pyglotaran$_{extras}$ Documentation

Release 0.7.2

Joris Snellenburg

Mar 22, 2024

CONTENTS:

CHAPTER

ONE

PYGLOTARAN-EXTRAS

Supplementary package for pyglotaran with (example) plotting code for use with the pyglotaran package. Can be installed as a python package or from sources.

1.1 Installation

Prerequisites:

- Python 3.10 or 3.11
- Python package `pyglotaran` v0.7.0 (or later)

1.1.1 Stable Release

To install pyglotaran-extras from PyPI, run this command in your terminal:

```
pip install pyglotaran-extras
```

If you want to install it via conda, you can run the following command:

```
conda install -c conda-forge pyglotaran-extras
```

1.1.2 From Source

To install pyglotaran-extras from sources, either clone this repository or download the latest release, then run this command in your terminal:

```
git clone https://github.com/glotaran/pyglotaran-extras.git
cd pyglotaran-extras
pip install -e .
```

or directly

```
pip install git+https://github.com/glotaran/pyglotaran-extras.git
```

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install pyglotaran_extras, run this command in your terminal:

```
$ pip install pyglotaran_extras
```

This is the preferred method to install pyglotaran_extras, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for pyglotaran_extras can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/glotaran/pyglotaran-extras
```

Or download the tarball:

```
$ curl -OJL https://github.com/glotaran/pyglotaran-extras/tarball/main
```

Once you have a copy of the source, you can install it with:

```
$ pip install .
```

CHAPTER

THREE

USAGE

To use pyglotaran_extras in a project:

```
import pyglotaran_extras
```

CHAPTER

FOUR

INNER WORKINGS

This is the detailed documentation of the inner workings of `pyglotaran_extras`.

pyglotaran_extras	Pyglotaran extension package with convenience functionality such as plotting.

4.1 pyglotaran_extras

Pyglotaran extension package with convenience functionality such as plotting.

Modules

pyglotaran_extras.deprecation	Module containing deprecation functionality.
pyglotaran_extras.inspect	Module with analysis inspection functionality.
pyglotaran_extras.io	Package containing io convenience functionality.
pyglotaran_extras.plotting	Package containing plotting functionality.
pyglotaran_extras.types	Module containing type definitions.

4.1.1 deprecation

Module containing deprecation functionality.

Modules

pyglotaran_extras.deprecation.deprecation_utils	Module containing deprecation functionality.

deprecation_utils

Module containing deprecation functionality.

Functions

Summary

check_overdue	Check if a deprecation is overdue for removal.
parse_version	Parse version string to tuple of three ints for comparison.
pyglotaran_extras_version	Version of the distribution.
warn_deprecated	Raise deprecation warning with change information.

check_overdue

check_overdue(*deprecated_qual_name_usage: str*, *to_be_removed_in_version: str*) → None

Check if a deprecation is overdue for removal.

Parameters

- **deprecated_qual_name_usage** (*str*) – Old usage with fully qualified name e.g.: `'glotaran.read_model_from_yaml(model_yml_str)'`
- **to_be_removed_in_version** (*str*) – Version the support for this usage will be removed.

Raises

OverDueDeprecation – If the current version is greater or equal to `to_be_removed_in_version`.

parse_version

parse_version(*version_str: str*) → tuple[int, int, int]

Parse version string to tuple of three ints for comparison.

Parameters

version_str (*str*) – Fully qualified version string of the form 'major.minor.patch'.

Returns

Version as tuple.

Return type

tuple[int, int, int]

Raises

- **ValueError** – If `version_str` has less that three elements separated by `.`.
- **ValueError** – If `version_str` 's first three elements can not be casted to int.

pyglotaran_extras_version

pyglotaran_extras_version() → str

Version of the distribution.

This is basically the same as `pyglotaran_extras.__version__` but independent from pyglotaran_extras. This way all of the deprecation functionality can be used even in `pyglotaran_extras.__init__.py` without moving the import below the definition of `__version__` or causing a circular import issue.

Returns
The version string.

Return type
str

warn_deprecated

warn_deprecated(*, *deprecated_qual_name_usage: str*, *new_qual_name_usage: str*, *to_be_removed_in_version: str*, *stacklevel: int = 2*) → None

Raise deprecation warning with change information.

The change information are old / new usage information and end of support version.

Parameters

- **deprecated_qual_name_usage** (*str*) – Old usage with fully qualified name e.g.: `'glotaran.read_model_from_yaml(model_yml_str)'`
- **new_qual_name_usage** (*str*) – New usage as fully qualified name e.g.: `'glotaran.io.load_model(model_yml_str, format_name="yml_str")'`
- **to_be_removed_in_version** (*str*) – Version the support for this usage will be removed.
- **stacklevel** (*int*) – Stack at which the warning should be shown as raise. Default: 2

Raises
OverDueDeprecation – If the current version is greater or equal to `to_be_removed_in_version`.

Exceptions

Exception Summary

OverDueDeprecationError	Error thrown when a deprecation should have been removed.
PyglotaranExtrasApiDeprecationWarning	Warning to give users about API changes.

OverDueDeprecationError

exception OverDueDeprecationError

Error thrown when a deprecation should have been removed.

See also:

`deprecate`, *warn_deprecated*, `deprecate_module_attribute`, `deprecate_submodule`, `deprecate_dict_entry`

PyglotaranExtrasApiDeprecationWarning

exception PyglotaranExtrasApiDeprecationWarning

Warning to give users about API changes.

See also:

warn_deprecated

4.1.2 inspect

Module with analysis inspection functionality.

Modules

pyglotaran_extras.inspect.a_matrix	Module containing a-matrix render functionality.
pyglotaran_extras.inspect.utils	Inspection utility module.

a_matrix

Module containing a-matrix render functionality.

Functions

Summary

a_matrix_to_html_table	Create HTML multi header table from a-matrix.
show_a_matrixes	Show all a-matrixes of a result grouped by dataset and megacomplex name.

a_matrix_to_html_table

a_matrix_to_html_table(*a_matrix: xr.DataArray*, *megacomplex_suffix: str*, ***, *normalize_initial_concentration: bool = False*, *decimal_places: int = 3*) → str

Create HTML multi header table from a-matrix.

Parameters

- **a_matrix** (*xr.DataArray*) – DataArray containing the a-matrix values and coordinates.
