

# VERAView User's Manual

June 13, 2022

Andrew Godfrey<sup>1</sup>, Ronald Lee<sup>1</sup>, Travis Lange<sup>2</sup>, and Erik Walker<sup>1</sup>

<sup>1</sup>Oak Ridge National Laboratory

<sup>2</sup>Idaho National Laboratory

**Approved for public release.  
Distribution is unlimited.**

## DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via US Department of Energy (DOE) SciTech Connect.

**Website** [www.osti.gov](http://www.osti.gov)

Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service  
5285 Port Royal Road  
Springfield, VA 22161  
**Telephone** 703-605-6000 (1-800-553-6847)  
**TDD** 703-487-4639  
**Fax** 703-605-6900  
**E-mail** [info@ntis.gov](mailto:info@ntis.gov)  
**Website** <http://classic.ntis.gov>

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information  
PO Box 62  
Oak Ridge, TN 37831  
**Telephone** 865-576-8401  
**Fax** 865-576-5728  
**E-mail** [reports@osti.gov](mailto:reports@osti.gov)  
**Website** <https://www.osti.gov/>

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.



## VERAVIEW USER'S MANUAL

Andrew Godfrey<sup>1</sup>, Ronald Lee<sup>1</sup>, Travis Lange<sup>2</sup>, and Erik Walker<sup>1</sup>

<sup>1</sup>Oak Ridge National Laboratory

<sup>2</sup>Idaho National Laboratory

Date Published: June 13, 2022

Prepared by  
OAK RIDGE NATIONAL LABORATORY  
Oak Ridge, TN 37831-6283  
managed by  
UT-Battelle, LLC  
for the  
US DEPARTMENT OF ENERGY  
under contract DE-AC05-00OR22725

# VERAView User's Manual

## Revision Log

Revision	Date	Affected Pages	Revision Description
0	6/13/2022	All	New template and minor editorial updates for VERA 4.3 release. This version supersedes previous document version CASL-U-2016-1058-003.

## Document pages that are:

Export Controlled:	None
IP/Proprietary/NDA Controlled:	None
Sensitive Controlled:	None
Unlimited:	All



# VERAView User's Manual

## Approvals:

*Erik Walker*

---

Erik Walker, VERAIO Product Software Manager

08/12/2022

---

Date

*Aaron Graham*

---

Aaron Graham, Independent Reviewer

08/12/2022

---

Date

## 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, 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

EXECUTIVE SUMMARY . . . . .	v
LIST OF FIGURES . . . . .	viii
LIST OF TABLES . . . . .	ix
ABBREVIATIONS . . . . .	x
1. OVERVIEW . . . . .	1
2. DESIGN PHILOSOPHY . . . . .	2
3. SYSTEM COMPONENTS . . . . .	3
4. DATA FORMAT . . . . .	4
5. INSTALLATION . . . . .	6
6. COMMON USER INTERACTIONS . . . . .	7
7. BASIC LAYOUT AND OPERATIONS . . . . .	8
7.1 Resizing the Widget Grid . . . . .	9
7.2 Widget Toolbar . . . . .	9
7.3 Controlling Global Selections . . . . .	10
7.4 Unlocking from Global Selections . . . . .	11
8. SELECTING DATASETS . . . . .	12
9. CURRENT WIDGETS . . . . .	14
9.1 Core 2D View . . . . .	14
9.2 Assembly 2D View . . . . .	15
9.3 Core Axial 2D View . . . . .	16
9.4 Axial Plots . . . . .	18
9.5 Time Plots . . . . .	20
9.6 Detector View . . . . .	20
9.7 Vessel Core 2D View . . . . .	22
9.8 Vessel Core Axial 2D View . . . . .	22
9.9 Table View . . . . .	23
9.10 Volume Slicer 3D View . . . . .	23
9.11 Volume 3D View . . . . .	24
9.12 Common Widget Operations . . . . .	25
9.13 Clipboard Copy Operations . . . . .	25
9.14 Dataset Operations . . . . .	26
9.15 Display Operations . . . . .	28
9.16 Image Save Operations . . . . .	29
9.17 Show in New Window . . . . .	29
10. Main Menu Operations . . . . .	30
10.1 File Menu . . . . .	30
10.2 Edit Menu . . . . .	32
10.3 Data Menu . . . . .	33
10.4 Window Menu . . . . .	37
11. CREATING DERIVED DATASETS . . . . .	38
12. CREATING DIFFERENCE DATASETS . . . . .	39
13. OPENING MULTIPLE FILES . . . . .	40
13.1 Congruence . . . . .	40
13.2 Resolving Time and Axial Levels . . . . .	40
14. SUMMARY . . . . .	42
15. ACKNOWLEDGEMENTS . . . . .	43

REFERENCES . . . . .	44
----------------------	----

## LIST OF FIGURES

1	General Layout and Widget Grid with 2x2 Widgets . . . . .	8
2	Resize the Widget Grid (3x2 Selection) . . . . .	9
3	Widget Toolbar . . . . .	10
4	Sample Fuel Lattice Locked (Right) and Unlocked (Left) from Global Axial Selections . . .	11
5	Dataset Selection Menu . . . . .	12
6	Multifile Dataset Menu . . . . .	12
7	Set Visible Datasets Dialog . . . . .	13
8	Pin-wise (Left) and Assembly-wise (Right) Data in the Core 2D View . . . . .	15
9	Channel-wise Data as Nominal (Left) and Zoomed (Right) in the Core 2D View . . . . .	15
10	2D Fuel Lattice Data Nominal (Left) and Zoomed (Right) in the Assembly 2D View . . . .	16
11	Coolant Channel Data Nominal (Left) and Zoomed (Right) in the Channel Assembly 2D View	16
12	Lattice View with Secondary Pin Selections . . . . .	17
13	2D Axial Pin Data Showing X Axis (Left) and Y Axis (Right) in the Core Axial 2D View . .	17
14	1D Axial Plot Showing Current Pin and Channel Data . . . . .	18
15	DataSet Plot Properties . . . . .	19
16	Axial Plot With Secondary Pin Selections . . . . .	19
17	1D Time Plot for Scalar Data (vs. Exposure) . . . . .	20
18	In-core Detector Data with 5 Inoperable Locations Presented with the Detector View . . . .	21
19	Detector Widget Numeric Display . . . . .	21
20	Vessel Core 2D View Showing Binned Fluence Tallies . . . . .	22
21	Vessel Core Axial 2D View . . . . .	23
22	Table View Widget . . . . .	24
23	3D Visualization Using Volume Slicer Widget . . . . .	24
24	3D Visualization Using Volume View Widget . . . . .	25
25	Widget Menus . . . . .	25
26	Custom Data Scale Dialog . . . . .	26
27	Scale options . . . . .	27
28	Colormap options . . . . .	27
29	Dataset Properties for Axial (Left) and Time (Right) Plots . . . . .	28
30	File Menu . . . . .	30
31	File Manager Dialog . . . . .	31
32	Message for Incongruent File . . . . .	31
33	Edit Menu With Pullrights . . . . .	32
34	Select Scale Mode . . . . .	33
35	Data Menu . . . . .	34
36	Create Derived Datasets Dialog . . . . .	34
37	Derived Dataset Methods . . . . .	34
38	Difference Dataset Dialog . . . . .	35
39	Difference Dataset Calculation . . . . .	35
40	Difference Dataset in Menu . . . . .	36
41	Manage Thresholds Selections . . . . .	36
42	Threshold Filter . . . . .	37
43	Widget With Threshold Applied . . . . .	37
44	Derived Pullright Menu . . . . .	38
45	Menus After Deriving a Dataset . . . . .	38

## LIST OF TABLES

1	Current Python Module Versions . . . . .	3
2	Dataset Categories/Types . . . . .	5

## ABBREVIATIONS

API	application programming interface
CASL	Consortium for Advanced Simulation of Light Water Reactors
CSV	comma separated values
EFPD	effective full power days
GIF	Graphics Interchange Format
GUI	graphical user interface
HDF	Hierarchical Data Format
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
PWR	pressurized water reactor
VERA	Virtual Environment for Reactor Applications
VERA-CS	VERA Core Simulator
VTK	Visualization Toolkit

## 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 [1] and VisIt [2] 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 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 Core Simulator (VERA-CS) output. Its primary requirements from the early stages were ease-of-use, flexibility, and maintainability. It directly reads the VERA Output HDF5 file specification format [3], 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 intuitive and can be interrogated using typical modern GUI interactions, such as left and right mouse clicks, mouse rollovers, 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.

The VERAView Code (beginning with version 2.1a11 and up) shall meet the following software requirements:

1. Open a single file and multiple files simultaneously
2. Demonstrate an ability to display values and plot values in multiple categories
3. Calculate and display value differences between files
4. Exporting data related to specific areas of focus into multiple different file formats

Images used in this manual were generated using VERAView versions starting with 2.1a11, released in early May 2017, through version 2.4.4, released at the VERA Users Group training in February 2019.

For specific questions about the use of VERAView, the licensing of the code, or to report bugs users are encouraged to send an email to vera-support@ornl.gov. When reporting bugs, users are requested to attach the problematic input to the email and to provide information in the body of the email about the code version, machine, runtime environment and any other relevant details to permit debugging.



## 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 VERA Output Specification [3] defines 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 VERA Output 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 VERA Output 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. 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 and minimum values have already been implemented, as have capabilities for displaying differences between files.
- VERAView allows inspection of data in multiple dimensions and visualization by the user in different perspectives in order 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 and quickly 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 has been the language and runtime environment for VERAView through version 2.4.5. Specifically, Continuum’s cross-platform Anaconda2 [4] environment has been chosen for development, testing, and distribution with VERAView.<sup>1</sup> Through each prototype iteration VERAView has been tested under Windows, Mac OS X, and Linux. Anaconda2 provides a consistent and stable environment across the three platforms. Several Python modules are required by VERAView. VERAView installers for each platform include the required Python runtime environment and packages as shown in Table 1. Note these are top level packages which in turn have other dependent packages.

**Table 1. Current Python Module Versions**

Module	VERAView Version <sup>a</sup>
h5py	2.5.0
matplotlib	1.4.3
mayavi	4.4.0
numpy	1.9.3
Pillow	4.2.1
pyparsing	2.0.3
Scipy	0.17.1
wxPython	3.0

<sup>a</sup>Anaconda module version numbers

The wxPython [5] module was chosen for the windowing toolkit. It is a Python wrapper around the wxWidgets [6] cross-platform GUI library based on native components. At present, all 2D plots are created using matplotlib [7].

The h5py module [8] is a Python wrapper around the Hierarchical Data Format (HDF) 5 C application programming interface (API) and library. It is used for all processing of VERA Output files. Since h5py uses numpy [9], the latter is used for all dataset vector/array manipulations.

Mayavi [10] is used for 3D visualizations. It is an open source capability maintained by Enthought that uses the Visualization Toolkit (VTK) Python wrapper.

Additional details for the system components and environment are documented in chapter 1 of the *VERAView Programmer’s Guide*.

---

<sup>1</sup>A port to Python-3 is underway, and some future version of VERAView will use the Anaconda-3 environment.

## 4. DATA FORMAT

The data files which can be interpreted by VERAView must conform to the VERA Output Specification described in Reference [3]. 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
- NDETAX – Number of detector axial planes
- NFDETAX – Number of fixed detector axial planes

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(4,NAX, NASS): 3D array representing nodal 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 “derived” 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. Datasets can have an attribute (named “factor”) specifying the `/CORE` dataset to be used as factors for the dataset. Derived datasets are created on the “Select Dataset to View” toolbar menu 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. Table 2 lists the dataset types or categories recognized by VERAView. Additional details on the data management are documented in chapter 3 of the *VERAView Programmer’s Guide*.

**Table 2. Dataset Categories/Types**

Category/Type	Shape	Derivable
channel	(nchany, nchanx, nax, nass)	n
detector	(ndetax, ndet)	n
fixed_detector	(nfdetax, ndet)	n
fluence <sup>a</sup>	(fluence.nz, fluence.ntheta, fluence.nr)	n
pin	(npiny, npinx, nax, nass)	n
radial_detector	(ndet)	n
scalar	() or (1, )	n
:assembly	(1, 1, nax, nass)	Y
:axial	(1, 1, nax, 1)	Y
:chan_radial	(nchany, nchanx, 1, nass)	Y
:core	(1, 1, 1, 1)	Y
:node	(1, 4, nax, nass)	Y
:radial	(npiny, npinx, 1, nass)	Y
:radial_assembly	(1, 1, 1, nass)	Y
:radial_node	(1, 4, 1, nass)	Y

<sup>a</sup>Only available in VERA Output files with fluence outputs.

Note the fluence values are as follows:

NZ – Number of axial levels

NTHETA – Number of angles

NR – Number of radii

## 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 Continuum's Anaconda2 product [4], a free Python development and runtime environment available for Windows, Mac OS X, and Linux. Anaconda2 and the modules required by VERAView can be downloaded and installed without administrative privileges.

The VERAView GitHub site, <https://github.com/CASL/VERAview/>, includes instructions for creating the VERAView runtime environment. Moreover, single-click installers for Windows and Mac OS X and a self-contained install script for Linux are available at <https://kepler.ornl.gov/casl/>, with links on the GitHub project page. Information on availability is provided on the GitHub site. The single-click installers will include a Python runtime environment with required modules bundled with the VERAView installation.

For user support or to receive updates about new VERAView versions and bug fixes please contact [vera-support@ornl.gov](mailto:vera-support@ornl.gov). Because of the current development stage and rapid prototyping, new versions and bug fixes are released often, sometimes weekly. However, single-click installers are only created for relatively stable and well-tested versions.

