IMASViz Documentation

Release 2018

Dec 20, 2018

CONTENTS:

CHAPTER

ONE

INTRODUCTION

1.1 Description

IMASViz is a visualization tool developed within **Integrated Modelling Analysis Suite** (**IMAS**) for the purposes of visualizing static and dynamic IMAS data, stored within IMAS **Interface Data Structures** (**IDSs**). While the tool itself is already available for use it is still under active development, and various features, GUI improvements etc. are still being implemented.

For additional support or if any issues are found with IMASViz please contact the developers via e-mail or submit a ticket on JIRA ITER Webpage.

While submitting the ticket please use the options listed below

Field	Required Option
Project	IMAS (IMAS)
Components	VIZ

Developers:

- **Ludovic Fleury** (CEA Cadarache, Research Institute for Magnetics fusion, e-mail: Ludovic.FLEURY@cea.fr)
- **Dejan Penko** (University of Ljubljana, Mech.Eng., LECAD Lab, e-mail: dejan.penko@lecad.fs.uni-lj.si)

The tool uses the following Python packages:

1. PyQt5

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android.

Qt is set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems.

For more on **PyQt5** see PyQt5 webpage. For more on **Qt** see Qt webpage.

2. pyqtgraph

Pyqtgraph is a graphics and user interface library for Python that provides functionality commonly required in engineering and science applications. Its primary goals are a) to provide fast, interactive graphics for displaying data (plots, video, etc.) and b) to provide tools to aid in rapid application development (for example, property trees such as used in Qt Designer).

PyQtgraph makes heavy use of the Qt GUI platform (via PyQt or PySide) for its high-performance graphics and numpy for heavy number crunching. In particular, pyqtgraph uses Qt's GraphicsView framework which is a highly capable graphics system on its own; we bring optimized and simplified primitives to this framework to allow data visualization with minimal effort.

For more on **pyqtgraph** see pyqtgraph webpage.

3. matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

For more on **matplotlib** see matplotlib webpage.

4. sphinx

Sphinx is a tool originally created as a Python documentation generator but it allows also generating documentation in form of html, latex etc.

For more on **Sphinx** see Sphinx webpage.

IMASViz tool is available on **ITER git repository** (access permission is required) under project **Visualization/VIZ**, branch **viz2.0_develop**.

Direct link to the **IMASViz** git.iter repository: IMASViz.

1.2 Release notes

1.2.1 Version 2.0

Released:

/

Changes:

- **Full GUI migration from wxPython and wxmPlot to PyQt and pyqtgraph Python libraries**
- Basic plot feature performance improved greatly. Quick comparison for plotting 17 plots to a single panel using default plotting options:
  - wxPython IMASViz: ~13s
  - PyQt5 IMASViz: less than 1s (more than **13x speed improvement**!)
- Superior plot export possibilities
- GUI improvements
- Database tree browser interface display improvements
- Added first 'node contents display' feature (displayed in the *Node Documentation* Widget)
- Reduced the number of separate windows, introduce docked widgets
- Introduce first GUI icons
- MultiPlot feature relabeled to TablePlotView
- SubPlot feature relabeled to StackedPlotView
- Add support for IMAS versions 3.20.0, 3.21.0 and 3.21.1
- Included **documentation + manual** (~60 pages in PDF) in a form of reStructuredText source files for document generation (single source can be generated into multiple formats e.g. PDF, HMTL. . .)
- In-code documentation greatly improved and extended
- and more. . .

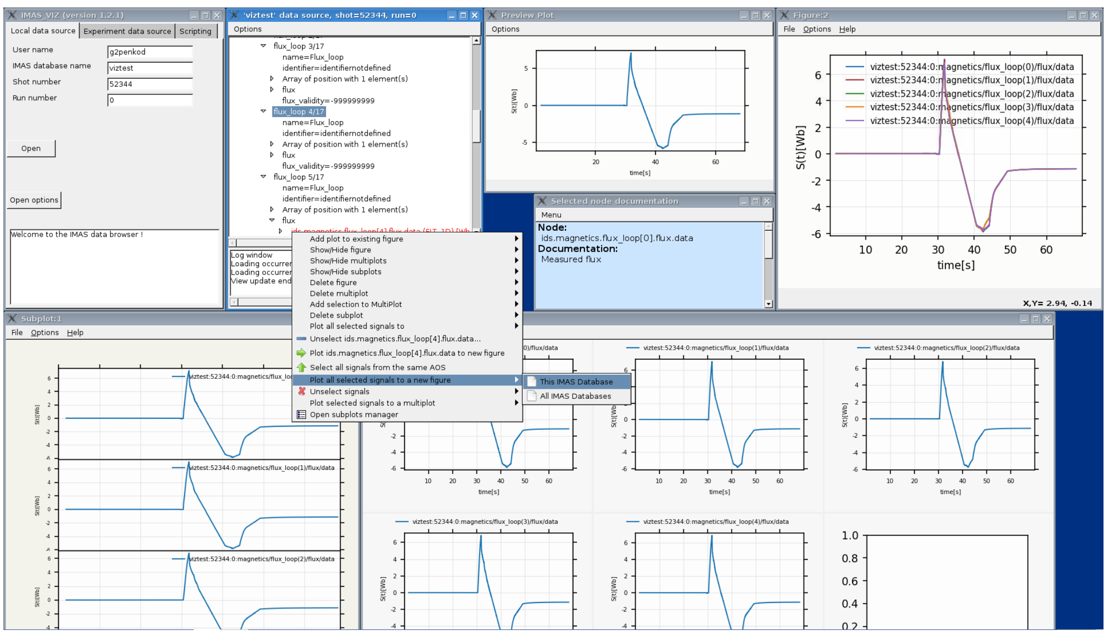
Short summary of files and line changes count (ignoring generated files and scripts):

- 193 commits,
- 268 files changed,
- 13316 insertions(+),
- 10162 deletions(-)

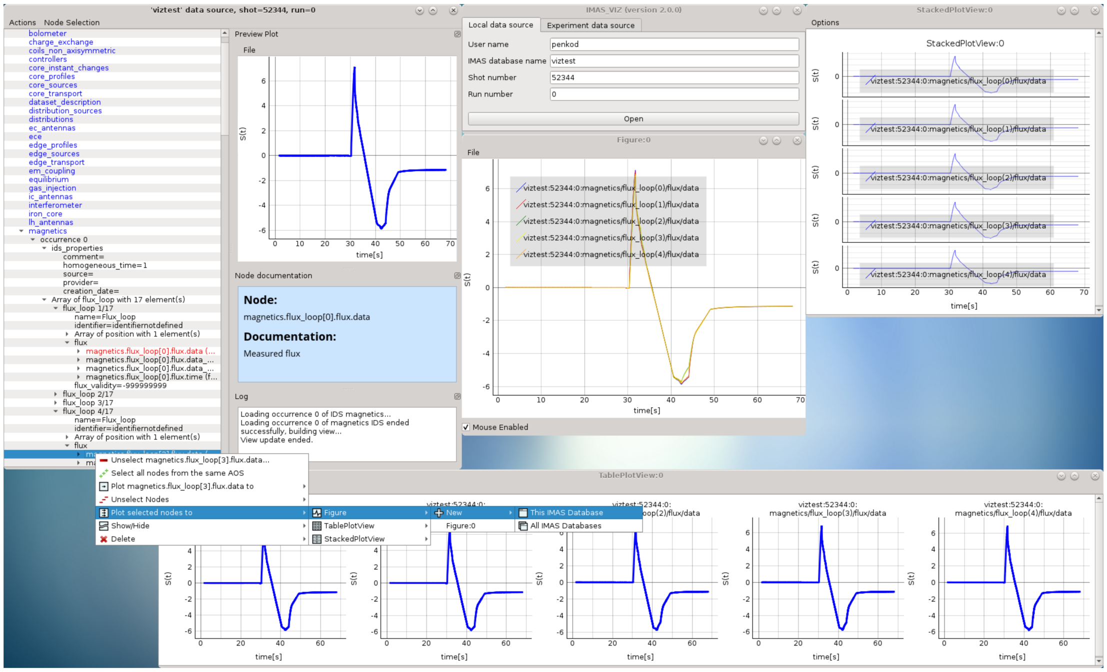
Note: The migration to PyQt5 due to IMASViz containing a large code sets is not yet fully complete. List of known features yet to migrate to IMASViz 2.0: `Equilibrium plugin`, `Add selected nodes to existing TablePlotView`, and `StackedPlotView manager`.

A quick GUI comparison between the **previous** and the **new** IMASViz GUI is shown below.

Overview of IMASViz 1.2 GUI:



Overview of IMASViz 2.0 GUI:



1.2.2 Version 1.2

Released:

24.8.2018

Changes:

- New functionality: selection command of nodes belonging to same parent AOS (Array of Structures)
- MultiPlot and SubPlot design improvements
- Adding support for IMAS versions 3.19.1

1.2.3 Version 1.1

Released:

8.6.2018

Changes (since March 2017):

- Bugs fixes & performance improvement
- Code migration to Python3
- GUI improvements
- UDA support for visualizing remote shots data
- Reuse of plots layout (multiplots customization can be saved as a script file to be applied for any shot)
- A first plugins mechanism has been developed which allows developers to integrate their plugins to IMASViz
- The 'Equilibrium overview plugin' developed by Morales Jorge has been integrated into IMASViz
- Concerning UDA, WEST shots can be accessed if a SSH tunnel can be established to the remote WEST UDA server.

- Introducing MultiPlot and SubPlot features
- Add support for IMAS version 3.18.0

CHAPTER

TWO

USER MANUAL

Note: The manual presented is executed on the **Gateway HPC**.

2.1 Getting Started

This section describes setting the environment configuration required to run the IMASViz tool and how to run the application itself.

2.1.1 Running IMASViz as a module on The GateWay

The procedure below describes how to use IMASViz if it is available as a module on the HPC cluster.

2.1.1.1 Setting the Environment

In a new terminal, execute the following command in order to load the required modules:

```
module load cineca
module load imasenv   # or any other specific imasenv module version
module load imas-viz/2.0.0

# The next few modules should be loaded together with imas-viz/2.0.0
# Listing them all here in case of module related issues.
# module load itm-gcc/6.1.0
# module unload itm-python/2.7
# module load itm-python/3.6
# module load itm-qt/5.8.0
```

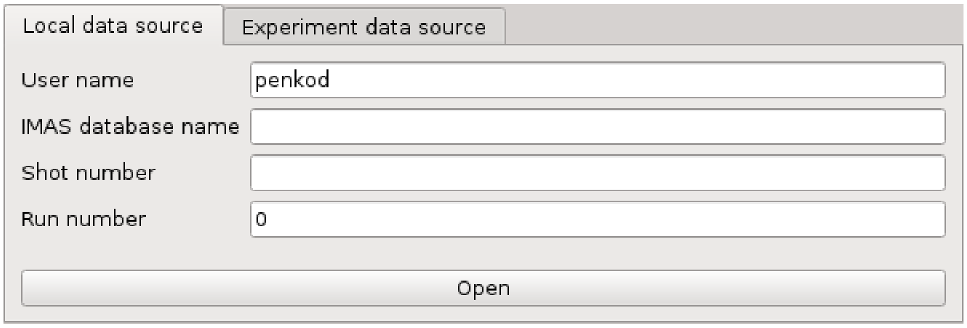
Warning: IMPORTANT! IMAS databases (IDSs) were written using specific version of IMAS. In order to open these IDSs the **same IMAS module version** should be used due to possible IDS database structure changes through different versions. Any tools or utilities that work with IDSs, including `IMASViz`, cannot work properly if this "IMAS version mismatch" is too great (!).

2.1.1.2 Running IMASViz

With the environment set, run the IMASviz by simply typing the following command:

```
viz
```

The main GUI window of IMAS_VIZ should display, as shown in the figure below:



The description of the above input parameters is as follows:

GUI Fields	Description
User name	Creator/owner of the IMAS IDSs database
IMAS database name	IMAS database label, usually device/machine name of the IMAS IDS database (i. e. iter, aug, west. . .)
Shot number	Pulse shot number
Run number	Pulse run number

2.1.2 Available benchmark IMAS databases

On the GateWay HPC there are a few **benchmark IMAS IDS cases** available. These databases are the main source of data used for IMASViz testing purposes and were also included in writing the this documentation. Users can freely use them for examples and practice purposes.

Note: There IMAS IDS cases are confirmed to work with IMAS versions **3.19.1** - **3.20.0**.

Available IMAS IDS Case Parameters			
Parameters	Case 1	Case 2	Case 3
User name	g2penkod	g2penkod	g2penkod
IMAS database name	viztest	viztest	viztest
Shot number	52344	52682	53223
Run number	0	0	0

2.1.3 Running IMASViz from source

The procedure below describes how to run IMASViz from source.

2.1.3.1 Requirements

The fundamental requirements in order to locally run IMASViz are:

- IMAS
- **Python3 and Python libraries:**

- PyQt5
- pyqtgraph
- matplotlib
- Sphinx (**pip3 install sphinx**)
- Sphinx RTD theme (**pip3 install sphinx_rtd_theme**)

2.1.3.2 Obtaining the source code

To obtain the IMASViz code source the next two steps are required:

1. Clone repository from **git.iter.org** (permissions are required!).

   Direct link to the **IMASViz** git.iter repository: IMASViz.

2. Switch to IMASViz2.0 branch (required if master branch is not updated yet)

   ```
   git fetch # optional
   git branch -r # optional
   git checkout viz2.0_develop
   ```

2.1.3.3 Setting the environment

To set the environment, go to `viz` directory and set *VIZ_HOME* and *VIZ_PRODUCTION* environment variables by running the next commands in the terminal:

```
cd viz
# bash
export VIZ_PRODUCTION=0
export VIZ_HOME=$PWD
# csh
setenv VIZ_PRODUCTION 0
setenv VIZ_HOME $PWD
```

Then proceed with the next instructions.

GateWay HPC

Load next modules:

```
module load cineca
module load imasenv
module load itm-gcc/6.1.0
module load itm-python/3.6
module load itm-qt/5.8.0
```

ITER HPC

Load next module:

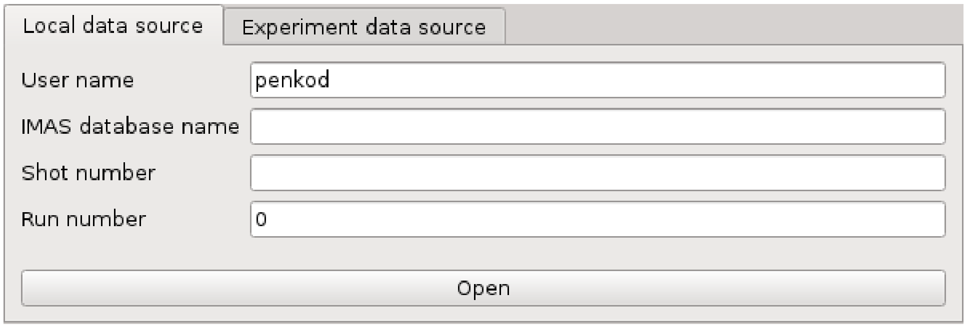
```
module load IMAS/3.20.0-3.8.3
```

2.1.3.4 Running IMASViz

To run IMASViz, run the next commands in terminal:

```
python3 $VIZ_HOME/imasviz/VizGUI/QtVIZ_GUI.py
```

The main GUI window of IMAS_VIZ should display, as shown in the figure below:



The description of the above input parameters is as follows:

GUI Fields	Description
User name	Creator/owner of the IMAS IDSs database
IMAS database name	IMAS database label, usually device/machine name of the IMAS IDS database (i. e. iter, aug, west…)
Shot number	Pulse shot number
Run number	Pulse run number

2.1.4 Latest documentation and manual

The documentation provided on other sources (confluence pages etc.) than the project repository might not be up to date. To get the latest documentation, first obtain the IMASViz source code (see *Obtaining the source code*).

Then navigate to

```
cd $VIZ_HOME/doc
```

and run

```
# for PDF documentation
module load texlive
make pdflatex
xdg-open build/latex/IMASViz.pdf
# for HTML documentation
make html
firefox build/html/index.html
```

Note: Additional prerequisites for generating the documentation: **Sphinx** and **Sphinx RTD** theme (listed in *Requirements*)

2.2 Loading IDS from IMAS local data source

This section describes and demonstrates how to load the **IMAS IDS case** within **IMASViz** and open one of the **IDS nodes**.

Note: The procedure below is executed on the **GateWay HPC** and thus the **IMAS IDS cases** available on the GateWay are used.

2.2.1 Loading IMAS IDS

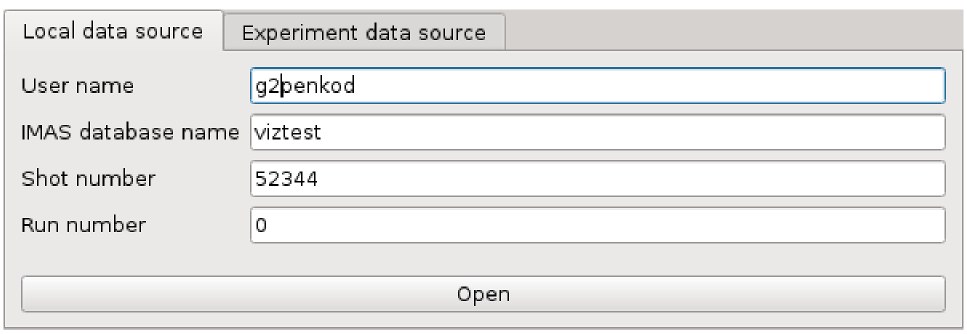
The procedure to load the IDS is as follows:

- In the main *IMASViz GUI*, select the first tab - *Local data source*.
- Enter the following parameters, listed below, to the appropriate text fields.

IMAS IDS case	
Parameters	Values
User name	g2penkod
IMAS database name	viztest
Shot number	52344
Run number	0

By default, the data source is a pulse file located in $HOME/public/imasdb/<IMAS database name>/3/0/ directory. In this case, the ~public/imasdb/viztest/3/0/ directory of user g2penkod.

