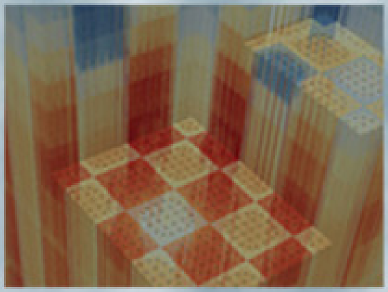
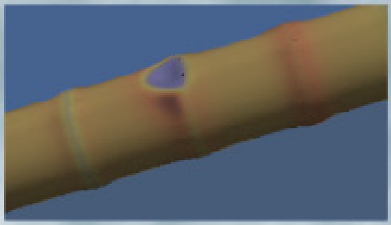
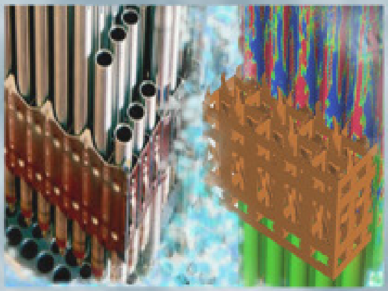
CASL-U-2016-1058-000







VERAView User's Guide

Andrew Godfrey
Ronald Lee
Oak Ridge National Laboratory

March 31, 2016

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0	3/31/16	All	Initial Release

Document pages that are:

Export Controlled _None____

IP/Proprietary/NDA Controlled_None____

Sensitive Controlled_None____

Requested Distribution:

To: N/A

Copy: N/A

EXECUTIVE SUMMARY

VERAView has been developed as an interactive graphical interface for the visualization and engineering analyses of output data from VERA. The python-based software is easy to install and intuitive to use, and provides instantaneous 2D and 3D images, 1D plots, and alpha-numeric data from VERA multi-physics simulations. This document provides a brief overview of the software and some description of the major features of the application, including examples of each of the encapsulated 'widgets' that have been implemented thus far. VERAView is still under major development and large changes in the software and this document are still anticipated.

CONTENTS

FIGURES

TABLES

ACRONYMS

CASL	Consortium for Advanced Simulation of Light Water Reactors
CSV	Comma Separated Values file format
CTF	COBRA-TF subchannel thermal-hydraulics code
GUI	Graphical User Interface
EFPD	Effective Full Power Days
PHI	Physics Integration
PWR	pressurized water reactor
VERA	Virtual Environment for Reactor Applications

1. OVERVIEW

Prior to 2015, the Consortium for Advanced Simulation of Light Water Reactors (CASL) was primarily focused on building and validating its virtual reactor capabilities, the Virtual Environment for Reactor Applications (VERA). As the capability for high fidelity reactor simulation became a reality for larger problems and multiple fuel cycles, the need for a post-processing analysis tool became significant. Tools like ParaView [2] and VisIt [3] were the main applications for visualization at the time, but they required significant training, some tribal knowledge, and could not natively interpret the reactor-specific geometry in the VERA.

VERAView was born from a rapid prototyping development process with frequent stakeholder feedback and a narrow application-driven scope of enabling reactor analysis (mainly fuel rod and coolant channel distributions) from VERA-CS output. Its primary requirements from the early stages were ease-of-use, flexibility, and maintainability. It directly reads the VERAOut HDF5 file specification format [1], and interprets reactor data generally based on dataset size and shape. This important aspect of the design allows VERAView to display almost ANY data in this format and does not limit it to the codes in VERA. For instance, CASL has already used VERAView to compare processed results from non-CASL codes such as KENO-VI and MCNP, as well as industry lattice physics and nodal codes. This extensibility gives VERAView stand-alone value for the nuclear industry in addition to CASL's advanced multi-physics capabilities.

To support CASL's multi-physics tools, VERAView seamlessly integrates multiple physics capabilities and supports extensibility to the future needs and applications of its users. A modular design of 'widgets' interconnected through a common container and event sequences allows for an unlimited expansion of tools, data views, and calculations. Additionally the selection of python for the source language gives VERAView instant access to thousands of available libraries and add-ons, as well as provides an easy-to-use platform for future developers to create custom widgets as new needs arise. Tapping into existing tools and existing skills of current and future engineers also aligns very well with CASL goals.

This document provides a brief overview of the design and components of VERAView. As a graphical user interface (GUI), VERAView is best learned by personal interaction with relevant data. The interface is fairly intuitive and can be interrogated using typical modern GUI interactions, such as left mouse clicks, right mouse clicks, click-and-drags, etc. Additionally, VERAView is still under significant develop and large changes are still forthcoming, so overly detailed documentation is not practical at this time.

All of the images used in this manual were generated using VERAView Build 34, released for testing in late March, 2016.

2. DESIGN PHILOSOPHY

VERAView has been designed and developed to be as useful and flexible as possible, while attempting to minimize development and maintenance. Its current features are a result of requirements derived from the rapid prototyping process by which it has been and continues to be developed. At a high level, these are:

- VERAView is specifically designed to interpret the output data from VERA codes. The VERAOut specification [1] provides structure for how reactor data is structured in a HDF5 file. Understanding this structure, VERAView implicitly displays the data in the form of simplified pressurized water reactor (PWR) geometry.

- Though VERAView can ONLY process files in the VERAOut format, it is NOT connected directly to VERA codes and does not require any specific data from the physics software. It is currently limited to common physical geometries such as fuel rods, coolant channels, fuel assemblies, etc., and does not display any more specific information (though this may be an option in the future). The result is that data from ANY reactor methods can be displayed by first converting the data to the VERAOut format. Numerous converting codes have been created for this purpose.

- VERAView is designed to be used with nearly no experience or training. While applications such as ParaView and VisIt are powerful, general-purpose data analysis and visualization tools, setting up data sources and rendering pipelines requires a fair amount of expertise and can be complex. VERAView provides an alternative, simplified interaction for engineers and students that is designed from the beginning for reactor analysis.

- VERAView is to be a true multi-physics analysis tool, specifically for the simultaneous visualization of neutronics, thermal-hydraulics, and fuel performance data. Generally the fidelity of the displayed data is in the form of fuel rod or coolant channel quantities, but VERAView can also provide coarser quantities derived from rods and channels (such as assemblies or axial distributions), and can also display in-core detector data as well. Currently VERAView does not support methods-specific geometries, meshes, or intra-pin distributions, though these features could be added in the future.

- The purpose of VERAView is to go beyond visualization and provide numeric data for engineering analyses. Features such as labels, plots, tool tips, image extraction, and comma separated values (CSV) file export allow VERAView to be extremely useful for interactions with Microsoft Excel and PowerPoint. Functions such as finding the maximum values have already been implanted and features such as displaying differences between files will be developed in the future.

- VERAView allows inspection of data in multiple dimensions, and visualization by the user in different perspectives, to accommodate various types of reactor data and quantities. It also incorporates time (or exposure) as a fourth dimension into this philosophy in a seamless manner, so that the user can discover critical aspects of data as a function of space and/or time, quickly able to observe trends and distributions.

- The analysis capabilities are designed to be extensible to new codes, features, or data as needed. This is accomplished through the use of the custom widgets, which could be added by developers or even implemented and integrated by users.

- VERAView is intended to be executed locally on the user's personal computer, and is supported on Windows, Mac OS X, and Linux.

3. SYSTEM COMPONENTS

Python-2.7 has been chosen as the language and environment for VERAView development. Through each prototype iteration, VERAView has been tested under Windows, Mac OS X, and Linux.