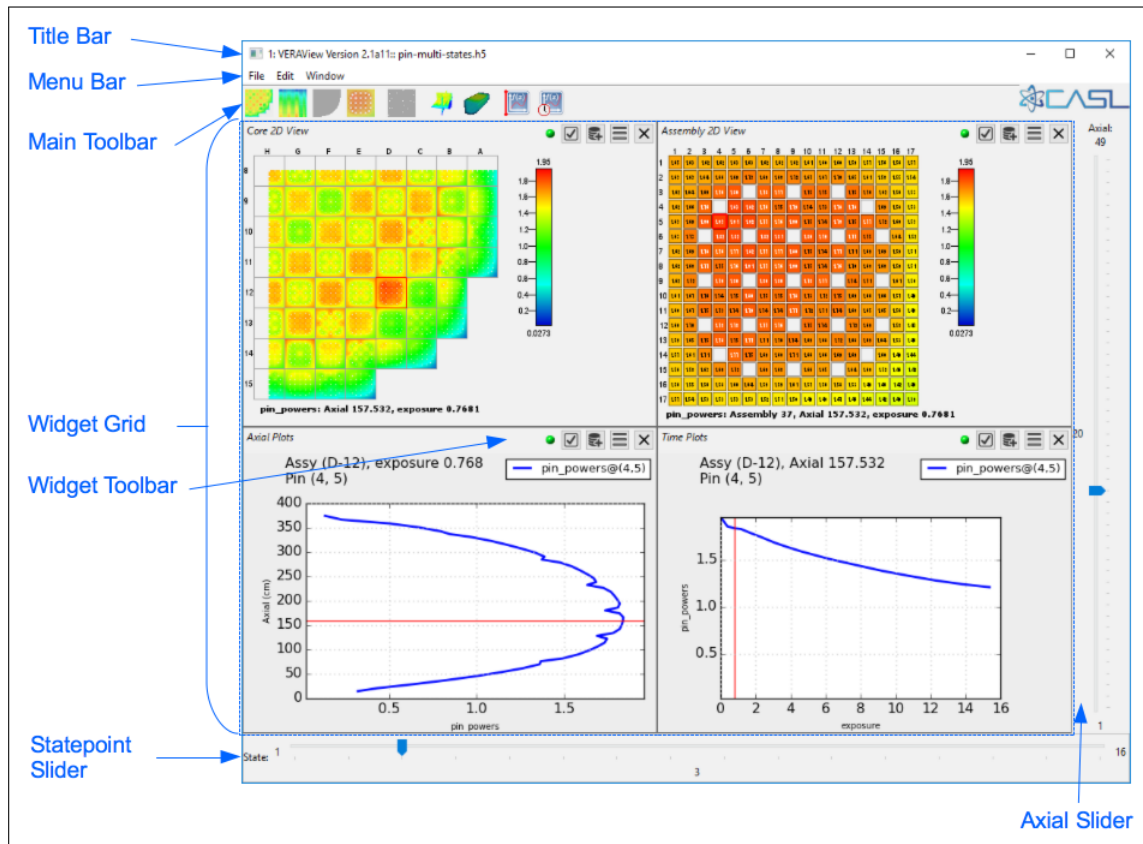
## 6. COMMON USER INTERACTIONS

VERAView is a GUI designed with common Python tools and 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 window
  - Select/change processing options
- VERAView can be minimized, maximized, or closed like most windows by using the buttons in the title bar.
- There is a main toolbar in each window 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 clipboard 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.)
- Color schemes defined by matplotlib for representing the data can be selected for each widget from the widget menu. By default, the data range covers the entire dataset and all statepoints (i.e. the maximum value occurs typically at one place in the entire file/simulation).
- A custom data range or scale can be defined in some widgets.
- Units are displayed in the legend if specified in the HDF5 data at this time. VERA Output files produced by more recent VERA versions are more likely to include the units.

## 7. BASIC LAYOUT AND OPERATIONS

Figure 1 illustrates 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 more than one assembly is represented in the data, an **Assembly 2D View** widget will open up automatically. A **Time Plots** widget is opened if there is more than one statepoint in the file, and an **Axial Plots** widget opens if there is more than one axial level in the data. More details are provided below. An in-depth discussion of the widget properties and framework are documented in chapter 4 of the *VERAVIEW Programmer's Guide*.



**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 accomplished with File New on the Menu Bar)
- The *Widget Grid* is an 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 column or row will be added to the grid, and the new widget will be placed in the first column of a new row. Currently there is no capability to move widgets from one grid location to another, but a widget may be 'popped out' into a separate window via the Show in New Window option on the widget menu.

- 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 level 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 **X** icon button (rightmost) 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 (also available via the **<Control>-G** keyboard shortcut<sup>2</sup>). This produces a dialog as shown in Figure 2, where the user can select the desired grid size and layout (shown in red).

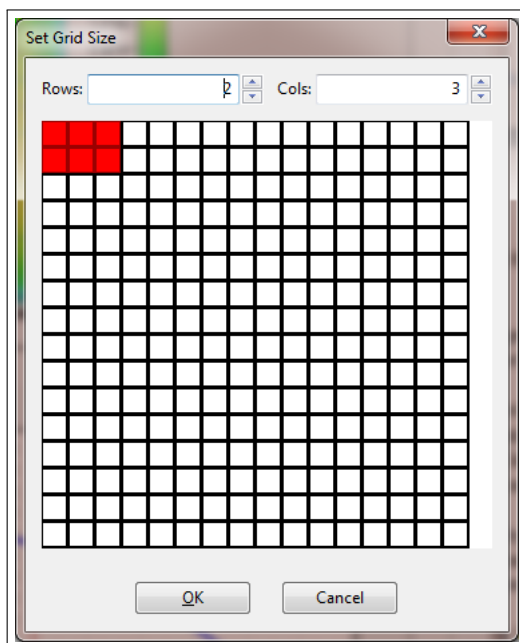


Figure 2. Resize the Widget Grid (3x2 Selection)

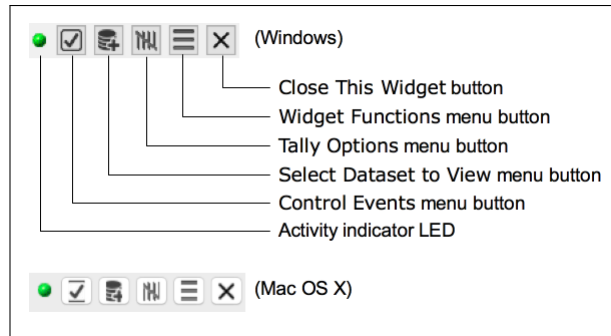
## 7.2 WIDGET TOOLBAR

Figure 3 shows a widget toolbar as displayed under Windows and Mac OS X with a description of each button. Note some buttons are not applicable to all widgets.

Most buttons have an associated popup menu.

<sup>2</sup>For the Mac the key combination is **<Cmd>-G**.





**Figure 3. Widget Toolbar**

### 7.3 CONTROLLING GLOBAL SELECTIONS

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

- Currently selected dataset
- Currently selected fuel rod (pin) / coolant channel
- Currently selected fuel assembly
- Currently selected axial plane
- Currently selected time (maps to a statepoint for each file)
- Currently selected detector dataset (if applicable)
- Currently selected node (applicable to nodal datasets)
- Current set of secondary fuel rod / coolant channel selections
- Current set of secondary node selections (applicable to nodal datasets)

In general, each widget can send and receive events 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 Plots* 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 locations (such as *Axial Plots* or *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. The same is true for assembly data. The datasets themselves are only selected via the *Widget Toolbars* (or *Edit* → *Select Dataset* on the main menu bar) 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 communicating the location of the data it is presenting, whether it is the global selection or not.

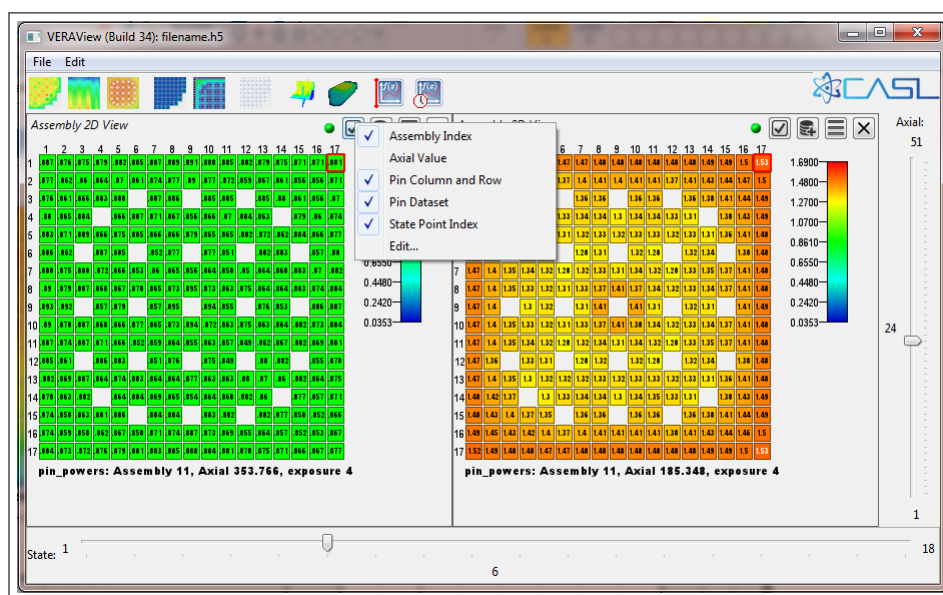
## 7.4 UNLOCKING FROM GLOBAL SELECTIONS

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, after which the widget communicates the change 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. On the *Widget Toolbar* menu for most widgets there is a button for unlocking individual selections and an Edit item for locking/unlocking multiple selections. By default, all the control events are checked and the widget is fully locked.

For example, if the user unchecks Statepoint/Time, that widget will no longer update when the global time is changed, and if that widget can change its own statepoint, such changes 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 Axial Value. Further changes in the globally selected axial plane will no longer update the “unlocked” widget.

Figure 4 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.

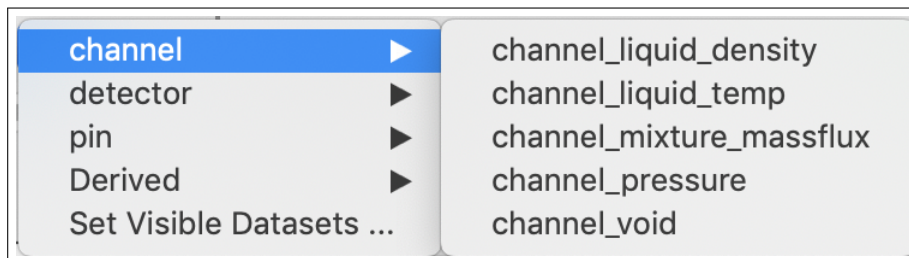


**Figure 4. Sample Fuel Lattice Locked (Right) and Unlocked (Left) from Global Axial Selections**

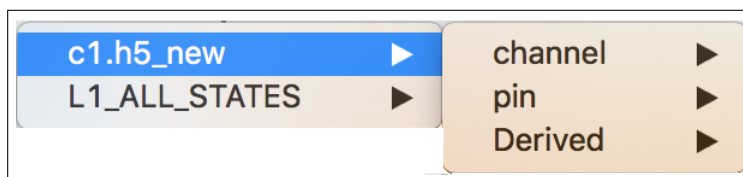
A final example of unlocking is if a user wants a particular widget to display a particular dataset no matter what other datasets are selected in the application. In this case, the user can simply unlock that widget by unchecking Selected Dataset and then select a different dataset without impacting any other widgets.

## 8. SELECTING DATASETS

VERAView widgets will display data for any valid dataset in a VERA Output file. Although the core and assembly widgets can display only a single dataset at a time, plot widgets can show multiple datasets simultaneously. Each widget has a **Select Dataset** button on its toolbar that pops up a dataset selection menu. If a single file is open, the menu has pullrights for each category or type of datasets available; refer to Figure 5. If multiple files have been opened (refer to multi-file capability description below), the dataset menu has pullright items for each open file, as shown in Figure 6.



**Figure 5. Dataset Selection Menu**



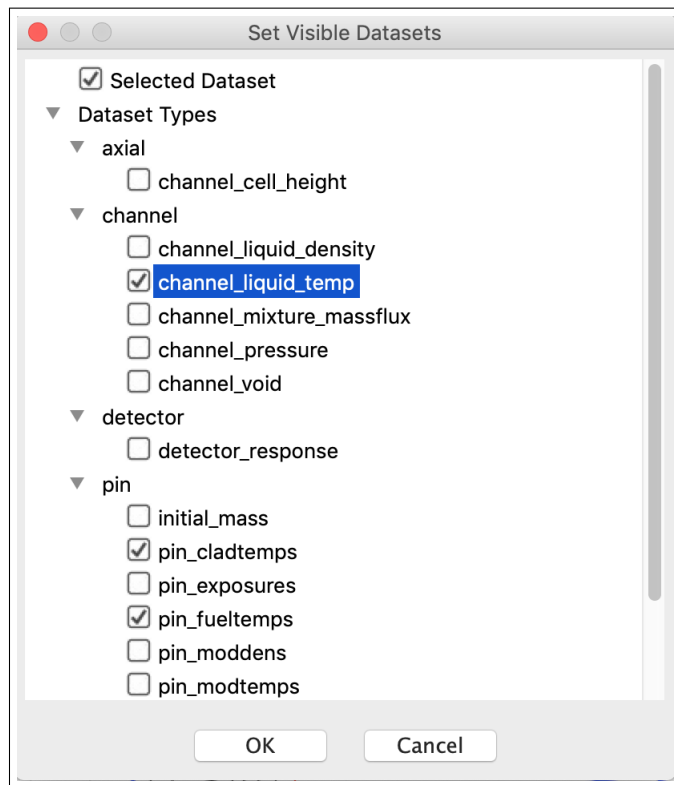
**Figure 6. Multifile Dataset Menu**

For widgets that display multiple datasets, the visibility of a dataset is indicated by a check in the menu item. For single-dataset widgets, the item checked represents the currently displayed dataset.

Note also that the current dataset can be selected from a widget **Select Dataset** toolbar button or via **Edit** → **Select Dataset** on the window menu bar (application menu bar for Mac OS X).

Widgets capable of displaying multiple datasets will have an additional item **Set Visible Datasets...** item at the bottom of the dataset selection menu. Selecting this item brings up a **Set Visible Datasets** dialog with a tree control (refer to Figure 7.)

The tree contains nodes for each dataset category or type with individual datasets listed as leaves. If multiple files are open, the first level of nodes represent each respective file. Multiple datasets can be toggled on/off by checking or unchecking the checkboxes.



**Figure 7. Set Visible Datasets Dialog**

## 9. 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, thereby users and developers to extend functionality to new applications and concepts.

Each widget is developed to know what categories of data it can display (Section 8) 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 Core 2D View 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. Most widgets also provide a means for the user to make a “current” selection, such as the current assembly, pin/channel, axial level, time, and so on.

This section summarizes 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. Additional details documenting the underlying framework for various user-widget interactions are documented in the *VERAView Programmer’s Guide*. Standardized tests are documented the *VERAView Software Requirements, Test Plan, and Test Report* and serve as a good walk-through introduction to basic capabilities in VERAView.