- **megacomplex_suffix** (*str*) – Megacomplex suffix used for the a-matrix data variable and coordinate names.
- **normalize_initial_concentration** (*bool*) – Whether or not to normalize the initial concentration. Defaults to False.
- **decimal_places** (*int*) – Decimal places to display. Defaults to 3.

Returns
Multi header HTML table representing the a-matrix.

Return type
str

show_a_matrixes

show_a_matrixes(*result: ResultLike*, ***, *normalize_initial_concentration: bool = False*, *decimal_places: int = 3*, *a_matrix_min_size: int | None = None*, *expanded_datasets: tuple[str, ...] = ()*, *heading_offset: int = 2*) → MarkdownStr

Show all a-matrixes of a result grouped by dataset and megacomplex name.

Each dataset is wrapped in a HTML details tag which is by default collapsed.

Parameters

- **result** (*ResultLike*) – Result or result dataset.
- **normalize_initial_concentration** (*bool*) – Whether or not to normalize the initial concentration. Defaults to False.
- **decimal_places** (*int*) – Decimal places to display. Defaults to 3.
- **a_matrix_min_size** (*int | None*) – Defaults to None.
- **expanded_datasets** (*tuple[str, ...]*) – Names of dataset to expand the details view for. Defaults to empty tuple () which means no dataset is expanded.
- **heading_offset** (*int*) – Number of heading level to offset the headings. Defaults to 2 which means that the first/top most heading is h3.

Returns
Markdown representation of the a-matrixes used in the optimization.

Return type
MarkdownStr

utils

Inspection utility module.

Functions

Summary

pretty_format_numerical	Format value with with at most `decimal_places` decimal places.
pretty_format_numerical_iterable	Pretty format numerical values in an iterable of numerical values or strings.
wrap_in_details_tag	Wrap `details_content` in a html details tag and add summary if `summary_content` set.

pretty_format_numerical

pretty_format_numerical(*value: float | int*, *decimal_places: int = 1*) → str

Format value with with at most `decimal_places` decimal places.

Used to format values like the t-value.

TODO : remove after raise pyglotaran dependency to 0.7.0 Forward port of https://github.com/glotaran/pyglotaran/pull/1192

Parameters

- **value** (*float | int*) – Numerical value to format.
- **decimal_places** (*int*) – Decimal places to display. Defaults to 1.

Returns
Pretty formatted version of the value.

Return type
str

pretty_format_numerical_iterable

pretty_format_numerical_iterable(*input_values: Iterable[str | float]*, *decimal_places: int | None = 3*) → Generator[str | float, None, None]

Pretty format numerical values in an iterable of numerical values or strings.

Parameters

- **input_values** (*Iterable[str | float]*) – Values that should be formatted.
- **decimal_places** (*int | None*) – Number of decimal places a value should have, if None the original value will be used. Defaults to 3.

See also:

pretty_format_numerical

Yields
str | float – Formatted string or initial value if `decimal_places` is None.

wrap_in_details_tag

wrap_in_details_tag(*details_content: str*, *, *summary_content: str | None = None*, *summary_heading_level: int | None = None*, *is_open: bool = False*) → str

Wrap `details_content` in a html details tag and add summary if `summary_content` set.

Parameters

- **details_content** (*str*) – Markdown string that should be displayed when the details are expanded.
- **summary_content** (*str | None*) – Summary test that should be displayed. Defaults to None so the summary is `Details`.
- **summary_heading_level** (*int | None*) – Level of the heading wrapping the summary if it is not None. Defaults to None.
- **is_open** (*bool*) – Whether or not the details tag should be initially opened. Defaults to False.

Return type
str

4.1.3 io

Package containing io convenience functionality.

Modules

pyglotaran_extras.io.load_data(result[, ...])	Extract a single dataset from a `DatasetConvertible` object.
pyglotaran_extras.io.setup_case_study([...])	Quickly get folders for a case study.
pyglotaran_extras.io.utils	Io utility module.

load_data

load_data(*result: Dataset | DataArray | str | Path | Result*, *dataset_name: str | None = None*, *, *_stacklevel: int = 2*) → Dataset

Extract a single dataset from a `DatasetConvertible` object.

Parameters

- **result** (*DatasetConvertible | Result*) – Result class instance, xarray Dataset or path to a dataset file.
- **dataset_name** (*str | None*) – Name of a specific dataset contained in `result`, if not provided the first dataset will be extracted. Defaults to None.
- **_stacklevel** (*int*) – Stacklevel of the warning which is raised when `result` is of class `Result`, contains multiple datasets and no `dataset_name` is provided. Changing this value is only required if you use this function inside of another function. Defaults to 2

Returns
Extracted dataset.

Return type
xr.Dataset

