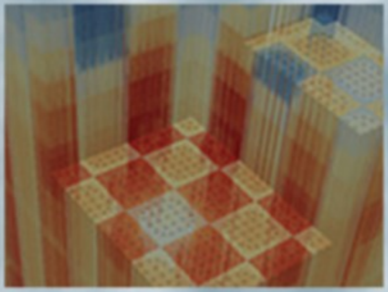
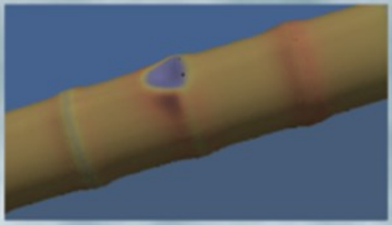
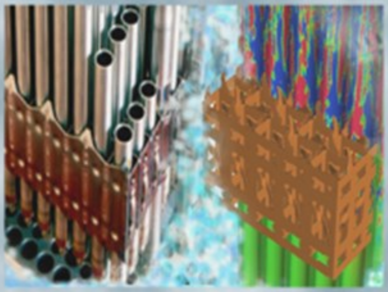
CASL-U-2016-1099-001

A DOE Energy Innovation Hub

# File-Based One-Way BISON Coupling Through VERA: User's Manual

Shane Stimpson
*Oak Ridge National Laboratory*

**February 28, 2017**

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

## REVISION LOG

| Revision | Date | Affected Pages | Revision Description |
|---|---|---|---|
| 0 | 06/03/2016 | All | Original Report |
| 1 | 02/28/2017 | All | Updated to reflect new capabilities |
| | | | |
| | | | |

**Document pages that are:**

Export Controlled ____________________NO

IP/Proprietary/NDA Controlled _______NO

Sensitive Controlled ________________NO

Approved for Public Release _________YES

# CONTENTS

## ACRONYMS

| | |
|---|---|
| ASCII | American Standard Code for Information Interchange |
| CASL | Consortium for Advanced Simulation of Light Water Reactors |
| CS | core simulator |
| CSV | comma-separated value |
| CTF | COBRA-TF |
| CZP | cold zero power |
| EFPD | effective full power day |
| HDF | hierarchical data format |
| HFP | hot full power |
| HZP | hot zero power |
| IFBA | integral fuel burnable absorber |
| INL | Idaho National Laboratory |
| JFNK | Jacobian-free Newton-Krylov |
| MC | Monte Carlo |
| MOOSE | Multiphysics Object Oriented Simulation Environment |
| MPI | message passing interface |
| MW | megawatt |
| ORNL | Oak Ridge National Laboratory |
| SNA | Supercomputing in Nuclear Applications |
| VERA | Virtual Environment for Reactor Applications |
| WBN1 | Watts Bar Nuclear Unit 1 |
| XML | extensible markup language |

# 1. INTRODUCTION

Activities to incorporate fuel performance capabilities into the Virtual Environment for Reactor Applications (VERA) are receiving increasing attention [1–6]. The multiphysics emphasis is expanding as the neutronics (MPACT) and thermal-hydraulics (CTF) packages are becoming more mature. Capturing the finer details of fuel phenomena (swelling, densification, relocation, gap closure, etc.) is the natural next step in the VERA development process since these phenomena are currently not directly taken into account. While several codes could be used to accomplish this, the BISON fuel performance code [8,9] being developed by the Idaho National Laboratory (INL) is the focus of ongoing work in the Consortium for Advanced Simulation of Light Water Reactors (CASL). Built on INL's MOOSE framework [10], BISON uses the finite element method for geometric representation and a Jacobian-free Newton-Krylov (JFNK) scheme to solve systems of partial differential equations for various fuel characteristic relationships. There are several modes of operation in BISON, but, this work uses a 2D azimuthally symmetric (R-Z) smeared-pellet model.

This manual is intended to cover (1) the procedure pertaining to the standalone BISON one-way coupling from VERA and (2) the procedure to generate BISON fuel temperature tables that VERA can use. Section 2 covers the process to generate BISON inputs based on input/output information from VERA, starting from the ASCII VERA input [11] through generation of a separate BISON input file for each rod in the problem. Section 3 covers the details pertaining to the fuel temperature table generation. Section 4 provides some brief background and guidelines on running the inputs. And Section 5 covers the current post-processing capabilities which heavily leverage the VERAView visualization tool [12].

# 2. GENERATING STANDALONE BISON INPUTS

There are four steps to generate the standalone BISON inputs:

1) Create a VERA ASCII input
2) Convert the ASCII input to extensible markup language (XML) (and XML to CTF)
3) Execute VERA to generate hierarchical data format (HDF5) output files
4) Run XML2MOOSE to produce BISON inputs

## 2.1 Creating a VERA ASCII Input

### 2.1.1 General Overview

The VERA ASCII input is intended to be set up quickly and easily, even for complex reactor designs. A detailed overview of the input specification will not be covered here, but it can be found in the VERAIn Common Input Manual [11]. However, enough background will be presented here to provide a general idea of the key components in the VERA ASCII input.

The top block(s) in each file are the STATE blocks, which specify a number of state-specific parameters such as power (%), soluble boron concentration (ppm), coolant inlet temperature, system pressure, and depletion time steps. Very simple histories can be specified using only one STATE block, but several blocks are typically used to capture various changes throughout the cycle. For example, in simulating Watts Bar Nuclear Unit 1 (WBN1), 30–40 statepoints is the typical number per cycle [13]. A new input file is required per cycle, which will become more apparent in the next section when the shuffling specification is covered.

Next, the CORE block outlines the general core shape, the layout of assemblies (using the ASSEMBLY indexes specified in subsequent ASSEMBLY blocks), the layout of burnable poison configurations (using indexes from the INSERT block), and the control rod bank layout (using indexes from the CONTROL block). Also included here are the rated core power and flow, along with the specifications for upper and lower core plates and reflector regions. Lastly, various materials definitions can be included here to be used in the ASSEMBLY blocks.