The filled GUI should then look as shown in the next figure:

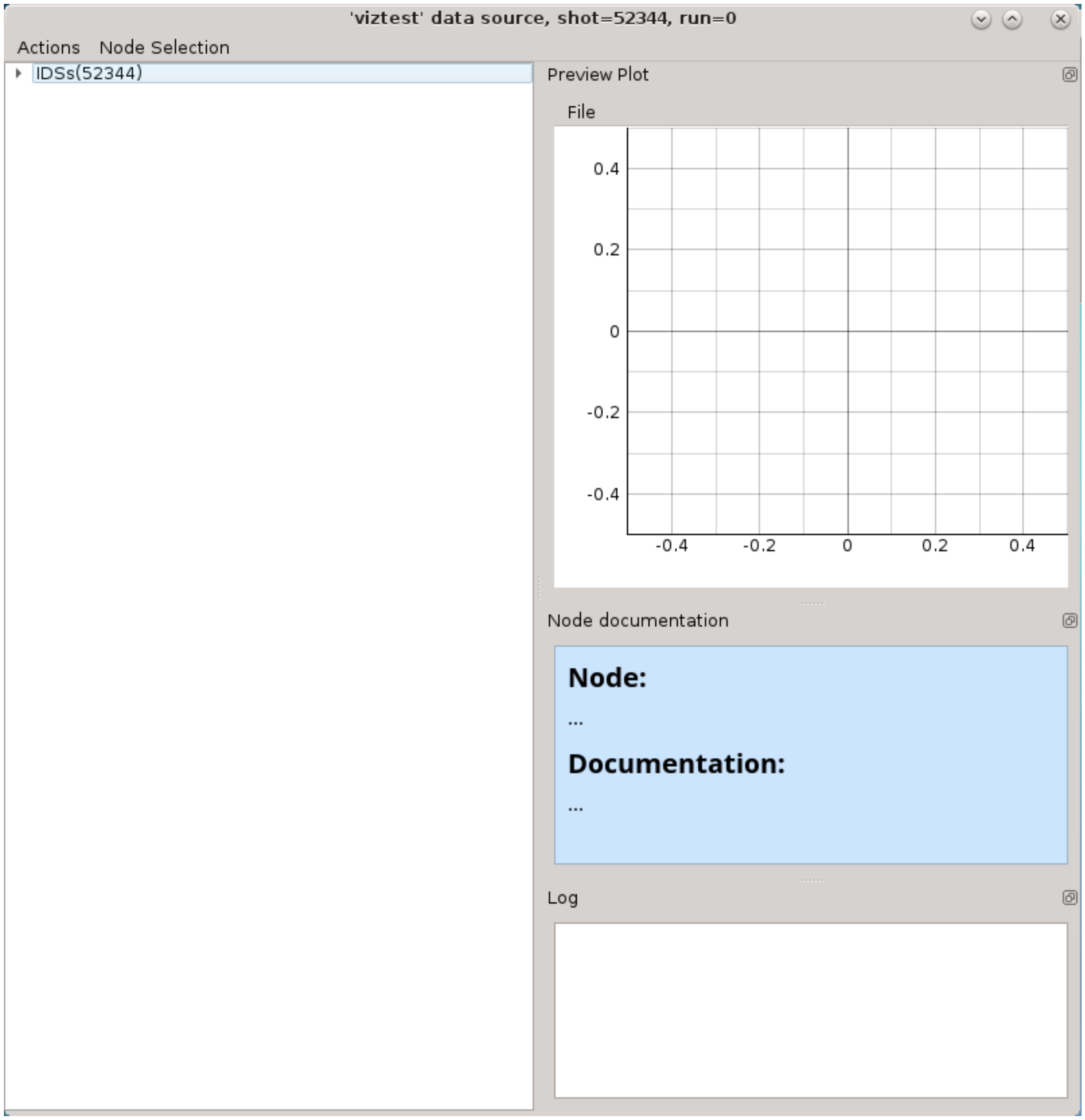


2.2.2 Open IDS

The procedure to open any IDS is the same. In this manual, the procedure will be shown on **magnetics IDS**.

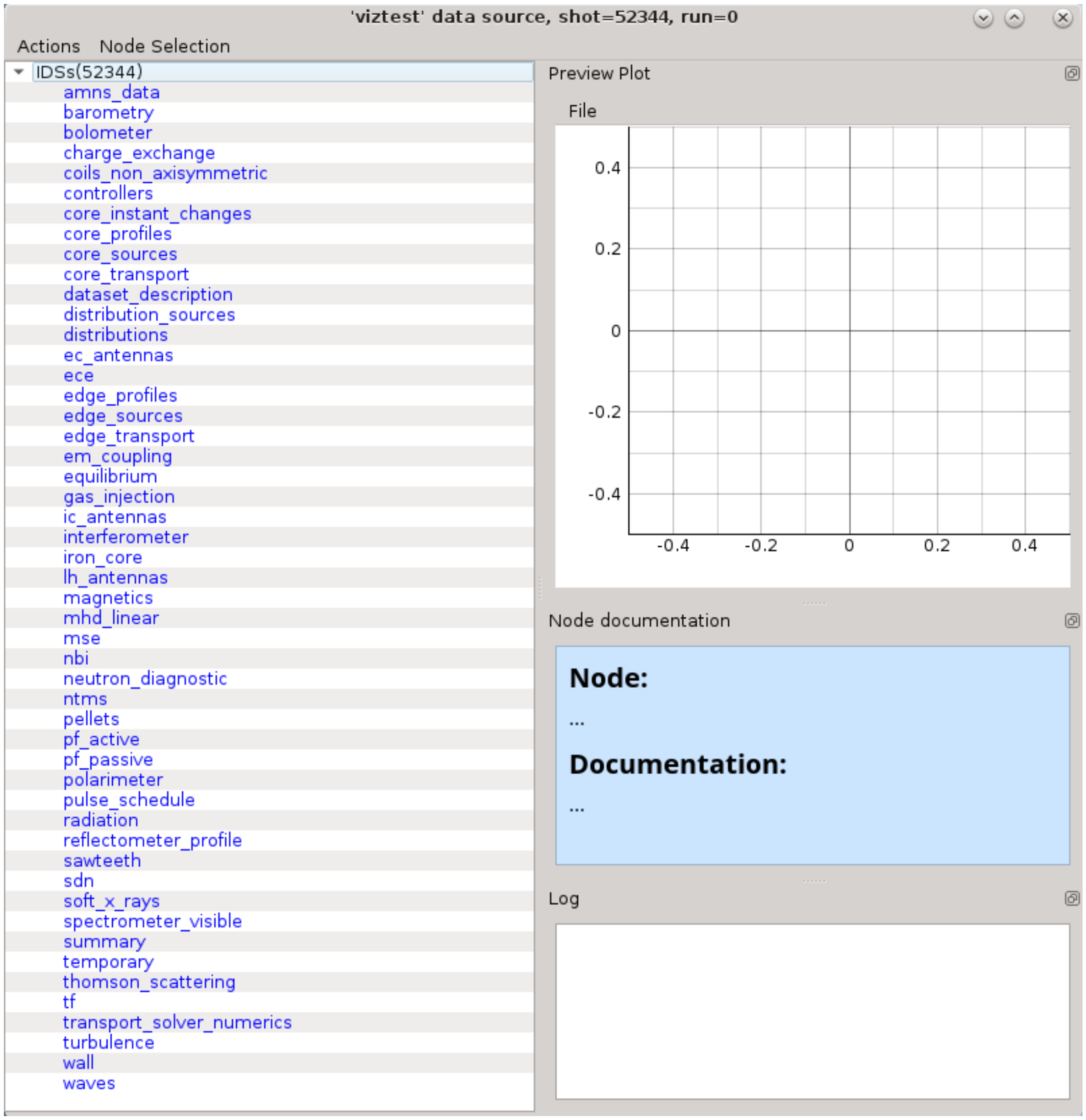
1. Click *Open* button to open the IDS.

   A navigation tree window will open, as shown in the figure below.

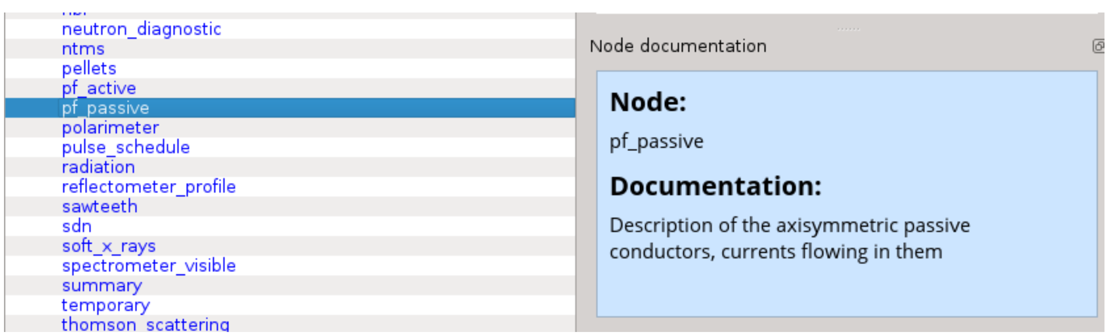


2. Press the **arrow button**  on the left side of the **IDS root node**.

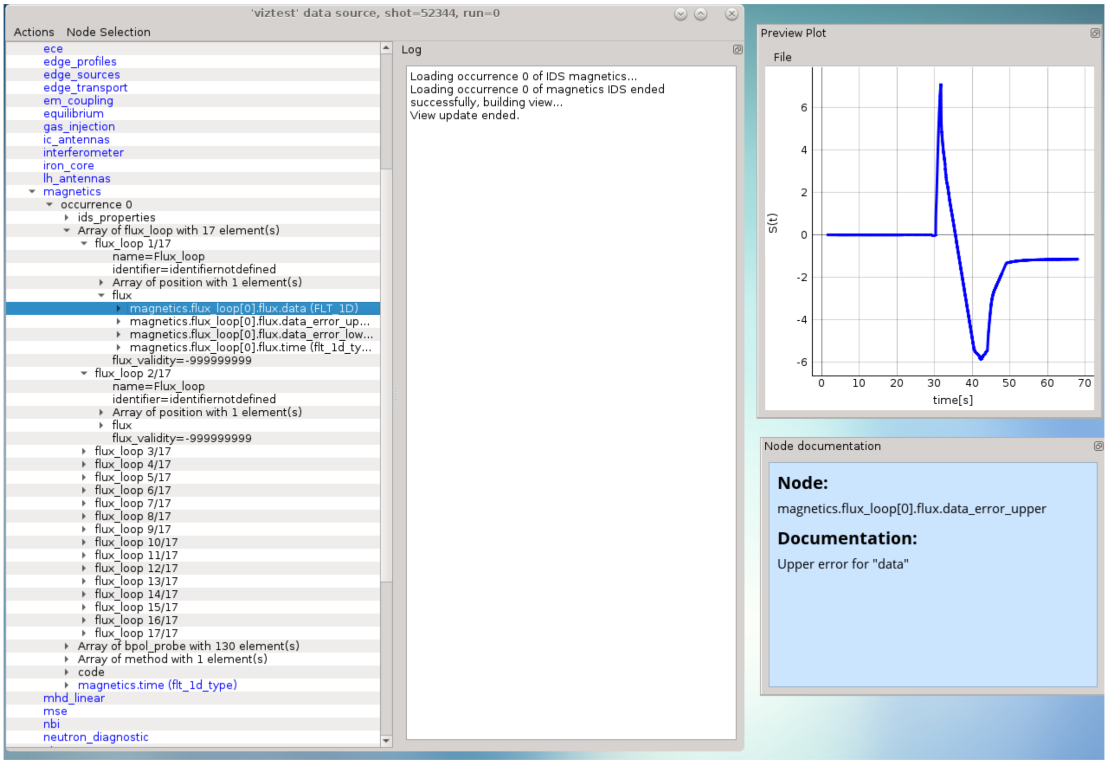
   This will expand the navigation *tree window* and display a list of all IDSs. The tree will allow browsing data for the specific shot number which is displayed by the root node (`IDSs(52344)`).



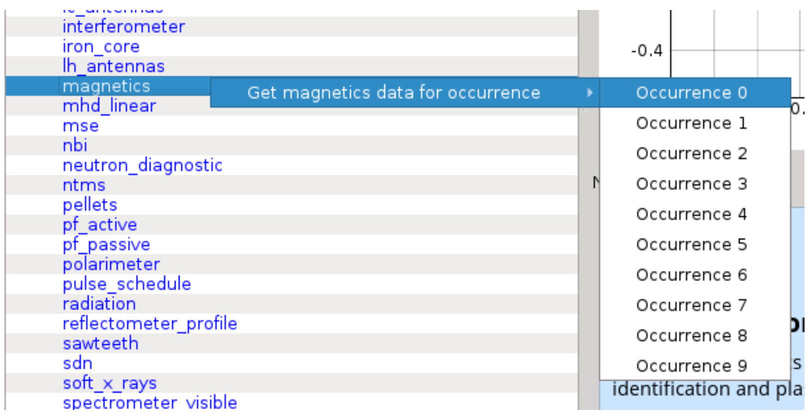
When IDS or node label is selected the *Node documentation* widget will display the basic information (name and documentation) of the node, as shown below.



The *Node Documentation* widget can be freely taken out from the main window by clicking *undock* button  the and positioned anywhere on the screen. The same thing goes for the *Preview Plot* and *Log* widget.

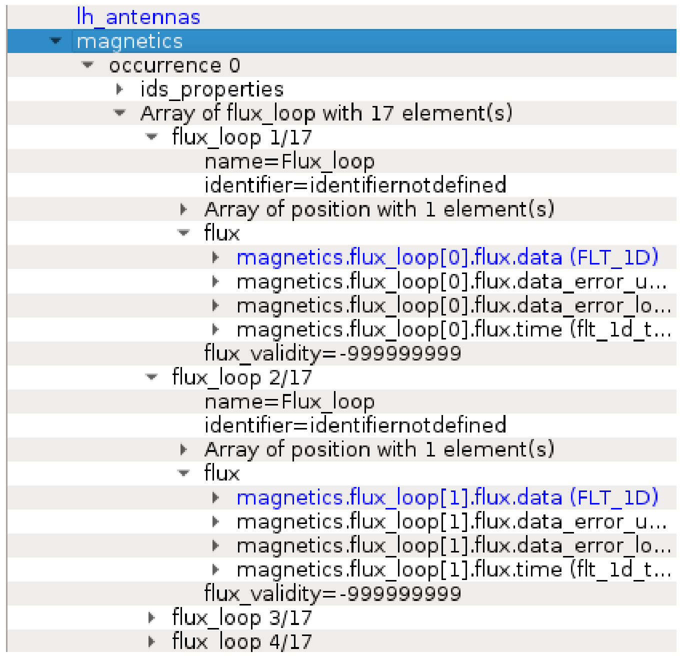


3. Open **magnetics IDS** by right-clicking on the **magnetics** node and selecting the command *Get magnetics data* (occurrence 0) as shown in the figure below.



Note: Alternative: Double-clicking on the **IDS node label** -> **occurrence 0** (default) of the selected IDS will load.

The magnetics IDS nodes are displayed as new nodes in the tree, as shown in the figure below. Nodes of an IDS are organized according to the **IMAS data dictionary**. Inside the **magnetics** tree, plottable **FLT_1D** nodes are colored blue (array length > 0).



2.3 Node selection features

IMASViz offers the user the ability to set or mark a selection of plottable arrays (nodes) as once. This way plotting multiple plots to the same *Figure* or to a *MultiPlot View* is more convenient and faster, avoiding "one-by-one" plotting.

Note: How to plot selection is described later in section *Plotting 1D arrays*.

In the continuation of this section different methods of node selection are described.

2.3.1 Select One-by-one

To select nodes one by one, first, right-click on the wanted node. From the shown pop-up menu, select the command *Select <node name>* ✚.

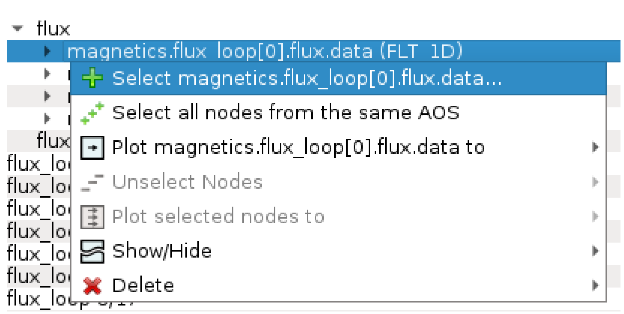


Fig. 1: Selecting a plottable node.

The selected node label gets colored red, indicating that it is added to the selection.

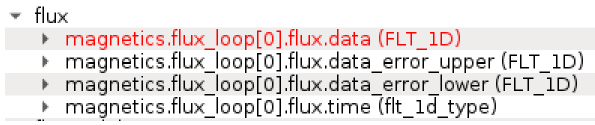


Fig. 2: Node colored red -> node is selected.

Repeat that procedure until all wanted nodes are selected.

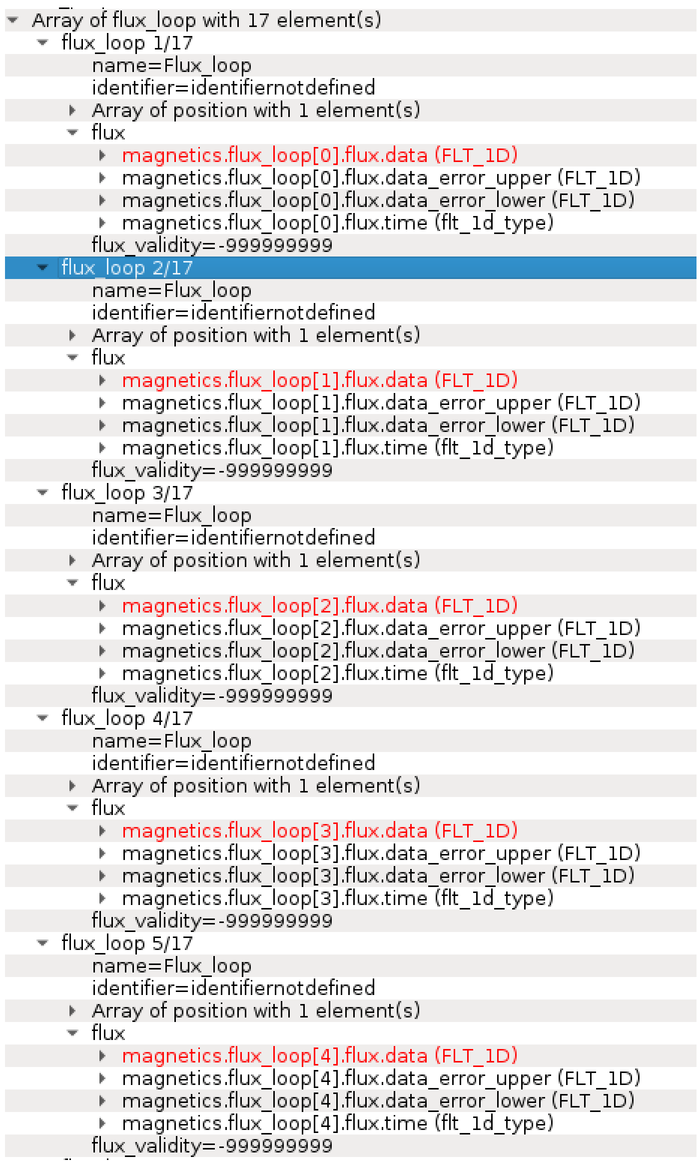


Fig. 3: Example of multiple nodes selection.