Dependent Python modules are represented below.:

```
VERAView
  |
  +-- wxPython (wxWidgets)
  |
  +-- h5py
  |     |
  +-- numpy
  |
  +-- Pillow
  |
  +-- matplotlib
        |
        +-- pyparsing
        |
        +-- python-dateutil
        |
        +-- six
  |
  +-- mayavi
        |
        +-- apptools
        |
        +-- envisage
        |
        +-- traitsui
        |
        +-- VTK
```

The versions used in the initial distribution are as follows.

Table 1: Current Python Module Versions

Module	Min Version	Used Version
apptools	4.3.0	4.3.0
envisage	4.4.0	4.4.0
hp5py	2.3.1	2.4.0
mayavi	4.4.2-1	4.4.2-1
matplotlib	1.3.1	1.4.3
numpy	1.8.0	1.9.2
Pillow	2.5.2	2.7.0
pyparsing	2.0.1	2.3.0
python-dateutil	1.5	2.4.1
six	1.4.1	1.9.0
traitsui	4.5.1	4.5.1
wxPython/wxWidgets	3.0.1.0	3.0.2.0

The wxPython [4] module was chosen for the windowing toolkit. It is a Python wrapper around the wxWidgets [5] cross-platform GUI library and is include in many scientific Python environments such as Anaconda and Canopy.

The h5py module [6] is a Python wrapper around the HDF5 C application programming interface (API) and library. It is used for all processing of the VERA output file. Since h5py uses numpy [7], the latter is used for all dataset vector/array manipulations.

Pillow [8] is the Python Imaging Library (PIL) implementation used for image creation in raster-based 2D views. At present, all 2D plots are created using matplotlib [9].

Mayavi [10] is used for 3D visualizations. It is an open source capability maintained by Enthought that uses the VTK Python wrapper. Enthought includes Mayavi as an optional package in the Canopy Python environment they maintain and provide for Windows, Mac OS X, and Linux.

4. DATA FORMAT

The data files which can be interpreted by VERAView must conform to the VERAOut specification described in Reference 1. It is an HDF5 hierarchical binary data format. It consists of data groups, one of which is the ***/CORE*** group that contains the general reactor geometry data (core shape, fuel assembly locations, axial mesh, etc.), and the others representing multiple statepoints of the reactor. For instance, a reactor depletion with 20 timesteps can be represented with 20 groups named ***/STATE_0001*** through ***/STATE_0020***. In each of these statepoints are data which can change over time. Typical VERA-CS output contains statepoint data for fuel rod powers, exposures, temperatures, cladding temperatures, channel densities, etc.

Currently VERAView uses dataset shape and size matching to determine how the HDF5 data should be represented in the reactor. For this discussion, let's define the following dimensions:

- NASS – Number of fuel assemblies in the calculated geometry (quarter or full core)
- NAX – Number of axial planes in the core region (assume same for all data)
- NPIN – Number of fuel rods across a fuel assembly (assumes equal X and Y dimensions)
- NCHAN – Number of coolant channels across an assembly (assumes equal X and Y dimensions)
- NDET – Number of in-core instrument strings

Once the user provides an HDF5 file to VERAView, it attempts to determine these fundamental reactor dimensions, and returns an error if it is not successful. Typically the dataset ***/STATE_0001/pin_powers*** is used as a model to get these dimensions, but other methods are possible. Once the reactor geometry is determined successfully, VERAView identifies and categorizes the data in each statepoint on the HDF5 by size using the following types of rules (python array ordering shown):

- DATA(NPIN,NPIN,NAX,NASS): 4D array representing 3D fuel rod data
- DATA(NCHAN,NCHAN,NAX,NASS): 4D array representing 3D coolant channel data
- DATA(NPIN,NPIN,NASS): 3D array representing rod-wise axially-integrated (radial) quantities
- DATA(NAX,NASS): 2D array representing 3D assembly-wise data
- DATA(NAX,NDET): 2D array representing 3D detector signals
- DATA(NAX): 1D array representing radially-integrated (axial) distributions
- DATA(NASS): 1D array representing axially-integrated (radial) assembly-wise distributions
- DATA: Scalar quantity

This general categorization scheme allows VERAView to easily determine which datasets can be interpreted by which widgets, and provides the user the generic capability to visualize any dataset regardless of its name and without needed prior knowledge of its existence. For constructing files from other codes or methods, the user needs only to place the data on the files with the correct shape and it will automatically be available in VERAView.

Additionally, VERAView can derive the lower order data formats if they do not already exist on the HDF5 file. For axial pin powers, for instance, VERAView can perform the integration over the other dimensions to calculate the axial distribution and display it as a 'virtual' dataset. To do this, it uses a predefined weighted-averaging scheme, where the weights account for axial mesh heights and fuel rods or channels cut by the line of symmetry in quarter core models. VERAView can calculate its own weighting factors, or it can use the ***/CORE/pin_factors*** dataset if it exists. A similar methodology is employed for channel data. The derived datasets are created on the Select Dataset menu on the toolbars of each individual widget.

The primary benefit of this simplified approach is that VERAView will function with a very minimal set of data on the HDF5 file, which allows for custom files to be created by users very easily. The number of required datasets is very minimal.

5. INSTALLATION

VERAView is a python-based software product and can easily be installed and executed on a local computer. Python is a very widely-used scripting and programming language that is supported on many operating systems. VERAView requires a few more python modules than are typically installed on a system, and this configuration is now managed through Canopy (https://www.enthought.com/products/canopy/), a free python deployment and analysis package. Canopy can be downloaded and installed without administrative privileges.

At the time of this report, the current instructions for installing Canopy for VERAView are currently located at: https://newton.ornl.gov/~re7/xfer/casl/canopy-install.pdf . This location is publicly accessible. The user name is ***'casl'*** and the password is ***'rocks'***. As VERAView is more broadly deployed, this installation guide may be relocated or modified.

Once Canopy is installed, the latest version of VERAView can be downloaded and extracted. Current builds are placed at the same site: https://newton.ornl.gov/~re7/xfer/casl/; with the same username and password as above. Once the files are extracted, there are batch files and run scripts in the main folder to execute the program. Subsequent updates can just be extracted without any change to Canopy. This process is likely to change in the future.

A mailing list has also been created for user support and to receive updates for new VERAView versions and bug fixes. Information about this list is located at http://casl-dev.ornl.gov/mailman/listinfo/veraview, and its members can be contacted at veraview@casl-dev.ornl.gov . Because of the current development stage and rapid prototyping, new versions and bug fixes are often released weekly.

6. COMMON INTERACTIONS

VERAView is a GUI designed with common python tools and with the intent to conform to typical graphical operating system interfaces. The user is encouraged to explore the tool like any other GUI to learn about available options and features. In most cases these are self-evident and don't require an abundance of documentation. Some of these basic interactions are the following:

- There is a basic menu bar with typical operations such as:
    - Open a file
    - Exit the application
    - Minimize the application
    - Copy a value
- VERAView can be minimized, maximized, or closed like most windows by using the buttons in the title bar.
- There is a toolbar with buttons to add a new widget of the selected type to the window
- Most widgets support selection of a point in space and time with a left mouse click
- Most widgets have a context menu that pops up on the right mouse button click. This menu usually supports widget specific operations such as copy and changing options.
- Many widgets support the click-and-drag operation to zoom the image. The corners of the box created by the drag set the new extents of the image, and an "unzoom" option exists to zoom out, either on the right-click menu or the widget toolbars.
- Most widgets provide tooltips with local information when the user hovers over the widget with the mouse pointer.
- Most widgets support these common operations from their toolbars:
    - Dataset selection
    - Save or copy data or images
    - Close the widget and remove from the window
    - Disconnect the widget from sending and receiving control events (to be discussed later.)
- Currently there is only one color scheme for distributions (RGB) and its bounds cover the enter dataset and all statepoints (i.e. the maximum value is red and occurs typically at one place in the entire file/simulation).
- No units are displayed for any of the HDF5 data at this time. Because this information is not on the data file, VERAView just processes the datasets and passes the values through. This limitation will be addressed in future development.