The ASSEMBLY blocks define the pin cell and lattice geometries used to construct axial stacks of lattices comprising each assembly. The pin cell specifications include all radii (fuel pellet, inner clad, outer clad, etc.) and material layouts. Lattice specifications include the 2D radial layouts of the pin cells in each lattice. These are stacked together with axial bounds for each region to construct the 3D assembly. Also included in this block are the spacer grid specifications in which the type (such as Inconel or Zircaloy) and midpoint of each grid are stated.

Blocks are also used to outline code-specific information (MPACT, COBRATF, and BISON). Later in this manual, some guidance will be provided on what can be included in the BISON block. Details on what the MPACT and COBRATF blocks can contain are defined in the VERAIn Manual [6].

### 2.1.2 Shuffling Specification

Since the shuffling specification through VERA is a significant part of multicycle simulation, it is outlined here. As mentioned above, a separate input file is necessary for each cycle depletion being executed. Furthermore, separate files are necessary for each shuffling procedure between cycles. Restart files are used to write the isotopics at the end of each cycle; these are then read in and shuffled based on the shuffle_label map specified in the STATE block of the shuffle input file. Once the shuffle has been completed, a new restart file is written and used as input for the next cycle. Figure 1 shows a representative shuffle_label specification in which fuel from the previous cycle is specified using the X/Y label <A-R>-<1-15> (corresponding to its location in the previous cycle) and "+" indicates the location of fresh fuel. Additionally, any label with an integer prefix indicates that the assembly was last present in a cycle other than the previous one. For example, 3B-10 indicates the assembly was from the B-10 location in cycle 3. Note that this is not intended to be a real shuffle layout, merely an example.

```
shuffle_label           3B-10 4G-2  4H-1  4A-11 4H-13 4M-2  3K-14
            L-1   P-6   +     +     +     +     +     +     B-12  F-3
      K-2   +     +     +     F-14  +     N-4   +     J-4   +     +     +     P-10
      R-7   +     R-9   L-10  +     A-5   H-14  J-1   +     F-2   N-14  +     G-15
3N-6  +     +     B-7   +     R-11  +     H-3   +     D-3   +     K-1   +     +    3J-6
4A-6  +     C-8   +     C-14  C-3   R-5   +     E-1   P-7   F-13  +     P-13  +    4A-8
4C-2  +     +     D-13  +     P-12  +     F-7   +     C-12  +     P-4   +     +    4G-12
4P-11 +     K-9   M-13  B-4   +     M-7   E-15  N-13  +     B-5   G-10  G-6   +    N-8
4A-7  +     +     L-14  +     K-7   +     R-10  +     M-3   +     G-1   +     +    4D-7
4N-10 +     K-3   +     B-13  E-14  H-15  +     M-14  B-11  N-3   +     L-15  +    4J-2
3C-10 +     +     D-2   +     J-15  +     E-2   +     F-11  +     C-6   +     +    3J-12
      P-9   +     M-9   G-4   +     F-15  C-13  A-9   +     R-8   B-9   +     D-9
      F-9   +     +     +     L-2   +     E-6   +     P-3   +     +     +     N-2
            N-12  P-5   +     +     +     +     +     +     +     C-4   J-14
                        3K-13 4B-3  4J-10 4G-14 4B-6  4D-14 3K-5
```

**Figure 1. Representative Shuffle Map**

Additionally, an "op_date" indicating the date operation began/ended for the cycle is specified. These dates are used to determine the shutdown time during the shuffling procedure, and isotopes are decayed accordingly. The cycle start date is specified in the CORE block, and the end date is specified in the last STATE block.

For a basic example of how the shuffling procedure works, please consult the "bison_from_vera_multi_qtr_shuffle_mir" and "bison_from_vera_multi_qtr_shuffle_rot" regression tests available in MOOSEExt/test, with mirror and rotational symmetry, respectively. These tests demonstrate the shuffling capability for three cycles using a small test core. Further details of these tests are provided in Sect. 2.5.

### 2.1.3 The BISON Block

The BISON block in the VERA input is where many of the values from the template can be updated, but it also includes several other important flags. Below is a sample BISON block:

```
[BISON]
  solve_type standalone
  fuel_pin_input_file_template = ../bison.template
  power_file = c1.h5 c2.h5 c3.h5
  cycle_xml = c1.xml c2.xml c3.xml
  shuffle_xml = c2_shuffle.xml c3_shuffle.xml
  only_cycle = 3
  bc_type bulk_cool
  axial_shape constant
  fast_flux true
  mesh_clad_bot_gap_height = 1.52e-3
  executioner_start_time = 0.0
  executioner_dtmin = 0.1
  bcs_plenumpressure_plenumpressure_initial_pressure = 1.999480e+06
```

To cover some of these important flags, the discussion begins with `solve_type`. Valid entries for this include `STANDALONE`, `TEMP_TABLE`, `TIAMAT`, and `TIAMAT_INLINE`. This effectively directs `XML2MOOSE` to the mode for which inputs should be parsed. Not surprisingly, the mode for this manual is `STANDALONE`, which will populate the template file specified in `fuel_pin_input_file_template`. The standalone mode does not simulate guide tubes or burnable poison inserts. Tiamat, however, uses a different template when simulating guide tubes and a non-fuel template needs to be provided.

The next three flags are extremely important. `power_file` lists the HDF5 output files produced from running the cycle depletions (as will be generated in the process outlined in Sect. 2.3). Both relative and absolute paths can be used. `cycle_xml` lists the XML files for each cycle, and shuffle_xml contains the XML files used to execute the shuffling (from which the shuffle maps are pulled). Generation of XML files is covered in Sect. 2.2. The files for all cycles should always be entered consecutively, without skipping any. If results are desired for Cycle 7, enter all files for Cycles 1–7. XML2MOOSE will use the shuffling maps to determine the appropriate data to pull from each file.

