾

Documentation ▾

Help ▾



getting_started > [gitting_bison](#)

'Git'-ting Bison: Working with Git

These instructions are meant to help users of both local machines and HPC get started with Bison. The installation and build processes are the same for both with the exception of the HPC build, which is prebuilt on HPC.

NOTE

These instructions are designed for new users that have never forked a repository. If you already have a Bison fork and need to clone to a new machine, skip to the section on [Cloning Bison](#).

Getting Git Setup

Cloning Bison

Building Bison

Running Bison

Bison Input File

Mesh Script

Contributing

Requesting Access

Install the MOOSE Pack...

Adding your SSH keys

Git Overview

Fork Bison

Load Modules

Bison and MOOSE use git for code management and distribution. For that reason you will need to learn a little bit about git even if you are not planning on contributing to the code. You will also need access to the Bison repository and a license agreement. Instructions for this are below.

Requesting Access

Contact a member of the Bison team to get HPC access and a licensing agreement for your institution. This process may take a few weeks. Bison is housed at <https://hpcgitlab.inl.gov> which lives on the INL HPC system, therefore HPC access is required. Continue on to the next step once HPC access is obtained.



New User Focus: Examples and Detailed Tutorials

■ Overview of different functionalities available in Bison

■ Use existing example input files

- Represent common simulation options

■ Explain interaction among common input file blocks

- Highlight parameter settings

Examples and Tutorials

One of the best ways to become familiar with Bison is to start running simulations. To help new users, and returning users looking to brush-up their familiarity, explore Bison, the Bison team provides several examples and tutorials for the different facets of Bison capabilities in [projects/bison/examples](#).

Typical LWR Analysis Geometries

The geometry and dimension used in a fuel analysis problem represents an optimization between simulation speed and localized detail. Presented below are the three most common geometries used in BISON for LWR fuel rod performance analysis; click on the heading to go to the geometry specific tutorial pages.



Layered 1D

Provides a quick simulation of a simplified fuel analysis model. Layered 1D models are ideal for simulations in which runtime speed is a prime consideration and localized effects can be neglected.



2D Axisymmetric

The yeoman of Bison fuel analysis simulations. 2D Axisymmetric (2D-RZ) simulations are best suited for modeling the axial and radial variations common in fuel while retaining the computational efficiency of a 2D problem.



Complete 3D

Enables the study of localized effects on the performance of a fuel rod. Full 3D models are most beneficial for simulations of non-symmetric fuel rods, including manufacturing defects on a single fuel pellet surface or localized loading conditions.

The most suitable geometry to use in a fuel rod analysis depends on the details of the specific analysis requirements.

Additional Tutorials

Beyond the light water reactor examples given above, Bison includes tutorials and guidelines for advanced topics, including changing geometry, different fuel rod types, and updating input files to the latest mechanics capabilities. Click on the headers below to go to the tutorial sections.



Switching Dimensions

Bison's ability easily upscale and downscale among simulation dimensions



Advanced Fuel Examples

Bison includes the capability to model many other fuels beyond traditional LWR



Tensor Mechanics Migration

The guidelines presented in this section will assist in the process of converting



New User Focus: Examples and Detailed Tutorials

Introduction to Layered 1D Models

The term 'Layered1D' in Bison describes a model of cylindrical fuel geometry that is a collection of coupled 1D simulations used to represent the behavior of the fuel at given axial positions along the rod. The fuel rod is divided into several axial slices, and a 1D model of the physics, Eq. 1 and Eq. 2, is solved on each slice. An illustration of these 1D axial slices is shown in Figure 1.

The following equations for energy conservation and stress divergence, respectively, are solved simultaneously using 1D axisymmetric models for each axial slice:

$$\rho C_p \frac{\partial T}{\partial t} - \nabla \cdot k \nabla T - q = 0 \quad (1)$$

$$\nabla \cdot \sigma = 0 \quad (2)$$

where ρ is the mass density, C_p is the specific heat, T is the temperature, k is the thermal conductivity, q is the volumetric heat generation rate, and σ is the stress.