7. BASIC LAYOUT

Figure 1 provides the basic VERAView layout using four widgets in a 2x2 configuration. When a data file is loaded, VERAView uses a default sequence of opening widgets depending on what data is located on the file. For instance, if channel data is found on the file, the channel views will open up automatically. The whole core views will only be used if more than one assembly is found, etc. Following Figure 1, more details are provided.

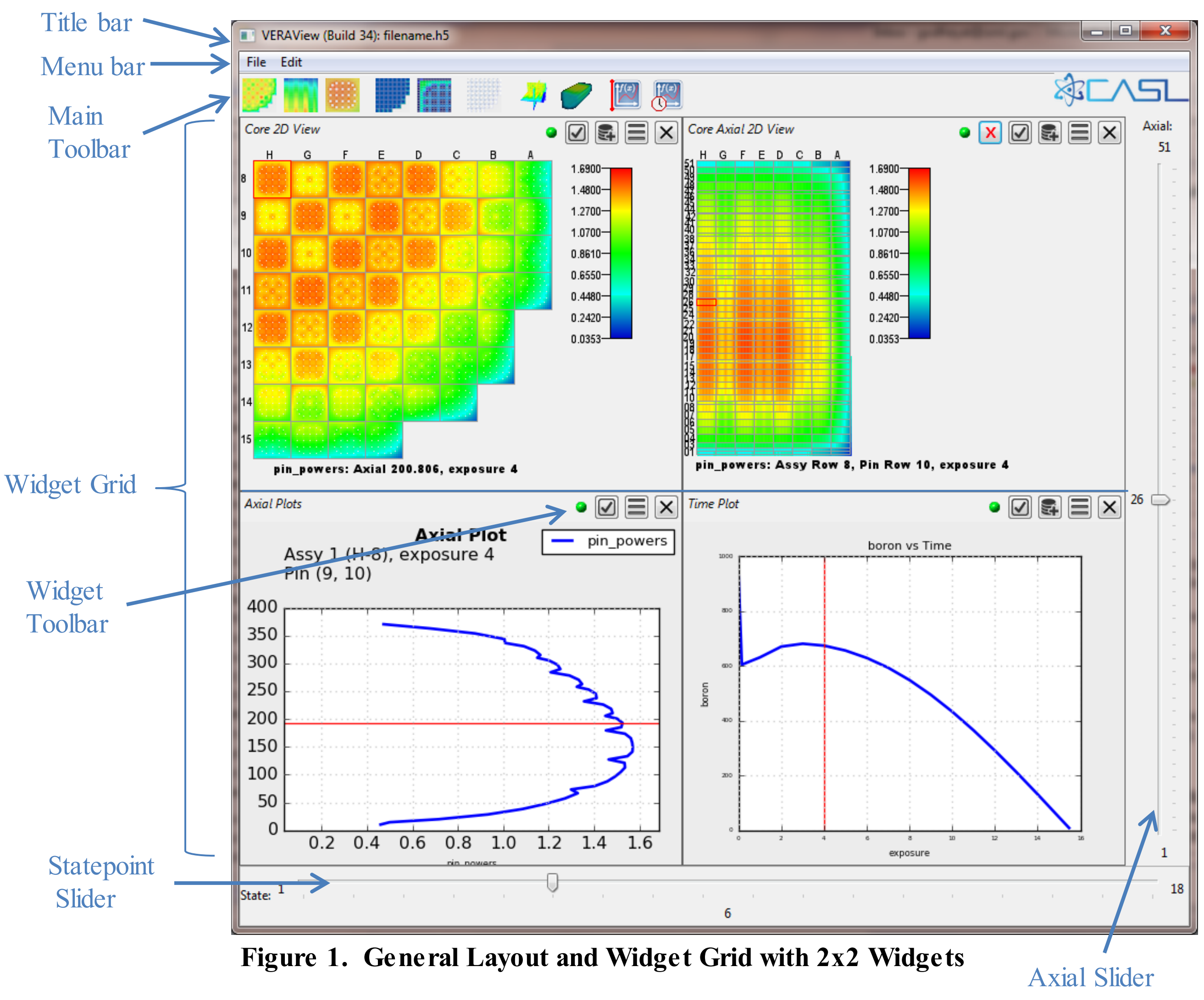


Figure 1. General Layout and Widget Grid with 2x2 Widgets

- The Title Bar contains the current build version, open file name, and the standard window control options.
- The Menu Bar has the File and Edit menus
- The Main Toolbar contains buttons to create/add widgets to the widget grid. This will be covered in more detail in subsequent sections. (This can also be done with **File → New** on the Menu Bar)
- The Widget Grid is a MxN sized container for the currently opened widgets. Each time a new widget is opened it is added to the grid in the next available location. If the grid is full, a new row will be added to the grid and the new widget will be placed in the first column of the new row. Currently there is no capability to move widgets from one grid location to another, or to 'pop out' a widget from the grid.

- The Statepoint Slider sets the current statepoint being displayed globally by VERAView. It can also be controlled by using the left and right arrow keys on the keyboard, or by clicking on a widget that supports statepoint selection.
- The Axial Slider sets the current axial index being displayed globally by VERAView. It can also be controlled by using the up and down arrow keys on the keyboard, or by clicking on a widget that supports axial selection.
- Each widget has its own Widget Toolbar to contain specific options and controls for that tool.

7.1 Resizing the Widget Grid

The user can add widgets to the view, or multiple instances of the same widget, as much as desired. The widgets can also be removed by clicking the upper right hand button in the Widget Toolbar. Typically the user will want to change the layout of the Widget grid after making such a change, but it currently does not occur automatically. To do the resize, use the **Edit → Resize** Grid option (**Ctrl-G**). This produces a prompt shown in Figure 2, where the user can select the desired grid size and layout (shown in red).

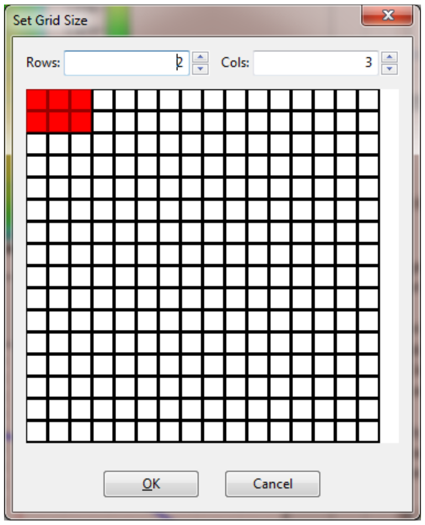


Figure 2. Resize the Widget Grid Prompt (showing a 3x2 selection)

7.2 Controlling Global Selections

For each widget in the Widget Grid, VERAView maintains global selection information, including:

- Currently selected fuel rod
- Currently selected coolant channel
- Currently selected fuel assembly
- Currently selected axial plane
- Currently selected statepoint
- Currently selected rod-dataset (also known as pin-wise)
- Currently selected channel dataset (if applicable)

In general, each widget can send and receive events for when these global selections change, if applicable to that particular widget. These are considered global because the value of the selection applies to all or most of the widgets, and can be altered in all widgets by just one interaction by the user (it is assumed that typically the user wants to change all widgets at once for a given selection).

Therefore, the Statepoint Slider selects the current statepoint (i.e. time, exposure) for all widgets that are displaying a single statepoint. Also, any widget which allows the user to select a statepoint (such as the Time Plot widget) will also set the global statepoint.