The following widgets not a basic capability and are not currently being tested in the *VERAView Software Requirements, Test Plan, and Test Report* document:

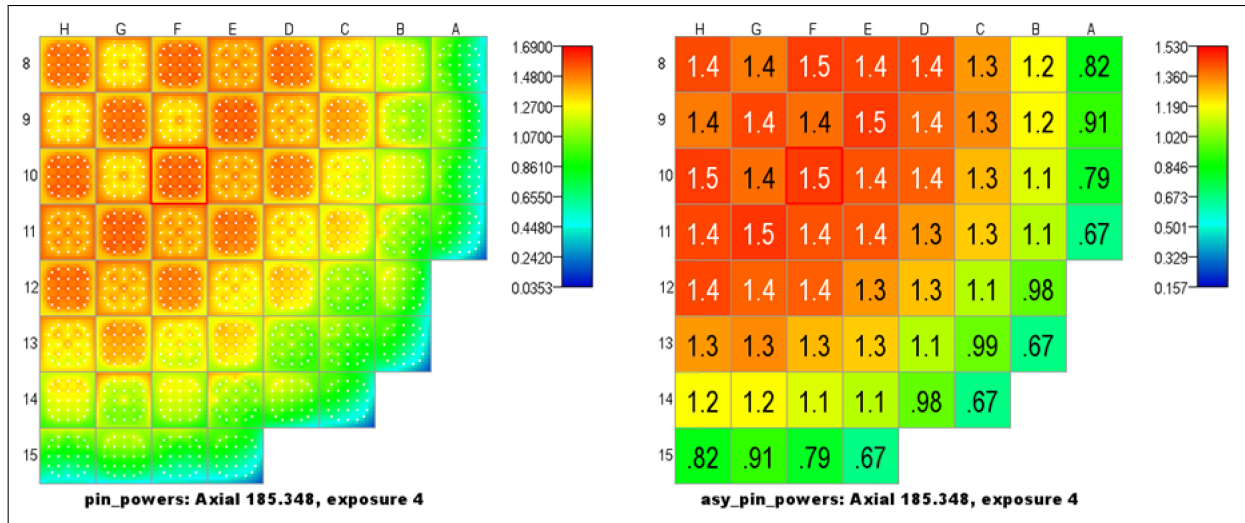
- Detector View
- Vessel Core 2D view
- Vessel Core Axial 2D View
- Table View
- Volume Slicer 3D View
- Volume 3D View

### 9.1 CORE 2D VIEW

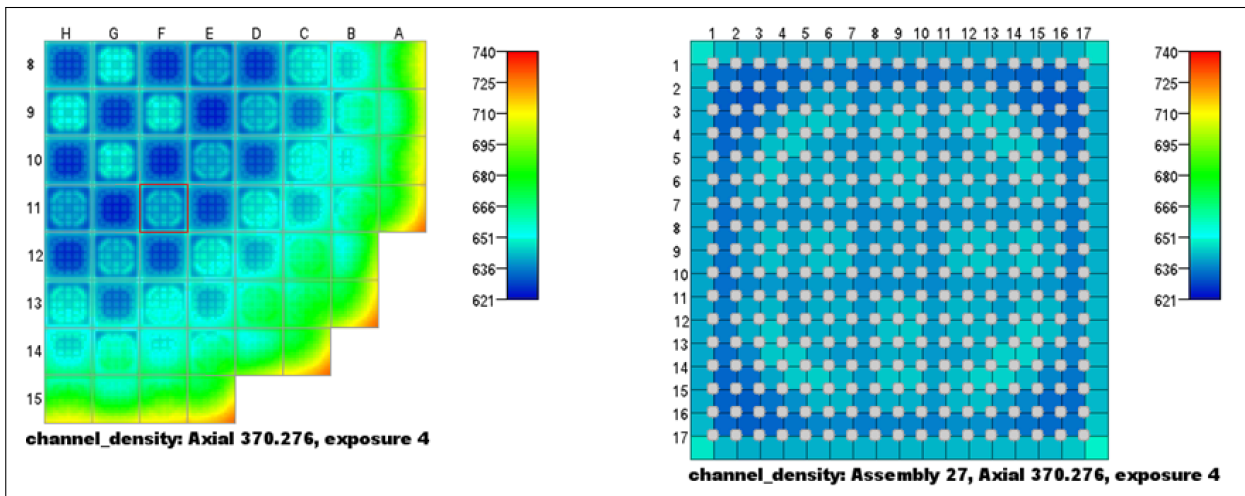
This widget, shown in Figure 8, provides a radial view of channel-wise, pin-wise, or assembly-wise distributions for the entire core for a specified axial location and statepoint. For quarter-core simulations, only that quarter is shown (values across the symmetry line are mirror reflected). For pin 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 8). This widget is capable of selecting global assembly and pin/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.

With this widget a user may select the current assembly by left-clicking a displayed assembly. The currently selected assembly is indicated with a red outline box. When nodal data are displayed, the current node is selected by clicking within an assembly. Secondary nodal selections are made by pressing **<Control>-<Shift>** while clicking.<sup>3</sup> The primary node is indicated with a red outline box, and secondary nodes have a yellow outline. Figure 9 illustrates display of channel data.

<sup>3</sup>For Mac OS X this is the **<Cmd>-<Shift>** key combination.



**Figure 8. Pin-wise (Left) and Assembly-wise (Right) Data in the Core 2D View**



**Figure 9. Channel-wise Data as Nominal (Left) and Zoomed (Right) in the Core 2D View**

## 9.2 ASSEMBLY 2D VIEW

This widget, shown in Figure 10, provides a radial view of pin-wise distributions or channel-wise distributions for a single assembly at a specified axial location and statepoint. The dataset values are shown in each fuel rod or channel location. This widget is capable of selecting primary and secondary global pin/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 rods/channels of interest.

Figure 11 illustrates display of channel data. Pins are indicated by small circles between channels. The display of pins can be toggled on/off via the Show/Hide Channel Data Pins item of the widget menu.

Similar to the Core 2D View, the primary pin/channel currently selected is indicated with a red outline box. Secondary pin/channel selections have a yellow outline. Left-clicking any pin/channel will select it as primary and will clear the list of secondary selections. Left-clicking a pin/channel while pressing



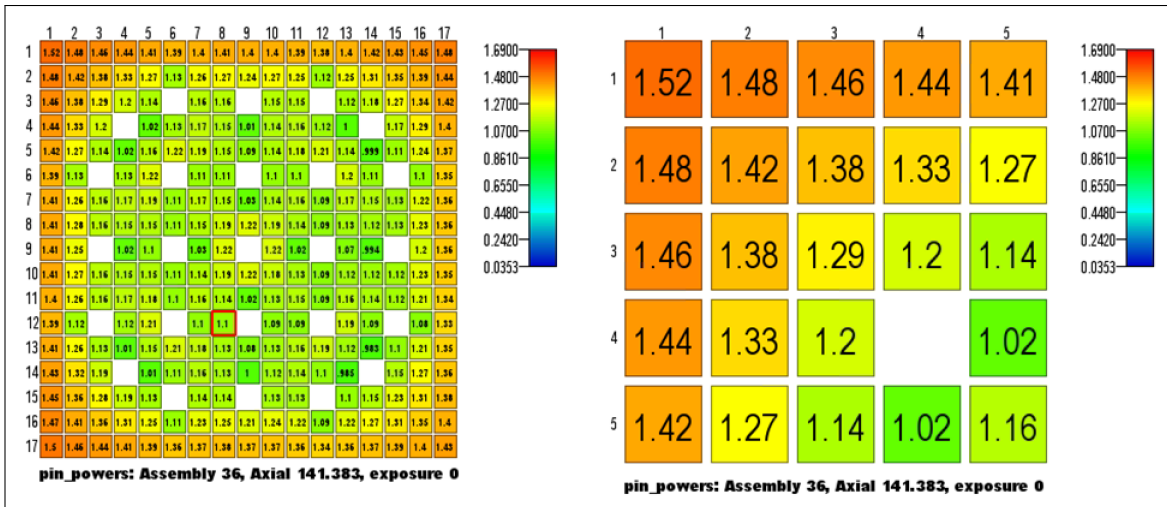


Figure 10. 2D Fuel Lattice Data Nominal (Left) and Zoomed (Right) in the Assembly 2D View

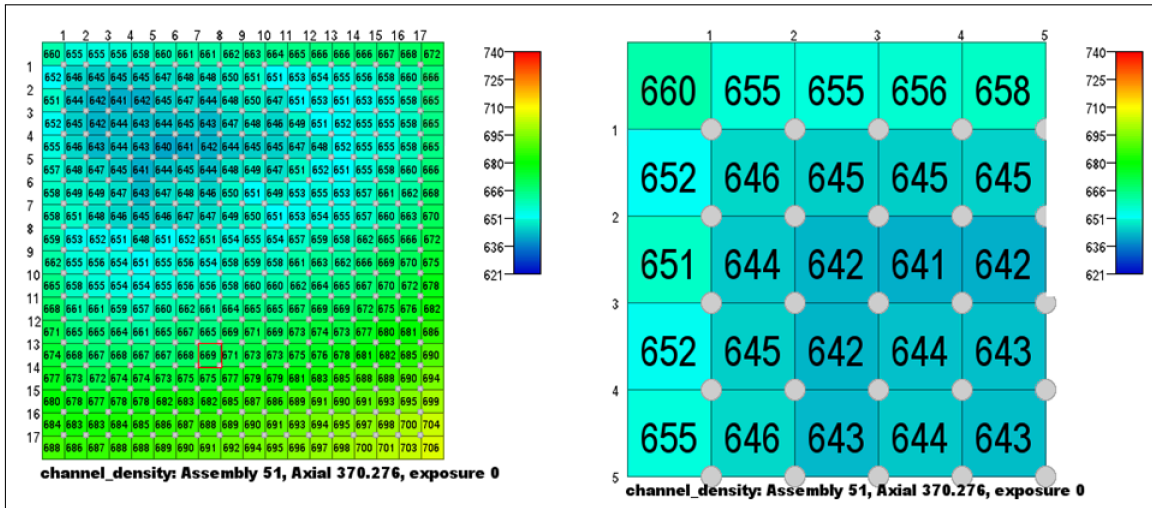


Figure 11. Coolant Channel Data Nominal (Left) and Zoomed (Right) in the Channel Assembly 2D View

<Control>-<Shift> adds it to the list of secondary selections. Figure 12 shows an Assembly 2D View widget with secondary pin selections.

### 9.3 CORE AXIAL 2D VIEW

This widget, shown in Figure 13, 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.

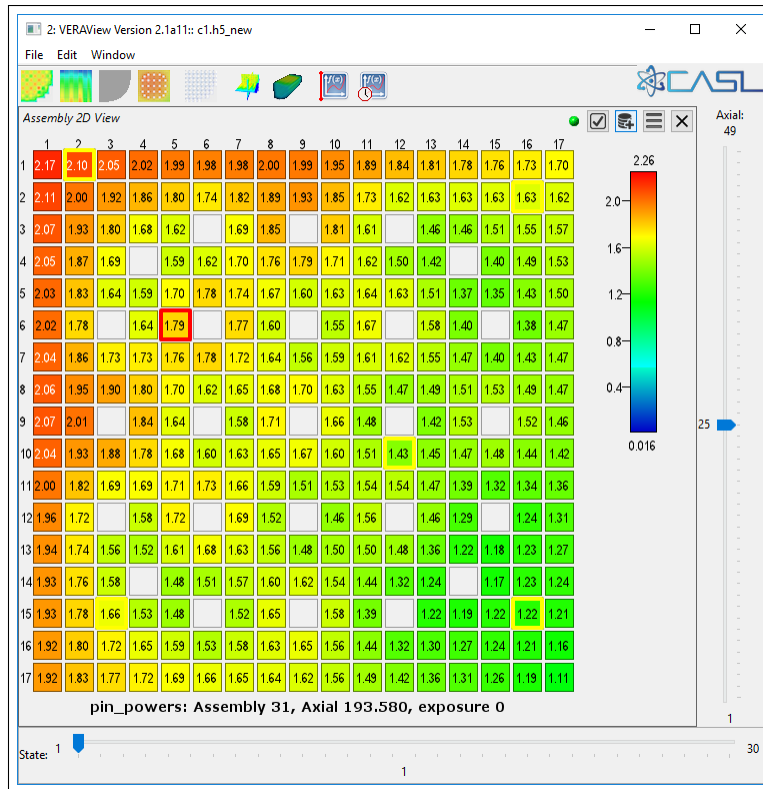


Figure 12. Lattice View with Secondary Pin Selections

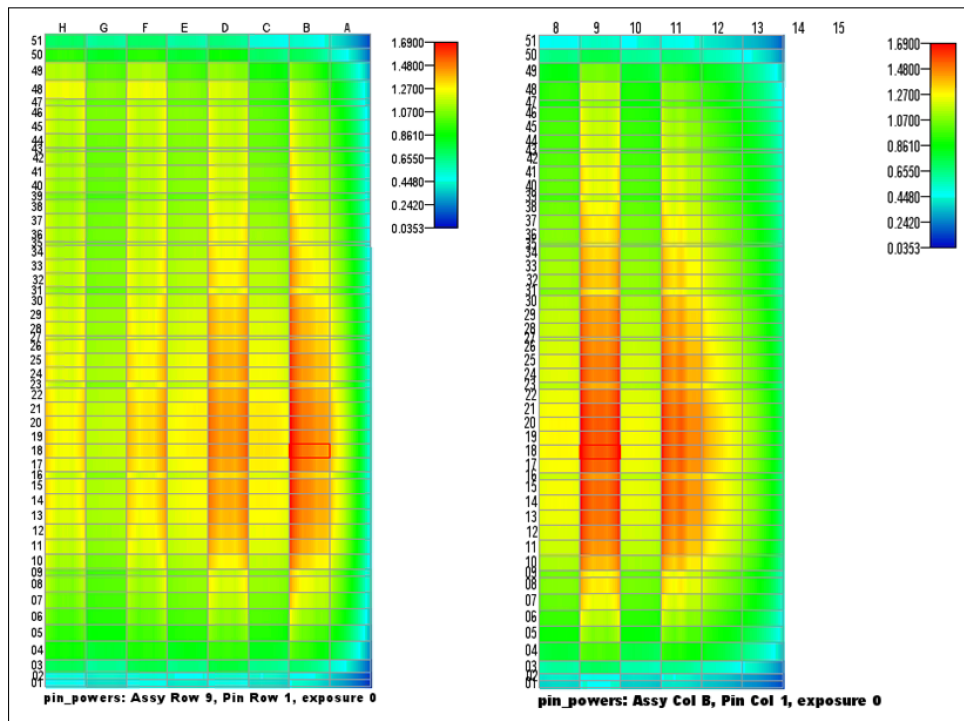


Figure 13. 2D Axial Pin Data Showing X Axis (Left) and Y Axis (Right) in the Core Axial 2D View



The user can toggle whether the X or Y plane is shown by clicking the **Toggle Slice to X/Y-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). This widget also supports channel data.