Next is the `only_cycle` flag, in which only the BISON inputs for the specified cycle are generated. However, the history of the rods from previous cycles will still be constructed as appropriate. If this flag is not included, XML2MOOSE will generate files for all cycles. At present, the `only_cycle` flag is highly recommended mode as it makes it easier to verify the number of inputs generated. It also makes the number of files easier to manage. In simulating WBN1, for example, there are roughly 15,000 input files generated per cycle. If several cycles are generated at once, the number of files created can be overwhelming to manage.

The next few flags that will be covered relate the some of the finer details of how BISON will solve each rod. `bc_type` controls the thermal boundary condition of the problem. Valid entries for this are currently `BULK_COOL` and `CLAD_OUTER`. As might be expected, `BULK_COOL` uses the coolant temperature from CTF as the boundary condition, allowing BISON to handle the convection from the clad to the coolant, and `CLAD_OUTER` uses the clad outer surface temperature reported by CTF.

`axial_shape` relates to the shaping that the power and temperature distributions from VERA use in BISON. Valid entries for this flag are `CONSTANT` and `MIDPOINT`. `MIDPOINT` is reflective of the initial capabilities offered, where the power/temperature from a particular axial mesh in VERA is assigned to the midpoint of the axial node, and BISON will linearly interpolate between the midpoints as appropriate. However, to more accurately represent the data reported from VERA, `CONSTANT` should be used, which constructs a staircase distribution of the data such that all points contained within the axial mesh received the same power/temperature, and no linear interpolation is performed.

Finally, the `fast_flux` flag is a Boolean than denotes whether or not clad fast flux data taken from MPACT will be used in BISON. If `FALSE`, the default fast flux factor in the BISON template will be used, which effectively applies a fast flux based on the local power distribution. This can inaccurately estimate the fast flux seen by the clad as the fast flux does not always correlate with the power distribution. If specified as `TRUE`, XML2MOOSE will read the fast flux data from the HDF5 and populate the template accordingly. It is very important to note that this last capability cannot be used unless the fast flux was edited from VERA. This can be done by specifying `edit pin_clad_fast_flux` in the first STATE block of the VERA ASCII input.

All other flags can be traced to locations in the BISON template file (see Sect. 2.4.1).

## 2.2 Converting the ASCII File to XML

Once the VERA ASCII input file has been created, it must be converted to XML using the react2xml script:

```
react2xml.pl vera.inp vera.xml
```

Codes such as MPACT can read the XML file directly, while codes like CTF and BISON require an additional preprocessor step to convert the XML file to a format they can read. The preprocessor for BISON (XML2MOOSE) will be covered in Sect. 2.4. While the preprocessor for CTF (XML2CTF) will not be covered in detail, it should be kept in mind when preparing to run VERA, and it can be run similarly:

```
xml2ctf --xmlfile=vera.xml
```

Because separate files are necessary for each cycle depletion and shuffle, this process must be repeated for each file.

## 2.3 Running VERA

Once all of the appropriate input files are preprocessed, they can be executed with MPACT/CTF to simulate the reactor through the cycle depletion. At the end of each cycle, separate executions are necessary to complete the shuffle procedure and move to the next cycle. Output from these simulations will be HDF5 files that contain the pin power and moderator temperature distributions for each rod, usually with 40–60 axial planes of resolution. These data are used by XML2MOOSE to construct each BISON input file.

It is worth noting that XML2MOOSE requires each HDF5 file to be of a complete cycle. Many cycle depletion cases require the use of at least one restart file to complete the simulation. The HDF5 files produced from the restarted cases must be appended together. This can be easily accomplished using HDF5 commands [14]. One caveat to this is that VERA reports power for each state on the HDF5 file, but the first statepoint reports the raw power in megawatts (MW), whereas subsequent statepoints report percent power. When appending HDF5 files together, it is important to ensure that the final appended file has only raw power for the first statepoint, as if it ran to completion without a restart. This requires changing the value from MW to % as appropriate.

## 2.4 XML2MOOSE

### 2.4.1 The BISON Template

The BISON template file is a crucial part of this process, and it is very important to understand what contained in the template and how it can be changed from the VERA input. The template can be found in the MOOSEExt/test directory as `bison.template`, and it contains several different categories of data:

1) VERA_DEFINED
2) VERA_MODIFIABLE
3) VERA_PREPARED
4) TIAMAT
5) BISON

VERA_DEFINED data are specified exclusively by XML2MOOSE and are not adjustable through the BISON block. These would include things such as geometry specification, fuel density/porosity, and other system characteristics.

VERA_MODIFIABLE inputs are data that can be manipulated in the BISON block. If not specified there, they either have a default value or they are omitted entirely. Each modifiable input has an additional flag indicating the variable in the BISON block that can be used to change the value. For example, `mesh_nx_p` (also show below as in the template) can be used to change the number of radial elements in the fuel.

```
#VERA_MODIFIABLE as mesh_nx_p  (omitted if file is specified)
```

One can set the value of `mesh_nx_p` in the BISON block, and it will be populated into the template. The an additional comment—"(omitted if file is specified)"— indicates that this flag is deleted from the input when an external mesh file is specified, as it completely controls the mesh specification.

VERA_PREPARED items are a combination of the VERA_DEFINED and VERA_MODIFIABLE. If no flag is entered from the BISON block, VERA will automatically populate it with a value determined from the input, but VERA will not use a default value. For example, the simulation end time is determined by the number of states and depletion specification in the input, but this can be overridden to truncate the simulation. This is not necessary for most modes of operation, but it useful for testing.

Data flagged as TIAMAT are only used when XML2MOOSE is generating inputs for TIAMAT; otherwise they are deleted from the template.

BISON inputs are only used for standalone BISON; otherwise they are omitted. Some flags will behave differently if they are being used for standalone BISON or TIAMAT. This behavior is documented in the template.