Likewise, the Axial Slider selects the current axial plane (or index) for all widgets that are displaying only data at a single axial plane, and widgets providing all axial location (such as the Axial Plot of the Core Axial 2D View) can provide the global axial plane selection as well. This technique allows the user to interact with the data freely and seamlessly and have all the information update at once.

The global fuel rod and coolant channel are selected by the 2D widgets showing pin and channel-wise data, respectively. Likewise for assembly data. The datasets themselves are only selected via the Widget Toolbars but by default will apply to all relevant widgets (i.e. widgets who know how to interpret the selected data).

In all respects each widget is individually responsible for communicated the location of the data it is presenting, whether it is the global selection or not.

7.3 Unlocking from Global Selections

By default each Widget is "locked" to the global selection of the Widget Grid. This means that it sends and receives events about the global selections. Receiving occurs when one of the selection indices changes in another widget or by the container. Sending occurs when the user interacts with the current widget to change a selection and it must communicate that out to the rest of the widgets. However, at times the user may want to interrupt this communication and allow a widget to have a local selection separate from the global selection. This is referred to as being "unlocked" from the global selection changes. In most widgets, on the Widget Toolbar, there is a button for unlocking. By default, all the control events are checked and the widget is fully locked. If the user unchecks the "Statepoint Index", that widget will no longer update when the global statepoint is changed, and if that widget can change its own statepoint, it won't affect any other widgets. It essentially becomes separated from the others and has its own independent selection, and this can be done for any or all of the various selections.

Another example is if the user only wants to view the core exit thermal-hydraulic conditions. In this case, the user would change the global axial plane to be at the top of the core, and then unlock the "Axial Value". Further changes in the globally selected axial plane will no longer update the "unlocked" widget.

Figure 3 displays an example of unlocking the axial location in a single fuel assembly. The data displayed is a 2D fuel rod lattice at two axial locations in the assembly. On the right, the lattice is showing data at the global axial index of 24 (186 cm), but the left lattice has been unlocked from the global axial selection and is fixed at 353 cm.

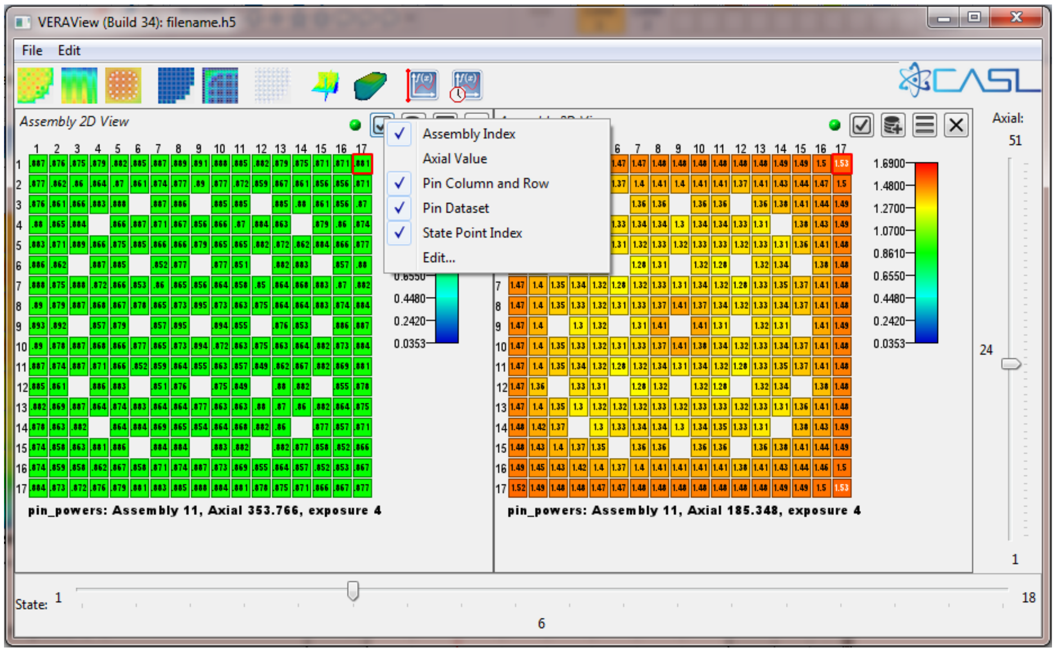

VERAView (Build 34): filename.h5
File Edit
Assembly 2D View
Assembly Index
Axial Value
Pin Column and Row
Pin Dataset
State Point Index
Edit...
pin_powers: Assembly 11, Axial 353.766, exposure 4
pin_powers: Assembly 11, Axial 185.348, exposure 4
Axial:
51
24
1
State:
1
6
18

Figure 3. Sample fuel lattice locked (right) and unlocked (left) from global axial selections

A final example of unlocking is if a user wants a particular widget to display a particular dataset no matter what other datasets are selected in the application. In this case, the user can simply unlock that widget by unchecking 'Pin Dataset' for instance (in the Control Events menu of the specific Widget Toolbar), and then can select a different dataset without impacting any other widgets.

8. CURRENT WIDGETS

VERAView already has many useful widgets that have each been developed with a specific application in mind. As was described earlier, savvy users may extend the current widgets, derive new ones from existing ones, or create totally new applications. The interface for the widgets is standardized resulting in a relatively simple model to extend functionality to new applications and concepts.

Each widget is developed to know what categories of data it can display (Section 3) and to send and receive events about the current local and global selections. Furthermore, each widget implements consistent zoom/unzoom and copy/paste functionalities where possible. For instance, the 2D core widget can display 2D or 3D pin or assembly data, can change the global pin or assembly selection, and can update its axial index or statepoint based on the global selection. As long as each widget is designed with this interoperability, it is very much self-contained otherwise.

This section summaries the available widgets and provides examples. Again the user is encouraged to interrogate the application themselves for a better understanding of how to interact with each.

8.1 Core 2D View

This widget, shown in Figure 4, provides a radial view of pin-wise or assembly-wise distributions for the entire core for a specified axial location and statepoint. For quarter-core simulations, only that quarter is shown (values across the symmetry line are mirror reflected). For pin datasets, the pin distribution is shown without labels. For assembly-wise data, the assembly is shown as a solid color with a numeric label for the value (right side of Figure 4). This widget is capable of selecting global assembly and pin (if applicable) and will update its axial plane and statepoint based on global changes. The user can zoom into a sub-region of the core by clicking and dragging a box around the region of interest.