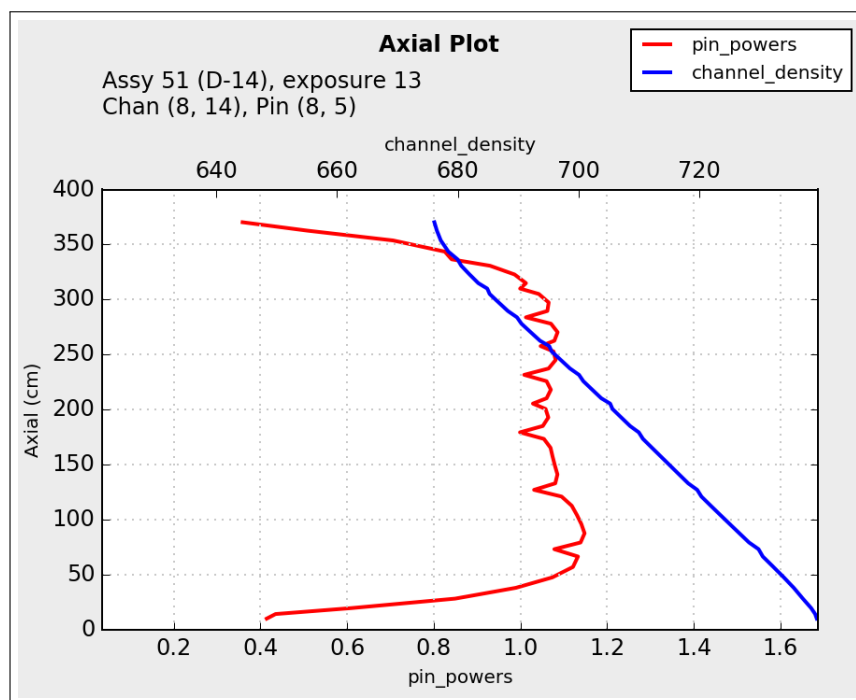
When an assembly or node is left-clicked in this widget, two values are selected:

- The current assembly corresponding to the column (Y-axis view) or row (X-axis view) displayed horizontally,
- The current axial level corresponding to the row

## 9.4 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 dataset. As additional datasets are toggled visible in the plot, they are added to the bottom axis. Figure 14 shows two datasets plotted, one using each respective axis. Individual datasets are toggled on/off via the dataset menu (refer to Section 8).



**Figure 14. 1D Axial Plot Showing Current Pin and Channel Data**

As the mouse is moved within the widget, a dotted black line representing the axial level will track the mouse. A left-click will select the axial level. The current axial level is represented with a horizontal, solid red line.

### 9.4.1 Axis Assignment

Datasets can be assigned to the bottom or top axis individually via the **Edit Dataset Properties** item of the widget menu. As shown in Figure 15, each currently visible dataset is shown with radio buttons for selecting the top or bottom axis and a scale factor. The scale factor can prove useful in bringing a dataset's value range in line with other datasets displayed on the same axis. 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.

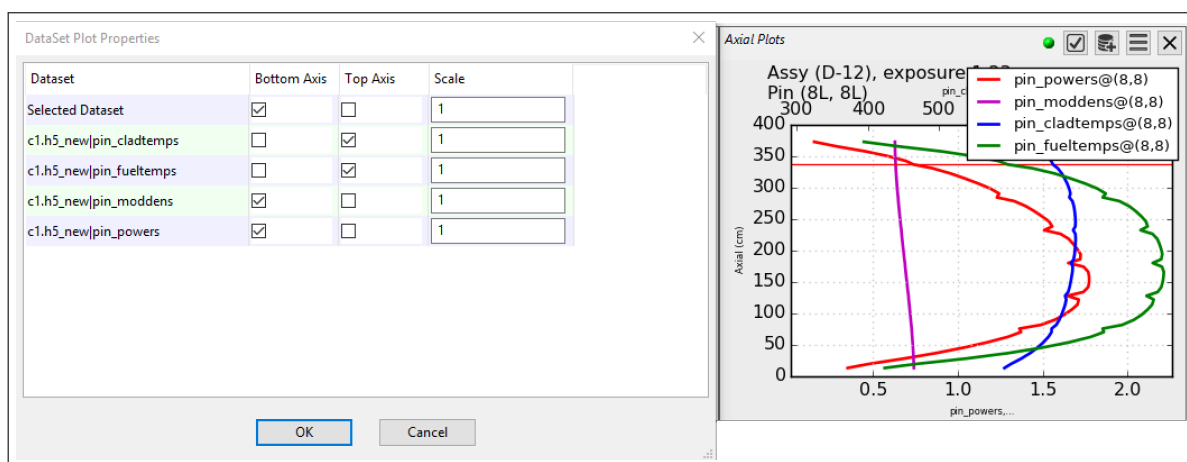


Figure 15. DataSet Plot Properties

### 9.4.2 Secondary Selections

Another feature of the **Axial Plots** widget is the ability to plot dataset values at secondary pin/channel selections from an **Assembly 2D View** widget. Figure 16 shows an example of such a plot.

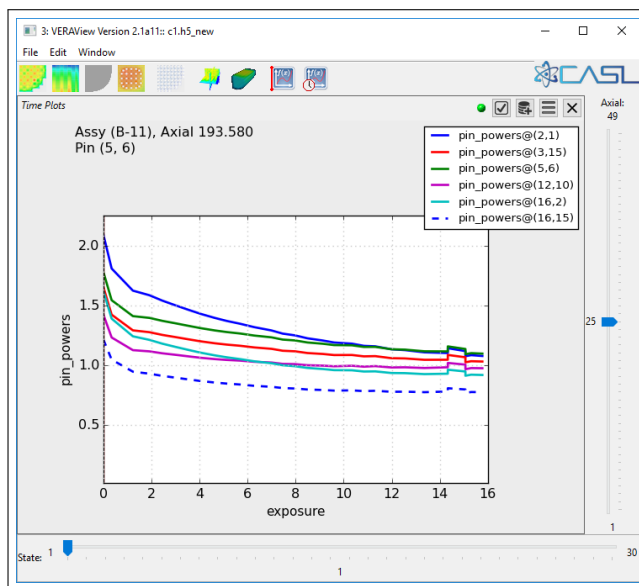
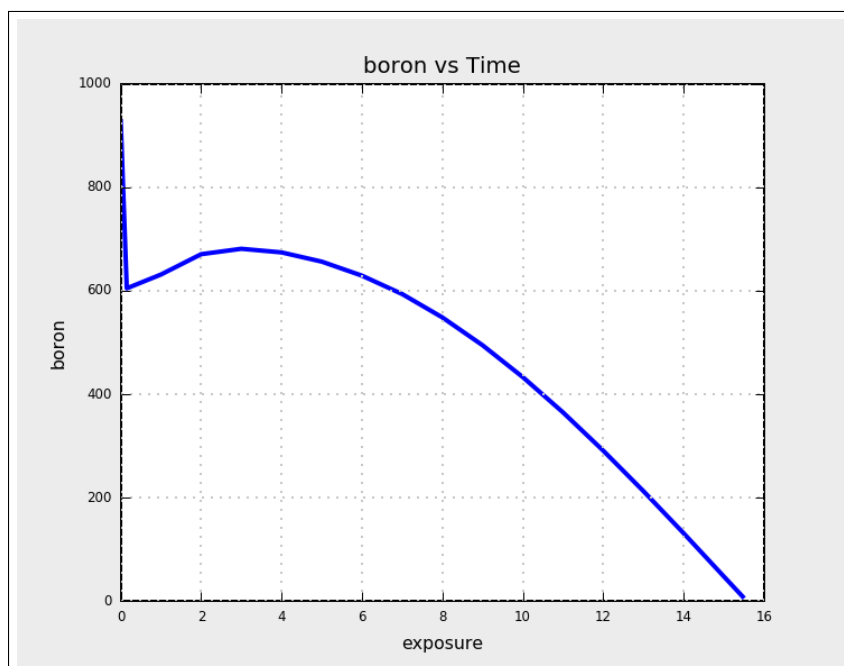


Figure 16. Axial Plot With Secondary Pin Selections

## 9.5 TIME PLOTS

The time plot widget provides the capability to plot scalar data versus 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 Plots widget is shown in Figure 17. This widget can also be used as a global statepoint selector. As the mouse moves within this widget, a dotted black line representing the time tracks the mouse. The currently selected time is represented with a vertical, solid red line. Like the Axial Plot widget, the **Edit Dataset Properties** item of the widget menu allows assignment of individual datasets to the left and right axis.



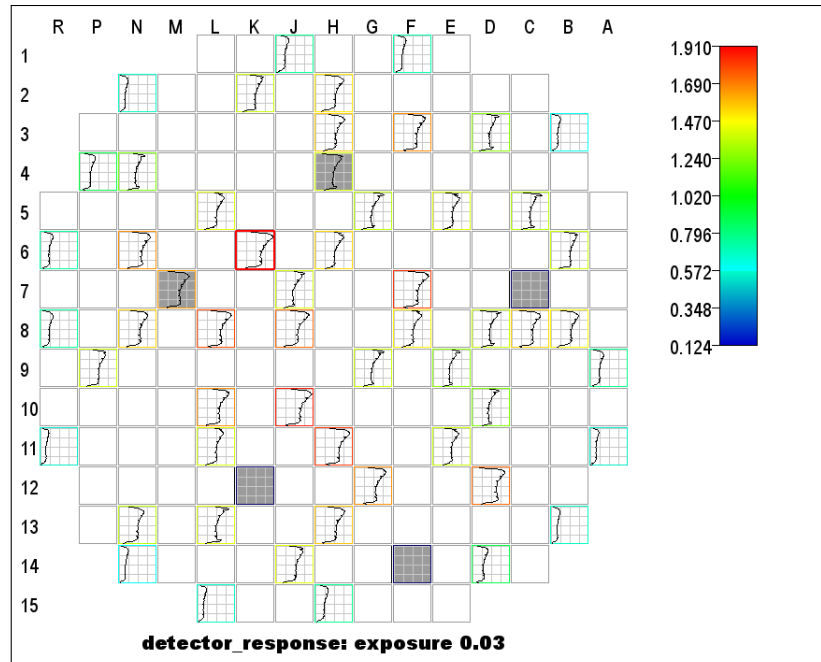
**Figure 17. 1D Time Plot for Scalar Data (vs. Exposure)**

## 9.6 DETECTOR VIEW



This widget, shown in Figure 18, 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 1/3<sup>rd</sup> 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.

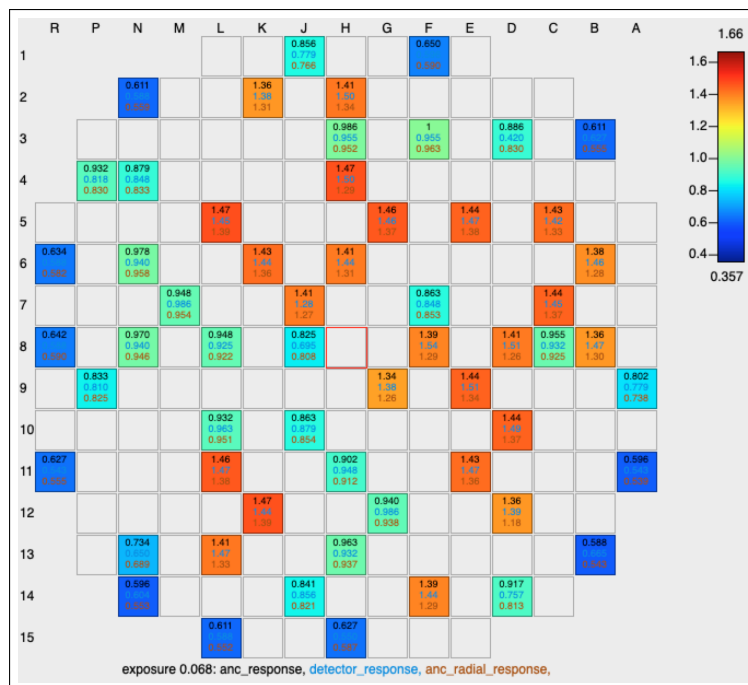
In each fuel assembly location, the axial data is presented based on axial elevation and relative magnitude, similar to dozens of axial plots but without the details such as axis labels, etc. The rectangular border of each assembly is colored based on the 3D detector signal value at the current selected 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. A left-click in an assembly in this widget selects the current assembly.



**Figure 18. In-core Detector Data with 5 Inoperable Locations Presented with the Detector View**

Detector widgets have an additional toolbar button for toggling between numeric () and plot () displays. Numeric displays are illustrated in Figure 19.



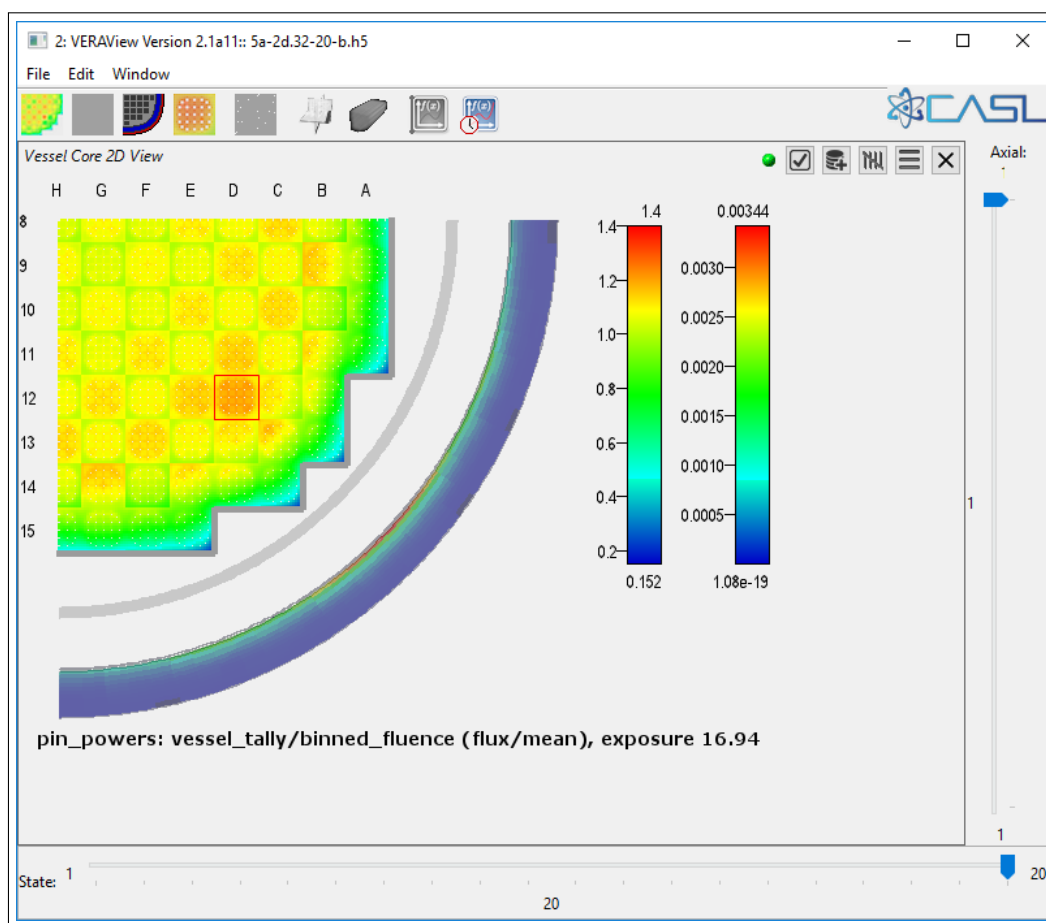
**Figure 19. Detector Widget Numeric Display**

This widget not a basic capability and is not currently being tested in the *VERAView Software Requirements, Test Plan, and Test Report* document.