Note: At the same time, nodes from other opened IDS databases too can be selected.

2.3.2 Select All Nodes of the same Structure (AOS)

To select all nodes of the same structure (same node structure type), right-click on one of the nodes and from the shown popup-menu select the option *Select All Nodes From The Same AOS* .

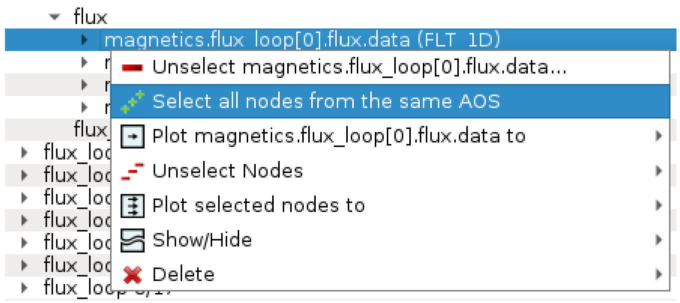


Fig. 4: Selecting plottable nodes of the same structure/type.

All nodes of the same structure will be selected and their label will be colored to red, indicating that they were added to the selection.

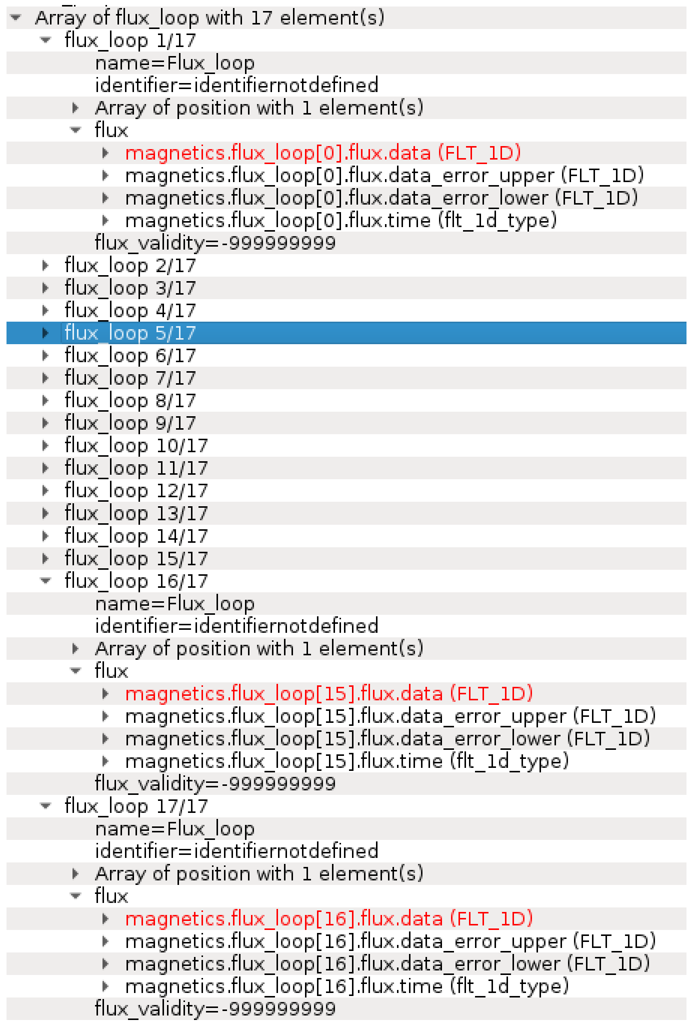


Fig. 5: Node colored red -> node is selected. All plottable nodes of the same structure/type are selected, in this case, 17 nodes.

2.3.3 Save Node Selection Configuration

Any node selection can be saved to a configuration file and used later with any opened IMAS database. To save a selection, follow the next steps:

1. In the main tree browser menu navigate to **Node Selection** -> **Save Node Selection**.
2. In opened GUI window type the name of the configuration.

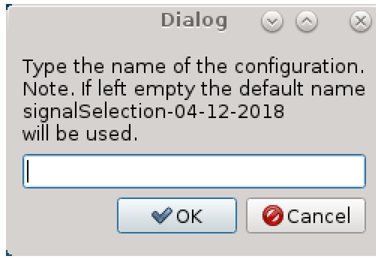


Fig. 6: Save Node Selection Dialog.

3. Press **OK** button.

Note: The configurations are saved to `$HOME/.imasviz` folder.

2.3.4 Apply Selection From Saved Configuration

Applying saved node selection can be performed using both *Node Selection Configuration* and *MultiPlot Configuration*.

2.3.4.1 Apply Selection From Saved Node Selection Configuration

To apply selection from *Node Selection Configuration*, follow the next steps:

1. In the main tree browser menu navigate to **Actions** -> **Apply Configuration**. In the shown window switch to *Available Node Selection Configurations* tab.

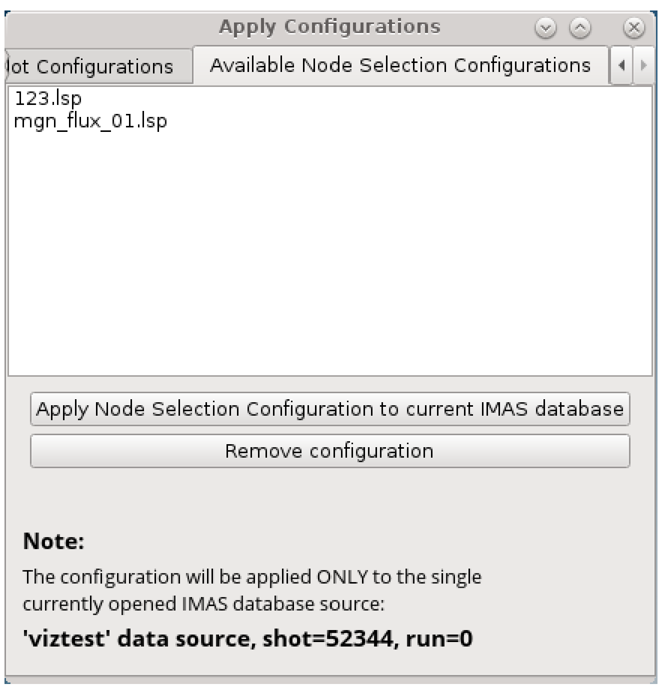


Fig. 7: *Apply Node Selection Configuration* tab.

2. Select the configuration from the list.
3. Press **Apply selection only**.

   The signal nodes, found in the configuration file, will then be selected.

2.3.4.2 Apply Selection From MultiPlot Configuration

To apply selection from *MultiPlot Configuration*, follow the next steps:

See also:

How the create MultiPlot Configuration is described in *Table Plot View*.

1. In Main Tree View Window menu navigate to **Actions** -> **Apply Configuration**. In the shown window switch to *Apply Plot Configuration* tab.

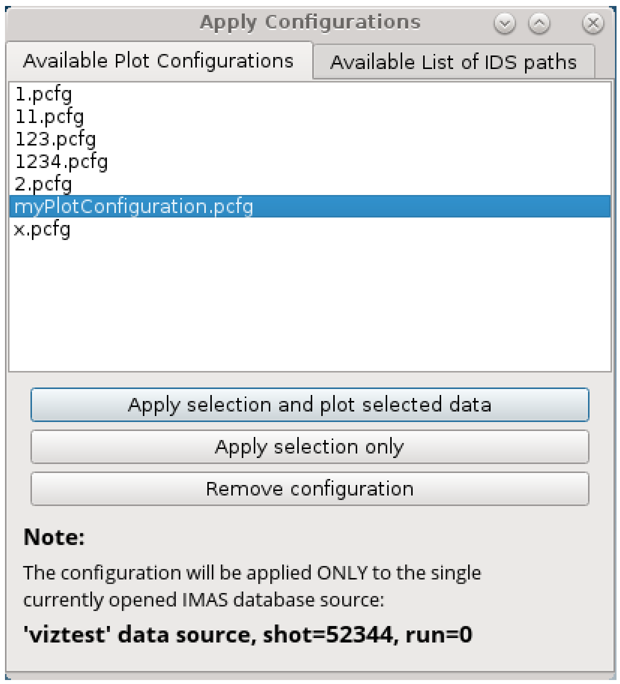


Fig. 8: *Apply Plot Configuration* tab.

2. Select the configuration from the list.
3. Press **Apply selection only**.

   The signal nodes, found in the configuration file, will then be selected.

2.3.5 Unselect selected Node Signals

There are few features that allow node signal unselection.

2.3.5.1 Unselect One-by-one

To unselect nodes one by one, first, right-click on the selected node. From the shown pop-up menu, select the command *Unselect <node name>* .

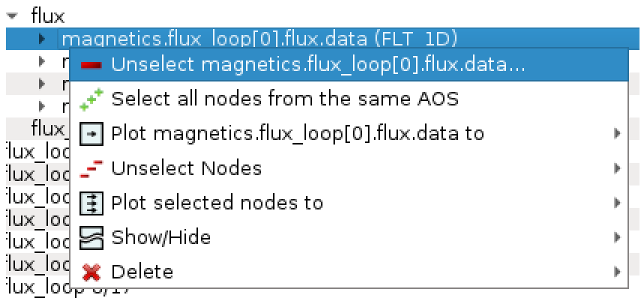


Fig. 9: Unselecting plottable node.

2.3.5.2 Unselect All

To unselect all selected nodes, first, right-click on the selected node. From the shown pop-up menu, select the command *Unselect Nodes* -> *This IMAS Database* or *All IMAS Databases*.

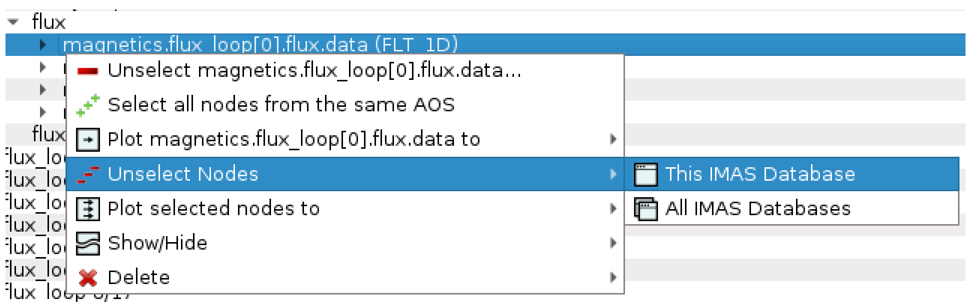


Fig. 10: Unselecting multiple plottable nodes at once.

2.4 Plotting 1D arrays

The plotting of 1D arrays option and plot handling is the main feature and purpose of the *IMASViz*.

This section describes the basics of plotting a 1D array, stored in the IDS, and how to handle the existing plots.

2.4.1 Plotting a single 1D array to plot figure

The procedure to plot 1D array is as follows:

1. Navigate through the **magnetics IDS** and search for the node containing **FLT_1D** data, for example **magnetics.flux_loop[0].flux.data**. Plottable FLT_1D nodes are colored **blue** (array length > 0).

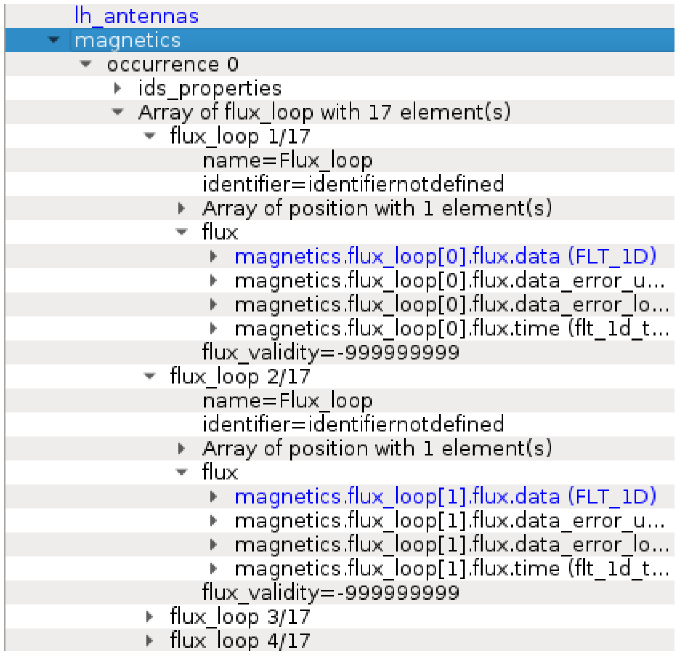


Fig. 11: Example of plottable FLT_1D node.

By clicking on the node the preview plot will be displayed in the *Preview Plot*, located in the main window. This feature helps to quickly check how the data, stored in the FLT_1D, looks when plotted.

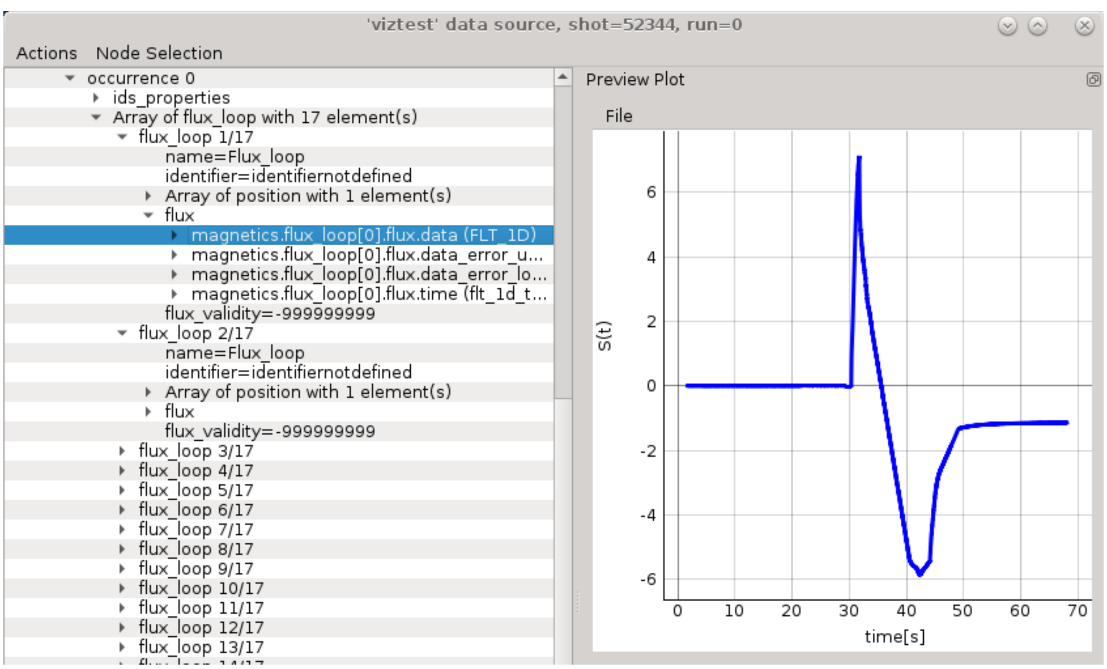


Fig. 12: Preview Plot

2. Right-click on the **magnetics.flux_loop[0].flux.data (FLT_1D)** node.
3. From the pop-up menu, select the command *Plot ids.magnetics.flux_loop[0].flux.data to* -> *figure* -> *New*

.

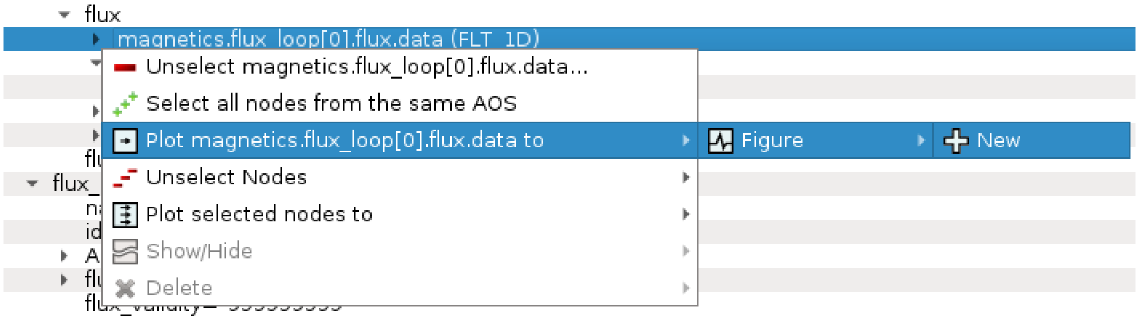


Fig. 13: Navigating through right-click menu to plot data to plot figure.

The plot should display in plot figure as shown in the image below.

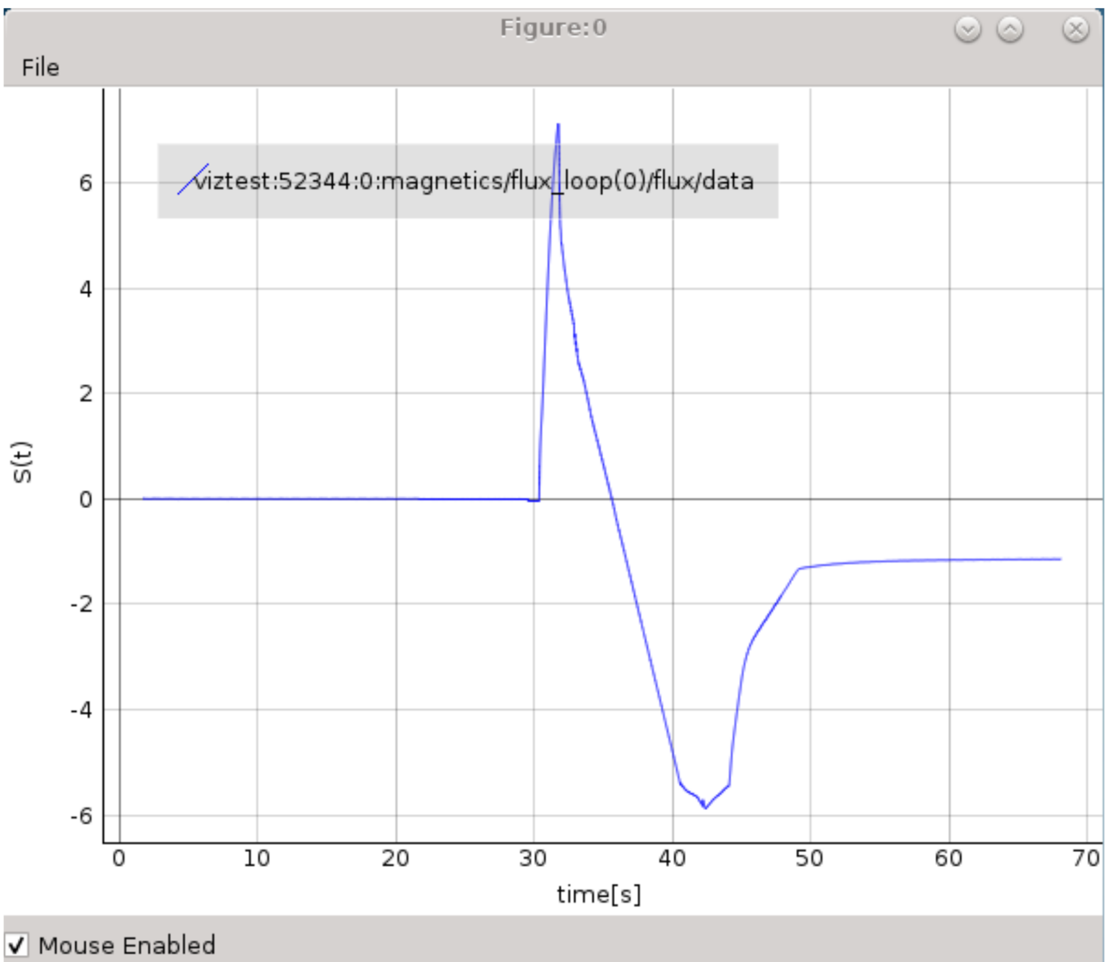


Fig. 14: Basic plot figure display.