### 2.4.2 Execution

Most of the work is performed in the XML2MOOSE preprocessor step, particularly creation of the BISON input files. Using the same XML file that was used to perform the VERA neutronics/thermal-hydraulics simulations with an additional BISON block, XML2MOOSE can be run as in the following command, where the XML file created is vera.xml:

```
xml2moose –c vera
```

This will create separate BISON inputs for each rod by populating the BISON template file.

### 2.4.3 File Naming Convention

Each file created by XML2MOOSE will conform to the naming convention given below to convey the starting cycle index and location in the core:

```
vera_C03_ASSYB097_P009_6.5.1.4.5.6.bison.i
```

where `vera` indicates the base filename executed with XML2MOOSE, `C03` indicates that the rod originated from cycle 3, `ASSYB097` indicates the assembly name in the VERA input is ASSYB and it starts at index 97, `P009` means it is at pin index 9, and `6.5.1.4.5.6` are the corresponding cell indexes from the VERA input that comprise the axial stack of the rod.

The assembly indexes for a core such as WBN1 with 193 assemblies is shown below. Future modifications may replace the assembly index with the X-Y label, which might make identification easier.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | | |
| | | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | | |
| | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | |
| | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | |
| 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 |
| 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 |
| 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 |
| 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 |
| 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 |
| | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | |
| | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | |
| | | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | | |
| | | | | 187 | 188 | 189 | 190 | 191 | 192 | 193 | | | | |

**Fig. 2. Assembly indexing with a jagged core.**

Similarly, the corresponding figure of pin indexes in a 17 × 17 assembly is shown below. This indexing is more straightforward if there are no jagged boundaries, which are present in a core layout for the assembly index. Currently, more complicated designs such as CE 16 × 16 and BWRs are not yet supported.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 |
| 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 |
| 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 |
| 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 |
| 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 |
| 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 |
| 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 |
| 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 |
| 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 |
| 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 |
| 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 |
| 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |
| 256 | 257 | 258 | 259 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 |
| 273 | 274 | 275 | 276 | 277 | 278 | 279 | 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 |

**Fig. 3. Pin indexing in an assembly.**

### 2.4.4 Imposed Transition Times

Because BISON can encounter convergence issues if simulation parameters change too rapidly, several restrictions are imposed to ensure that there is ample time when transitioning, particularly between zero- and full-power conditions. Initially, all rods start at cold zero power (CZP) and transition to (HZP) conditions over an imposed 3-hour time frame. From HZP, the transition to hot full power (HFP) occurs over a 24-hour period. From here, the reactor operates as specified until the end of the cycle is reached, when it transitions back to HZP over 24 hours and then to CZP over 3 hours, where it remains for the duration of the cycle outage. This process is repeated for the number of cycles required.

Sometimes an instantaneous power change is imposed in VERA. A similar 24-hour period is imposed in BISON to robustly handle this transition.

It is worth noting that if explicit startup/shutdown periods are more explicitly modelled in VERA, these will also be appropriately reflected in the BISON power history.

## 2.5 Examples and Regression Testing

Eight regressions tests are currently used to ensure performance of this capability, and they are highly recommended as a starting point for examples of how to use it. For this reason, each test will be briefly described.

### 2.5.1 bison_from_vera

The bison_from_vera test is the simplest regression test, as it is a basic single pin case that was run out to 300 effective full power days (EFPDs). It has a 3.1% $^{235}$U enrichment and a simple power history of 95% until it reaches 10 EFPDs. Then it goes up to 110% until it reaches 100 EFPDs. At that point it operates at 120% until it reaches 300 EFPDs. While this power history is unlikely, it is the way to test the capability and ensure that power changes are accounted for correctly.

### 2.5.2 bison_from_vera_clad

This case is identical to the bison_from_vera case except that it uses the clad outer surface temperature as the boundary condition instead of the bulk coolant temperature.

### 2.5.3 bison_from_vera_fast

This case is also identical to the bison_from_vera case except that it uses the fast flux data reported by MPACT instead of the fast flux factor available in the BISON template file.

### 2.5.4 bison_from_vera_ifba

This test builds on bison_from_vera by adding an integral fuel burnable absorber (IFBA) coating to the solid fuel pellets. It also includes annular fuel pellets in the blanket regions. Currently, BISON does not account for annular pellets with the internal mesh generator, so the preprocessor ensures that the volume of these annular pellets is accounted for in the plenum.

By comparing the BISON inputs generated for this and bison_from_vera, the minor differences imposed by being identified as an IFBA rod can be seen, such as the inclusion of the `he_prod` block in the input.

### 2.5.5 bison_from_vera_multi

bison_from_vera_multi is a 3 × 3 cross core with 3 × 3 pin assemblies, and it simulates the core without shuffling over three cycles. The depletion inputs for each cycle are included (c1.inp, c2.inp, and c3.inp), as well as the corresponding HDF5 files. XML2MOOSE is run using the bison.vera input, where the BISON block can be seen to reflect these cycles accordingly.

### 2.5.6 bison_from_vera_multi_qtr_shuffle_mir

This test also uses 3 × 3 pin assemblies, but in a full 3 × 3 core layout, and it simulates three cycles, including shuffling. From c3_shuffle.vera, it can also been observed that one assembly (1A-3) skips cycle 2 and comes from cycle 1. This test uses mirror symmetry, and a later test uses rotational symmetry.

### 2.5.7 bison_from_vera_multi_qtr_onlyC3

This test is identical to bison_from_vera_multi_qtr_shuffle_mir except with the `only_cycle` flag included to restrict the input generates only the files pertinent to cycle 3.

### 2.5.8 bison_from_vera_multi_qtr_shuffle_rot

This is the last test, and it is identical to bison_from_vera_multi_qtr_shuffle_mir, but with rotational boundary conditions along the symmetry line.