## 9.7 VESSEL CORE 2D VIEW

Figure 20 shows an example of this widget that displays fluence tally output from Shift. Vessel geometry is displayed to a reasonably accurate scale as specified in the data. Whereas the currently selected rod (pin-wise) or channel dataset is displayed in the core region, the selected fluence dataset is displayed in the vessel region. The Select Dataset menu for this widget has pullrights (labeled **core** and **fluence**) for selecting the respective datasets. A left-click in the vessel region will select the azimuth and radial positions in the fluence data. Data displayed are for the current statepoint and fluence axial level.

This widget not a basic capability and is not currently being tested in the *VERAView Software Requirements, Test Plan, and Test Report* document.

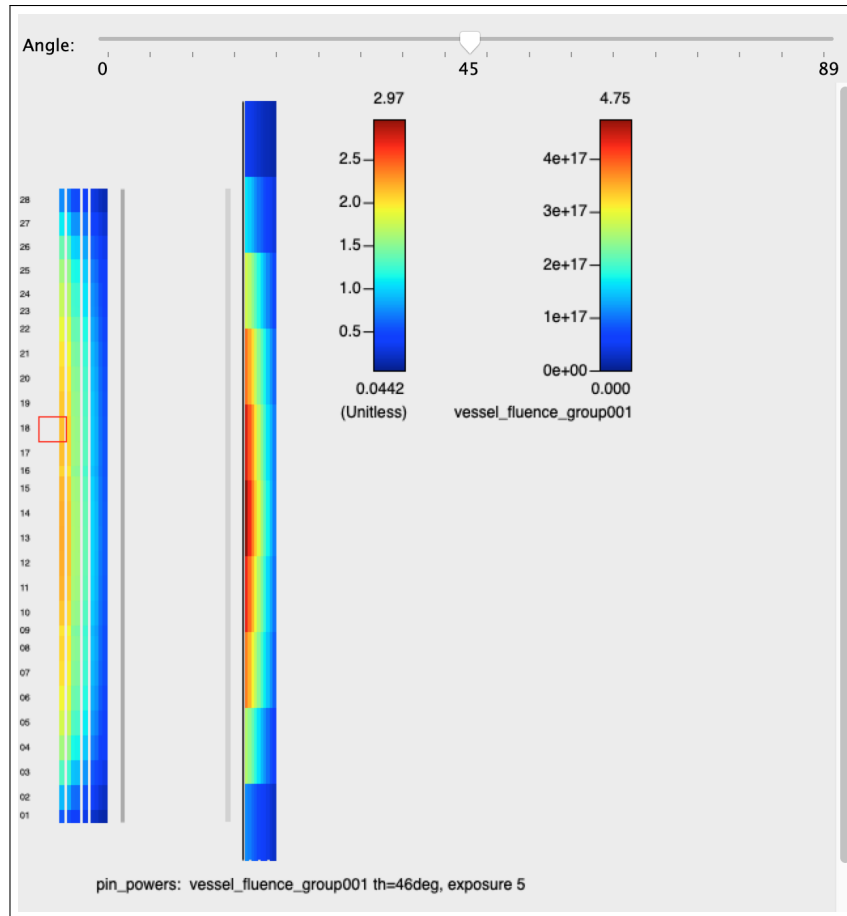


**Figure 20. Vessel Core 2D View Showing Binned Fluence Tallies**

## 9.8 VESSEL CORE AXIAL 2D VIEW

Figure 21 shows this widget, which displays an axial slice of the core and vessel regions. Fluence data displayed in the vessel region are for the current statepoint and azimuth, and the azimuth can be set via the slider above the Angle display. A left click in the core region will select the axial level.

This widget not a basic capability and is not currently being tested in the *VERAView Software Requirements, Test Plan, and Test Report* document.



**Figure 21. Vessel Core Axial 2D View**

## 9.9 TABLE VIEW

A recently added widget is the Table View. It displays dataset values for the current selections (assembly, pin/channel, axial level, statepoint, etc.). Since it displays multiple values it has a **Set Visible Datasets...** item on the **Select Dataset** menu to bring up a tree of available datasets for toggling visible/invisible. Each visible dataset is represented with a table row. Columns in the table represent secondary pin/channel (and node) selections, with the appropriate values displayed for the dataset row. When a dataset row is selected, any attributes stored with the dataset are displayed below the table. Refer to Figure 22.

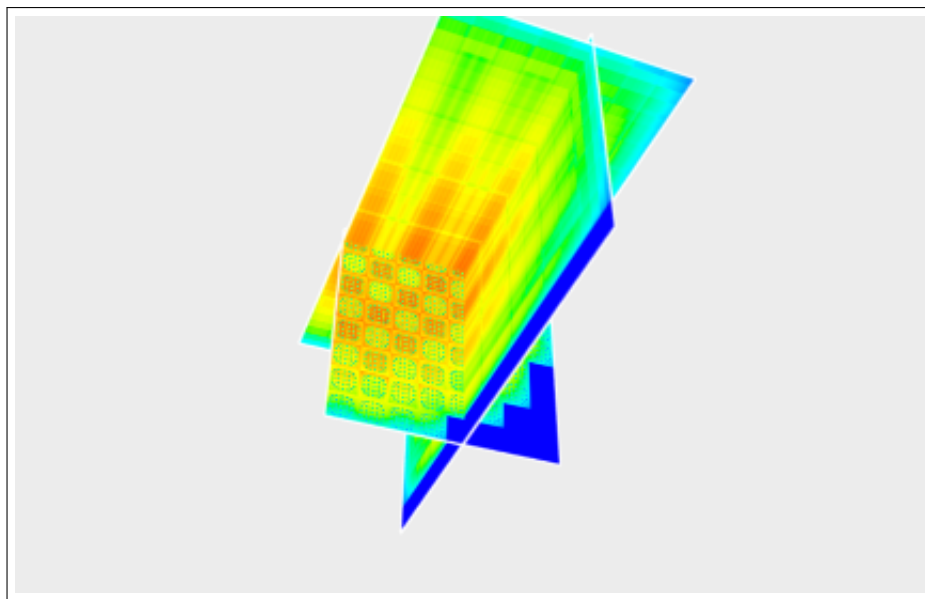
## 9.10 VOLUME SLICER 3D VIEW

The volume slicer provides a very unique capability for 3D visualization similar to that the “3D-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 **<Escape>** to exit full screen mode).

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 23.

Table View	
Assy (D-12); Axial 193.580; exposure 0	
Dataset	(8,8)
pin_powers	2.194607
boron	1296.157
channel_liquid_density	741.8218
channel_liquid_temp	291.6667
channel_mixture_massflux	3630.354
channel_pressure	155.7462
channel_void	1.036905e-06
exposure	0
c1.h5 new   channel_liquid_temp	
Attribute	Value
physical_units	['Celsius']

**Figure 22. Table View Widget**

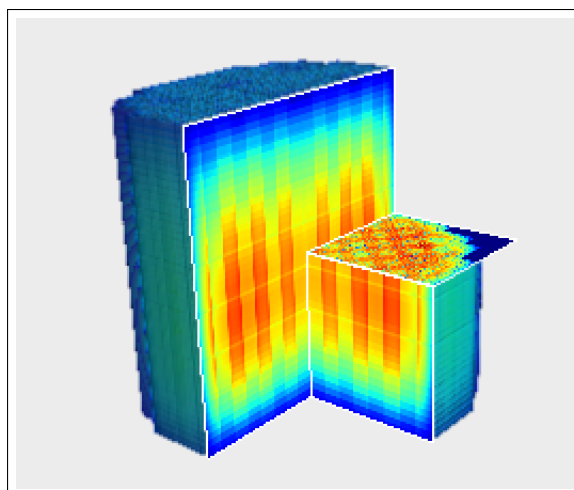


**Figure 23. 3D Visualization Using Volume Slicer Widget**

## 9.11 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 slicer, 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 **<Escape>** to exit full screen mode).

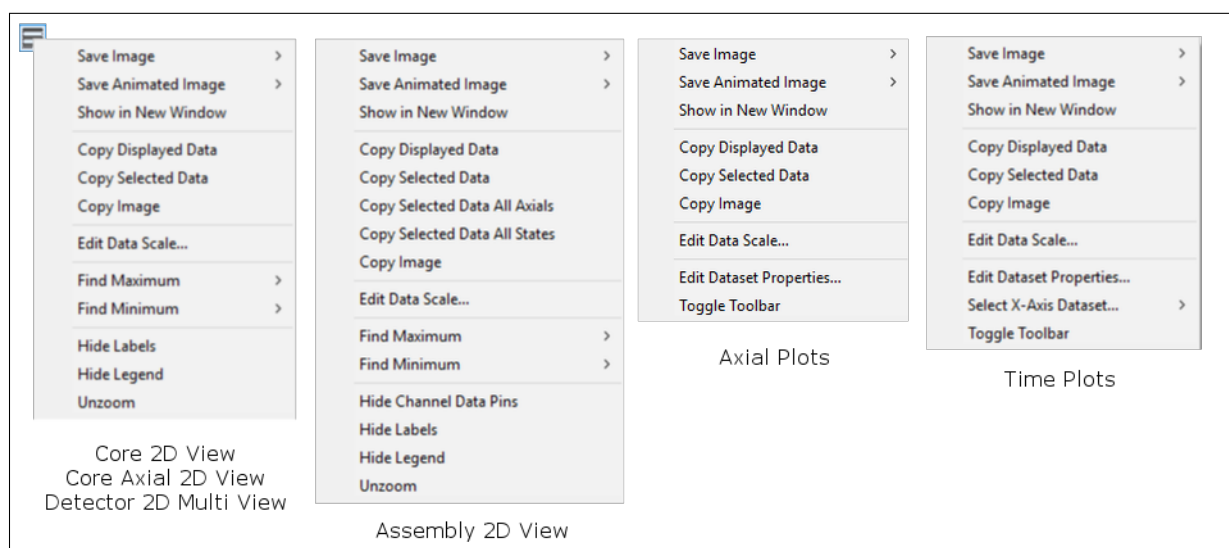
Currently, this widget only functions with pin-wise datasets but the capability to view channel data will be available in the future. A sample image from the widget is shown in Figure 24.



**Figure 24. 3D Visualization Using Volume View Widget**

## 9.12 COMMON WIDGET OPERATIONS

Each widget has a **Widget Functions** button on its toolbar that presents a menu of operations. Most of these are common to all widgets, but there are some widget-specific options. Figure 25 shows the menus displayed by widget. Menu items are described in groups below.



**Figure 25. Widget Menus**

## 9.13 CLIPBOARD COPY OPERATIONS

Each widget offers at least three clipboard copy options, two for copying data and one to copy the current display. Data copies are in CSV format suitable for input into Excel or another spreadsheet application. Headers and labels are added to the copied data when and where appropriate.

### 9.13.1 Copy Displayed Data

This option copies the data currently displayed in the widget. The current axial level, time, assembly, pins/channels, and so on are applied to select the data copied.



### 9.13.2 Copy Image

The current display/image of the widget at its current size is copied to the clipboard. Note this differs from the Save Image option which creates an image of optimal size and scale. However, this is offered as a quick way to use the clipboard to transfer the image.

### 9.13.3 Copy Selected Data

As documented in the widget descriptions above, what is displayed in most widgets is dictated by current selection values. Refer to Section 7.3 for a description of these values. With this option, only the currently selected data relative to the widget are copied to the clipboard. In some instances this will be a single value.

### 9.13.4 Copy Selected Data All Axials

The Assembly 2D View widget offers this option to include the selected data (current assembly, time, pin, and secondary pins) for all axial levels, not just the currently selected axial level.

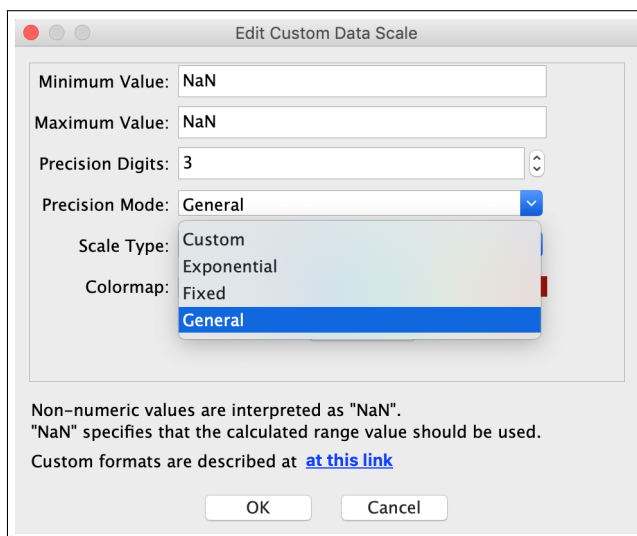
### 9.13.5 Copy Selected Data All States

The Assembly 2D View widget offers this option to include the selected data (current assembly, axial level, pin, and secondary pins) for all statepoints, not just the currently selected time.

## 9.14 DATASET OPERATIONS

### 9.14.1 Edit Data Scale

By default, a global data scaling mode is applied in each widget to the dataset(s) displayed in the widget, as described in Section 10.2.3. A custom data range or scale for the widget may be set via the Edit Data Scale item. Figure 26 shows the dialog that results from selecting the item.



**Figure 26. Custom Data Scale Dialog**

The Minimum Value and Maximum Value fields default to “NaN” (not a number), which is also the assumed value if the field is all blanks. NaN indicates the value should be derived from the data displayed. Enter a specific value to force the lower (minimum) or upper (maximum) range value.

In addition to setting the range, this dialog is used to control the precision and format of numbers displayed in the widget (Core 2D View, Core 2D Axial View, and Assembly 2D View). Four Precision Mode

options are offered: Custom, Exponential, Fixed and General, with the latter as the default. Precision Digits specifies the number of significant figures to display in Exponential or General mode and the number of decimal places in Fixed mode.