2.4.1.1 Basic plot display features

The below features are available for any *plot display*. Most of them are available in the right-click menu.

Note: Term *Plot Display* is used for any base subwindow for displaying plots. Following that the *Plot Figure* contains a single *Plot Display*, while *Table Plot View* and *Stacked Plot View* consist of multiple *Plot Displays*.

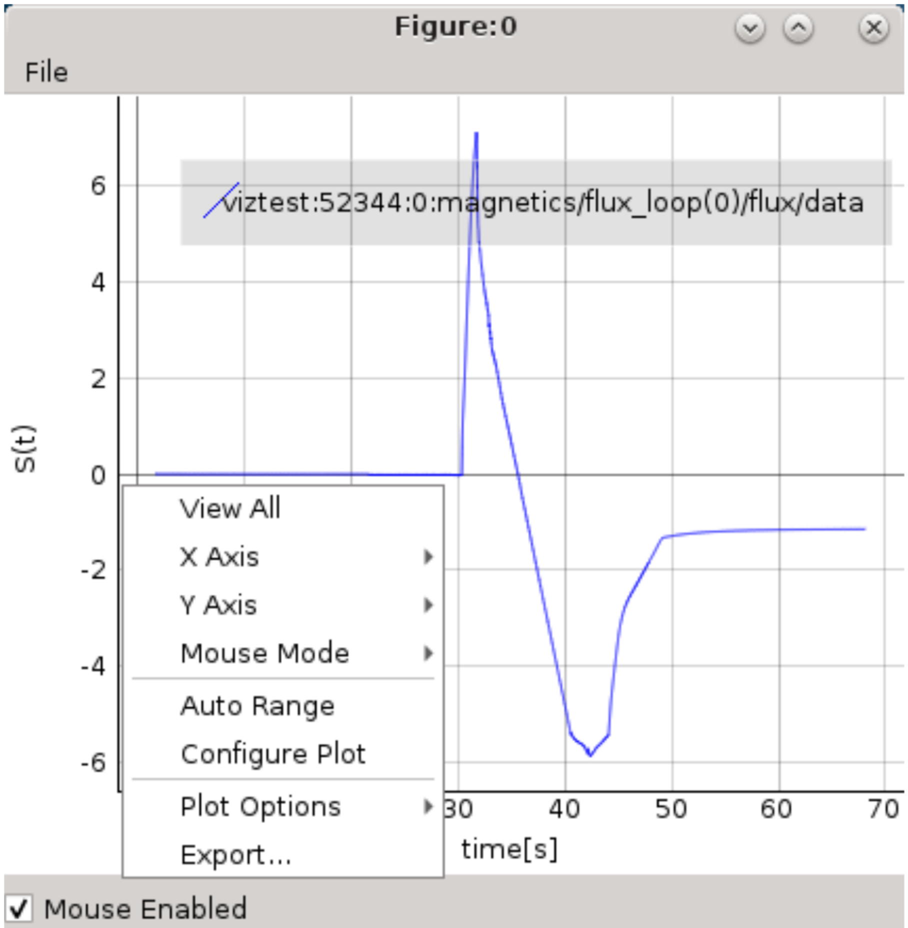


Fig. 15: Plot display window right-click menu.

View All

Zoom to view whole plot area.

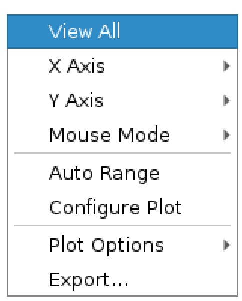


Fig. 16: *View All* feature in the right-click menu.

Auto Range

Similar to *View All* feature with the difference that it shows plot area between values `X_min` -> `X_max` and `Y_min` -> `Y_max`, without additional "plot margins" on the sides.

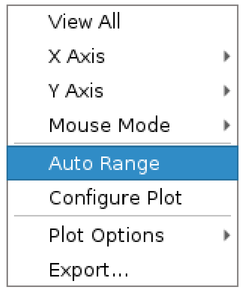


Fig. 17: *Auto Range* feature in the right-click menu.

Left Mouse Button Mode Change

Change between *Pan Mode* (move plot around) and *Area Zoom Mode* (choose selectable area to zoom into).

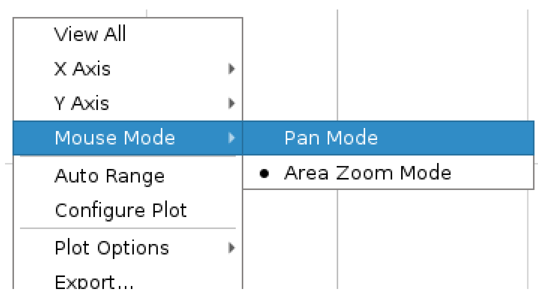


Fig. 18: *Mouse Mode* feature in the right-click menu.

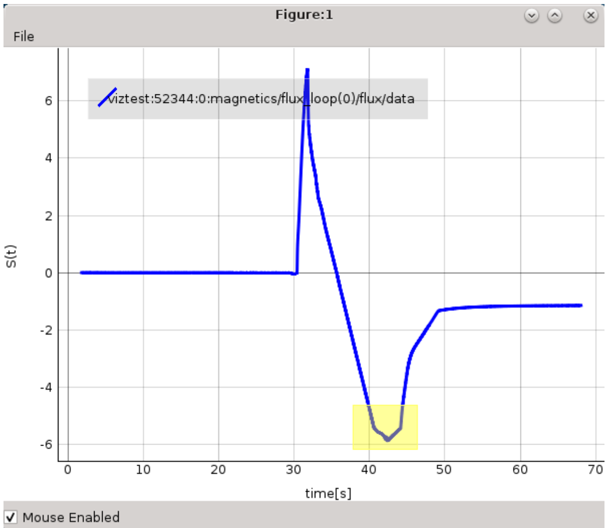


Fig. 19: *Area Zoom* example: Marking zoom area using.

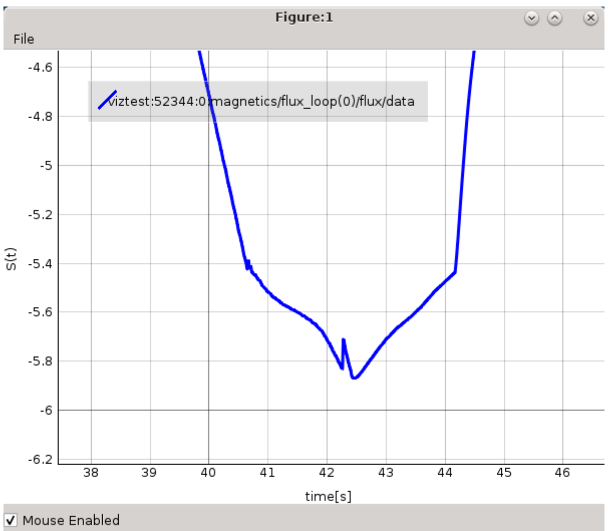


Fig. 20: *Area Zoom* example: Result.

Axis options

X and Y axis range, inverse, mouse enable/disable options and more.

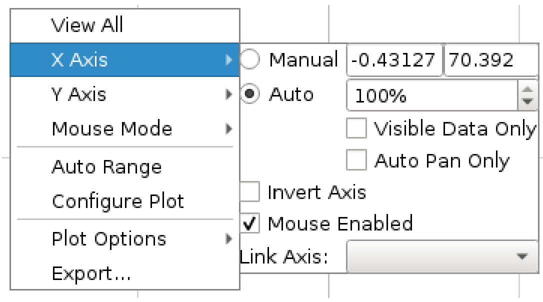


Fig. 21: *Axis Options* feature in the right-click menu.

Plot Configuration and Customization

Setting color and line properties of plots shown in the Plot Display.

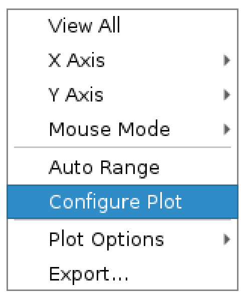


Fig. 22: *Configure Plot* feature in the right-click menu.

Each plot can be customized. By selecting this feature a separate GUI window will open, listing all plots within the plot display window and their properties that can be customized.

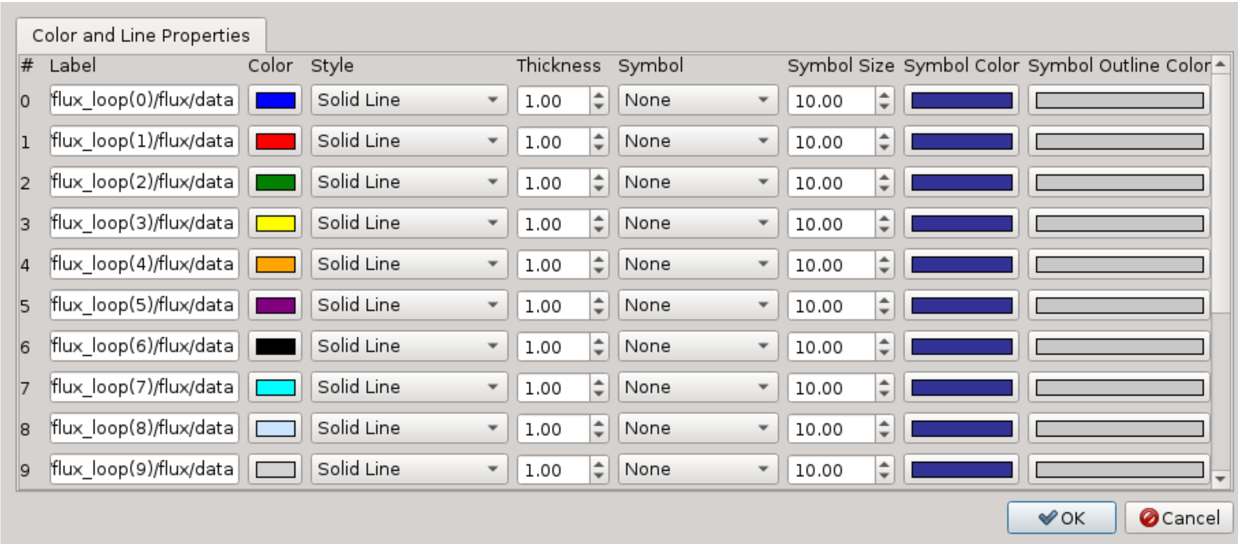


Fig. 23: *Configure Plot* GUI.

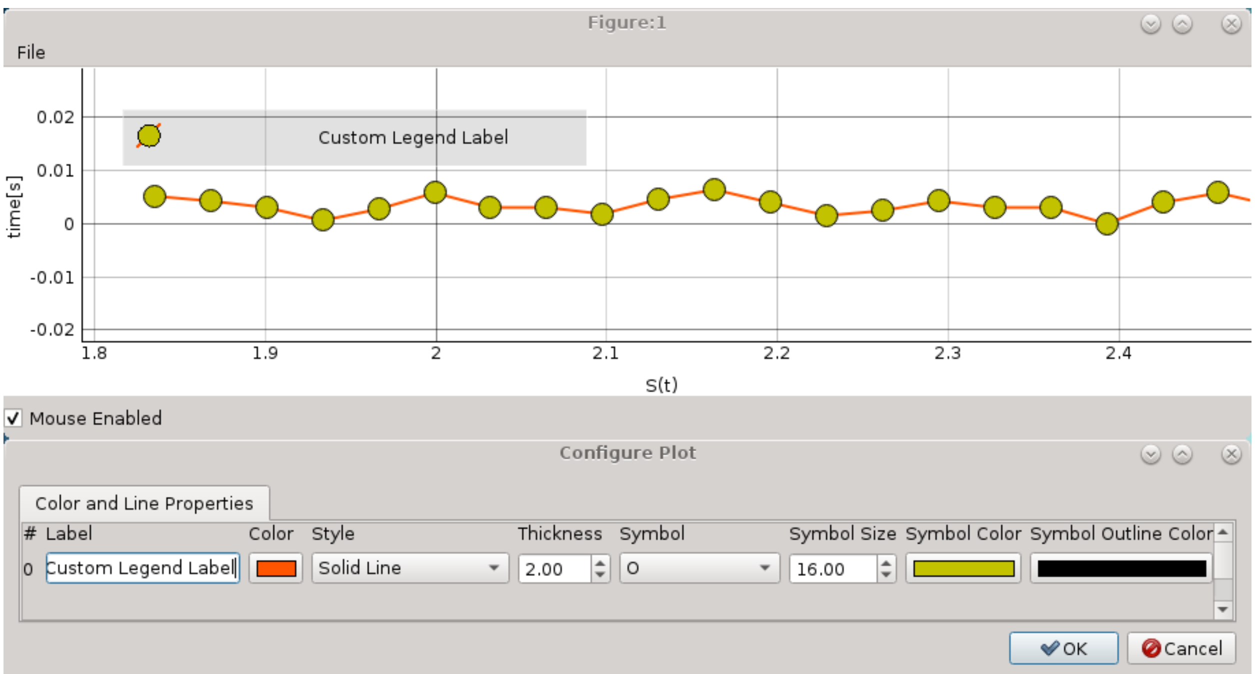


Fig. 24: Plot configuration example for single plot.

Plot options

Enable/Disable grid, log scale and more.

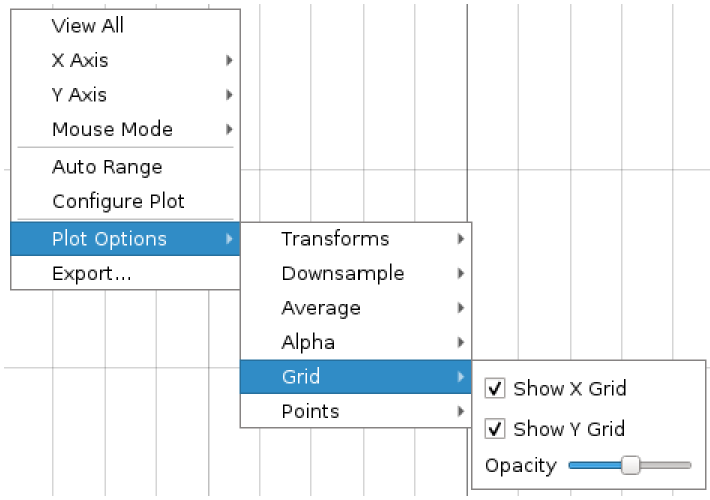


Fig. 25: *Plot Options* feature in the right-click menu.

Export feature

The *Plot Display* scene can be exported to:

- image file (PNG, JPG, . . .). A total of **16** image formats are supported.
- scalable vector graphics (SVG) file
- matplotlib window
- CSV file
- HDF5 file

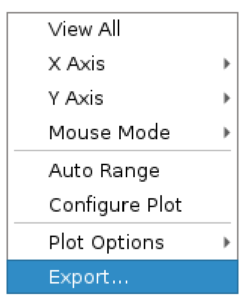


Fig. 26: *Export* feature in the right-click menu.

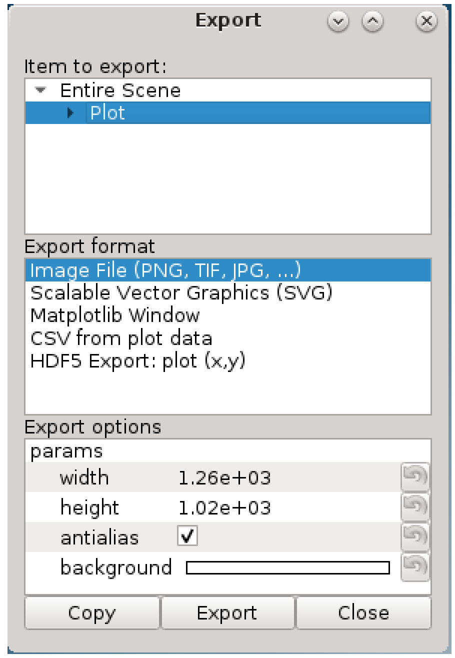


Fig. 27: Export GUI window.