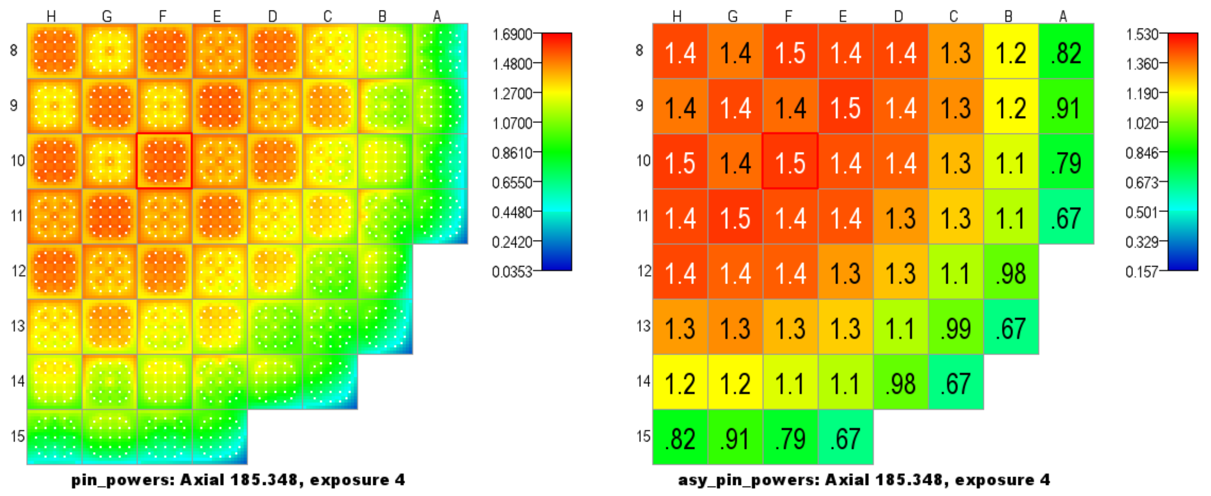


Figure 4. Pin-wise (left) and Assembly-wise (right) data in the Core 2D View

8.2 Channel Core 2D View

This widget, shown in Figure 5, is similar to the Core 2D View but displays channel data rather than pin data. It provides a radial view of channel-wise or assembly-wise distributions for the entire core for a specified axial location and statepoint. For quarter-core simulations, only that quarter is shown (values across the symmetry line are mirror reflected). For channels, the distribution is shown without labels. For assembly-wise data, the assembly is shown as a solid color with a numeric label for the value. This widget is capable of selecting a global assembly and channel (if applicable) and will update its axial plane and statepoint based on global changes. The user can zoom into a sub-region of the core by clicking and dragging a box around the region of interest.

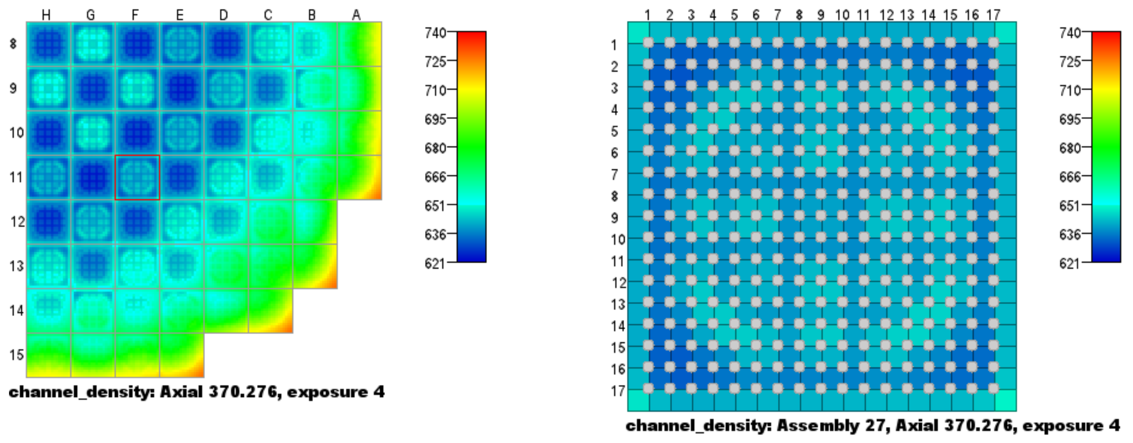


Figure 5. Channel-wise data as nominal (left) and zoomed (right) in the Channel Core 2D View

8.3 Assembly 2D View

This widget, shown in Figure 6, provides a radial view of pin-wise distributions (fuel lattice) for a single assembly at a specified axial location and statepoint. The dataset values are shown in each fuel rod location. This widget is capable of selecting global pin indices and will update its assembly location, axial plane, and statepoint based on global changes. The user can zoom into a sub-region of the assembly by clicking and dragging a box around the rods of interest.

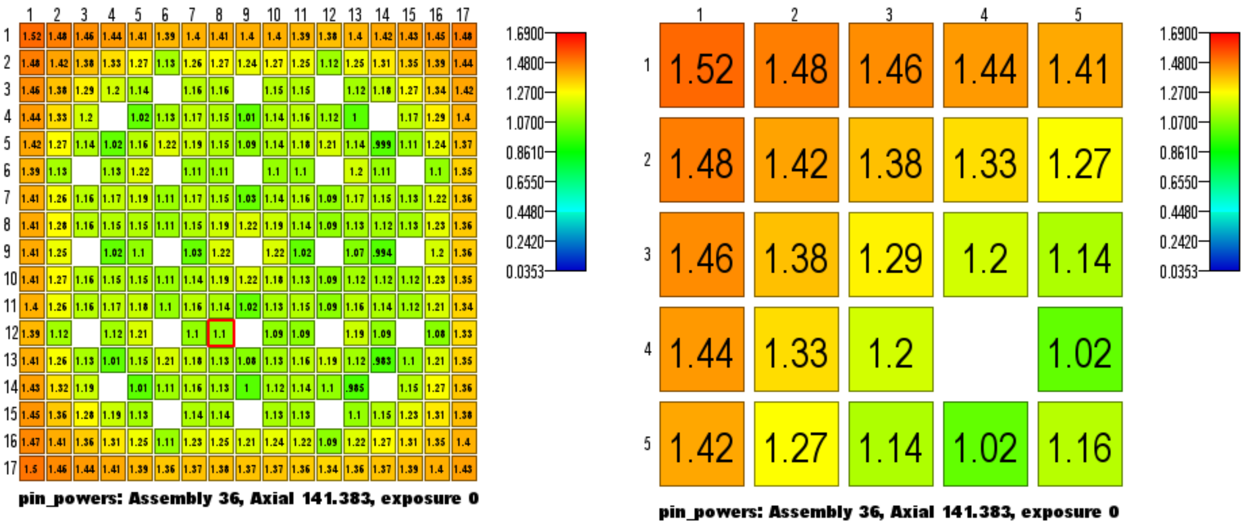


Figure 6. 2D fuel lattice data nominal (left) and zoomed (right) in the Assembly 2D View

8.4 Channel Assembly 2D View

This widget, shown in Figure 7, is nearly identical to the Assembly 2D View but it displays channel data rather than pin data. It provides a radial view of channel-wise distributions for a single assembly at a specified axial location and statepoint. The dataset values are shown in each coolant sub-channel location. This widget is capable of selecting global channel indices and will update its assembly location, axial plane, and statepoint based on global changes. The user can zoom into a sub-region of the assembly by clicking and dragging a box around the channels of interest.

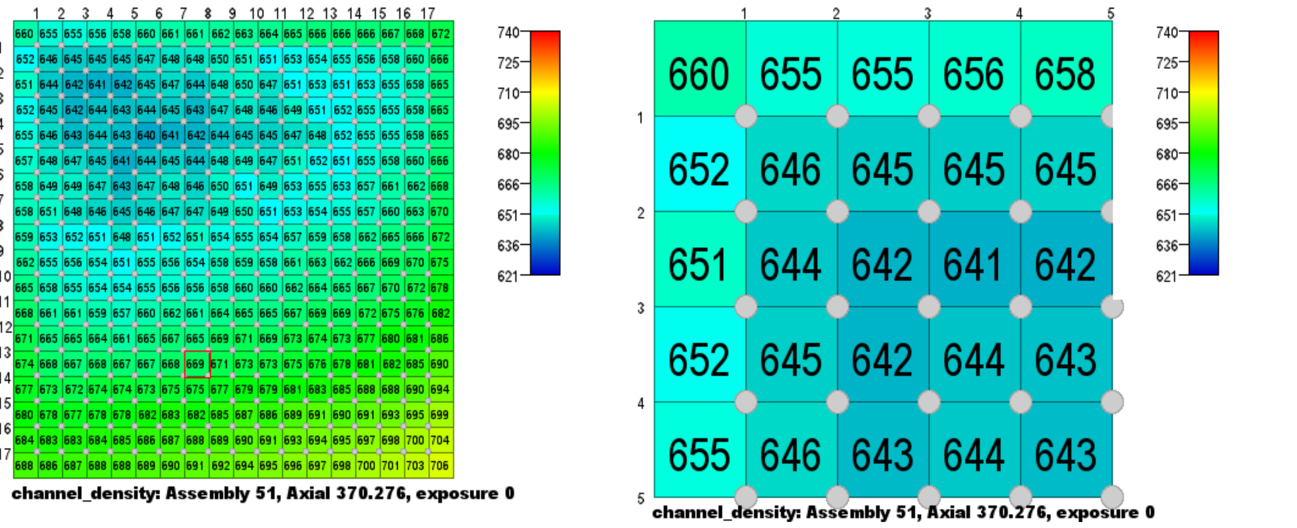


Figure 7. 2D coolant channel data nominal (left) and zoomed (right) in the Channel Assembly 2D View