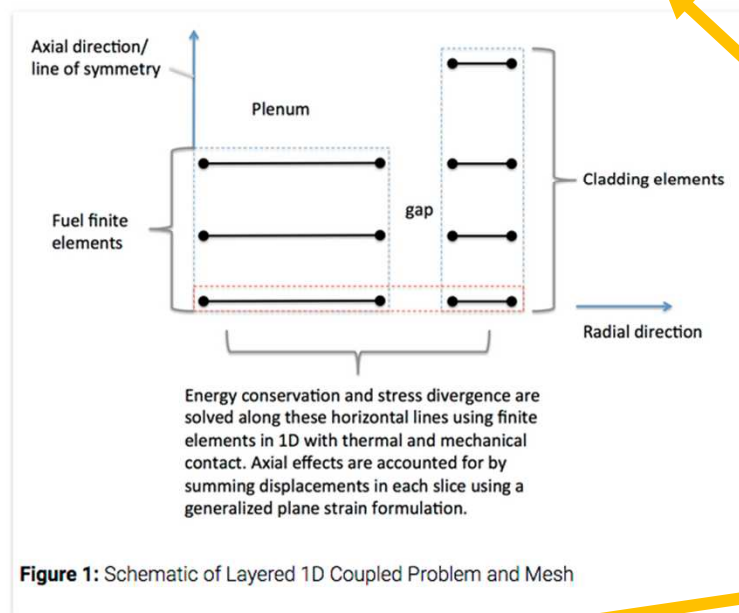
Each slice represents the behavior of a finite axial segment of the rod. The behavior is assumed to be uniform over the entirety each axial segment.

Layered 1D Specific Code

Because a given 1D model is only applicable for a finite length of the fuel rod, and generally the behavior of a rod varies significantly over its length, the ability to represent axial variations is necessary. This ability is added to Bison with a set of code classes that compute integral quantities specifically for Layered 1D models, including code used to calculate:

- Plenum volume ([InternalVolumeLayered1D](#))
- Fission gas released ([SideAverageValueLayered1D](#) computes the average temperature used to determine fission gas released)
- Heat flux ([SideFluxIntegralLayered1D](#))
- Burnup profiles ([Layered1DFuelPinGeometry](#) provides geometry information about the fuel rod geometry)
- Axial pressure contributions from the coolant and plenum pressure (out_of_plane_pressure parameter in [Layered1DMaster](#))

Introduction to Layere...
Layered 1 D Specific Co...
Coupling with Axial Out ...
Comparison with 2 D - ...
Best Use Cases for Lay...
References



Unified Documentation: Example Class Page

Internal Volume Layered1D

Computes the volume of an enclosed area by performing an integral over a user-supplied boundary.

Description

InternalVolumeLayered1D calculates the internal volume for the case of Layered1D. For the integration along the layer stacking direction, it uses the constant slice_height for each layer fetched from Layered1DFuelPinGeometry object.

$$V = \sum_i h_i \cdot A_i \quad (1)$$

where V is the total integrated volume, i is the layer index, h_i is the height of layer i , and A_i is the in-plane area of layer i .

Example Input Syntax

```
[./volume]
type = InternalVolumeLayered1D
fuel_pin_geometry = pin_geometry
boundary = 9
execute_on = 'initial timestep_end'
[../]
```

([test/tests/layered_1D/internal_volume.i](#))

Input Parameters

▼ Required Parameters

out_of_plane_strain The out-of-plane strain nodal variable

boundary The list of boundary IDs from the mesh where this boundary condition applies

fuel_pin_geometry Name of Layered1DFuelPinGeometry UserObject

- **Single complete documentation page:**
 - Theoretical information
 - Input file example syntax
 - Available input parameters