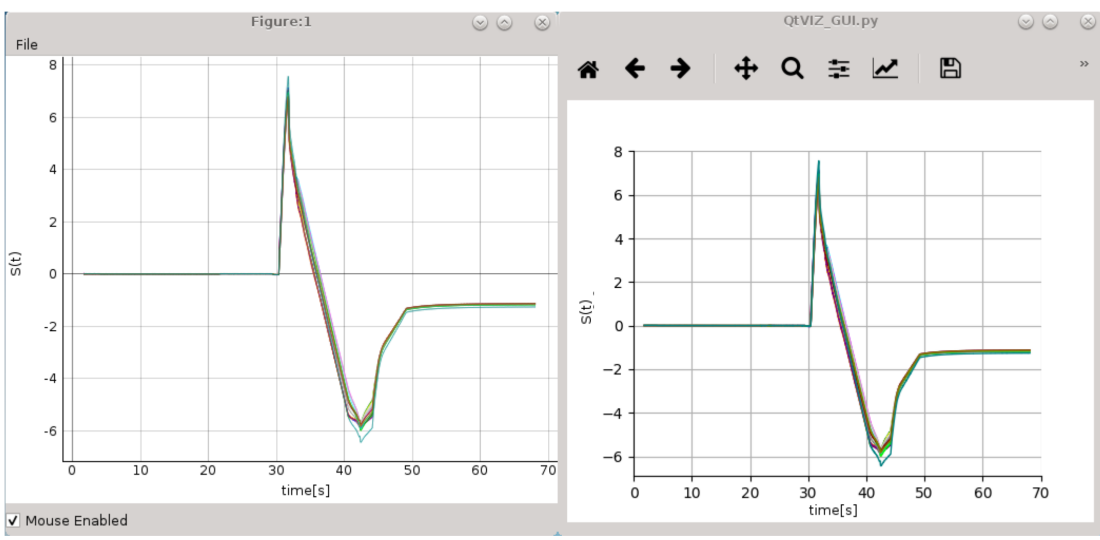


Fig. 28: Comparison of IMASViz *Plot Figure* and *matplotlib window*

2.4.2 Adding a plot to existing figure

The procedure of adding a plot to an already existing figure is as follows:

1. From the previous navigation tree, navigate to the wanted node, for example **ids.magnetics.flux_loop[16].flux.data**
2. Right-click on the node.
3. From the pop-up menu, navigate and select *Plot <node name> to* -> *Figure* -> *Figure:0*

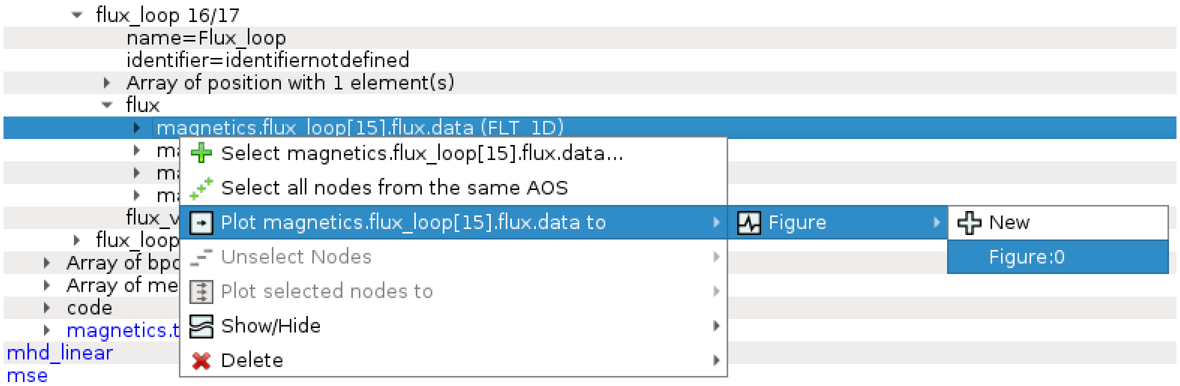


Fig. 29: Plotting to existing figure.

The plot will be added to the selected existing plot as shown in the image below.

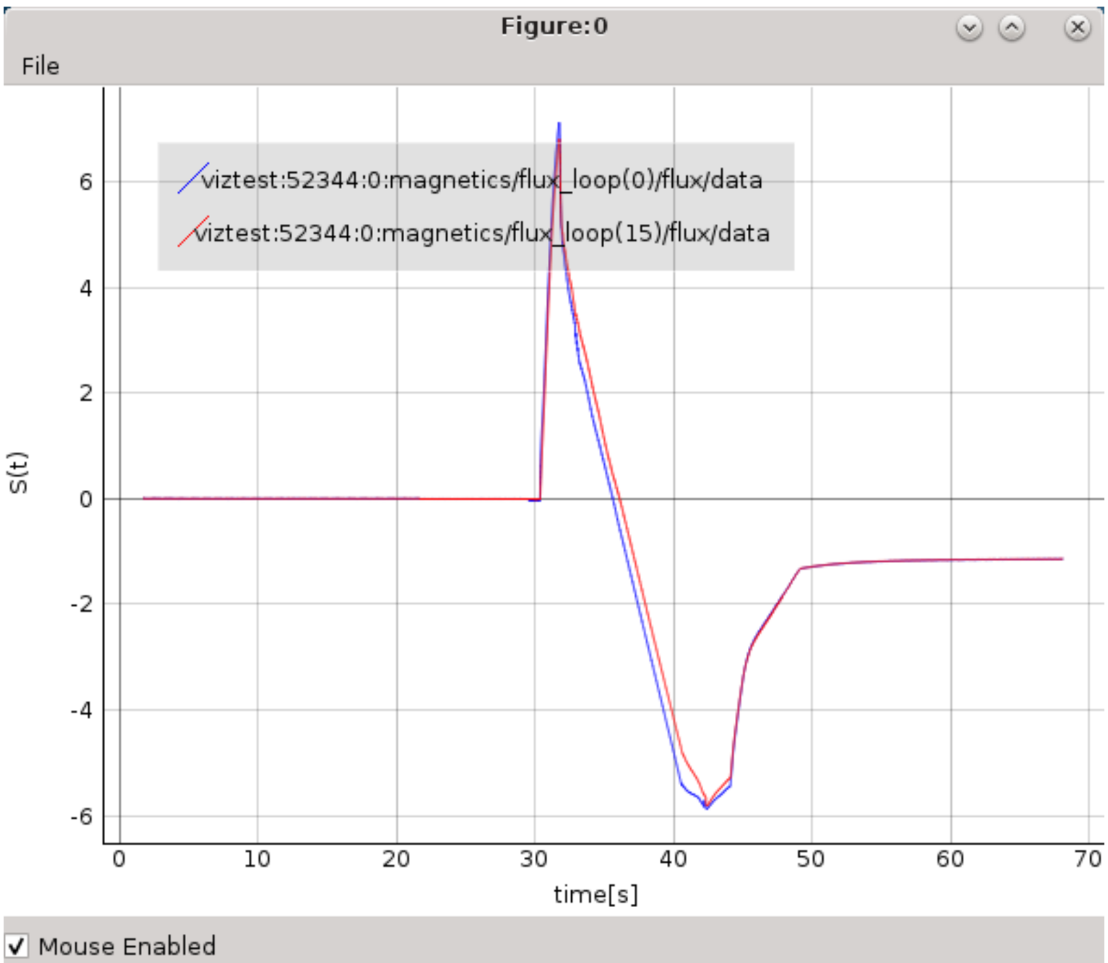


Fig. 30: Plotting to existing figure - result.

2.4.3 Comparing plots between two IDS databases

IMASViz allows comparing of FLT_1D arrays between two different IDS databases (different shots too). The procedure is very similar to the one presented in the section *Adding a plot to existing figure*:

1. Open another IMAS database, same as shown in section *Loading IDS from IMAS local data source*. In this manual this will be demonstrated using IDS with *shot* **52682** and *run* **0** parameters.

Manual IDS case	
parameters	values
User name	g2penkod
IMAS database name	viztest
Shot number	52682
Run number	0

2. Load **occurrence 0** of **magnetics** IDS
3. Navigate through the IDS search for the wanted node, for example **ids.magnetics.flux_loop[0].flux.data**.
4. Right-click on the node.
5. From the pop-up menu, navigate and select *Plot <node name> to* -> *Figure* -> *Figure:0*

The plot will be added to the existing plot as shown in the image below.

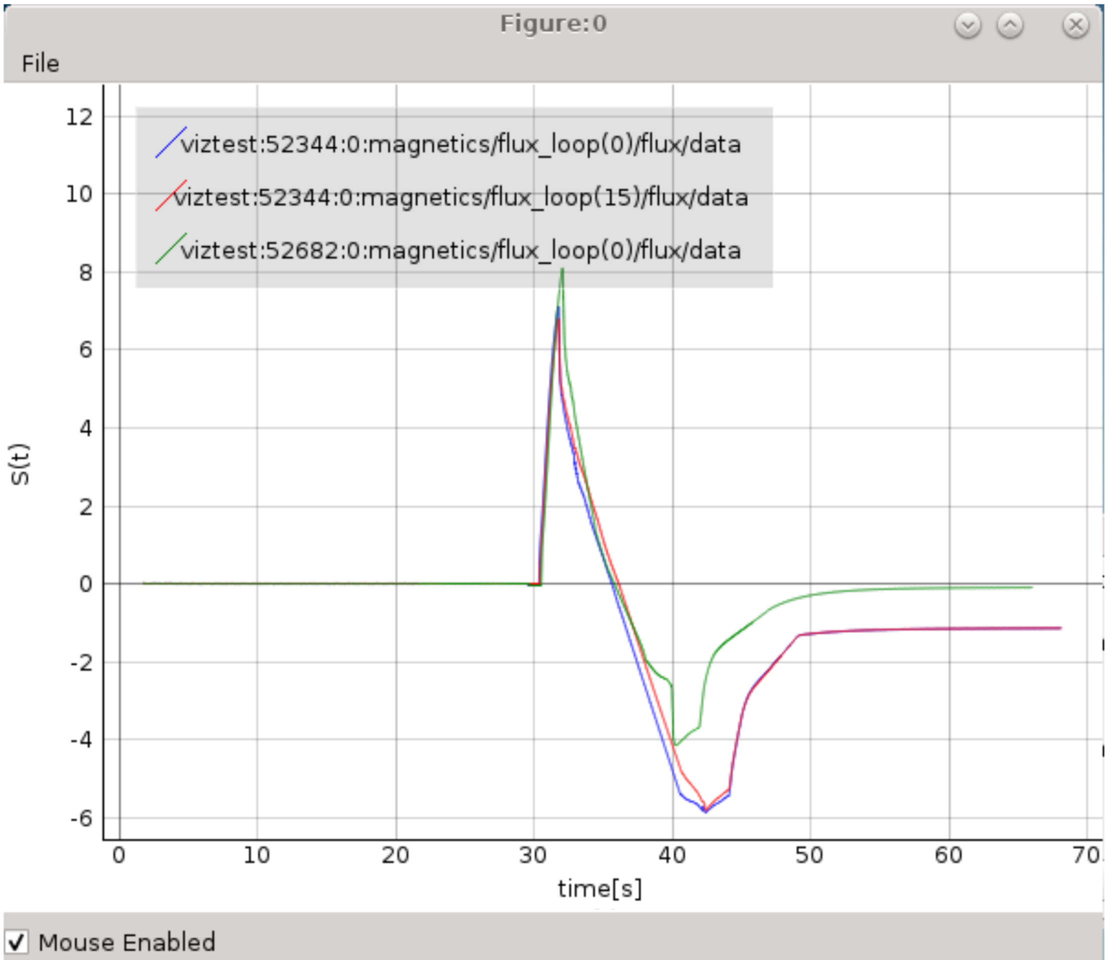


Fig. 31: Plotting from other IDS to existing figure - result.

2.4.4 Plotting a selection of 1D arrays to figure

The procedure of 1D arrays selection and plotting to figure is as follows:

1. In main tree view window set a selection of nodes holding 1D arrays.

Note: How to create a selection of arrays is described in section *Node selection features*.

2. When finished with node selection, either: - right-click on any FLT_1D node, or - click *Node Selection* menu on menubar of the main tree view window.
3. From the pop-up menu, navigate and select *Plot selected nodes to* -> *Figure* -> *New* -> *This IMAS database*.

Note: The same procedure applies plotting the selection to an existing figure.

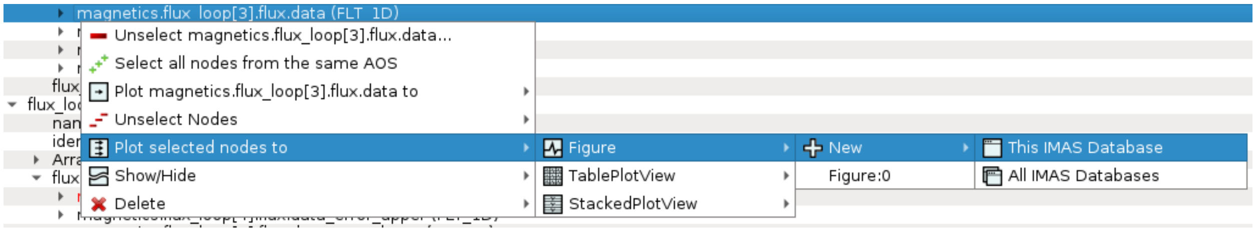


Fig. 32: Plotting selection to a new figure using selection from the currently opened IDS database.

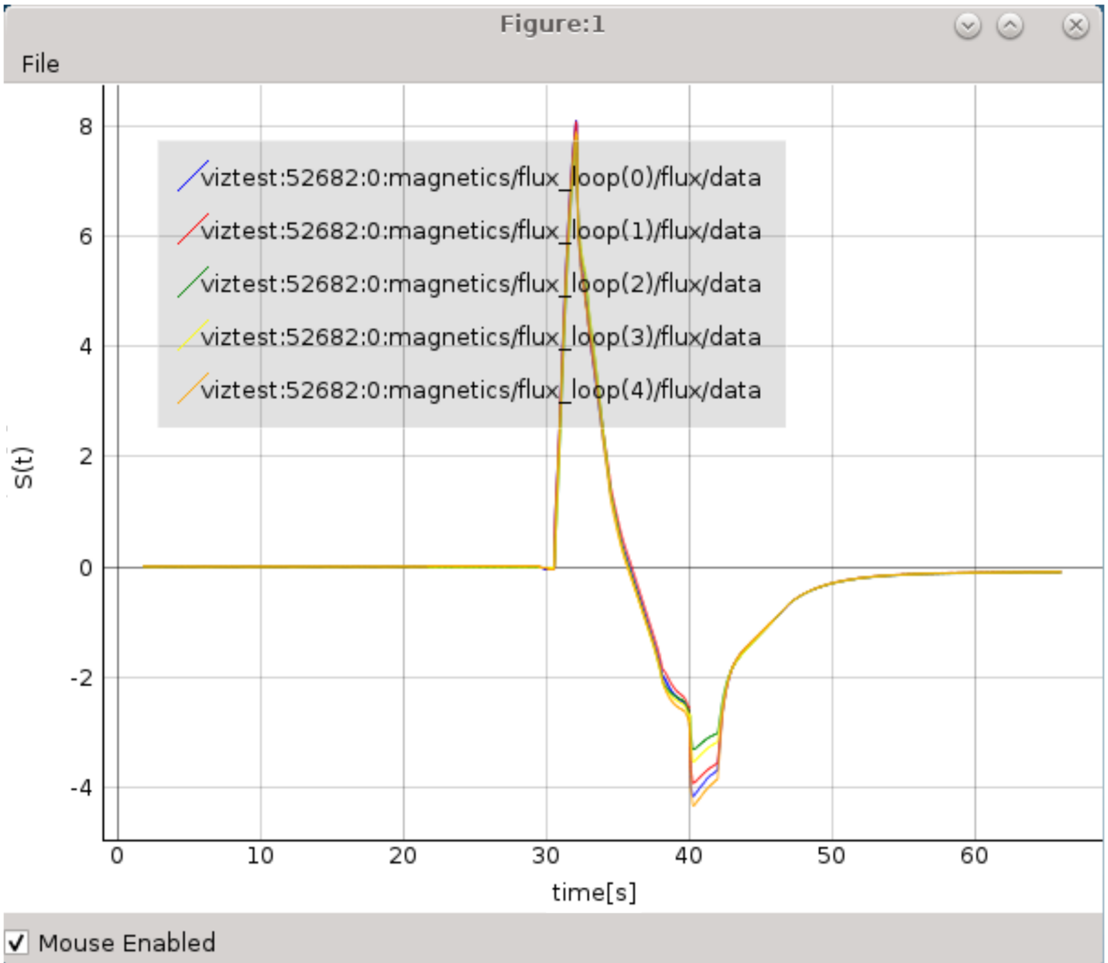


Fig. 33: Example of plot figure, created by plotting data from node selection.

2.4.5 Plotting 1D array as a function of coordinate1 along the time axis

One of the IMASViz features is plotting coordinate along the time axis. This is allowed for IDS nodes, located within **time_slice[:]** structure, and it is already set as a default plotting feature for such arrays.

The procedure to plot such 1D array is quite identical as in section *Plotting a single 1D array to plot figure*. The procedure is described and demonstrated on **equilibrium.time_slice[0].profiles_1d.phi** (Torodial Flux) array.

1. Navigate through the **equilibrium IDS** and search for the time slice node containing **FLT_1D** data, for example **equilibrium.time_slice[0].profiles_1d.phi**.

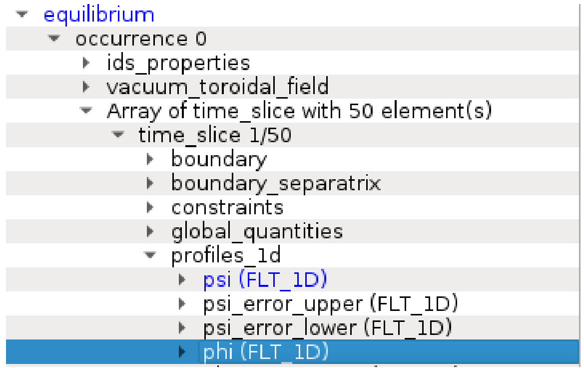


Fig. 34: Example of plottable FLT_1D time slice node.

2. Right-click on the **equilibrium.time_slice[0].profiles_1d.phi (FLT_1D)** node.
3. From the pop-up menu, select the command *Plot equilibrium.time_slice[0].profiles_1d.phi to* -> *figure* -> *New*.

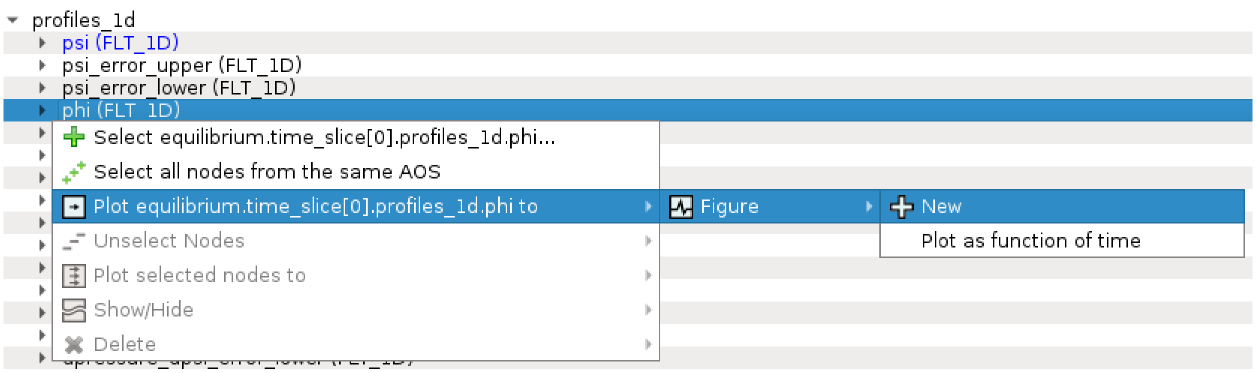


Fig. 35: Navigating through right-click menu to plot data to plot figure.

The plot should display in plot figure as shown in the image below. Note that **coordinate1 = ids.equilibrium.time_slice[0].profiles_1d.psi** for this FLT_1D data array.