# 3. GENERATING FUEL TEMPERATURE TABLES

## 3.1 History

Fuel temperature table development is something that began in VERA early in FY15. In the planning for the capability, it was decided to pursue a quadratic fit with respect to power, using burnup bins, providing coefficients for each bin:

$$\Delta T = T_{fuel} - T_{cool} = aP + bP^2$$

In the above equation, $\Delta T$ is the difference between the average fuel temperature ($T_{fuel}$, which is considered to be constant radially) and the bulk coolant temperature ($T_{cool}$). $P$ is the local linear heat rate, which is determined by the power profile in MPACT.

Meetings with the science council yielded the idea of using 9 predefined axial power shapes input to BISON to formulate the temperature table. These 9 axial profiles were intended to cover a range of linear heat rates by using several different rod powers along with top and bottom peaked distributions. A table based on this approach was constructed and used in the simulations of WBN1, Cycles 1-12 [13]. However, it was noted that the maximum burnup in the table was ~22.5 GWd/MT because of convergence issues observed in BISON. Additionally, the table was constructed using manual extraction by opening the output Exodus files in Paraview and determining the difference between the fuel and coolant temperatures by hand.

In the subsequent months since the report, attention has been given to try to extend the burnups available as well as make the data extraction easier. To help address both of these issues, a new approach has been adopted, which uses a uniform linear heat rate ranging from 1-15 kW/ft in 1 kW/ft increments with a coolant temperature boundary condition of 585 K. This effectively turns the problems into a 1D radial calculation as opposed to the 2D R-Z calculation, making data extraction available from the output CSV file, instead of relying on the Exodus file. It is worth noting that now that SEACAS and NetCDF have been incorporated, it would be possible to more easily automate based on the Exodus file. However, the new approach also allows data to be obtained at considerably higher burnups, particularly at lower linear heat rates.

From the initial testing, it appears that the new approach yields fairly comparable results. In fact, there were several trends in the results from the previous approach that did not seem to make physical sense, so there is increased confidence that this approach is demonstrating some of the expected behavior, particularly once contact is made. For example, in the original table, the temperature monotonically decreased with burnup, even out to 22.5 GWd/MT, indicating that contact likely was not made. Additionally, the original table was missing the sharp decrease in temperature in the first few GWd/MT, where substantial fuel performance physics are in play. In fact, the temperature was nearly constant, only slightly decreasing out to 10 GWd/MT.

## 3.2 Generating Uniform LHR Files With XML2MOOSE

To generate the uniform LHR files that must be executed to generate the table, one simply needs to change the `solve_type` flag in the BISON block to `TEMP_TABLE`. XML2MOOSE will then pick out all unique rods in the problem based on the cell indexes used to construct the rod and will create 15 inputs for each rods, corresponding to the 1-15 kW/ft LHRs in 1 kW/ft increments.

Once the inputs have been generated, they can be run in any fashion desired. In general, it is recommended to use a bash script to run them all in the background simultaneously. Typically, the inputs are run either until (1) BISON fails to converge or (2) the user has checked that the burnup reached is sufficient and manually terminates the simulation. While BISON will eventually fail to converge for most LHRs at a sufficiently high burnup, some can continue with small timesteps for a very long time, just on the edge of failing, so manually terminating is desired in some cases.

## 3.3 Post-Processing Results Into a Temperature Table

To generate a table with a quadratic fit, a python script has been developed that takes in a list of the BISON outputs with the corresponding LHR value. This script (`create_temp_table.py`) can be found in the MOOSEExt/src directory. After reading in the output data from BISON, the script uses linear interpolation of the data to determine a temperature value for each burnup point. Once the data is obtained, a least squares algorithm in python's NUMPY routines is leveraged to provide a quadratic fit for that burnup bin, accounting for the values from all LHRs. The coefficients from this fit are then printed to a file that VERA can read.

The script takes in a file describing several options as well as a list of the output CSV files along with their corresponding LHR value. Below is an example of such a file. In it, the first row in the table corresponds to (1) the number of files being read, (2) the burnup discretization (in GWd/MT), (3) the moderator temperature used in the uniform LHR cases, and (4) a Boolean denoting whether or not effective fuel temperatures will be calculated, using the following relation between centerline and fuel surface temperature:

$$T_{eff} = T_{surf} + \frac{4}{9}\left(T_{cent} - T_{surf}\right).$$

```
15  0.1 585 0
   1 A1_LHR01_7.9.1.8.9.7_output.csv 60
   2 A1_LHR02_7.9.1.8.9.7_output.csv 60
   3 A1_LHR03_7.9.1.8.9.7_output.csv 60
   4 A1_LHR04_7.9.1.8.9.7_output.csv 60
   5 A1_LHR05_7.9.1.8.9.7_output.csv 60
   6 A1_LHR06_7.9.1.8.9.7_output.csv 60
   7 A1_LHR07_7.9.1.8.9.7_output.csv 60
   8 A1_LHR08_7.9.1.8.9.7_output.csv 60
   9 A1_LHR09_7.9.1.8.9.7_output.csv 55
  10 A1_LHR10_7.9.1.8.9.7_output.csv 50
  11 A1_LHR11_7.9.1.8.9.7_output.csv 45
  12 A1_LHR12_7.9.1.8.9.7_output.csv 45
  13 A1_LHR13_7.9.1.8.9.7_output.csv 45
  14 A1_LHR14_7.9.1.8.9.7_output.csv 45
  15 A1_LHR15_7.9.1.8.9.7_output.csv 45
```

If effective temperature is not used, then the volume-averaged fuel temperature is reported. In the rest of the file, we see a list of the output files where the first item on each line is the LHR in kW/ft, the second is the output CSV file, and the third item is the burnup cutoff to be considered for that output. As mentioned, some of the BISON cases may continue for a long time, and the results can become unreliable past a certain point, so this value allows to user to truncate the data used in the table generation. The script will produce a figure showing the raw data used to generate the table, and users can use that as a guide to determine the appropriate value for the cutoff, and simply run the script again.