Raises
TypeError – If `result` isn't a `DatasetConvertible` object.

setup_case_study

setup_case_study(*output_folder_name: str = 'pyglotaran_results'*, *results_folder_root: None | str | PathLike[str] = None*) → tuple[Path, Path]

Quickly get folders for a case study.

This is an execution environment independent (works in python script files and notebooks, independent of where the python runtime was called from) way to get the folder the analysis code resides in and also creates the `results_folder` in case it didn't exist before.

Parameters

- **output_folder_name** (*str*) – Name of the base folder for the results. Defaults to "pyglotaran_results".
- **results_folder_root** (*None | str | PathLike[str]*) – The folder where the results named `output_folder_name` should be saved to. Defaults to None, which results in the users Home folder being used.

Returns

results_folder, script_folder:

results_folder:
Folder to be used to save results in of the pattern (`results_folder_root / output_folder_name / analysis_folder.parent`).

analysis_folder:
Folder the script or Notebook resides in.

Return type
tuple[Path, Path]

utils

Io utility module.

Functions

Summary

result_dataset_mapping	Convert a `ResultLike` object to a per dataset mapping of result like data.

result_dataset_mapping

result_dataset_mapping(*result: Result | Dataset | DataArray | str | Path | Mapping[str, Dataset | DataArray | str | Path] | Sequence[Dataset | DataArray | str | Path]*) → Mapping[str, Dataset]

Convert a `ResultLike` object to a per dataset mapping of result like data.

Parameters
result (*ResultLike*) – Data structure which can be converted to a mapping.

Returns
Per dataset mapping of result like data.

Return type
Mapping[str, xr.Dataset]

Raises

- **TypeError** – If any value of a `result` isn't of `DatasetConvertible`.
- **TypeError** – If `result` isn't a `ResultLike` object.

4.1.4 plotting

Package containing plotting functionality.

Modules

pyglotaran_extras.plotting.plot_coherent_artifact	Module containing coherent artifact plot functionality.
pyglotaran_extras.plotting.plot_concentrations	Module containing concentration plot functionality.
pyglotaran_extras.plotting.plot_data	Module containing data plotting functionality.
pyglotaran_extras.plotting.plot_doas	Module containing DOAS (Damped Oscillation) plotting functionality.
pyglotaran_extras.plotting.plot_guidance	Module containing guidance spectra plotting functionality.
pyglotaran_extras.plotting.plot_irf_dispersion_center	Module containing IRF dispersion plotting functionality.
pyglotaran_extras.plotting.plot_overview(...)	Plot overview of the optimization result.
pyglotaran_extras.plotting.plot_residual	Module containing residual plot functionality.
pyglotaran_extras.plotting.plot_spectra	Module containing spectra plotting functionality.
pyglotaran_extras.plotting.plot_svd	Module containing SVD plotting functionality.
pyglotaran_extras.plotting.plot_traces	Module containing functionality to plot fitted traces.
pyglotaran_extras.plotting.style	Module containing predefined plot styles.
pyglotaran_extras.plotting.utils	Module containing plotting utility functionality.

plot_coherent_artifact

Module containing coherent artifact plot functionality.

Functions

Summary

plot_coherent_artifact	Plot coherent artifact as IRF derivative components over time and IRFAS over spectral dim.

plot_coherent_artifact

plot_coherent_artifact(*dataset: DatasetConvertible | Result*, ***, *time_range: tuple[float, float] | None = None*, *spectral: float = 0*, *main_irf_nr: int | None = 0*, *normalize: bool = True*, *figsize: tuple[float, float] = (18, 7)*, *show_zero_line: bool = True*, *cycler: Cycler | None = None*, *title: str | None = 'Coherent Artifact'*) → tuple[Figure, Axes]

Plot coherent artifact as IRF derivative components over time and IRFAS over spectral dim.

The IRFAS are the IRF (Instrument Response Function) Associated Spectra.

Parameters

- **dataset** (*DatasetConvertible | Result*) – Result dataset from a pyglotaran optimization.
- **time_range** (*tuple[float, float] | None*) – Start and end time for the IRF derivative plot. Defaults to None which means that the full time range is used.
- **spectral** (*float*) – Value of the spectral axis that should be used to select the data for the IRF derivative plot this value does not need to be an exact existing value and only has effect if the IRF has dispersion. Defaults to 0 which means that the IRF derivative plot at lowest spectral value will be shown.
- **main_irf_nr** (*int | None*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks and is used to shift the time axis. If it is none `None` the shifting will be deactivated. Defaults to 0.
- **normalize** (*bool*) – Whether or not to normalize the IRF derivative plot. If the IRF derivative is normalized, the IRFAS is scaled with the reciprocal of the normalization to compensate for this. Defaults to True.