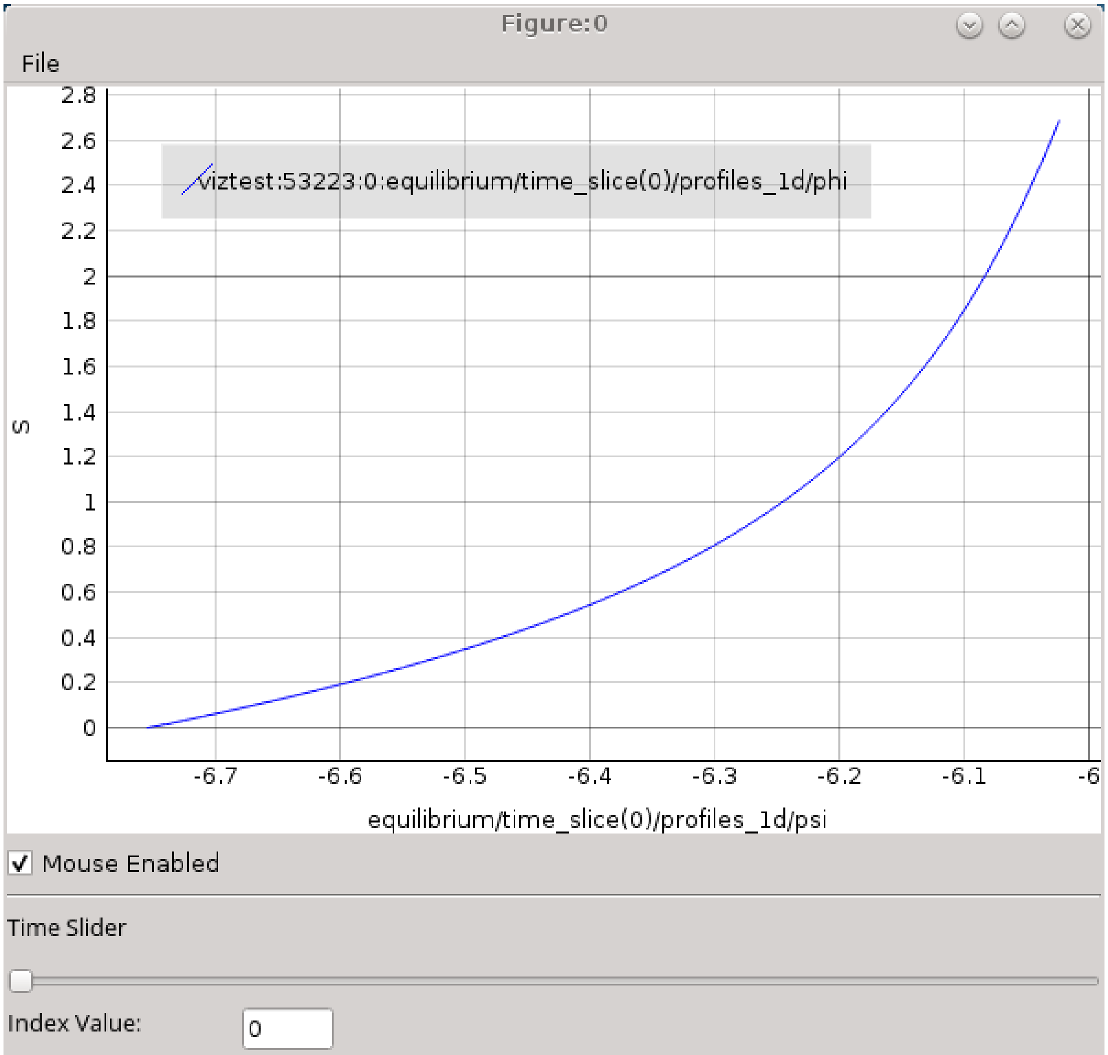


Fig. 36: Time slice plot figure display. The data are represented as a function of coordinate1 (**ids.equilibrium.time_slice[0].profiles_1d.psi**) for the first **phi** time slice (**ids.equilibrium.time_slice[0].profiles_1d.phi**).

The time slider allows you to move along the time axis and the plot will change accordingly.

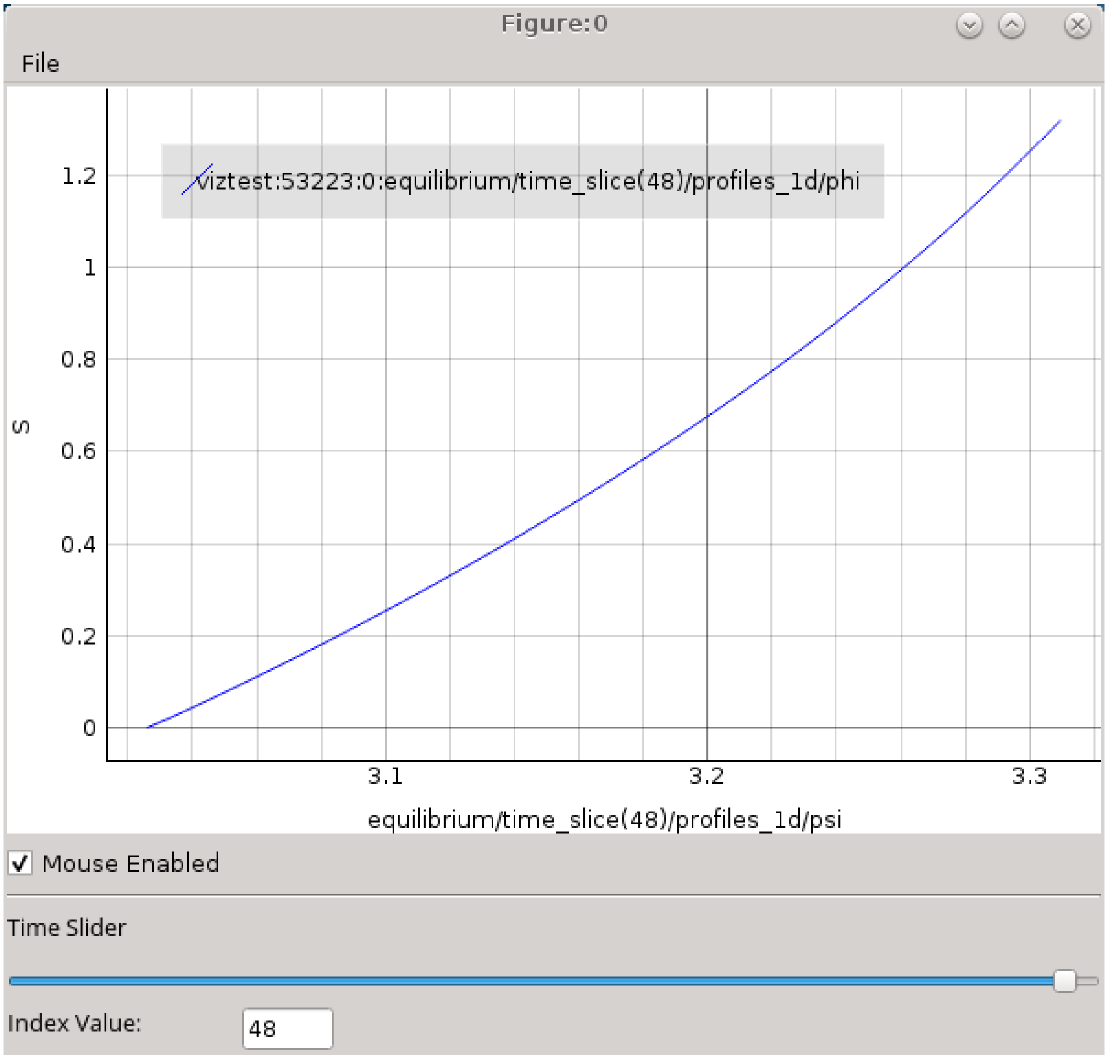


Fig. 37: Time slice plot figure display for **ids.equilibrium.time_slice[48].profiles_1d.phi**. The data are represented as a function of coordinate1 (**ids.equilibrium.time_slice[48].profiles_1d.psi**).

Note: Adding a plot (presented in *Adding a plot to existing figure*) to such existing plot might not work as expected, as the sliding through indexes works directly only the last added plot.

2.4.6 Plotting 1D arrays at index as a function of the time

One of the IMASViz features is plotting array values at certain index as a function of time. This is allowed for IDS nodes, located within **time_slice[:]** structure, and it is already set as a default plotting feature for such arrays.

The procedure is described and demonstrated on **equilibrium.time_slice[0].profiles_1d.f** (Torodial Flux) array.

1. Navigate through the **equilibrium IDS** and search for the time slice node containing **FLT_1D** data, for example **equilibrium.time_slice[0].profiles_1d.f**.

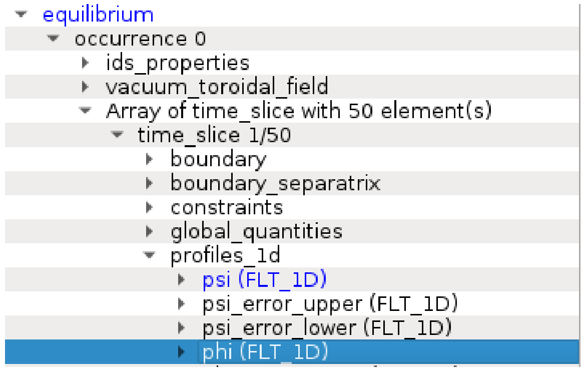


Fig. 38: Example of plottable FLT_1D time slice node.

2. Right-click on the **equilibrium.time_slice[0].profiles_1d.f (FLT_1D)** node.
3. From the pop-up menu, select the command *Plot equilibrium.time_slice[0].profiles_1d.f to* -> *figure* -> *New* .

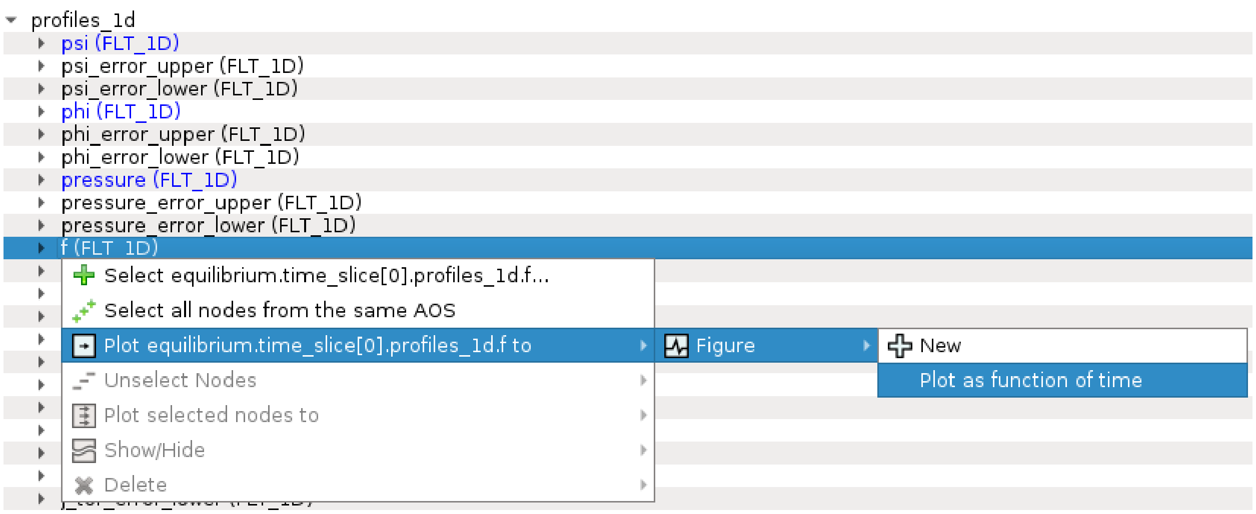


Fig. 39: Navigating through right-click menu to plot data to plot figure.

The plot should display in plot figure as shown in the image below. Note that Y-axis values are an array of **equilibrium.time_slice[:].profiles_1d.f[0]** values through all time slices (marked by **[:]**) and X-axis values are time values found in **equilibrium.time**.

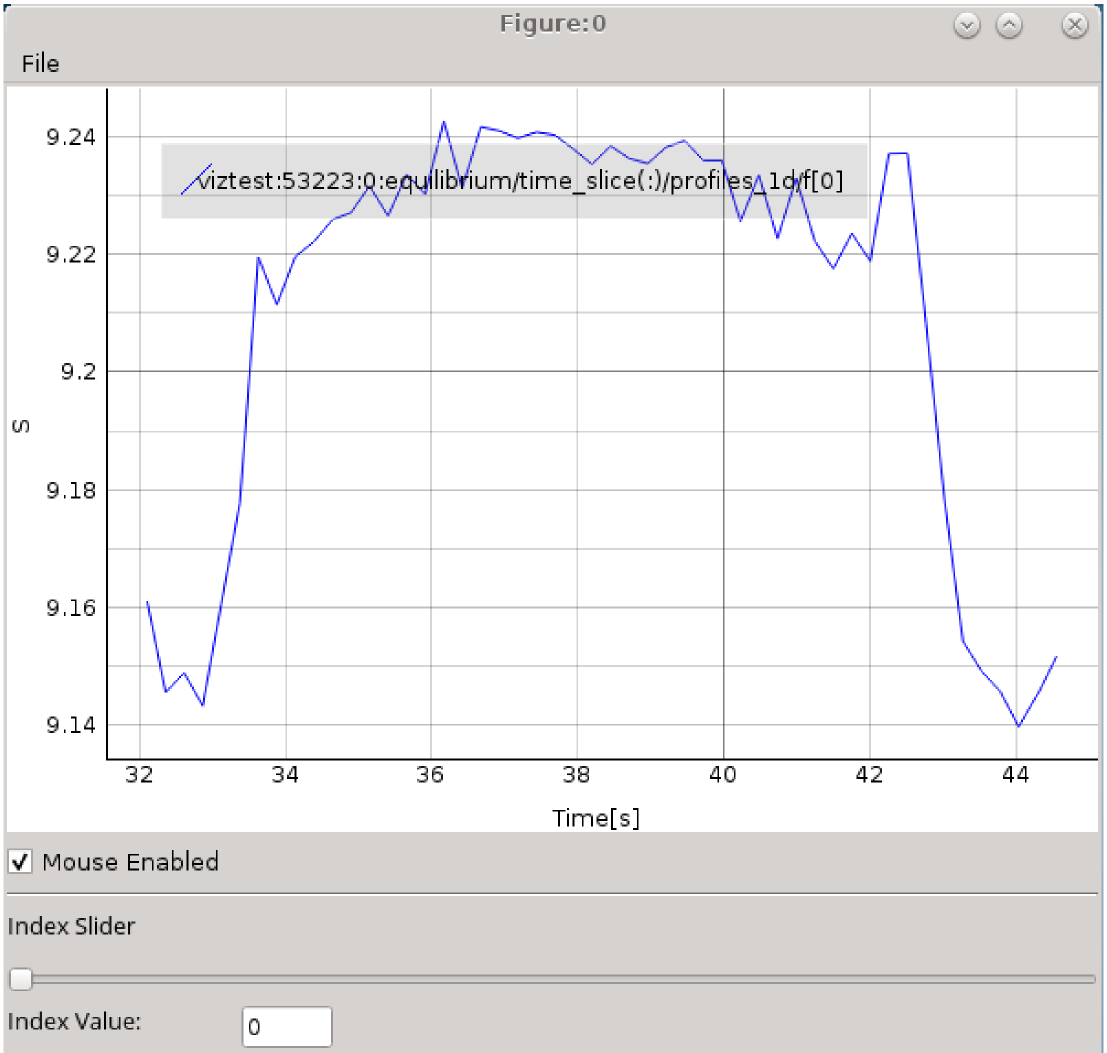


Fig. 40: Plot as a function of time. Y-axis values are an array of **equilibrium.time_slice[:].profiles_1d.f[0]** values through all time slices (marked by **[:]**) and X-axis values are time values found in **equilibrium.time**.

The time slider allows you to move along the array index and the plot will change accordingly.

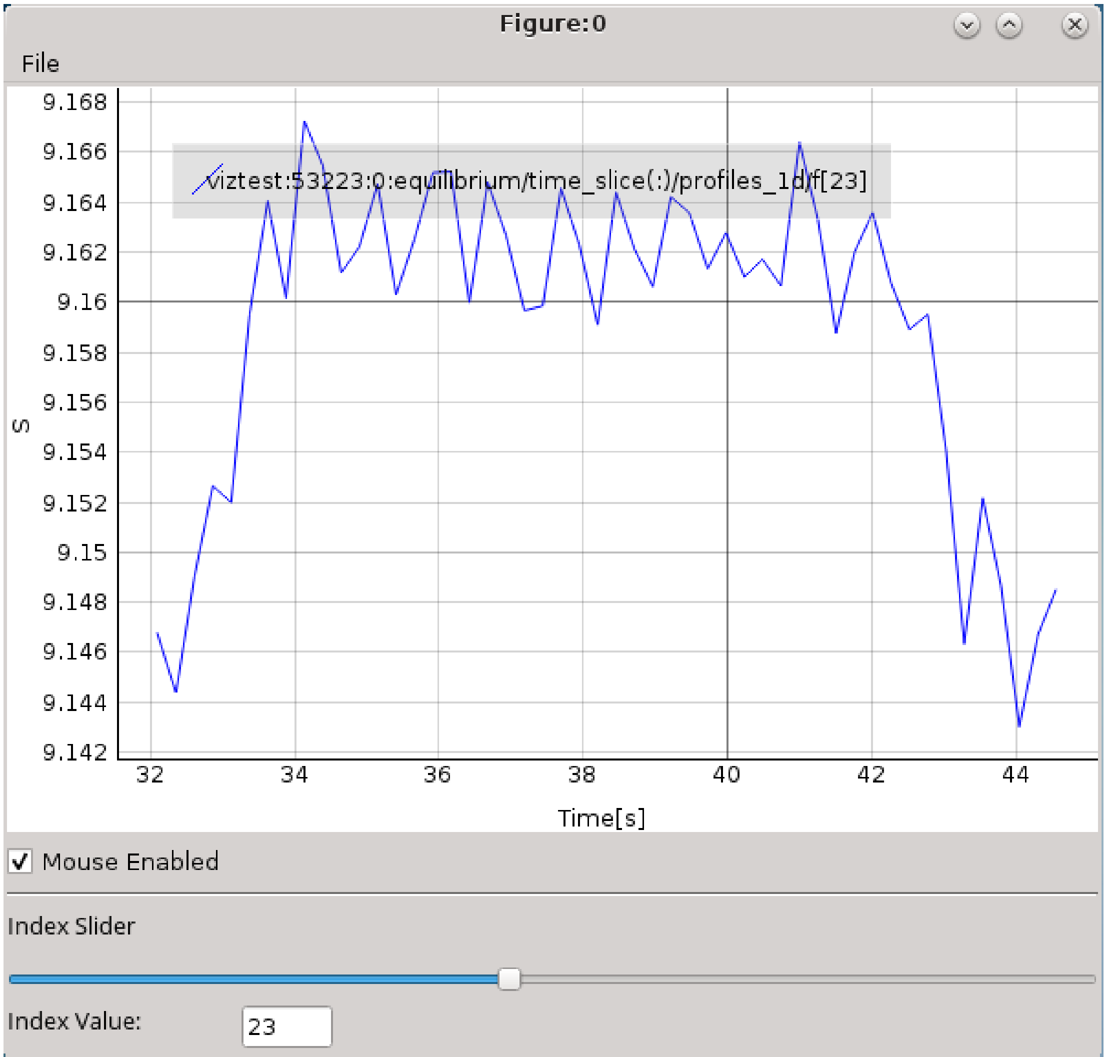


Fig. 41: Plot as a function of time. Y-axis values are an array of **equilibrium.time_slice[:].profiles_1d.f[23]** values through all time slices (marked by **[:]**) and X-axis values are time values found in **equilibrium.time** node (array of FLT_1D values).