## 3.4 Examples and Regression Testing

**bison_from_vera_multi_temp_table**

This test builds on the bison_from_vera_multi example, which contains two different enrichments. To cover both of these, it generates 30 files ranging from 1-15 kW/ft in 1 kW/ft increments (one for each enrichments).

# 4. EXECUTING BISON

With several thousand files generated by XML2MOOSE, it is a difficult task to execute them all. One possibility is to run them all in tandem using the MOOSE MultiApps capability by creating each pin as a separate MultiApp. However, there is an outstanding bug in BISON that prevents all outputs from being written when being run in this manner. The most success achieved at this point was by dividing the rods into batches and running them individually. For example, for each cycle in WBN1, 14,784 BISON inputs are created. These were divided into 616 jobs of 24 rods each, and each was executed using 12 message passing interface (MPI) processes in a total runtime of approximately 35,000 core-hours on Falcon. However, other systems have restrictions that will prevent this approach from being successful. On Titan and Eos, users are limited to two running jobs at once, so queueing 616 separate jobs is impractical. Also, sometimes there are restrictions on the number of tasks that can be combined into one job. Titan and Eos have a limit of 100 aprun (equivalent of mpirun) calls per job, so no more than 100 rods can be run in a single job.

Running all of the rods created by XML2MOOSE may be the most difficult task involved in this process. Future development to address some of these issues is necessary before considering widespread deployment.

## 5. POST-PROCESSING RESULTS

Once all of the BISON cases have been executed, results can be gathered and visualized using VERAView [12]. A post-processor program called `bison_post` can be called and will conglomerate the output data onto the HDF5 file(s) specified by `power_file`.

**NOTE:** It is highly recommended to back up the HDF5 file produced by VERA before attempting to use bison_post.

To run `bison_post`, all of the output CSV and/or Exodus files must be in the same directory as the input file. This means that if the inputs were moved or divided into batches before running, the output files must be gathered into one location. As when running XML2MOOSE, bison_post can be executed using the following:

```
bison_post -c vera
```

**TIP:** As stochastic fails have occasionally been encountered with BISON, it is sometimes useful to first run bison_post with just the CSV files available. This is quick and can be opened in VERAView to ensure that data is available from all rods at all statepoints. If the data seems adequate, `bison_post` can be rerun with the CSV AND Exodus files. This can save time as Exodus file processing is more time consuming. However, the VERA HDF5 file will need to be restored to avoid overwriting the 2D CSV datasets, which will cause an error.

Using data from the CSV files, the following 2D datasets will be added to the HDF5 file and will be registered in VERAView:

1) cumulative_damage_index
2) average_clad_temperature
3) bison_burnup
4) maximum_clad_hoop_stress
5) minimum_clad_hoop_stress
6) he_prod
7) maximum_clad_temperature
8) maximum_fuel_centerline_temperature
9) maximum_fuel_surface_temperature
10) minimum_gap_distance
11) minimum_clad_temperature
12) plenum_pressure
13) average_fuel_temperature
14) rod_input_power
15) rod_output_power

These can be found under the radial datasets, as in Fig. 4:

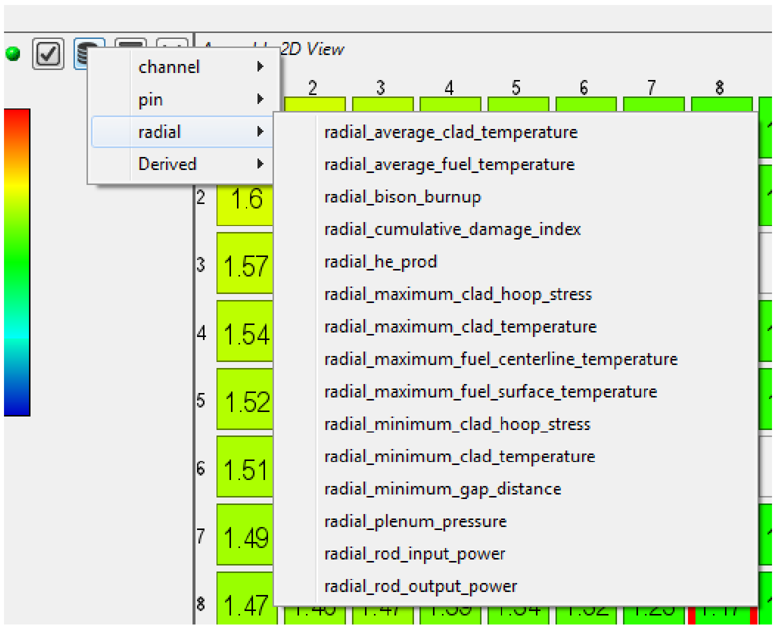

2D View
channel
pin
radial
Derived
radial_average_clad_temperature
radial_average_fuel_temperature
radial_bison_burnup
radial_cumulative_damage_index
radial_he_prod
radial_maximum_clad_hoop_stress
radial_maximum_clad_temperature
radial_maximum_fuel_centerline_temperature
radial_maximum_fuel_surface_temperature
radial_minimum_clad_hoop_stress
radial_minimum_clad_temperature
radial_minimum_gap_distance
radial_plenum_pressure
radial_rod_input_power
radial_rod_output_power
1.6
1.57
1.54
1.52
1.51
1.49
1.47

**Fig. 4. Radial Datasets in VERAView**

From Exodus files, the following data are automatically registered and can be found under the pin datasets:

1) max_clad_hoop_stress
2) average_temperature
3) avg_fuel-clad_gap_thickness
4) max_centerline_fuel_temperature

These can be found under the radial datasets, as in Fig. 5:

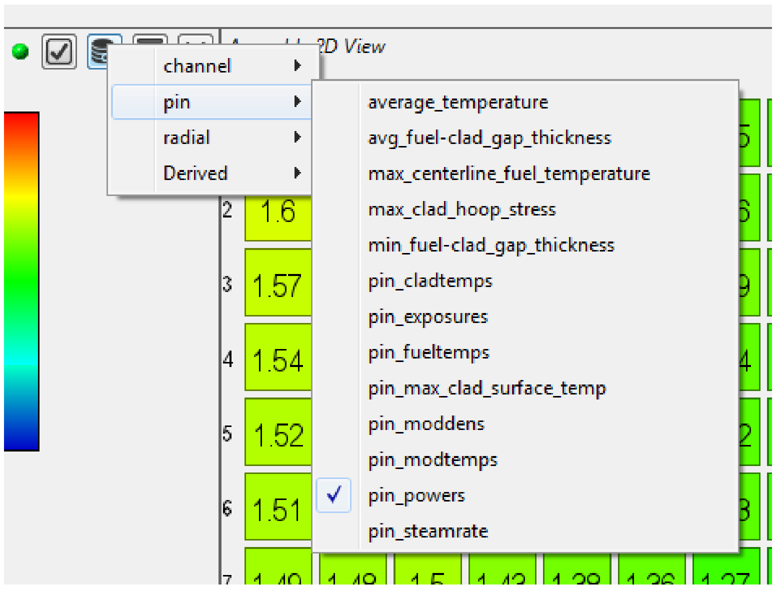
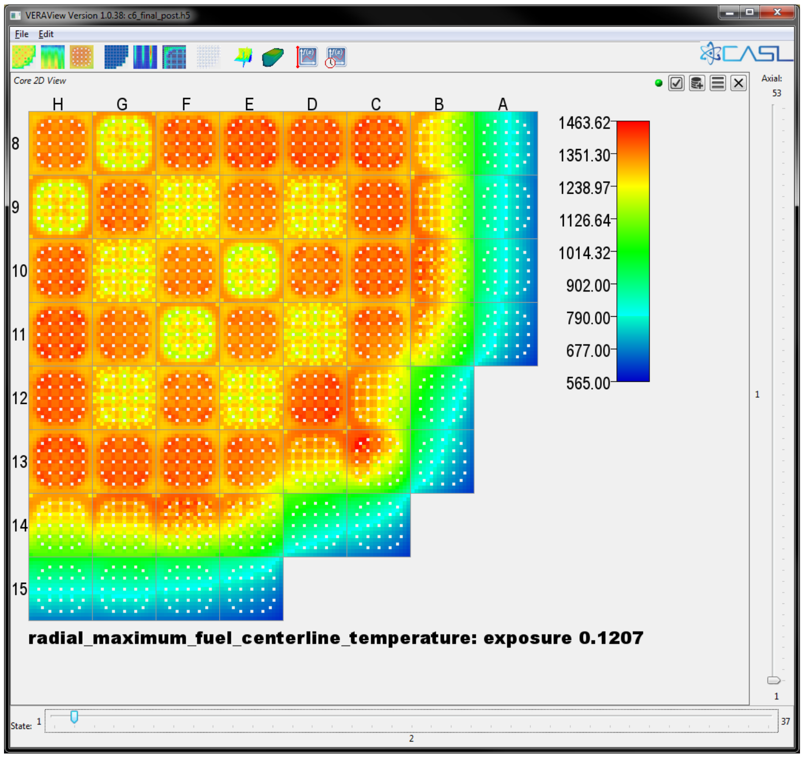


**Fig. 5. Pin Datasets in VERAView**

An example of the 2D maximum centerline temperature (K) distribution (obtained from CSV files) is presented below (Fig. 6), as well as an example 3D distribution generated by VisIt [15] (Fig. 7) from Exodus files:



radial_maximum_fuel_centerline_temperature: exposure 0.1207

**Fig. 6. VERAView visualization of 2D maximum centerline fuel temperature distribution.**

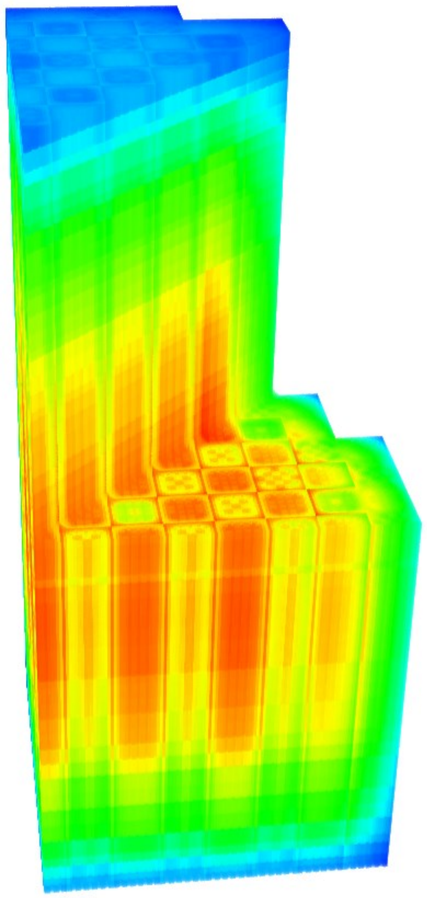


**Fig. 7. VisIt visualization of 3D maximum centerline fuel temperature.**

To generate plots in VisIt, the HDF5 files produced by VERA/BISON first need to be run through the `hdf5tosilo` tool, which will produce a SILO file for each state in the HDF5 file. These can then be opened in VisIt and the data is available, as in Fig. 8, which shows the data available to pseudocolor visualizations.