- **Parameters, descriptions, and defaults taken directly from source code**

- **Example input syntax block linked from regression tests**

Unified Documentation: Example Action Page

■ Connects action parameters to partial differential equation terms

Burnup Action System

Computes the radial distributions of power density, burnup, and concentrations of various heavy metal isotopes in UO2 and U3Si2 fuels for LWRs

Constructed Objects

Description
Constructed Objects
Example Input Syntax
Input Parameters
Associated Actions

Table 1: Correspondence Among Action Functionality and Moose/Bison Objects for the Burnup Action

Functionality	Replaced Classes	Associated Parameters
Calculate radial distributions of power density, burnup, and heavy metal concentrations	BurnupFunction	<p>fuel_inner_radius and fuel_outer_radius: values of the fuel radial dimensions</p> <p>a_lower and a_upper: values of the fuel axial dimensions</p> <p>num_radial: integer of the number of burnup-specific grid divisions in the radial direction</p> <p>num_axial : integer of the number of burnup-specific grid divisions in the axial direction</p> <p>rod_ave_lin_pow: a string containing the name of the average rod linear power function</p>
Retrieve and store the radial profiles of burnup and fission rate	BurnupGrid and AuxVariables	<p>burnup: string containing the name of the burnup auxvariable (created by default)</p> <p>fission_rate: string of the name of the fission rate auxvariable (created by default)</p> <p>average_burnup: string containing the name of the average burnup auxvariable</p>
Retrieve and store the radial profiles of heavy metal concentration (optional)	BurnupGrid and AuxVariables	<p>N235, N236, N238, N240, N241, N242: string containing the name of the heavy metal ion of interest</p>

Experienced User: Reference Index Pages

Bison Only Code Reference Manual

Given below are the reference pages for all of the Bison specific code. To see a complete list of all of the possible input parameter options and the associated reference pages for Bison and the MOOSE Modules code, view the [Complete Code Manual](#).

Click the blue links in the class names shown below to view the detailed description the class purpose, theoretical models, input file examples, and references.

AuxKernels

[Al2O3Aux](#) Calculates the Aluminum oxide thick

[BurnupAux](#) Compute burnup given fission_rate

[BurnupGrid](#) Retrieve burnup, fission_rate, conc

[BurnupMetalAux](#)

[CoolantAux](#)

[CumulativeDamageIndex](#) Calculates the cum

[EffectiveBurnupAux](#) Compute effective burnu

[EutecticThicknessFCCI](#) Calculates eutectic pe

[FastNeutronFluenceAux](#) Compute fast neutro

[FastNeutronFluxAux](#) Compute fast neutron fi

[FeCrAlOxideAux](#) Calculates the oxidation and corrosion of FeCrAl cladding

[FissionRateAux](#) Give a scaled value of a function that can be used to set fission rate. Similar to Fu...

[AuxKernels](#)

[AuxVariables](#)

[BCs](#)

[Burnup](#)

[CladdingHydrides](#)

[CoolantChannel](#)

[DefaultElementQuality](#)

[Functions](#)

[Bison](#)



[Getting Started](#)

[Examples & Tutorials](#)

[Theory](#)

[Documentation](#)

[Help](#)



[syntax](#)

[Bison Specific Manual](#)

[Complete Code Manual](#)

Complete Bison Input File Syntax

Listed below are all the reference pages for both Bison specific code and the MOOSE Modules code which can be used in the Bison input file. To see a list of only Bison specific code, view the [Bison Specific Code Manual](#).

Click the blue links in the class names shown below to view the detailed description the class purpose, theoretical models, input file examples, and references.

Click the blue icon link shown to the right of each main class type heading (e.g. Adaptivity) to see a more detailed description of the class type purpose within the MOOSE framework.