Note: Adding a plot (presented in *Adding a plot to existing figure*) to such existing plot might not work as expected, as the sliding through indexes works directly only the last added plot.

2.4.7 MultiPlot features

IMASViz provides few features that allow plotting a selection of plottable arrays to a single plot view window.

Currently there are two such features available:

- *Table Plot View* and
- *Stacked Plot View*.

Each of those Plot Views feature its own plot display layout and plot display window interaction features.

Note: In the old IMASViz, the *Table Plot View* is known as *MultiPlot* and the *Stacked Plot View* is known as *SubPlot*. The decision to rename those features was made due to the previous names not properly describing the feature itself and both of those features being a form of 'MultiPlot' - a window consisting of multiple plot displays.

2.4.7.1 Table Plot View

Table Plot View allows the user to create a multiplot window by plotting every array from selection to its own plot display. The plot displays are arranged to resemble a table layout, as shown in figure below.

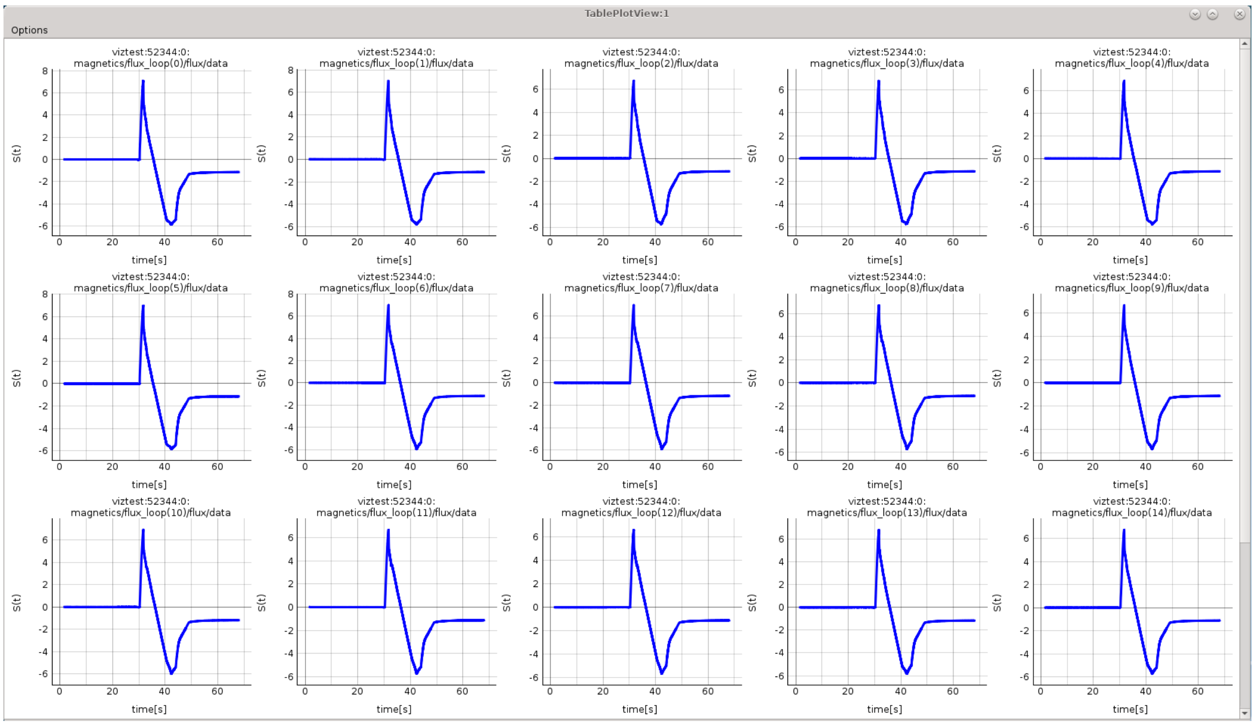


Fig. 42: MultiPlot - *Table Plot View* Example.

Creating New View

To create a new *Table Plot View*, follow the next steps:

1. Create a selection of nodes, as described in section *Plotting a selection of 1D arrays to figure*.
2. **When finished with node selection, either:**
    - right-click on any **FLT_1D** node or
    - click *Node Selection* menu on menubar of the main tree view window.
3. From the pop-up menu, navigate and select *Plot selected nodes to* -> *TablePlotView* -> *New* -> *This IMAS database* or *All IMAS databases* or.

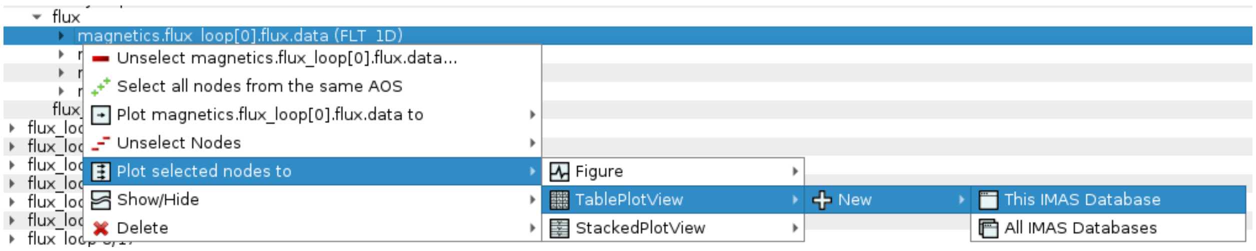


Fig. 43: Plotting selection to a new figure using selection from the currently opened IDS database.

The *Table Plot View* window will then be created and shown.

Note: Each plot can be customized individually by right-clicking to the plot d display and selecting option *Configure Plot*.

Note: Scrolling down the *Table Plot View* window using the middle mouse button is disabled as the same button is used to interact with the plot display (zoom in and out). Scrolling can be done by clicking the scroll bar on the right and dragging it up and down.

Save MultiPlot Configuration

MultiPlot configuration (currently available only for *Table Plot View* feature) allows the user to save the MultiPlot session and load it later.

To create MultiPlot configuration, follow the next steps:

1. Create a selection of nodes, as described in section *Plotting a selection of 1D arrays to figure*.
2. Create a *Table Plot View*, as described in *Table Plot View*.
3. In *Table Plot View* menubar navigate to **Options -> Save Plot Configuration**

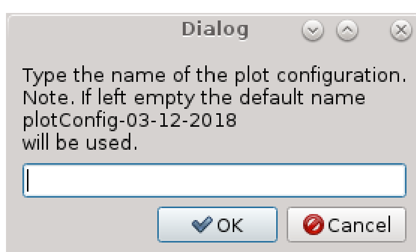


Fig. 44: Save Plot Configuration Dialog Window.

4. Type configuration name in the text area.
5. Press **OK**.

Note: The configurations files are saved to `$HOME/.imasviz` folder.

Applying MultiPlot configuration to other IMAS database

To apply MultiPlot configuration to any IMAS database, follow the next steps:

1. Open IMAS database.
2. In Main Tree View Window menu navigate to **Actions** -> **Apply Configuration**. In the shown window switch to *Apply Plot Configuration* tab.

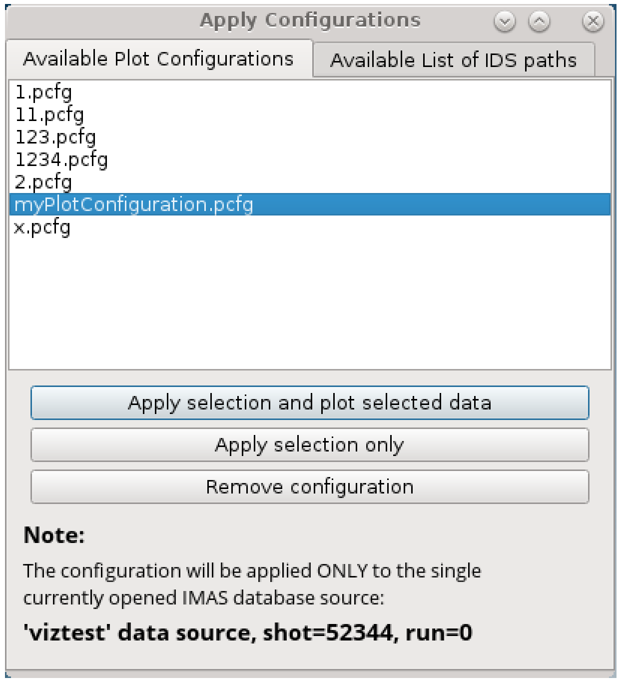


Fig. 45: Apply Plot Configuration GUI Window.

3. Select the configuration from the list.
4. Press **Apply selection and plot selected data**.

   The *Table Plot View* will be created using the data stored in the configuration file.

Note: Currently this feature will take all plot data from single (currently) opened IMAS database, event though MultiPlot configuration was made using plots from multiple IMAS databases at once. This feature is to be improved in the future.

Warning: The plots order depends on the order in which the data selection has been performed. First selected data will be the first plots in the *Table Plot View* window.

2.4.7.2 Stacked Plot View

Stacked Plot View allows the user to create a multiplot window by plotting every array from selection to its own plot display. The plot displays are arranged to resemble a stack layout, as shown in figure below. All plots displays always share the same X and Y range, even when using plot interaction features such as *Zoom in/out*, *Pan Mode*, *Area Zoom Mode* etc.

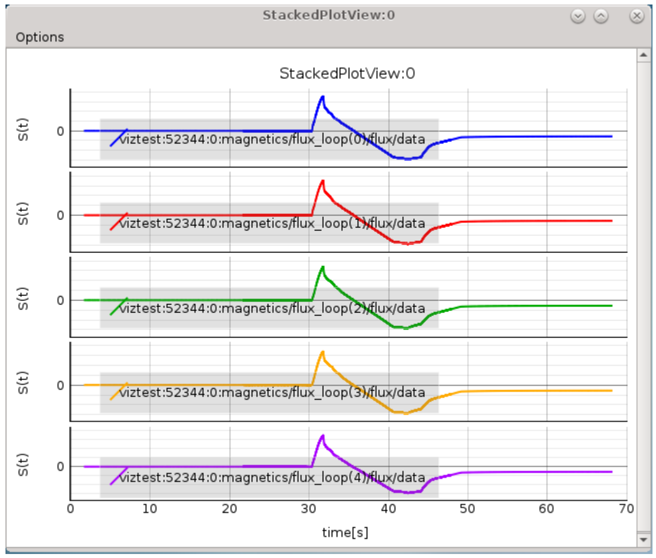


Fig. 46: MultiPlot - *Stacked Plot View* Example.

Creating New View

To create a new *Stacked Plot View*, follow the next steps:

1. Create a selection of nodes, as described in section *Plotting a selection of 1D arrays to figure*.
2. **When finished with node selection, either:**
   - right-click on any FLT_1D node or
   - click *Node Selection* menu on menubar of the main tree view window.
3. From the pop-up menu, navigate and select *Plot selected nodes to* -> *StackedPlotView* -> *New* -> *This IMAS database* or *All IMAS databases* or.

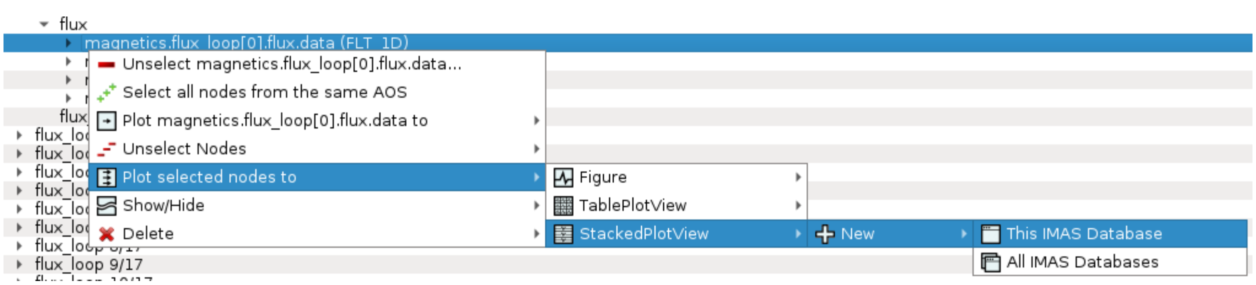


Fig. 47: Plotting selection to a new figure using selection from the currently opened IDS database.

The *Stacked Plot View* window will then be shown.

Note: Each plot can be customized individually; right click on a node and select 'Configure Plot'.

2.5 Other GUI features

Other *IMASViz* GUI features are listed here.

2.5.1 Hide/Show Plot Window

To **hide** or **show** any of the plot windows (*Figure*, *Table Plot View* etc.), first right-click on any signal node. From the shown pop-up menu, select the command *Show/Hide* and select the plot window you wish to show or hide.

Note: Shown plot windows will be hidden while hidden windows will be shown.

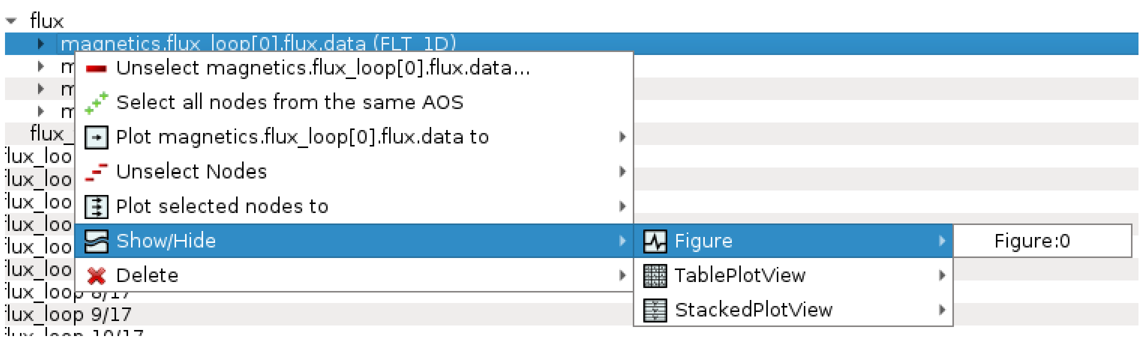


Fig. 48: Show/Hide plot window.

2.5.2 Delete Plot Window

To delete any of the plot windows, first right-click on any signal node. From the shown pop-up menu, select the command *Delete* and select the plot window you wish to delete or delete all plot windows of certain type (*Figure*, *Table Plot View* etc.).

Warning: The plot window will be deleted permanently.

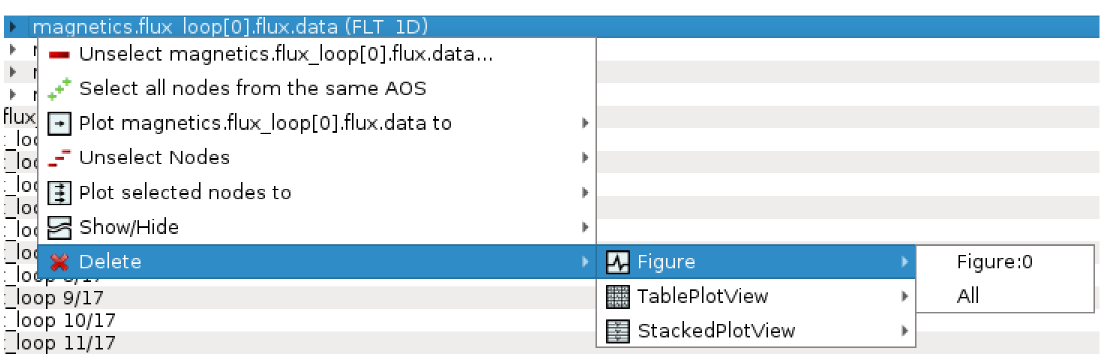


Fig. 49: Delete plot window.

2.6 Scripting

There are two main methods to open and browse IMAS IDS database using *IMASViz*. First is the standard way of running IMASViz (as described in *Getting Started*) and setting the parameters in the GUI, the second is through a script. Furthermore, some GUI actions (like node selection, plot action commands etc.) can be performed through scripting too. This way a simple IMASViz session can be set and populated on run.

This section describes the second method: how to create and use **user-made** Python3 scripts that run and populate *IMASViz*.

IMASViz scripting can be seen as an advanced alternative to the *Apply Configuration* features (described in *Apply Selection From Saved Configuration* and *Applying MultiPlot configuration to other IMAS database*).

The procedure below can be used with either **IMASViz module** and **IMASViz from source** but the environment **must be set accordingly to the method the IMASViz is used (!)** (as described in *Running IMASViz as a module on The GateWay* and *Running IMASViz from source*).

2.6.1 Adding IMASViz Path to PYTHONPATH

Warning: Before proceeding, make sure that the environment is properly pre-set! This way also the **VIZ_HOME** system variable, required in this manual section, is available.

The IMASViz home directory can be added to **PYTHONPATH** system variable by running in the terminal the command below (use the command that matches your shell - **c-shell** or **bash**):

Note: **PYTHONPATH** is a "list" of paths that tell Python where to look for sources, libraries etc.

```
# c-shell (csh)
setenv PYTHONPATH ${VIZ_HOME}:${PYTHONPATH}
# bash
export PYTHONPATH=${VIZ_HOME}:${PYTHONPATH}
```

2.6.2 Creating A Script

This subsection will cover the basic procedure of writing a Python3 script that can be used with *IMASViz*. Few such working script examples are shown in section *Script examples*. The same examples can be found in project GIT repository here.