- **figsize** (*tuple[float, float]*) – Size of the figure (N, M) in inches. Defaults to (18, 7).
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to None, which means that the matplotlib default style will be used.
- **title** (*str | None*) – Title of the figure. Defaults to "Coherent Artifact".

Returns

Figure object which contains the plots and the Axes.

Return type
tuple[Figure, Axes]

plot_concentrations

Module containing concentration plot functionality.

Functions

Summary

plot_concentrations	Plot traces on the given axis `ax`.

plot_concentrations

plot_concentrations(*res: xr.Dataset, ax: Axis, center_: float | None, linlog: bool = False, linthresh: float = 1, linscale: float = 1, main_irf_nr: int = 0, cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), title: str = 'Concentrations'*) → None

Plot traces on the given axis `ax`.

Parameters

- **res** (*xr.Dataset*) – Result dataset from a pyglotaran optimization.
- **ax** (*Axis*) – Axis to plot the traces on
- **center_** (*float | None*) – Center wavelength (in nm)
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **linscale** (*float*) – This allows the linear range (-linthresh to linthresh) to be stretched relative to the logarithmic range. Its value is the number of decades to use for each half of the linear range. For example, when linscale == 1.0 (the default), the space used for the positive and negative halves of the linear range will be equal to one decade in the logarithmic range. Defaults to 1.
- **main_irf_nr** (*int*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks. Defaults to 0.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().data_cycler_solid.
- **title** (*str*) – Title used for the plot axis. Defaults to "Concentrations".

See also:

`get_shifted_traces`

plot_data

Module containing data plotting functionality.

Functions

Summary

plot_data_overview	Plot data as filled contour plot and SVD components.

plot_data_overview

plot_data_overview(*dataset: DatasetConvertible | Result, title: str = 'Data overview', linlog: bool = False, linthresh: float = 1, figsize: tuple[float, float] = (15, 10), nr_of_data_svd_vectors: int = 4, show_data_svd_legend: bool = True, irf_location: float | None = None, cmap: str = 'PuRd', vmin: float | None = None, vmax: float | None = None, svd_cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), use_svd_number: bool = False*) → tuple[Figure, Axes] | tuple[Figure, Axis]

Plot data as filled contour plot and SVD components.

Parameters

- **dataset** (*DatasetConvertible | Result*) – Dataset containing data and SVD of the data.
- **title** (*str*) – Title to add to the figure. Defaults to "Data overview".
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **figsize** (*tuple[float, float]*) – Size of the figure (N, M) in inches. Defaults to (15, 10).
- **nr_of_data_svd_vectors** (*int*) – Number of data SVD vector to plot. Defaults to 4.
- **show_data_svd_legend** (*bool*) – Whether or not to show the data SVD legend. Defaults to True.

- **irf_location** (*float | None*) – Location of the `irf` by which the time axis will get shifted. If it is None the time axis will not be shifted. Defaults to None.
- **cmap** (*str*) – Colormap to use for the filled contour plot. Defaults to "PuRd" which is most suitable for emission data (previous default was "viridis").
- **vmin** (*float | None*) – Lower value to anchor the colormap. Defaults to None meaning it inferred from the data.
- **vmax** (*float | None*) – Lower value to anchor the colormap. Defaults to None meaning it inferred from the data.
- **svd_cycler** (*Cycler | None*) – Plot style cycler to use for SVD plots. Defaults to `PlotStyle().cycler`.
- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

Returns
Figure and axes which can then be refined by the user.

Return type
tuple[Figure, Axes] | tuple[Figure, Axis]

plot_doas

Module containing DOAS (Damped Oscillation) plotting functionality.

Functions

Summary

plot_doas	Plot DOAS (Damped Oscillation) related data of the optimization result.

plot_doas

plot_doas(*dataset: DatasetConvertible | Result, *, damped_oscillation: list[str] | None = None, time_range: tuple[float, float] | None = None, spectral: float = 0, main_irf_nr: int | None = 0, normalize: bool = False, figsize: tuple[float, float] = (20, 5), show_zero_line: bool = True, cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), oscillation_type: Literal['cos', 'sin'] = 'cos', title: str | None = 'Damped oscillations', legend_format_string: str = '{label}: $\\nu$={frequency:.0f}, $\\gamma$={rate:.1f}'*) → tuple[Figure, Axes]

Plot DOAS (Damped Oscillation) related data of the optimization result.

Parameters

- **dataset** (*DatasetConvertible | Result*) – Result dataset from a pyglotaran optimization.