8.5 Core Axial 2D View

This widget, shown in Figure 8, is similar to the Core 2D View but it displays data on the X-Z or Y-Z planes. It provides an axial view (from the side) of pin-wise or assembly-wise distributions for the entire core for a specified X or Y pin coordinate and statepoint. The axial dimension is presented as elevation based on the ***/CORE/axial_mesh*** dataset (in VERA output this is relative to the fuel assembly seating surface). For quarter-core simulations, only half of the core is shown (values across the symmetry line are mirror reflected). Currently this widget does not support labels for the assembly-wise values. This widget is capable of selecting global assembly and pin (if applicable, in one dimension) and axial plane, and will update the global pin index and statepoint based on global changes. The user can zoom into a sub-region of the core by clicking and dragging a box around the region of interest.

The user can toggle whether the X or Y plane is shown by clicking the Toggle Slice Axis button in the Widget Toolbar. If the X plane is shown, then assembly and pin coordinates in the X direction can be selected, and the Y pin coordinate of the X-Z plane is set by another widget (or by toggling this one). Likewise, if the Y plane is shown, then assembly and pin coordinates in the Y direction can be selected, and the X pin coordinate of the Y-Z plane is set by another widget (or by toggling this one)

Currently this widget does not support channel data, but this will be supported in the future.

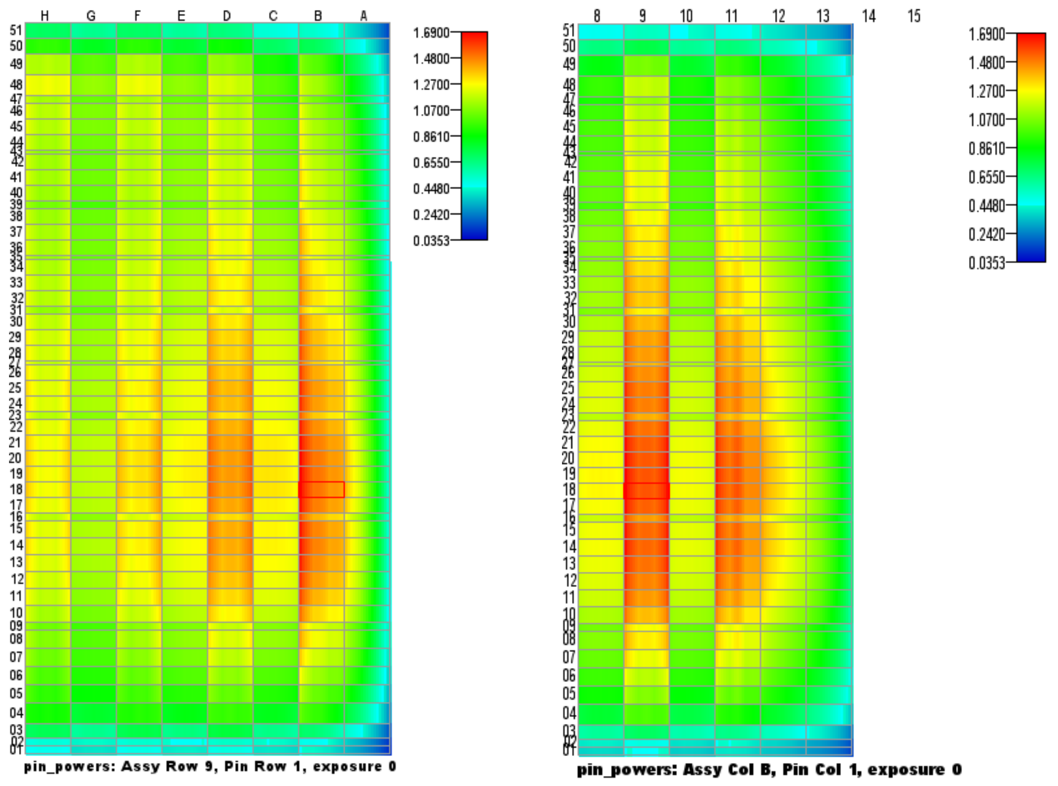


Figure 8. 2D Axial pin data showing X axis (left) and Y axis (right) in the Core Axial 2D View

8.6 Axial Plots

This widget provides the capability of plotting 1D data for any dataset in any statepoint with an axial dimension. For multi-dimensional data, the index of the other dimensions is specified by global selections. The plots are oriented physically such that axial elevation is plotted vertically and the values are horizontally. The widget also currently supports two horizontal axes (one on bottom, one on top). It provides the user with an axial plane selection and will update based on all other global selection changes.

By default, the Axial Plot widget displays the currently selected pin-based dataset and currently selected channel-based dataset simultaneously (if applicable). By default the pin data uses the lower axis and the channel data uses the upper axis. This is shown in Figure 9.

Additionally, the Axial Plot provides an advanced option for dataset selection. Selecting the 'Select Datasets' option of the Widget Functions menu, a dialog opens which permits the following additional options:

- Selection of any number of datasets (from a list of all axial or 3D data)
- Selection of which axis to use for that data (top or bottom)
- Ability to provide a scaling factor in order to fit three or more datasets with different scales on the two axes.

Figure 10 displays the dialog with a sample set of settings on the left, and the axial plot produced on the right. Regardless of how datasets are selected, they will continue to update based on selected pin/channel location and statepoint, unless they are unlocked from the global events.