[Adaptivity](#)

[AuxKernels](#)

[AuxScalarKernels](#)

[AuxVariables](#)

[BCs](#)

[Burnup](#)

[CladdingHydrides](#)

[Constraints](#)

[Contact](#)

[Controls](#)

Experienced User: Directly Embedded MOOSE Module Pages

- **Access information about each possible input file block**
- **Remain within the single unified documentation website**
 - Eliminates informal requirement to know which classes are Bison and which are MOOSE classes

Compute Axisymmetric RZ Finite Strain

Compute a strain increment for finite strains under axisymmetric assumptions.

Description

The material `ComputeAxisymmetricRZFiniteStrain` calculates the finite strain for 2D Axisymmetric systems.

Axisymmetric (cylindrical) materials are included in Tensor Mechanics for revolved geometries and assume symmetrical loading. These 'strain calculator' materials compute the strain within the appropriate coordinate system and rely on specialized AxisymmetricRZ kernels to handle the stress divergence.

WARNING: Symmetry Assumed About the z -axis

The axis of symmetry must lie along the z -axis in a (r, z, θ) or cylindrical coordinate system. This symmetry orientation is required for the calculation of the residual and of the jacobian. See [StressDivergenceRZTensors](#) for the residual equation and the germane discussion.

The AxisymmetricRZ material is appropriate for a 2D simulation and assumes symmetry revolved about the z -axis. A 2D formulation of an appropriate simulation problem can reduce the simulation run time while preserving key physics. Axisymmetric simulations are appropriate to problems in which a solid is generated by [revolving a planar area about an axis](#) in the same plane.

NOTE: Use RZ Coordinate Type

The coordinate type in the Problem block of the input file must be set to `COORD_TYPE = RZ`.

Axisymmetric Strain Formulation

The axisymmetric model employs the cylindrical coordinates, r , z , and θ , where the planar cross section formed by the r and z axes is rotated about the axial z axis, along the length of the cylinder, in the θ direction. The cylindrical coordinate system strain tensor for axisymmetric problems has the form

$$\begin{bmatrix} \epsilon_{rr} & \epsilon_{rz} & 0 \\ \epsilon_{rz} & \epsilon_{zz} & 0 \\ 0 & 0 & \epsilon_{\theta\theta} \end{bmatrix}$$

(1)

Experienced User: New Classes Require New Documentation

- **Enforce connection between documentation and source code by requiring markdown files for every class in Bison**
 - Merge requests will fail Civet testing without corresponding markdown files

```
[~/projects/bison/doc](devel)> ./moosedocs.py build --check
MooseDocs.extensions.appsyntax: Reading MOOSE application syntax.
MooseDocs.extensions.appsyntax: Building MOOSE class database.
MooseDocs.build: Cleaning destination /Users/pittsa/.local/share/moose/site
MooseDocs.commands.check: No documentation for /Materials/ComputeCrystalPlasticityStress.
  - The page should be located at /Users/pittsa/projects/moose/modules/tensor_mechanics/doc/content/source/
materials/ComputeCrystalPlasticityStress.md.
  - It is possible to generate stub pages using './moosedocs.py check --generate'.
```

- **Markdown files should contain theory geared toward the user**
 - Input parameters table and example usage blocks are automatically generated by MooseDocs
 - General documentation standards are available:
<http://www.mooseframework.org/moose/utilities/MooseDocs/generate.html>

Future Development Goals

- **To-date documentation development efforts have focused on light water reactor mechanics materials**
- **Complete tutorial sections with a focus on advanced fuel**
 - Including Metal fuel, TRISO fuel, and ATF
- **Integrate training workshop materials in an interactive manner**
- **Migrate assessment case information to documentation website**



U.S. DEPARTMENT OF
ENERGY

Nuclear Energy

Bison Documentation Website

<https://hpcsc.inl.gov/ssl/BISON/site/> (Requires a Bison license,
Built daily from Bison's master branch)

<https://bison.inl.gov/Documentation/index.aspx> (External)

With a bison-opt executable, build the documentation locally

```
cd ~/projects/bison/docs  
./moosedocs.py build --serve
```