- **damped_oscillation** (*list[str] | None*) – List of oscillation names which should be plotted. Defaults to None which means that all oscillations will be plotted.
- **time_range** (*tuple[float, float] | None*) – Start and end time for the Oscillation plot, if `main_irf_nr` is not None the value are relative to the IRF location. Defaults to None which means that the full time range is used.
- **spectral** (*float*) – Value of the spectral axis that should be used to select the data for the Oscillation plot this value does not need to be an exact existing value and only has effect if the IRF has dispersion. Defaults to 0 which means that the Oscillation plot at lowest spectral value will be shown.
- **main_irf_nr** (*int | None*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks and is used to shift the time axis. If it is none `None` the shifting will be deactivated. Defaults to 0.
- **normalize** (*bool*) – Whether or not to normalize the DOAS spectra plot. If the DOAS spectra is normalized, the Oscillation is scaled with the reciprocal of the normalization to compensate for this. Defaults to False.
- **figsize** (*tuple[float, float]*) – Size of the figure (N, M) in inches. Defaults to (20, 5)
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler
- **oscillation_type** (*Literal["cos", "sin"]*) – Type of the oscillation to show in the oscillation plot. Defaults to “cos”
- **title** (*str | None*) – Title of the figure. Defaults to “Damped oscillations”
- **legend_format_string** (*str*) – Format string for each entry in the legend of the oscillation plot. Possible values which can be replaced are `label` (label of the oscillation in the model definition), `frequency` () and `rate` (). Use `""` to remove the legend. Defaults to `r"{label}: $\nu$={frequency:.0f}, $\gamma$={rate:.1f}"`

Returns
Figure object which contains the plots and the Axes.

Return type
tuple[Figure, Axes]

See also:

`calculate_ticks_in_units_of_pi`

plot_guidance

Module containing guidance spectra plotting functionality.

Functions

Summary

plot_guidance	Plot overview for a guidance spectrum.

plot_guidance

plot_guidance(*result: DatasetConvertible | Result, figsize: tuple[float, float] = (15, 5), title: str = 'Guidance Overview', y_label: str = 'a.u.', cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>])*) → tuple[Figure, Axes]

Plot overview for a guidance spectrum.

Parameters

- **result** (*DatasetConvertible | Result*) – Result from a pyglotaran optimization as dataset, Path or Result object.
- **figsize** (*tuple[float, float]*) – Size of the figure (N, M) in inches. Defaults to (15, 5)
- **title** (*str*) – Title to add to the figure. Defaults to "Guidance Overview"
- **y_label** (*str*) – Label used for the y-axis of each subplot. Defaults to "a.u."
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.

Returns
Figure and axes which can then be refined by the user.

Return type
tuple[Figure, Axes]

plot_irf_dispersion_center

Module containing IRF dispersion plotting functionality.

Functions

Summary

plot_irf_dispersion_center	Plot the IRF dispersion center over the spectral dimension for one or multiple datasets.

plot_irf_dispersion_center

plot_irf_dispersion_center(*result: ResultLike, ax: Axis | None = None, figsize: tuple[float, float] = (12, 8), cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), irf_location: float | None = None*) → tuple[Figure, Axis] | None

Plot the IRF dispersion center over the spectral dimension for one or multiple datasets.

Parameters

- **result** (*ResultLike*) – Data structure which can be converted to a mapping.
- **ax** (*Axis | None*) – Axis to plot on. Defaults to None which means that a new figure and axis will be created.
- **figsize** (*tuple[float, float]*) – Size of the figure (N, M) in inches. Defaults to (12, 8).
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler
- **irf_location** (*float | None*) – Location of the `irf` by which the time axis will get shifted. If it is None the time axis will not be shifted. Defaults to None.

Returns

Figure object which contains the plots and the Axis, if `ax` is not None nothing will be returned.

Return type

tuple[Figure, Axis] | None

plot_overview

plot_overview(*result: DatasetConvertible | Result, center_: float | None = None, linlog: bool = True, linthresh: float = 1, linscale: float = 1, show_data: bool | None = False, main_irf_nr: int = 0, figsize: tuple[float, float] = (18, 16), cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), figure_only: bool | None = None, nr_of_data_svd_vectors: int = 4, nr_of_residual_svd_vectors: int = 2, show_data_svd_legend: bool = True, show_residual_svd_legend: bool = True, show_irf_dispersion_center: bool = True, show_zero_line: bool = True, das_cycler: Cycler | None | UnsetType = Unset, svd_cycler: Cycler | None | UnsetType = Unset, use_svd_number: bool = False*) → tuple[Figure, Axes]

Plot overview of the optimization result.