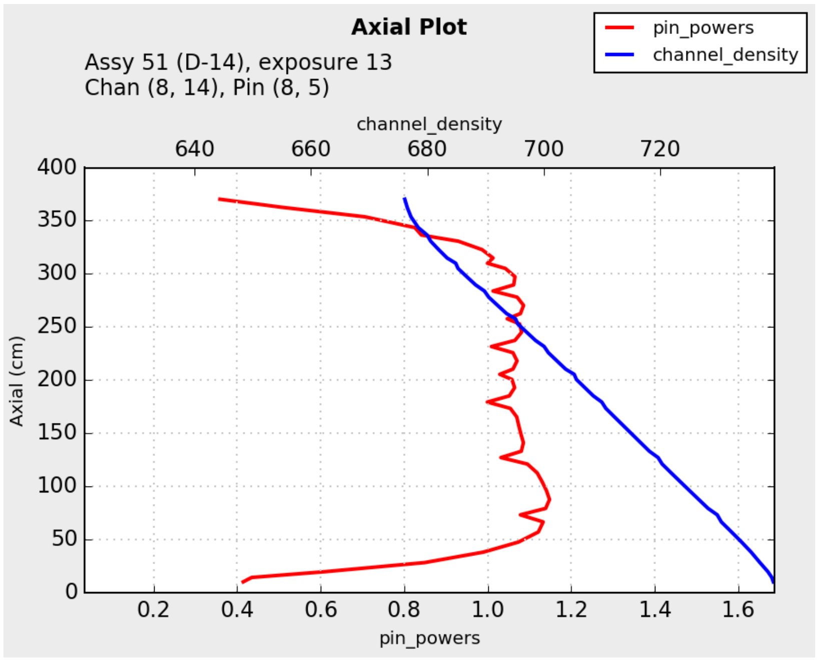
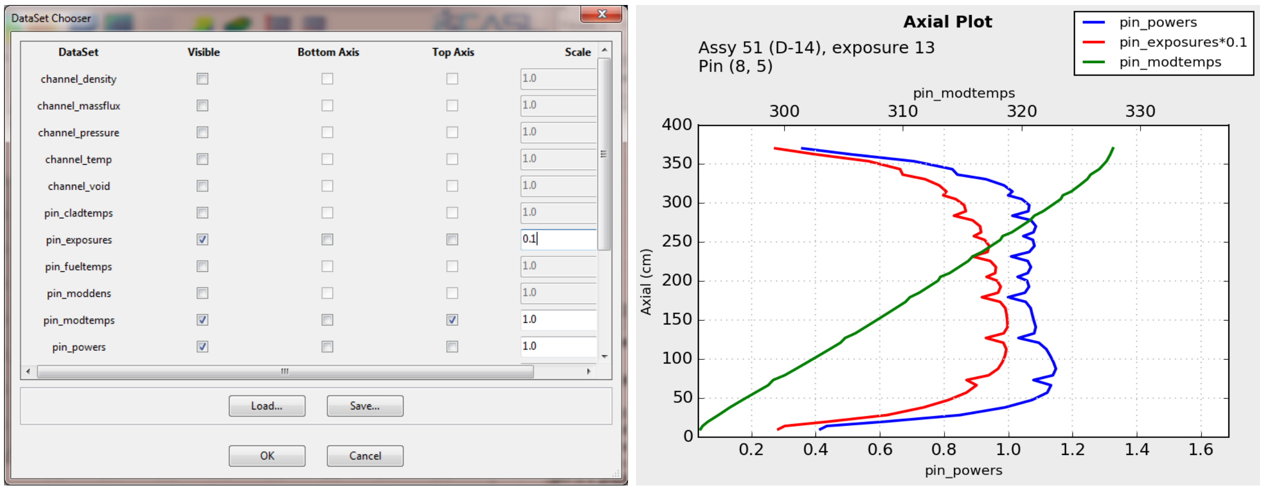


Figure 9. 1D axial pin data showing current pin and channel data Axial Plots widget



DataSet Chooser

DataSet	Visible	Bottom Axis	Top Axis	Scale
channel_density	☐	☐	☐	1.0
channel_massflux	☐	☐	☐	1.0
channel_pressure	☐	☐	☐	1.0
channel_temp	☐	☐	☐	1.0
channel_void	☐	☐	☐	1.0
pin_cladtemps	☐	☐	☐	1.0
pin_exposures	✓	☐	☐	0.1
pin_fueltemps	☐	☐	☐	1.0
pin_moddens	☐	☐	☐	1.0
pin_modtemps	✓	☐	✓	1.0
pin_powers	✓	☐	☐	1.0

Figure 10. Custom axial pin data (right) resulting from advanced dialog setting (left)

8.7 Time Plots

The time plot widget provides the capability to plot scalar data verses time on a chart. The widget is currently limited but will be greatly expanded in the future. Typical scalar data includes boron concentration, power level, k-effective, etc. By default the scalars are plotted against cycle exposure, if available. Otherwise, the user can choose alternate variables to use for the x-axis using **Edit → Select Time Dataset** on the Menu Bar, including exposure in effective full power days (EFPD), hours, and statepoint index.

A sample Time Plot is shown in Figure 11. This widget can also be used as a global statepoint selector but currently does not receive updates from global selection changes.

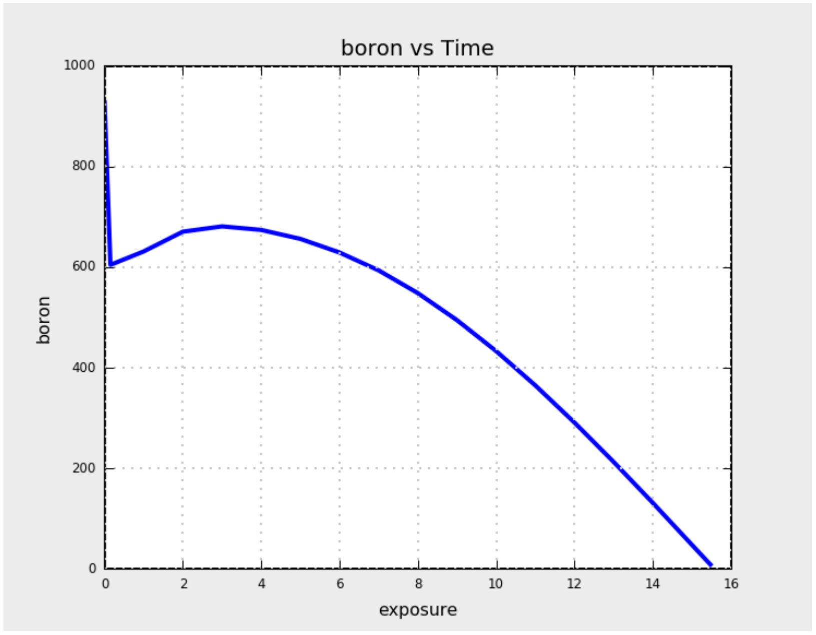


Figure 11. 1D Time plot for scalar data (vs. exposure)

8.8 Detector View

This widget, shown in Figure 12, provides 3D assembly-wise results in the form of 1D axial plots embedded in a 2D radial core arrangement. This is particularly designed for displaying in-core detector data, which are measured neutron flux traces in about 1/3rd of the fuel assemblies in some PWRs. Because CASL is using these traces for validation of its multi-physics models, this type of display provides a very useful way to perform comparisons.

In each fuel assembly location, the axial data is presented based on axial elevation and relative magnitude, similar to dozens of axial plots but without the details such as axis labels, etc. The rectangular border of each assembly is colored based on the 3D detector signal value at the current selected axial location. If the corresponding assembly element in dataset ***/STATE_000X/detector_operable*** is non-zero, then the widget grays out that assembly location and assumes the signal is bad for one reason or another.

The widget provides a global selection capability by assembly and updates on changes in statepoints and axial plane.

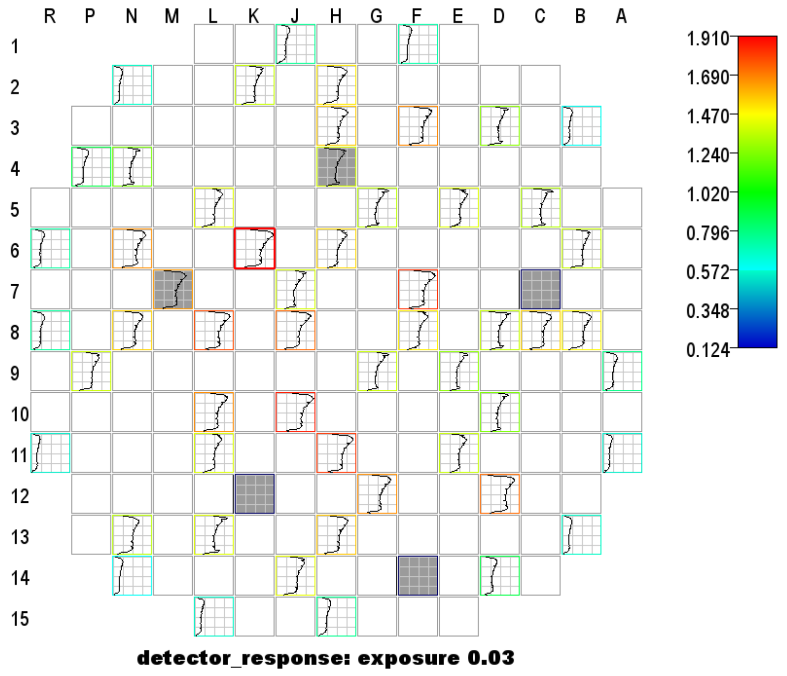


Figure 12. In-core detector data with 5 inoperable locations presented with the Detector View

8.9 Volume Slicer 3D View

The volume slicer provides a very unique capability for 3D visualization similar to that the "3-slice" view in VisIt or ParaView, but embedded within the VERAView Widget Grid. The user can position or spin the 3D image using the mouse, zoom in and out by holding the right mouse button, and change the location of the three planes by changing the global selected fuel rod coordination and axial planes. The Widget is not itself a selector but it does perform updates for changes in global coordinates, statepoint, and dataset. It also has its own embedded toolbar which allows some various viewing angles and permits a full screen mode (use ESC to exit).