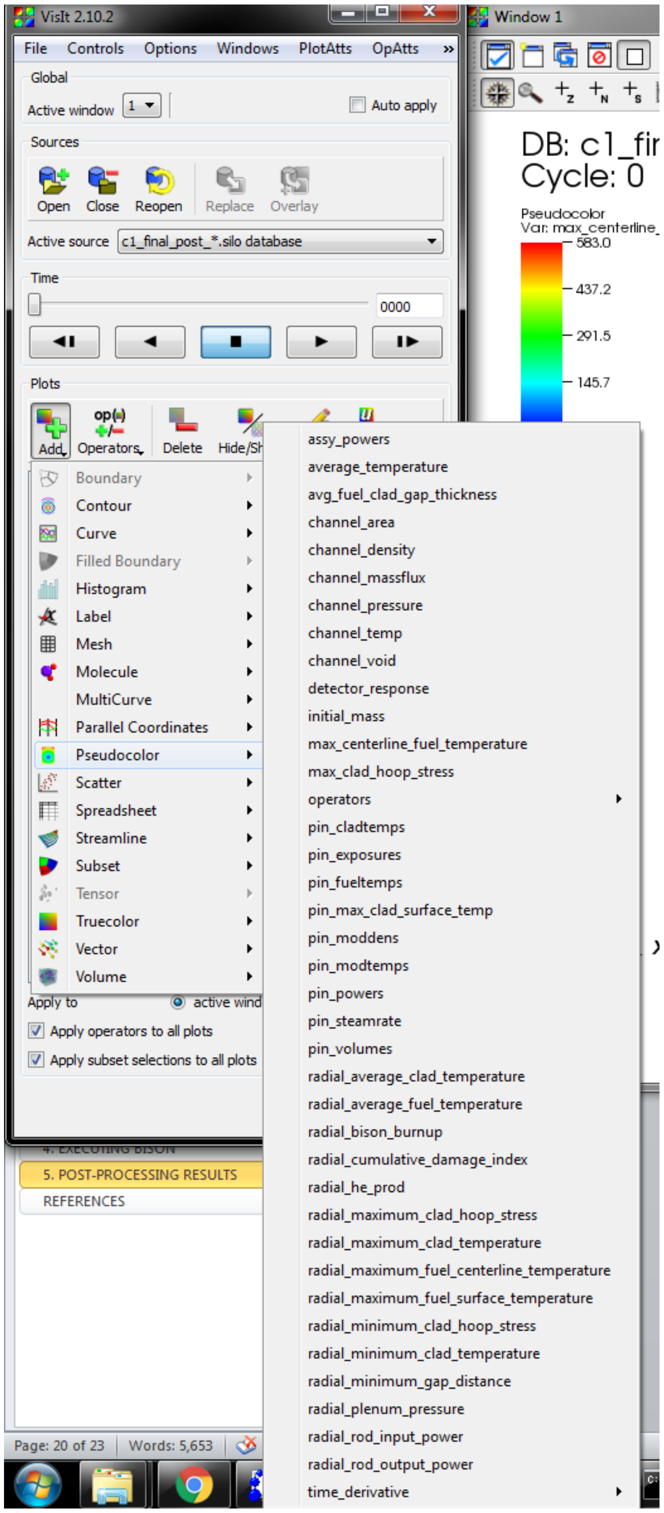


**Fig. 8. VisIt dataset examples through pseudocolor plots.**

## REFERENCES

1. K. T. Clarno et al., "High fidelity modeling of pellet-clad interaction using the CASL virtual environment for reactor applications," in: *Proceedings of the ANS Joint International Conference on Mathematics and Computation (M&C 2015), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*. Nashville, TN, USA (2015).

2. J. Powers et al., *Integrating BISON into VERA and Fuel Temperature Calculations*, Technical Report CASL-U-2016-1059-000, Oak Ridge National Laboratory (2016).

3. S. Stimpson et al., *Standalone BISON Fuel Performance Results for Watts Bar Unit 1, Cycles 1–3*, Technical Report CASL-U-2015-1010-001, Oak Ridge National Laboratory (2016).

4. S. G. Stimpson et al., "Assessment of Pellet-Clad Interaction Indicators in Watts Bar Unit 1, Cycles 1-3 Using VERA," *Proc. PHYSOR 2016*, Sun Valley, Idaho, USA (May 1-5, 2016).

5. S. G. Stimpson et al., "Assessment of Pellet-Clad Interaction with VERA: WBN1, Cycles 6-7," Proc. Top Fuel 2016, Boise, Idaho, USA (September 11-16, 2016).

6. N. Capps et al., "Demonstrating BISON Analysis Capability for Core-Wide PWR Pellet-Clad Interaction (PCI) Screening," CASL-U-2016-1200-000, Oak Ridge National Laboratory (September 30, 2016).

7. J. Powers et al., "Use and Assessment of BISON Within VERA in Support of PCI Challenge Problem," CASL-X-2016-1190-000, Oak Ridge National Laboratory (September 15, 2016).

8. R. Williamson et al. "Multidimensional multiphysics simulation of nuclear fuel behavior." *Journal of Nuclear Materials*, **423**: pp. 149–163 (2012).

9. J. D. Hales et al., *BISON Theory Manual: The Equations behind Nuclear Fuel Analysis*, Technical Report, Idaho National Laboratory (2015).

10. D. Gaston et al., "Moose: A parallel computational framework for coupled systems of nonlinear equations." *Nuclear Engineering Design*, **239** (2009): 1768–1778.

11. S. Palmtag and A. Godfrey, *VERA Common Input User Manual*, CASL-U-2014-0014-002, Revision 2, Oak Ridge National Laboratory. Available online at http://www.casl.gov/docs/CASL-U-2014-0014-002.pdf (February 23, 2015).

12. A. Godfrey and R. Lee, *VERAView User's Guide*, Technical Report CASL-U-2016-1058-000, Oak Ridge National Laboratory (2016).

13. A. Godfrey et al., *VERA Benchmarking Results for Watts Bar Nuclear Plant Unit 1 Cycles 1–12*, Technical Report CASL-U-2015-0206-000, Oak Ridge National Laboratory, Available online at http://www.casl.gov/docs/CASL-U-2015-0206-000.pdf (2015).

14. The HDF Group. *Hierarchical Data Format*, version 5, 1997–2016. http://www.hdfgroup.org/HDF5/.

15. H. Childs et al., "VisIt: An End-User Tool for Visualizing and Analyzing Very Large Data," High Performance Visualization – Enabling Extreme-Scale Scientific Insight, pp. 357-372 (2012).