Parameters

- **result** (*DatasetConvertible | Result*) – Result from a pyglotaran optimization as dataset, Path or Result object.
- **center_** (*float | None*) – Center wavelength (in nm)
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **linscale** (*float*) – This allows the linear range (-linthresh to linthresh) to be stretched relative to the logarithmic range. Its value is the number of decades to use for each half of the linear range. For example, when linscale == 1.0 (the default), the space used for the positive and negative halves of the linear range will be equal to one decade in the logarithmic range. Defaults to 1.
- **show_data** (*bool | None*) – Whether to show the input data or residual. If set to `None` the plot is skipped which improves plotting performance for big datasets. Defaults to False.
- **main_irf_nr** (*int*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks. Defaults to 0.
- **figsize** (*tuple[float, float]*) – Size of the figure (N, M) in inches. Defaults to (18, 16).
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **figure_only** (*bool | None*) – Deprecated please remove this argument for you function calls. Defaults to None.
- **nr_of_data_svd_vectors** (*int*) – Number of data SVD vector to plot. Defaults to 4.
- **nr_of_residual_svd_vectors** (*int*) – Number of residual SVD vector to plot. Defaults to 2.
- **show_data_svd_legend** (*bool*) – Whether or not to show the data SVD legend. Defaults to True.
- **show_residual_svd_legend** (*bool*) – Whether or not to show the residual SVD legend. Defaults to True.
- **show_irf_dispersion_center** (*bool*) – Whether to show the the IRF dispersion center as overlay on the residual/data plot. Defaults to True.

- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero to the plots of the spectra. Defaults to True.
- **das_cycler** (*Cycler | None | UnsetType*) – Plot style cycler to use for DAS plots. Defaults to `Unset` which means that the value of `cycler` is used.
- **svd_cycler** (*Cycler | None | UnsetType*) – Plot style cycler to use for SVD plots. Defaults to `Unset` which means that the value of `cycler` is used.
- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

Return type
tuple[Figure, Axes]

plot_residual

Module containing residual plot functionality.

Functions

Summary

plot_residual	Plot data or residual on a 2D contour plot.

plot_residual

plot_residual(*res: xr.Dataset, ax: Axis, linlog: bool = False, linthresh: float = 1, show_data: bool | None = False, cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_irf_dispersion_center: bool = True, irf_location: float | None = None*) → None

Plot data or residual on a 2D contour plot.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **show_data** (*bool | None*) – Whether to show the input data or residual. If set to `None` the plot is skipped which improves plotting performance for big datasets. Defaults to False.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.

- **show_irf_dispersion_center** (*bool*) – Whether to show the the IRF dispersion center as overlay on the residual/data plot. Defaults to True.
- **irf_location** (*float | None*) – Location of the `irf` by which the time axis will get shifted. If it is None the time axis will not be shifted. Defaults to None.

plot_spectra

Module containing spectra plotting functionality.

Functions

Summary

plot_das	Plot DAS (Decay Associated Spectra) on `ax`.
plot_norm_das	Plot normalized DAS (Decay Associated Spectra) on `ax`.
plot_norm_sas	Plot normalized SAS (Species Associated Spectra) on `ax`.
plot_sas	Plot SAS (Species Associated Spectra) on `ax`.
plot_spectra	Plot spectra such as SAS and DAS as well as their normalize version on `axes`.

plot_das

plot_das(*res: xr.Dataset, ax: Axis, title: str = 'DAS', cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_zero_line: bool = True*) → None

Plot DAS (Decay Associated Spectra) on `ax`.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **title** (*str*) – Title of the plot. Defaults to “DAS”.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True.

plot_norm_das

plot_norm_das(*res: xr.Dataset, ax: Axis, title: str = 'norm DAS', cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_zero_line: bool = True*) → None

Plot normalized DAS (Decay Associated Spectra) on `ax`.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **title** (*str*) – Title of the plot. Defaults to "norm DAS".
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True.

plot_norm_sas

plot_norm_sas(*res: xr.Dataset, ax: Axis, title: str = 'norm SAS', cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_zero_line: bool = True*) → None

Plot normalized SAS (Species Associated Spectra) on `ax`.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **title** (*str*) – Title of the plot. Defaults to "norm SAS".
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True.