If Custom is selected as the Precision Mode, the Precision Digits field is replaced with a Custom Format field, the default value for which is “g” (general). This field specifies a format code for the Python string. Formatter class and should only be applied by users with a working knowledge of Python string formatting [11].

Two additional controls, Scale Type and Colormap, allow control of the coloring of values in displays. Options for these controls are shown in Figure 27 and Figure 28. Colormaps are provided by the matplotlib package and are selected via a popup menu. The colors are categorized into three pullright menus: Miscellaneous, Qualitative, and Sequential. Qualitative colormaps emphasize contrast and generally make no attempt to associate scale with color. Sequential colormaps progress from low/high to high/low intensity values, and Miscellaneous includes all remaining colormaps available.

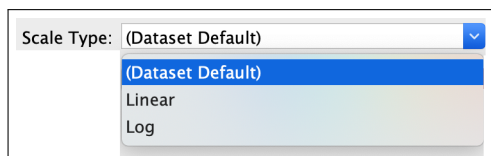


Figure 27. Scale options

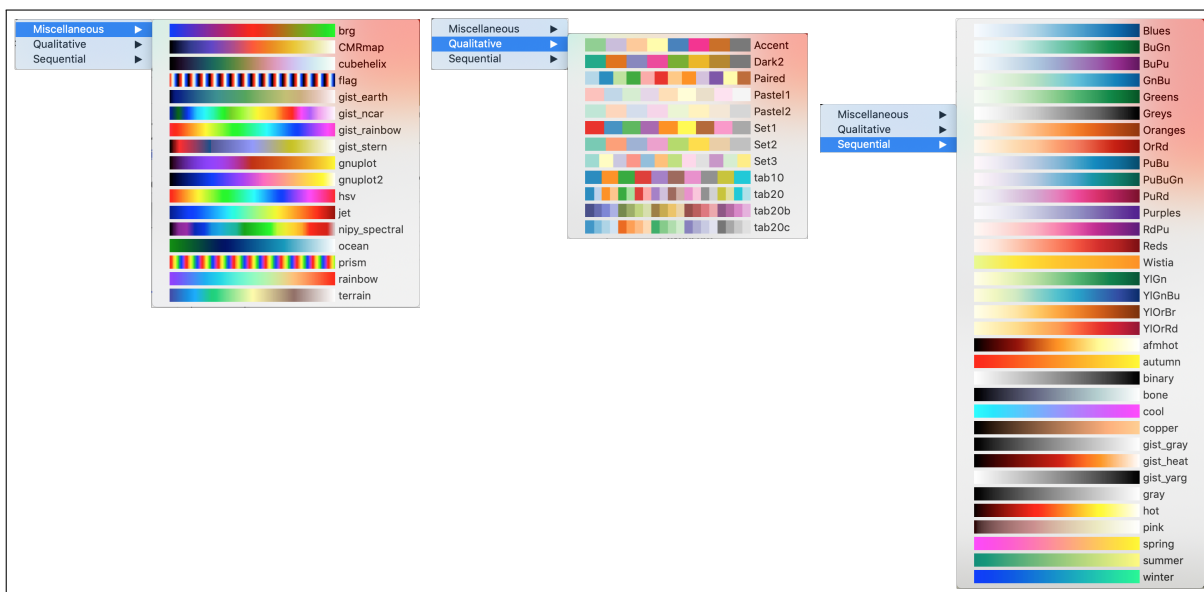
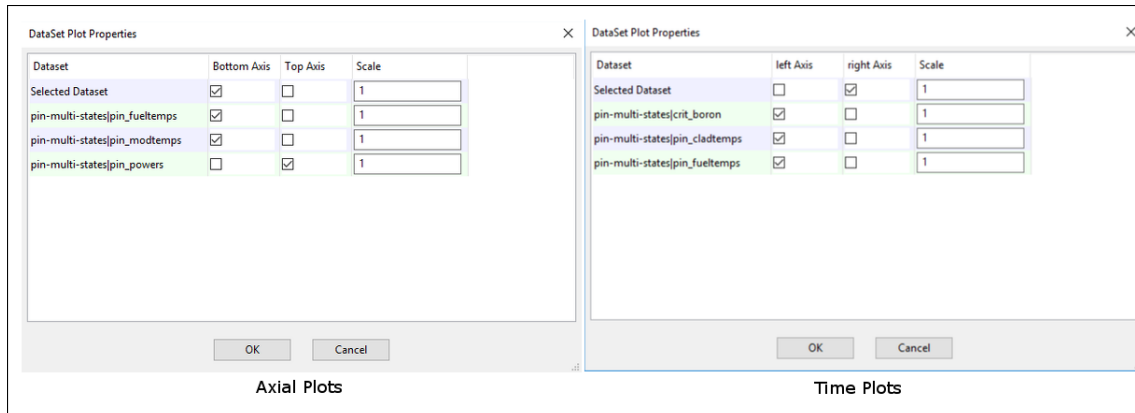


Figure 28. Colormap options

### 9.14.2 Edit Dataset Properties

This item is specific to plot widgets and exists to allow assignment of each visible dataset to an axis. For the Axial Plots widget the assignment is to the top or bottom axis, with bottom as the default. For the Time Plots widget either the left or right axis may be assigned. Figure 29 shows example dialogs for Axial Plots and Time Plots, respectively.



**Figure 29. Dataset Properties for Axial (Left) and Time (Right) Plots**

### 9.14.3 Find Maximum

This option is only available for the raster (not plot) widgets since they display a single dataset at a time. A pullright menu offers **All State Points** and **Current State Point** items. If the former is selected, the search is for the currently displayed dataset over all statepoints (times). The latter limits the search to the current statepoint. The assembly, pin/channel, axial level, and statepoint (if the former menu item is selected) with the maximum value become the current values and are propagated to other widgets.

### 9.14.4 Find Minimum

This option behaves the same as **Find Maximum** except the minimum value is located instead of the maximum.

### 9.14.5 Select X-Axis Dataset

This option is unique to the **Time Plots** widget. It offers pullrights for selecting available datasets and an item for using the current **Time Dataset**, which is the default. With this option it is possible to plot datasets against another dataset.

## 9.15 DISPLAY OPERATIONS

### 9.15.1 Hide/Show Channel Data Pins

This option is unique to the **Assembly 2D View** widget. When displaying a channel dataset, the default behavior is for pins/rods to be represented with circles. This item allows those circles to be toggled visible or invisible.

### 9.15.2 Hide/Show Labels

Most raster widgets include assembly or pin address labels. This item allows those labels to be toggled visible or invisible.

### 9.15.3 Hide/Show Legends

All raster widgets include a legend in the display. This item allows the legend to be toggled visible or invisible.

#### **9.15.4 Toggle Toolbar**

Plot widgets are implemented using matplotlib. This item toggles display of the matplotlib toolbar which has options for zooming and panning within the plot.

#### **9.15.5 Unzoom**

Raster widgets include this option to unzoom one level on the stack of previous zooms. It is also available on the context (right-click) menu.

### **9.16 IMAGE SAVE OPERATIONS**

#### **9.16.1 Save Image**

This function is supported by most widgets and displays a pullright menu with two options: **Transparent Background** and **White Background**. Depending on the application, imported images fare better with transparent backgrounds in some cases and white backgrounds in others. Thus, the options are provided. Regardless of the selection, a file dialog will be shown to specify the path and name of the file to create.

#### **9.16.2 Save Animated Image**

This option also presents a pullright menu with two possible options: **Pin Axial Levels** and **State Points**. Which options (if any) are available are determined by the VERA Output file(s) that have been opened and the specific widget. If there is only one axial level for all open files, **Pin Axial Levels** will never be offered because there would be nothing to animate. Similarly, if there is a single statepoint, no animation across state points is possible.

For either selection, a file dialog is displayed for entering the path and name of the animation file to create. A progress dialog will then appear to show progress of the operation. When finished, three files are created with extensions **.gif**, **.gif.images.zip**, and **.gif.html**. The first is the animated image file in Graphics Interchange Format (GIF). The second is a Zip file with all the individual frame images in the animation, and the third is a Hypertext Markup Language (HTML) file that references the animated GIF, suitable for viewing in a Web browser.

### **9.17 SHOW IN NEW WINDOW**

This option is not currently available for 3D widgets, but a two-step workaround is to use the **File → New Window** option on the main menu bar and then add the desired 3D widget via the Window ToolBar. For 2D widgets this is a convenient method for creating a new window that contains the widget as its sole widget. The widget is effectively moved from the current window to a new window. All settings and options should be preserved.

Note also that any widget can be moved between windows by dragging the widget title onto another window.

## 10. MAIN MENU OPERATIONS

Under Windows and Linux each VERAView window includes a menu bar. As with other Mac OS X applications, when running on a Mac there is a single menu bar for VERAView that applies to the currently active window.

### 10.1 FILE MENU

Figure 30 shows the File menu with the New pullright expanded. File operations are described below.

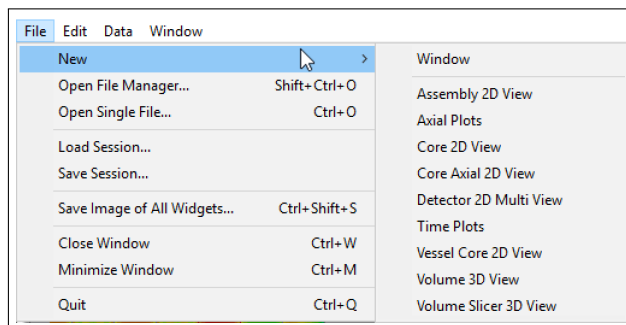


Figure 30. File Menu

#### 10.1.1 File → New

This pullright offers a New item as well as items for each available widget. Note some widget items might be grayed out (disabled) if the types of datasets required for the widget are not available in the open files. Selecting a widget item is the same as left-clicking the corresponding widget icon/button on the window *Main Toolbar*.

Selecting the Window item creates a new, empty window.

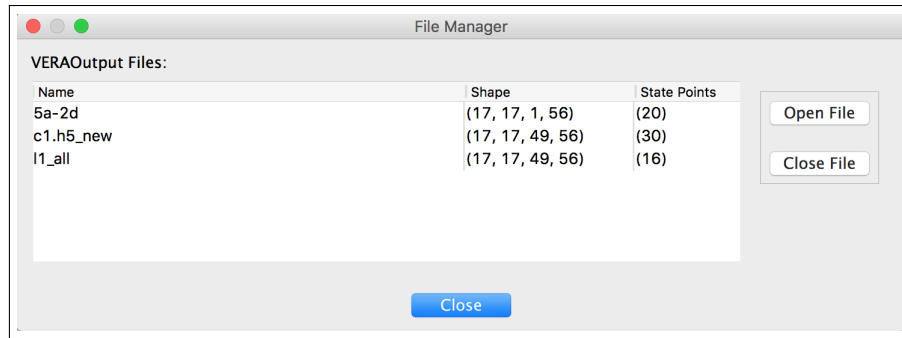
#### 10.1.2 File → Open File Manager

VERAView supports having multiple VERA Output files open simultaneously provided their respective geometries are congruent, as described in Section 13. Selecting this item will bring up the File Manager Dialog, shown in Figure 31. All currently opened files are listed in the dialog, and Open File and Close File buttons are available for adding to or removing from the list, respectively. Each file's 4D shape and number of state points are shown (refer to Section 4).

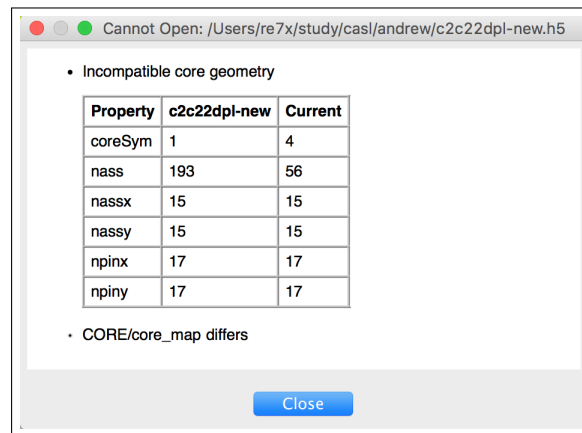
Activating the Open File button brings up a file dialog. If the selected file does not have a congruent geometry, and error/information message dialog is displayed, as illustrated in Figure 32. Otherwise, the file is added. To close and remove a file, select it in the list and activate the Close File button. Activating the Close button at the bottom of the dialog will remove it and cause an update of all VERAView windows. Buttons on the *Main Toolbar* and widget items in the File → New pullright menu will be enabled/disabled as appropriate for available datasets.

#### 10.1.3 File → Open Single File

This item provides the more familiar operation of closing any existing open file(s) and opening a single VERA Output file. Note the same can be accomplished via the File → Open File Manager item if the File Manager dialog is closed with a single open file.



**Figure 31. File Manager Dialog**



**Figure 32. Message for Incongruent File**

#### 10.1.4 File → Load Session

VERAView supports the concept of a “session” in which all active widgets, current state value selections, and the position and size of the VERAView window is serialized to a file. Previously saved session files can be specified on the command line (`--load-session` option) or opened with this menu item. The various objects are deserialized and realized. Sessions are saved in JavaScript Object Notation (JSON) format with a default file extension of `vview`.

Note there are limitations in session support with the current version of VERAView. These limitations follow:

- Multiple windows are not supported, and only the active window’s contents will be saved.
- 3D widgets are not supported.
- Derived datasets are not re-created and a selected derived dataset will be substituted.

However, sessions can be used effectively under many circumstances.

#### 10.1.5 File → Save Session

Select this item to save the current session to a file. The serialized session can be loaded later via the **File → Load Session** menu item.

### 10.1.6 File → Save Image of All Widgets

Select this item to create an image showing all the widgets displayed in the window. Keep in mind that different widgets scale differently and have different aspect ratios. Whereas height tends to be greater than width for axial raster widgets, the converse is true for non-axial raster widgets. Thus, the convenience offered by this item might be offset by unexpected scaling of the widgets in the resulting image.

### 10.1.7 File → Close Window

This item is especially useful when multiple windows are open. It has an equivalent keyboard accelerator (<Control>-W for Windows and Linux, <Cmd>-W for Mac OS X). If the last window is closed, VERAView is terminated, which is equivalent to selecting File → Close.

### 10.1.8 File → Minimize Window

Selecting this item iconifies or minimizes the window. The corresponding keyboard accelerator is <Control>-M for Windows and Linux and <Cmd>-M for Mac OS X.

### 10.1.9 File → Quit

Select this item to terminate VERAView. Note that the session is saved to a default, user-specific location.

## 10.2 EDIT MENU

The edit menu with all pullrights in shown in Figure 33. Items are described below.