1. **First, the imports are required:**

   - constants, functions, and methods of the Python interpreter,
   - PyQt5 classes and routines, and
   - IMASViz classes and routines.

```
# !/usr/bin/python
# A module providing a number of functions and variables that can be used to
# manipulate different parts of the Python runtime environment.
import sys
# PyQt library imports
from PyQt5.QtWidgets import QApplication
```

(continues on next page)

(continued from previous page)

```
# IMASViz source imports
from imasviz.Viz_API import Viz_API
from imasviz.VizDataSource.QVizDataSourceFactory import QVizDataSourceFactory
from imasviz.VizUtils.QVizGlobalOperations import QVizGlobalOperations
from imasviz.VizGUI.VizGUICommands.VizMenusManagement.QVizSignalHandling \
    import QVizSignalHandlingds.VizMenusManagement.QVizSignalHandling \
        import QVizSignalHandling
```

2. Set object managing the PyQt GUI application's control flow:

```
app = QApplication(sys.argv)
```

3. Check if necessary system variables are set

```
QVizGlobalOperations.checkEnvSettings()
```

4. Set Application Program Interface

```
api = Viz_API()
```

5. Set data source retriever/factory

```
dataSourceFactory = QVizDataSourceFactory()
```

6. Load IMAS database and build the data tree view

```
f1 = api.CreateDataTree(dataSourceFactory.create(shotNumber=52344,
                                                 runNumber=0,
                                                 userName='g2penkod',
                                                 imasDbName='viztest'))
```

7. Add the build data tree view (DTV) to a list (!)

```
f = [f1]
```

8. Set the list of node paths

```
pathsList1 = []
for i in range(0, 5):
    pathsList1.append('magnetics/flux_loop(' + str(i) + ')/flux/data')
```

9. Select signals corresponding to the list of node paths

```
api.SelectSignals(f1, pathsList1)
```

10. Show the data tree window

```
f1.show()
```

11. Plot selected nodes

```
f = [f1]
api.PlotSelectedSignalsFrom(f)
```

12. Plot data from the first data source (f1) to Table Plot View

```
QVizSignalHandling(f1.dataTreeView).onPlotToTablePlotView(all_DTV=False)
```

13. Plot data from the first data source (f1) to Stacked Plot View

```
QVizSignalHandling(f1.dataTreeView).onPlotToStackedPlotView(all_DTV=False)
```

14. Keep the application running

```
sys.exit(app.exec_())
```

The final script is available below.

```
# !/usr/bin/python
# A module providing a number of functions and variables that can be used to
# manipulate different parts of the Python runtime environment.
import sys
# PyQt library imports
from PyQt5.QtWidgets import QApplication
# IMASViz source imports
from imasviz.Viz_API import Viz_API
from imasviz.VizDataSource.QVizDataSourceFactory import QVizDataSourceFactory
from imasviz.VizUtils.QVizGlobalOperations import QVizGlobalOperations
from imasviz.VizGUI.VizGUICommands.VizMenusManagement.QVizSignalHandling \
    import QVizSignalHandling

# Set object managing the PyQt GUI application's control flow and main
# settings
app = QApplication(sys.argv)

# Check if necessary system variables are set
QVizGlobalOperations.checkEnvSettings()

# Set Application Program Interface
api = Viz_API()

# Set data source retriever/factory
dataSourceFactory = QVizDataSourceFactory()

# Load IMAS database and build the data tree view
f1 = api.CreateDataTree(dataSourceFactory.create(shotNumber=52344,
                                                 runNumber=0,
                                                 userName='g2penkod',
                                                 imasDbName='viztest'))

# Add data tree view frame to list (!)
f = [f1]

# Set the list of node paths that are to be selected
pathsList1 = []
for i in range(0, 5):
    pathsList1.append('magnetics/flux_loop(' + str(i) + ')/flux/data')

# Select signal nodes corresponding to the paths in pathsList
api.SelectSignals(f1, pathsList1)

# Show the data tree view window
f1.show()
```

(continues on next page)

(continued from previous page)

```
# Plot signal nodes
# Note: Data tree view does not need to be shown in order for this
#       routine to work
api.PlotSelectedSignalsFrom(f)

# Plot data from the data source to Table Plot View
QVizSignalHandling(f1.dataTreeView).onPlotToTablePlotView(all_DTV=False)

# Plot data from the data source to Stacked Plot View
QVizSignalHandling(f1.dataTreeView).onPlotToStackedPlotView(all_DTV=False)

# Keep the application running
sys.exit(app.exec_())
```

2.6.3 Running the script

With the environment set (done in *Adding IMASViz Path to PYTHONPATH*) and script completed (done in *Creating A Script*), the script can be run using the basic Python3 terminal command:

```
python3 <path_to_script>/<script_name>.py
```

By running this script all *Data Tree Views*, *Plot Widgets* and *MultiPlot Views*, previously set in the script, should show, as shown in the figure below.

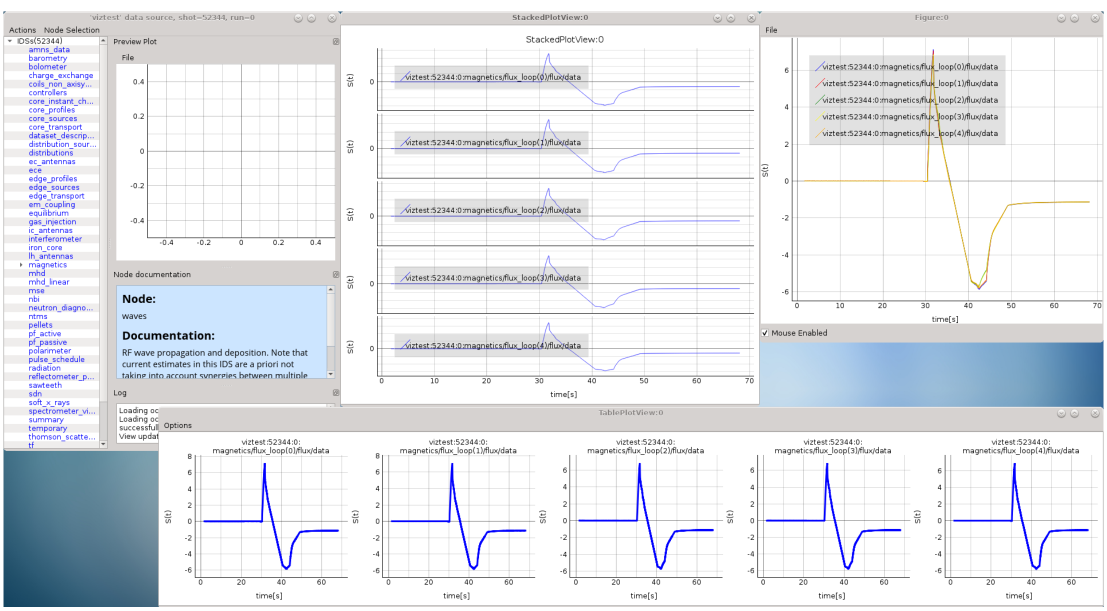


Fig. 50: The result of running the script example: *Data Tree View (DTV)*, *Plot Widget*, *Table Plot View* and *Stacked Plot View* containing multiple plots.

2.6.4 Script examples

Few complete script examples are shown below. The same examples can be found in project GIT repository here.

2.6.4.1 Example 1

```
#!/usr/bin/python
"""This example demonstrates the procedure of plotting multiple arrays to
a single plot, Table Plot View and Stacked Plot View, using IMAS IDS databases
located on the GateWay HPC.
"""

# A module providing a number of functions and variables that can be used to
# manipulate different parts of the Python runtime environment.
import sys
# PyQt library imports
from PyQt5.QtWidgets import QApplication
# IMASViz source imports
from imasviz.VizUtils.QVizGlobalOperations import QVizGlobalOperations
from imasviz.Viz_API import Viz_API
from imasviz.VizDataSource.QVizDataSourceFactory import QVizDataSourceFactory
from imasviz.VizUtils.QVizGlobalValues import QVizGlobalValues

# Set object managing the PyQt GUI application's control flow and main
# settings
app = QApplication(sys.argv)

# Check if necessary system variables are set
QVizGlobalOperations.checkEnvSettings()

# Set Application Program Interface
api = Viz_API()

# Set data source retriever/factory
dataSourceFactory = QVizDataSourceFactory()

# Load IMAS database
dataSource = dataSourceFactory.create(
                                 dataSourceName=QVizGlobalValues.IMAS_NATIVE,
                                 shotNumber=52702,
                                 runNumber=0,
                                 userName='imas_public',
                                 imasDbName='west')

# Database on the GateWay HPC (comment the above dataSource code and uncomment
# the one below)
# dataSource = dataSourceFactory.create(shotNumber=52344,
#                                       runNumber=0,
#                                       userName='g2penkod',
#                                       imasDbName='viztest')
#

# Build the data tree view frame
f = api.CreateDataTree(dataSource)

# Set the list of node paths that are to be selected
paths = []
for i in range(0,6):
    paths.append('magnetics/flux_loop(' + str(i) + ')/flux/data')

# Select signal nodes corresponding to the paths in paths list
```

(continues on next page)

(continued from previous page)

```
api.SelectSignals(f, paths)

# Plot signal nodes
# Note: Data tree view does not need to be shown in order for this routine to
#       work
api.PlotSelectedSignals(f)

# Keep the application running
app.exec()
```

2.6.4.2 Example 2

```
# !/usr/bin/python
"""This example demonstrates the procedure of plotting multiple arrays to a
single plot, Table Plot View and Stacked Plot View, using IMAS IDS databases
located on the GateWay HPC.
"""

# A module providing a number of functions and variables that can be used to
# manipulate different parts of the Python runtime environment.
import sys
# PyQt library imports
from PyQt5.QtWidgets import QApplication
# IMASViz source imports
from imasviz.Viz_API import Viz_API
from imasviz.VizDataSource.QVizDataSourceFactory import QVizDataSourceFactory
from imasviz.VizUtils.QVizGlobalOperations import QVizGlobalOperations
from imasviz.VizGUI.VizGUICommands.VizMenusManagement.QVizSignalHandling \
    import QVizSignalHandling

# Set object managing the PyQt GUI application's control flow and main
# settings
app = QApplication(sys.argv)

# Check if necessary system variables are set
QVizGlobalOperations.checkEnvSettings()

# Set Application Program Interface
api = Viz_API()

# Set data source retriever/factory
dataSourceFactory = QVizDataSourceFactory()

# Load IMAS database and build the data tree view frame
f1 = api.CreateDataTree(dataSourceFactory.create(shotNumber=52344,
                                                 runNumber=0,
                                                 userName='g2penkod',
                                                 imasDbName='viztest'))

# Load IMAS database and build the data tree view frame
f2 = api.CreateDataTree(dataSourceFactory.create(shotNumber=52682,
                                                 runNumber=0,
                                                 userName='g2penkod',
                                                 imasDbName='viztest'))

```

(continues on next page)

(continued from previous page)

```
# Add data tree view frames to list (!)
f = [f1, f2]
# Set the list of node paths that are to be selected
pathsList1 = []
for i in range(0, 5):
    pathsList1.append('magnetics/flux_loop(' + str(i) + ')/flux/data')
pathsList2 = []
for i in range(0, 6):
    pathsList2.append('magnetics/bpol_probe(' + str(i) + ')/field/data')

# Select signal nodes corresponding to the paths in paths list
api.SelectSignals(f1, pathsList1)
api.SelectSignals(f2, pathsList2)
# Might use also
# QVizSelectSignals(f1.dataTreeView, pathsList1).execute()
# QVizSelectSignals(f2.dataTreeView, pathsList2).execute()

# Show the data tree view window
f1.show()
f2.show()

# Plot signal nodes
# Note: Data tree view does not need to be shown in order for this routine to
#       work
api.PlotSelectedSignalsFrom(f)


# Plot data from the first data source (f1) to Table Plot View
QVizSignalHandling(f1.dataTreeView).onPlotToTablePlotView(all_DTV=False)

# Plot data from the first data source (f1) to Stacked Plot View
QVizSignalHandling(f1.dataTreeView).onPlotToStackedPlotView(all_DTV=False)

# Keep the application running
sys.exit(app.exec_())
```

2.6.4.3 Example 2b

```
#!/usr/bin/python
"""This example demonstrates the procedure of plotting multiple arrays from
two IMAS IDS databases to a single plot.
"""

# A module providing a number of functions and variables that can be used to
# manipulate different parts of the Python runtime environment.
import sys
# PyQt library imports
from PyQt5.QtWidgets import QApplication
# IMASViz source imports
from imasviz.util.GlobalOperations import GlobalOperations
from imasviz.Viz_API import Viz_API
from imasviz.VizDataSource import DataSourceFactory
from imasviz.VizUtils.QVizGlobalValues import QVizGlobalValues

# Set object managing the PyQt GUI application's control flow and main
```

(continues on next page)

(continued from previous page)

```
# settings
app = QApplication(sys.argv)

# Check if necessary system variables are set
GlobalOperations.checkEnvSettings()

# Set Application Program Interface
api = Viz_API()

# Set data source retriever/factory
dataSourceFactory = DataSourceFactory()

# Set and empty list for listing data tree view frames
f = []
# Set list of shots
n_shot = [52702, 52703]

for i in range(0, 2):
    # Load IMAS databases
    dataSource = dataSourceFactory.create(dataSourceName=QVizGlobalValues.IMAS_NATIVE,
                                          shotNumber=n_shot[i],
                                          runNumber=0,
                                          userName='imas_public',
                                          imasDbName='west')
    # Append data tree view frame to list
    f.append(api.CreateDataTree(dataSource))

# Set the list of node paths (for both databases) that are to be selected
paths1 = []
for i in range(1, 3):
    paths1.append('magnetics/flux_loop(' + str(i) + ')/flux/data')
paths2 = []
for i in range(1, 3):
    paths2.append('magnetics/bpol_probe(' + str(i) + ')/field/data')

# Select signal nodes corresponding to the paths in paths list
api.SelectSignals(f[0], paths1)
api.SelectSignals(f[1], paths2)
# Plot signal nodes
# Note: Data tree view does not need to be shown in order for this routine to
#       work
api.PlotSelectedSignalsFrom(f)

# Show the data tree view window
f[0].show()
f[1].show()

# Keep the application running
app.exec()
```

2.6.4.4 Example 3

```
#!/usr/bin/python
"""This example demonstrates the procedure of plotting multiple arrays to a
single Table Plot View.
```

(continues on next page)

(continued from previous page)

```python
"""

# A module providing a number of functions and variables that can be used to
# manipulate different parts of the Python runtime environment.
import sys
# PyQt library imports
from PyQt5.QtWidgets import QApplication
# IMASViz source imports
from imasviz.VizUtils.QVizGlobalOperations import QVizGlobalOperations
from imasviz.Viz_API import Viz_API
from imasviz.VizDataSource.QVizDataSourceFactory import QVizDataSourceFactory
from imasviz.VizUtils.QVizGlobalValues import QVizGlobalValues

# Set object managing the PyQt GUI application's control flow and main
# settings
app = QApplication(sys.argv)

# Check if necessary system variables are set
QVizGlobalOperations.checkEnvSettings()

# Set Application Program Interface
api = Viz_API()

# Set data source retriever/factory
dataSourceFactory = QVizDataSourceFactory()

# Load IMAS database
dataSource = dataSourceFactory.create(
                                    dataSourceName=QVizGlobalValues.IMAS_NATIVE,
                                    shotNumber=52682,
                                    runNumber=0,
                                    userName='imas_public',
                                    imasDbName='west')

# Database on the GateWay HPC (comment the above dataSource code and uncomment
# the one below)
# dataSource = dataSourceFactory.create(shotNumber=52344,
#                                        runNumber=0,
#                                        userName='g2penkod',
#                                        imasDbName='viztest')

# Build the data tree view frame
f = api.CreateDataTree(dataSource)

# Set the list of node paths that are to be selected
pathsList = []
for i in range(0, 5):
    pathsList.append('magnetics/flux_loop(' + str(i) + ')/flux/data')

# Select signal nodes corresponding to the paths in paths list
api.SelectSignals(f, pathsList)

# Plot the set of signal nodes selected by the user to a new Table Plot View.
api.PlotSelectedSignalsInTablePlotViewFrame(f)

# Show the data tree view window
api.ShowDataTree(f)
```

(continues on next page)

(continued from previous page)

```

# Keep the application running
app.exec()
```

2.6.4.5 Example 4

```
#!/usr/bin/python

import os
# A module providing a number of functions and variables that can be used to
# manipulate different parts of the Python runtime environment.
import sys
# PyQt library imports
from PyQt5.QtWidgets import QApplication
# IMASViz source imports
from imasviz.VizUtils.QVizGlobalOperations import QVizGlobalOperations
from imasviz.Viz_API import Viz_API
from imasviz.VizDataSource.QVizDataSourceFactory import QVizDataSourceFactory
from imasviz.VizGUI.VizGUICommands.VizDataSelection.QVizSelectSignals import QVizSelectSignals
from imasviz.VizGUI.VizGUICommands.VizDataSelection.QVizUnselectAllSignals import QVizUnselectAllSignals
from imasviz.VizUtils.QVizGlobalValues import QVizGlobalValues

# Set object managing the PyQt GUI application's control flow and main
# settings
app = QApplication(sys.argv)

# Check if necessary system variables are set
QVizGlobalOperations.checkEnvSettings()

# Set Application Program Interface
api = Viz_API()

# Set data source retriever/factory
dataSourceFactory = QVizDataSourceFactory()

# Set user (get current user)
userName = os.environ['USER']

# Load IMAS database
dataSource = dataSourceFactory.create(
                                    dataSourceName=QVizGlobalValues.IMAS_NATIVE,
                                    shotNumber=52344,
                                    runNumber=0,
                                    userName='imas_public',
                                    imasDbName='west')

# Database on the GateWay HPC (comment the above dataSource code and uncomment
# the one below)
# dataSource = dataSourceFactory.create(shotNumber=52344,
#                                        runNumber=0,
#                                        userName='g2penkod',
#                                        imasDbName='viztest')

```

(continues on next page)

(continued from previous page)

```
# Build the data tree view frame
f = api.CreateDataTree(dataSource)

# Set configuration file
configFilePath = os.environ['HOME'] + "/.imasviz/configuration_name.pcfg"

# Extract signal paths from the config file and add them to a list of
# paths
pathsList = QVizGlobalOperations.getSignalsPathsFromConfigurationFile(
    configFile=configFilePath)

# First unselect all signals (optional)
# QVizUnselectAllSignals(dataTreeView=f.dataTreeView).execute()

# Select the signals, defined by a path in a list of paths, in the
# given Data Tree View (DTV) window
QVizSelectSignals(dataTreeView=f.dataTreeView,
                  pathsList=pathsList).execute()

# Plot the set of the signal nodes selected using plot configuration file to
# a new Table Plot View and apply plot configurations (colors, line width etc.)
api.ApplyTablePlotViewConfiguration(f, configFilePath=configFilePath)

# Keep the application running
app.exec()
```