Currently, this widget only functions with pin-wise datasets but the capability to view channel data will be available in the future. A sample image from the widget is shown in Figure 13.

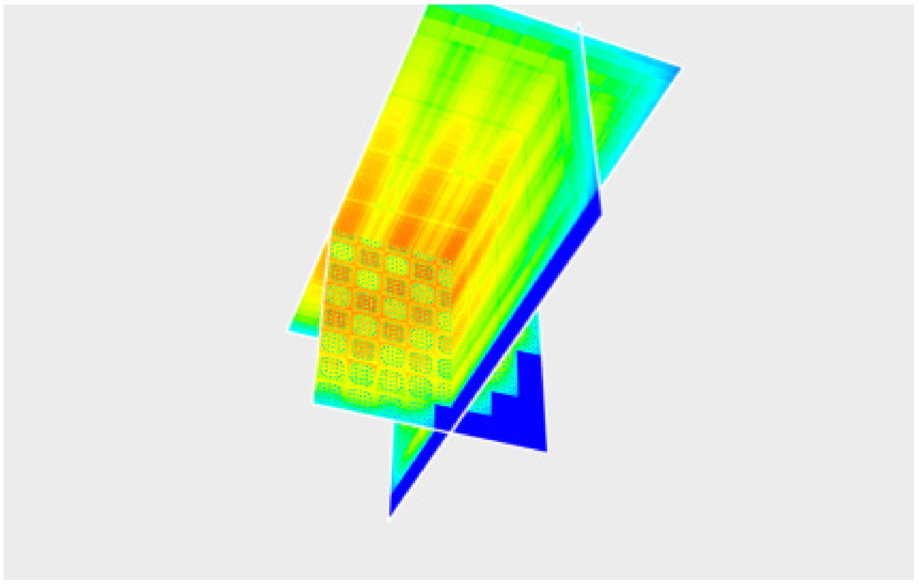


Figure 13. 3D visualization using Volume Slicer Widget

8.10 Volume 3D View

The 3D volume view provides another very unique capability for 3D visualization similar to what has previously only been available with VisIt or ParaView. The user can position or spin the 3D image using the mouse, and zoom in and out by holding the right mouse button. Currently there is no additional capability but this widget will be enhanced in the future. Like the 3D slider, the widget updates for changes in global statepoint, and selected pin dataset. It also has its own embedded toolbar which allows some various viewing angles and permits a full screen mode (use ESC to exit).

Currently, this widget only functions with pin-wise datasets but the capability to view channel data will be available in the future. A sample image from the widget is shown in Figure 14.

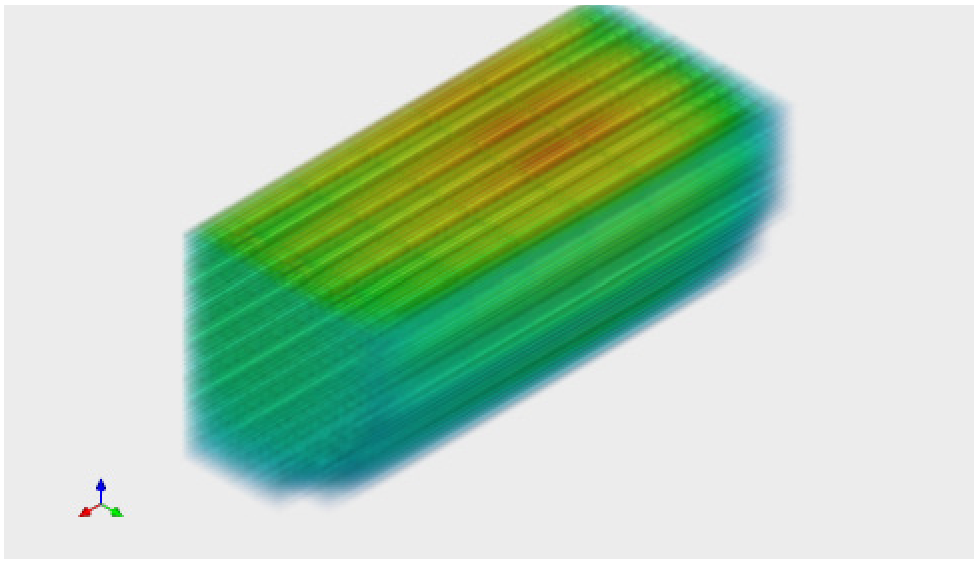


Figure 14. 3D visualization using Volume View Widget

9. SUMMARY

The python application VERAView has been quickly developed by CASL to enable multi-dimensional visualization, interaction, and engineering analyses of multi-physics results produced by VERA. The rapid prototyping development has demonstrated high efficiency, resulting in a high functioning tool for current use in the testing and validation of VERA in a very short development cycle. Design decisions have prioritized flexibility, extensibility, value to industry, and minimum investment. Ten modular mini-tools called widgets have already been created and embedded within VERAView and are at various stages of development, with the latest being the 3D views of the core data. Together this software significantly improves the user's ability to understand the large amount of data being produced by VERA, find potential problems or errors, and communicate those finding with others.

The VERAView tool is very easy to use and intuitive to inspect the reactor data. It opens up the VERAOut formatted HDF5 directly with no conversions and loads data relatively quickly. These features make this tool ideal for training and workshops when students need to focus on using the physics tools and interpreting results, not spend large amounts of time learning how to use the more advanced but generic visualization tools such as ParaView and VisIt. Most importantly VERAView is designed to support the work efforts of typical engineers, providing numeric results,1D line plots, tabular output to Microsoft Excel, and copy/paste images for emails and reports (like this one).

The use of python has allowed the tool to grow in capability very quickly, leveraging other libraries and plug-ins for visualization and reducing the amount of new code being written for each specific application. Python is also a simple language to learn and its use will increase the pool of students and engineers who are capable of enhancing the code and creating their own custom widgets for different applications.

Finally, VERAView development will continue to improve the current features and add additional widgets as the needs arise in CASL. Long term goals include visualization of physics models, the ability to compare multiple files, scripting capability, and multi-cycle capability. With the current design, the possible extensions are limitless.

10. REFERENCES

1. Godfrey, A., et. al., "VERAOut – VERA HDF5 Output Specification", CASL-U-2014-0043-001, Revision 1, CASL, May 2014. http://www.casl.gov/docs/CASL-U-2014-0043-001.pdf
2. http://www.paraview.org/
3. https://visit.llnl.gov/
4. http://wxpython.org/
5. https://www.wxwidgets.org/
6. http://www.h5py.org/
7. http://www.numpy.org/
8. https://python-pillow.github.io/
9. http://matplotlib.org/
10. https://www.enthought.com