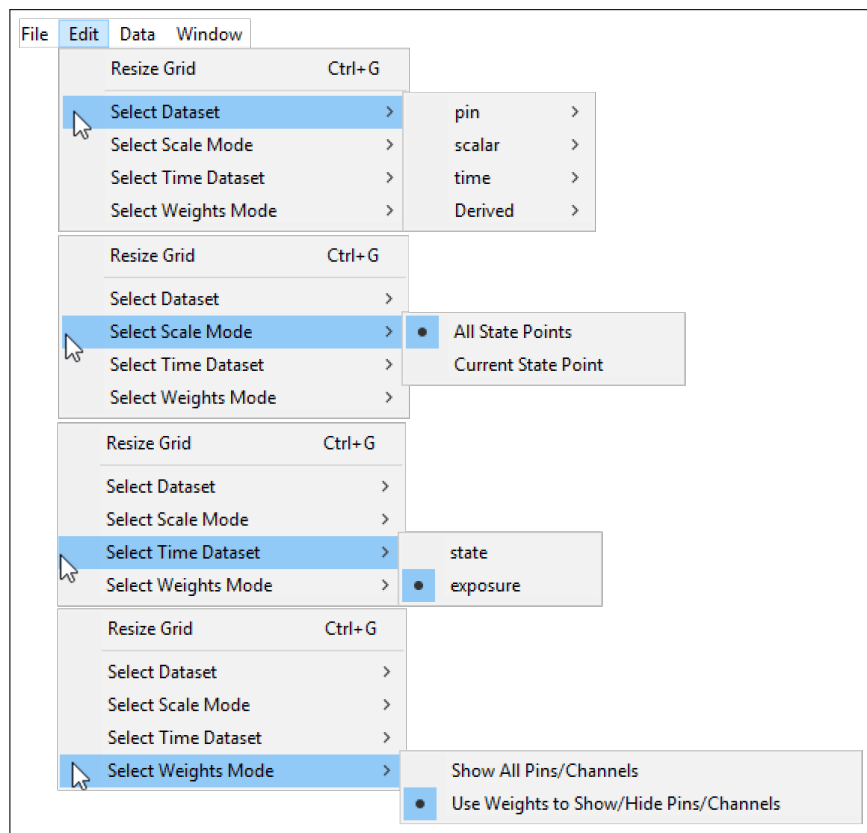


Figure 33. Edit Menu With Pullrights

### 10.2.1 Edit → Resize Grid

Selecting this item brings up the Set Grid Size dialog. Refer to Section 7.1 for a description of this operation. This item's keyboard shortcut is <Control>-G for Windows and Linux and <Cmd>-G for Mac OS X.

### 10.2.2 Edit → Select Dataset

This item presents a dataset selection pullright menu very similar to a Select Dataset to View widget toolbar button. This item differs from a widget dataset menu in that all available datasets are shown, not just those viewable in a specific widget. As is the case with any dataset selection, widgets which are tied to the dataset selection will ignore the dataset if it is not viewable by the widget. Nonetheless, this menu provides a global means of selecting the current dataset.

### 10.2.3 Edit → Select Scale Mode

By default, a global data scaling mode is applied in each widget to the dataset(s) displayed. The global scale is selected via Edit → Select Scale Mode on the main menu bar, which presents the two scale mode options as a pullright. The colormap used in each widget may be applied across the range of dataset values in the current statepoint only (Current State Point) or across all statepoints (All State Points). The latter is the default. Refer to Figure 34.

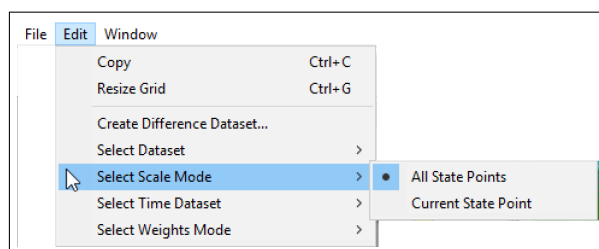


Figure 34. Select Scale Mode

### 10.2.4 Edit → Select Time Dataset

VERAView recognizes certain scalar datasets as representing time and thus available for use as the time value. Specifically, the dataset names in decreasing order of priority are “msecs”, “exposure”, “exposure\_efpd”, “hours”, and “transient\_time”. The order indicates which dataset is chosen as the default time dataset. If none of these datasets are present in a VERA Output file, the statepoint index is used as the time indicator.

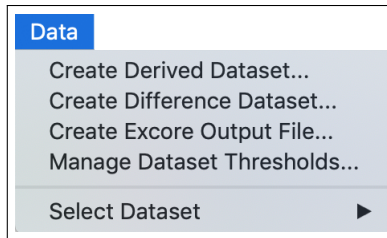
### 10.2.5 Edit → Select Weights Mode

With this item, one of two modes for dealing with weights or factors are applied to raster widgets. The default mode is Use Weights to Show/Hide Pins/Channels, meaning pins or channels with a weight factor of zero are not drawn. The alternative is Show All Pins/Channels, in which all pins or channels are shown regardless of the corresponding weight factor.

## 10.3 DATA MENU

This menu was introduced in the later alpha releases of VERAView version 2.1. In prior VERAView versions the items on the menu appeared in the Edit menu. Figure 35 illustrates the Data menu with its three items, described below.

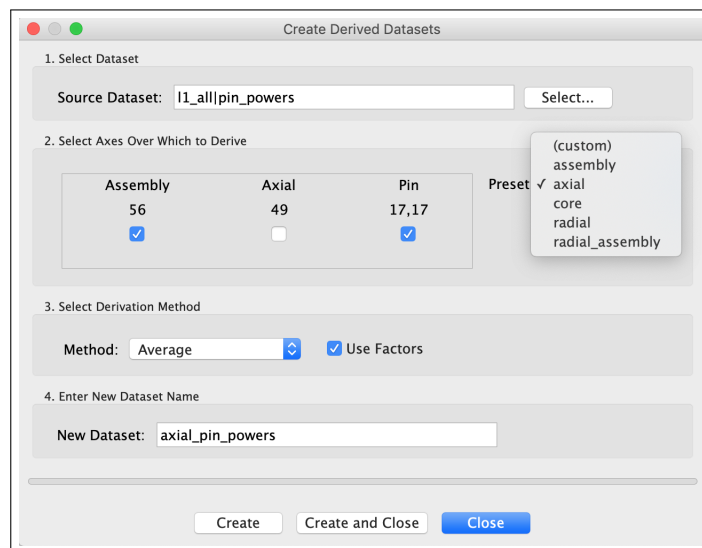




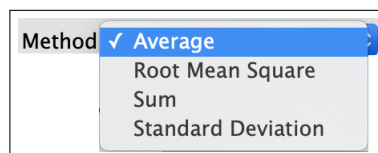
**Figure 35. Data Menu**

### 10.3.1 Data → Create Derived Dataset

VERAView provides a couple of ways to create derived datasets. A quick way is made available on dataset selection menus as described in Section 11. Complete control over the process is available via this menu selection, which allows derivation from any dataset, including a previously-derived one. The Create Derived Datasets dialog is shown in Figure 36. The sequence of inputs is numbered in control groups from top to bottom, with the first step being selection of the source dataset for the derivation. The Select... button brings up a dataset selection menu similar to the dataset selection button in widget toolbars. Once a dataset has been selected, the axes over which to apply the derivation must be specified. Individual checkboxes can be toggled, or one of the Presets can be selected. Next, a derivation method must be selected, with options as shown in Figure 37. Average is the default. Finally, a name for the new dataset must be specified. A default name will be provided in the New Dataset field, but this can (and probably should) be overridden.



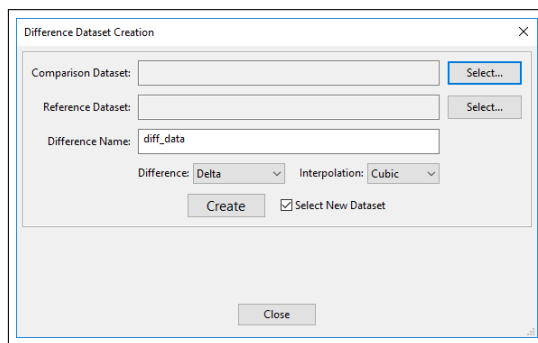
**Figure 36. Create Derived Datasets Dialog**



**Figure 37. Derived Dataset Methods**

### 10.3.2 Data → Create Difference Dataset

VERAView will compute the difference between two datasets, even datasets from different VERA Output files that have been opened (refer to Section 10.1.2). Selecting this menu item brings up the **Difference Dataset Creation** dialog, shown in Figure 38. The dialog presents **Select** buttons for identifying the comparison and reference datasets. The created difference dataset will be stored in the same file as the comparison dataset.<sup>4</sup> The result is the comparison minus the reference.

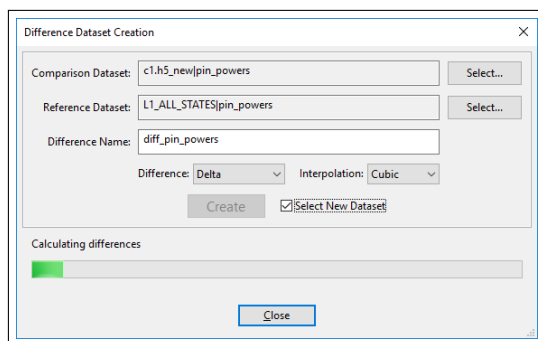


**Figure 38. Difference Dataset Dialog**

The **Select** buttons present pullright dataset menus similar to the **Edit → Select Dataset** menu. If the selected dataset names are the same (from different files), the **Difference Name** field will be filled automatically with the dataset name with a “diff\_” prefix.

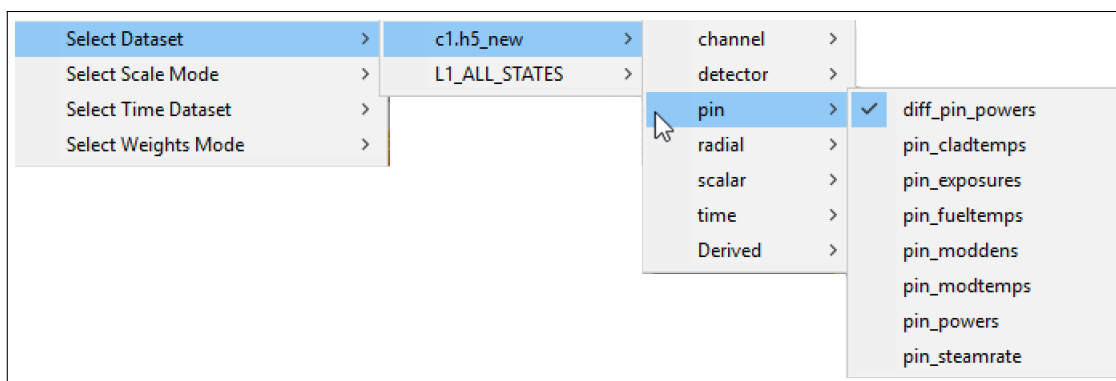
The **Difference** pulldown menu offers two options: **Delta** (default) and **Pct Difference**. Unless the axial meshes for the selected datasets are equivalent, the reference dataset is interpolated on the comparison dataset axial mesh. Three **Interpolation** options are offered on a pulldown menu in order of decreasing order accuracy and increasing calculation time: **Cubic**, **Quadratic**, and **Linear**. The **Select New Dataset** checkbox determines if the created dataset is set as the selected dataset upon completion.

Finally, activating the **Create** button launches the difference calculation. Note multiple difference datasets may be created before closing the dialog via the **Close** button (or the Window manager icon for closing a dialog window). Figure 39 shows the dialog while a difference calculation is in progress. The newly created dataset(s) will appear in dataset pullright menus in the appropriate type or category, as shown in Figure 40.



**Figure 39. Difference Dataset Calculation**

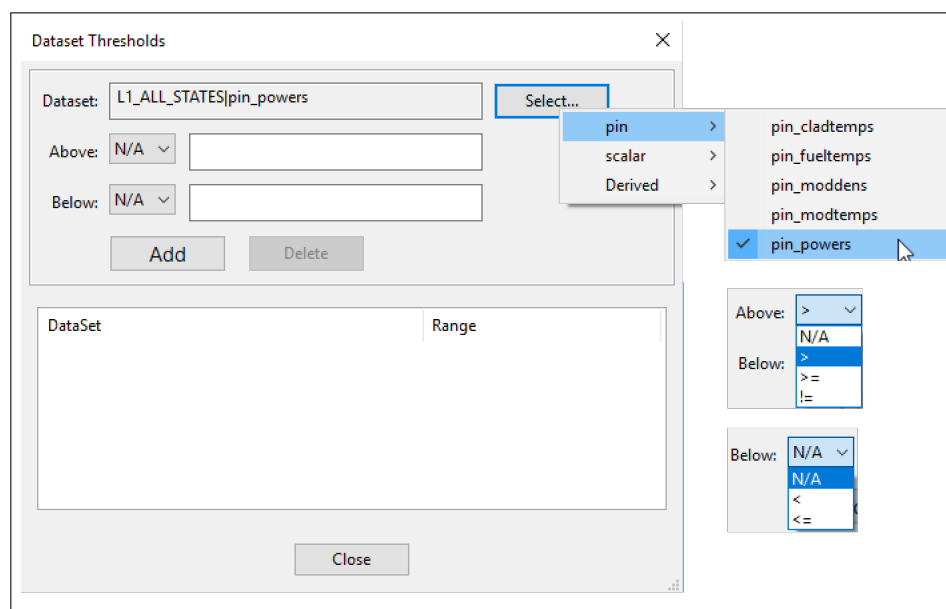
<sup>4</sup>Note the VERA Output HDF file is not modified. Rather, a temporary HDF5 corresponding to the opened file is created to hold derived and difference datasets.



**Figure 40. Difference Dataset in Menu**

### 10.3.3 Data → Manage Dataset Thresholds

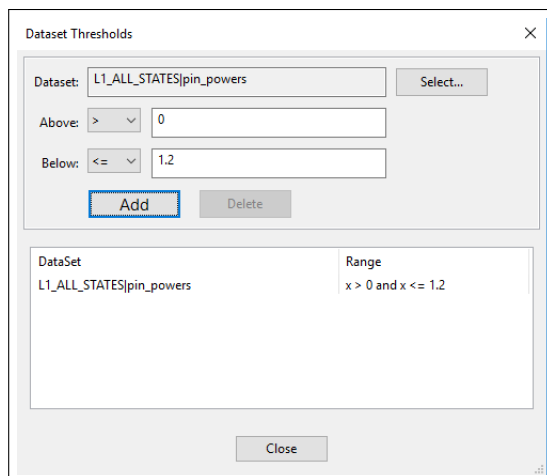
With this option a filter can be specified to control the range of data values to be displayed. First, one selects the dataset to which to apply the filter. Next, lower and/or upper range filtering operators are selected via the Above and Below menus, respectively, as shown in Figure 41.



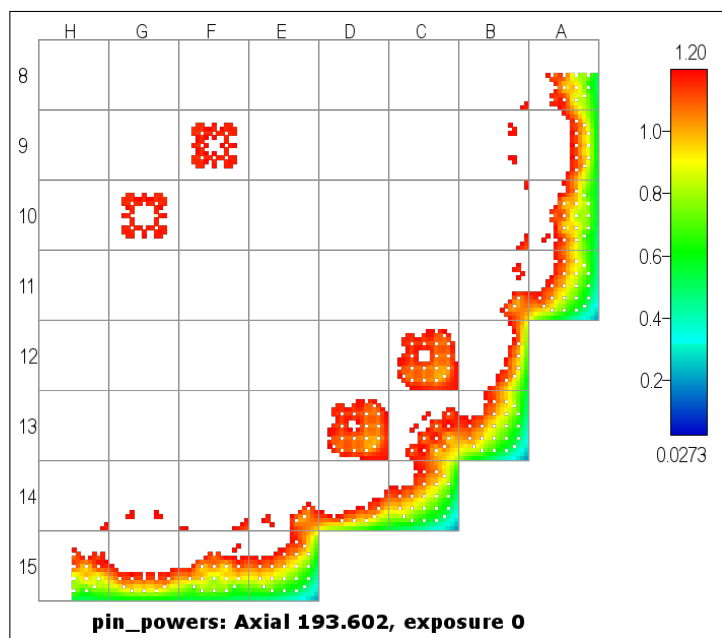
**Figure 41. Manage Thresholds Selections**

The lower (Above) operators are >, >=, !=, and N/A to apply no lower bound. Note != is a not specifically a lower bound but must be selected from the Above menu. Upper bound (Below) operators are <, <=, and N/A for no upper bounds. Floating point or integer values are entered in the text fields corresponding to the thresholds. When the Add button is activated, the range expression is added and shown in the window, as shown in Figure 42.

When the dialog is closed via the Close button, any widgets currently displaying the datasets specified will apply the threshold, as illustrated in Figure 43.



**Figure 42. Threshold Filter**



**Figure 43. Widget With Threshold Applied**

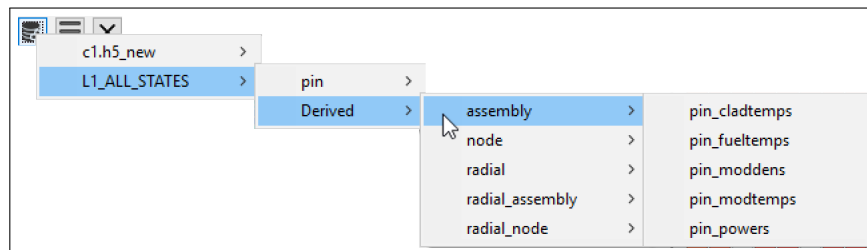
## 10.4 WINDOW MENU

This menu is provided to help manage multiple windows and is divided into two parts. The first part has a single item, **Tile Windows**, which will arrange all open windows visibly as tiles using the entire desktop. Note for many windows this will probably result in less-than-optimal window sizes.

After a separator, the second part of the menu lists each open window. Selecting a window causes it to be brought to the front and focused. Note this is a convenience mostly useful when several windows are open or as an alternative to whatever window manager options allow selection of open windows. With Windows and most Linux window managers, **<Alt>-<Tab>** (holding **<Alt>** down while pressing **<Tab>**) can be used to cycle between all open windows. Under Mac OS X, the **<Cmd>-'** sequence (holding **<Cmd>** down while pressing **'**) cycles between windows of the current application. This works for VERAView windows as well. **<Cmd>-<Tab>** may be used to cycle between applications to make VERAView current.

## 11. CREATING DERIVED DATASETS

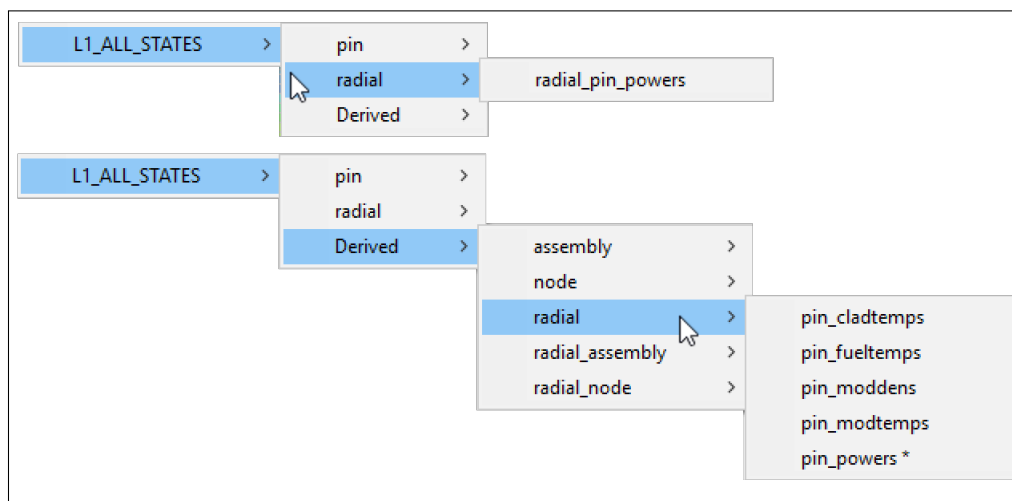
Table 2 in Section 4 describes the mapping between dataset types or categories and 4D shape. VERAView offers an additional, quick option for calculating derived datasets from datasets present in open VERA Output files. The last item on the dataset pullright menu for an open file (or the top menu level if a single file is open) is **Derived**, which is itself a pullright offering derivable categories for the file. Refer to Figure 44. The pullright for each derivable category/type lists the datasets from which the type can be derived. Select an item in the right-most menu to create the derived dataset.



**Figure 44. Derived Pullright Menu**

Section 10.3.1 describes a more general option for creating derived datasets via the Create Derived Datasets dialog.

When a derived dataset is added, it will appear in the dataset menu. The **Derived** menu will indicate that the derived dataset has been added by appending a star (\*) to the source dataset name. Note derivable datasets can be provided in the dataset originally, in which case they will appear in the dataset menu from the start. Refer to Figure 45 for an example of updated menus after deriving a dataset. Once created, derived datasets may be selected for display in widgets which support them.



**Figure 45. Menus After Deriving a Dataset**

Note that derived datasets are not preserved between VERAView executions. They must be re-created upon a subsequent execution, and they will not be available when loading from a saved session.

## 12. CREATING DIFFERENCE DATASETS

Section 10.3.2 describes the mechanism for creating difference datasets. Differences between datasets in the same or different files may be calculated. If the datasets come from different files and those files have different axial meshes, the reference dataset must be interpolated on the comparison dataset mesh before subtracting it from the comparison. Interpolation is implemented using the `scipy.interpolate.interp1d()` function using a cubic, quadratic, or linear spline. Regions outside the comparison mesh are linearly interpolated. Other interpolation options may be explored in the future and possibly offered as options.

Once created, difference datasets may be selected as current and displayed in widgets which support the corresponding category/type. Note that difference datasets are not preserved between VERAView executions. They must be re-created upon a subsequent execution, and they will not be available when loading from a saved session.

## 13. OPENING MULTIPLE FILES

Section 10.1.2 describes the process of managing multiple open VERA Output files in VERAView. An additional file must be congruent with currently open files before being opened. Further, when multiple files are open, items for each file will comprise the first level of a dataset selection menu, each file item being a pullright with items for the categories/types available in the file. Refer to Figure 40 for an example. Having multiple files open also changes the behavior of the Statepoint Slider and Axial Slider.

### 13.1 CONGRUENCE

As relates to opening multiple files in VERAView, “congruence” is defined as having the same core geometry, symmetry, and core assembly map. That is, the following /CORE values must be equivalent: `nass`, `nassx`, `nassy`, `npinx`, `npiny`, `core_sym`, and `core_map`. Note the `axial_mesh` can be different and will be used as a basis for interpolation if necessary when differencing two datasets. An attempt to open an incongruent file will result in an error message shown in Figure 32. The first open file defines the values upon which congruence will be determined for all other files.

### 13.2 RESOLVING TIME AND AXIAL LEVELS

When a single file is open, the increments on the Statepoint Slider and Axial Slider correspond to times and axial levels, respectively, in the file. However, with multiple open files, each increment on the sliders represents a value in one or more of the files. Times (statepoints) and axial levels are resolved across all open files to represent each distinct value on the corresponding slider. Thus, incrementing the time or axial level may not cause an update to a widget if the dataset(s) displayed in the widget do not have the selected value.

For example, suppose a file is opened with the following:

Fifty `axial_mesh` values resulting in 49 axial mesh centers:

```
11.951, 15.817, 24.028, 32.239, 40.450, 48.662, 56.873, 65.084,  
73.295, 77.105, 85.170, 93.235, 101.300, 109.365, 117.430, 125.495,  
129.305, 137.370, 145.435, 153.500, 161.565, 169.630, 177.695, 181.505,  
189.570, 197.635, 205.700, 213.765, 221.830, 229.895, 233.705, 241.770,  
249.835, 257.900, 265.965, 274.030, 282.095, 285.905, 293.970, 302.035,  
310.100, 318.165, 326.230, 334.295, 338.105, 346.026, 353.947, 361.869,  
369.790
```

Sixteen statepoint exposure values:

```
0.000000, 0.384058, 0.768116, 1.152173, 1.728260, 2.304347,  
3.072462, 3.840578, 4.608693, 6.144924, 7.681155, 9.217387,  
10.753618, 12.289849, 13.862080, 15.362311
```

Axial level indexes correspond directly to the axial mesh center values. For example, using 0-based indexes, the `axial_mesh_centers[3]` is 32.239, `axial_mesh_centers[10]` is 85.170, and so on. Similarly, `exposure[1]` is 0.384058, `exposure[15]` is 15.362311, etc.

Subsequently, a file is opened with the following:

Fifty axial\_mesh values with the following 49 centers:

10.315, 14.821, 23.389, 31.957, 40.526, 49.094, 57.662, 66.230,  
73.279, 77.089, 85.154, 93.218, 101.283, 109.347, 117.412, 125.476,  
129.286, 137.351, 145.415, 153.480, 161.544, 169.609, 177.673, 181.483,  
189.548, 197.612, 205.677, 213.741, 221.806, 229.870, 233.680, 241.745,  
249.809, 257.874, 265.938, 274.003, 282.067, 285.877, 293.942, 302.006,  
310.071, 318.135, 326.200, 334.264, 338.074, 345.674, 353.274, 360.875,  
368.475

Thirty statepoint exposure values:

0.000, 0.346, 1.230, 1.919, 2.457, 2.994, 3.562, 4.065,  
4.642, 5.139, 5.700, 6.237, 7.000, 7.463, 7.978, 8.493,  
9.140, 9.602, 10.344, 10.842, 11.314, 11.988, 12.552, 13.360,  
14.335, 14.335, 15.068, 15.068, 15.308, 15.774

The resolved axial mesh centers have the following 98 values:

10.315, 11.951, 14.821, 15.817, 23.389, 24.028,  
31.957, 32.239, 40.450, 40.526, 48.662, 49.094,  
56.873, 57.662, 65.084, 66.230, 73.279, 73.295,  
77.089, 77.105, 85.154, 85.170, 93.218, 93.235,  
101.283, 101.300, 109.347, 109.365, 117.412, 117.430,  
125.476, 125.495, 129.286, 129.305, 137.351, 137.370,  
145.415, 145.435, 153.480, 153.500, 161.544, 161.565,  
169.609, 169.630, 177.673, 177.695, 181.483, 181.505,  
189.548, 189.570, 197.612, 197.635, 205.677, 205.700,  
213.741, 213.765, 221.806, 221.830, 229.870, 229.895,  
233.680, 233.705, 241.745, 241.770, 249.809, 249.835,  
257.874, 257.900, 265.938, 265.965, 274.003, 274.030,  
282.067, 282.095, 285.877, 285.905, 293.942, 293.970,  
302.006, 302.035, 310.071, 310.100, 318.135, 318.165,  
326.200, 326.230, 334.264, 334.295, 338.074, 338.105,  
345.674, 346.026, 353.274, 353.947, 360.875, 361.869,  
368.475, 369.790

Thus, axial\_mesh\_centers[3] is now 15.187, and axial\_mesh\_centers[10] becomes 48.662. Conversely, the axial level value of 101.3 formerly was at index 12 but is at index 24 after the second file is opened.

The 45 resolved time values are as follows:

0.000, 0.346, 0.384, 0.768, 1.152, 1.230, 1.728, 1.919,  
2.304, 2.457, 2.994, 3.072, 3.562, 3.841, 4.065, 4.609,  
4.642, 5.139, 5.700, 6.145, 6.273, 7.000, 7.463, 7.681,  
7.978, 8.493, 9.140, 9.217, 9.602, 10.344, 10.754, 10.842,  
11.314, 11.988, 12.290, 12.552, 13.360, 13.826, 14.335, 14.335,  
15.068, 15.068, 15.308, 15.362, 15.774

After the new file is added, exposure[1] is 0.346, and exposure[15] is 4.609. The exposure value 7.681 was at index 10 but is now at index 23.



## 14. 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. Nine 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 findings with others.

VERAView tool easy to use for intuitive inspection of the reactor data. It reads VERA Output formatted HDF5 files 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 spending 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.

## 15. ACKNOWLEDGEMENTS

This research was supported by the US Department of Energy and the Nuclear Energy Advanced Modeling and Simulation Program.

## REFERENCES

- [1] Paraview, 2022. Available at <https://www.paraview.org/>.
- [2] VisIt, 2022. Available at <https://visit.llnl.gov/>.
- [3] Andrew Godfrey, Scott Palmtag, Greg Davidson, and Ben Collins. VERAOut - VERA HDF5 Output Specification. Technical Report CASL-U-2014-0043-001, Oak Ridge National Laboratory, 2014.
- [4] Anaconda, 2022. Available at <https://www.anaconda.com/>.
- [5] wxpython, 2022. Available at <https://wxpython.org/>.
- [6] wxwidgets, 2022. Available at <https://www.wxwidgets.org/>.
- [7] Matplotlib, 2022. Available at <https://matplotlib.org/>.
- [8] h5py, 2022. Available at <https://www.h5py.org/>.
- [9] Numpy, 2022. Available at <https://www.numpy.org/>.
- [10] Mayavi, 2022. Available at <https://docs.enthought.com/mayavi/mayavi/>.
- [11] Python string formatting, 2022. Available at <https://docs.python.org/2/library/string.html#string.Formatter>.