plot_sas

plot_sas(*res: xr.Dataset, ax: Axis, title: str = 'SAS', cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_zero_line: bool = True*) → None

Plot SAS (Species Associated Spectra) on `ax`.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **title** (*str*) – Title of the plot. Defaults to "SAS".
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True.

plot_spectra

plot_spectra(*res: xr.Dataset, axes: Axes, cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_zero_line: bool = True, das_cycler: Cycler | None | UnsetType = Unset*) → None

Plot spectra such as SAS and DAS as well as their normalize version on `axes`.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **axes** (*Axes*) – Axes to plot the spectra on (needs to be at least 2x2).
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True.
- **das_cycler** (*Cycler | None | UnsetType*) – Plot style cycler to use for DAS plots. Defaults to `Unset` which means that the value of `cycler` is used.

plot_svd

Module containing SVD plotting functionality.

Functions

Summary

plot_lsv_data	Plot left singular vectors (time) of the data matrix.
plot_lsv_residual	Plot left singular vectors (time) of the residual matrix.
plot_rsv_data	Plot right singular vectors (spectra) of the data matrix.
plot_rsv_residual	Plot right singular vectors (spectra) of the residual matrix.
plot_sv_data	Plot singular values of the data matrix.
plot_sv_residual	Plot singular values of the residual matrix.
plot_svd	Plot SVD (Singular Value Decomposition) of data and residual.

plot_lsv_data

plot_lsv_data(*res: xr.Dataset, ax: Axis, indices: Sequence[int] = range(0, 4), linlog: bool = False, linthresh: float = 1, cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_legend: bool = True, irf_location: float | None = None, use_svd_number: bool = False*) → None

Plot left singular vectors (time) of the data matrix.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **indices** (*Sequence[int]*) – Indices of the singular vector to plot. Defaults to range(4).
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_legend** (*bool*) – Whether or not to show the legend. Defaults to True.
- **irf_location** (*float | None*) – Location of the `irf` by which the time axis will get shifted. If it is None the time axis will not be shifted. Defaults to None.

- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

plot_lsv_residual

plot_lsv_residual(*res: xr.Dataset, ax: Axis, indices: Sequence[int] = range(0, 2), linlog: bool = False, linthresh: float = 1, cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_legend: bool = True, irf_location: float | None = None, use_svd_number: bool = False*) → None

Plot left singular vectors (time) of the residual matrix.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **indices** (*Sequence[int]*) – Indices of the singular vector to plot. Defaults to range(4).
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_legend** (*bool*) – Whether or not to show the legend. Defaults to True.
- **irf_location** (*float | None*) – Location of the `irf` by which the time axis will get shifted. If it is None the time axis will not be shifted. Defaults to None.
- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

plot_rsv_data

plot_rsv_data(*res: xr.Dataset, ax: Axis, indices: Sequence[int] = range(0, 4), cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_legend: bool = True, irf_location: float | None = None, use_svd_number: bool = False*) → None

Plot right singular vectors (spectra) of the data matrix.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **indices** (*Sequence[int]*) – Indices of the singular vector to plot. Defaults to range(4).
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_legend** (*bool*) – Whether or not to show the legend. Defaults to True.
- **irf_location** (*float | None*) – Location of the `irf` by which the time axis will get shifted. If it is None the time axis will not be shifted. Defaults to None.
- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

plot_rsv_residual

plot_rsv_residual(*res: xr.Dataset, ax: Axis, indices: Sequence[int] = range(0, 2), cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), show_legend: bool = True, irf_location: float | None = None, use_svd_number: bool = False*) → None

Plot right singular vectors (spectra) of the residual matrix.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **indices** (*Sequence[int]*) – Indices of the singular vector to plot. Defaults to range(4).
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **show_legend** (*bool*) – Whether or not to show the legend. Defaults to True.
- **irf_location** (*float | None*) – Location of the `irf` by which the time axis will get shifted. If it is None the time axis will not be shifted. Defaults to None.
- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

plot_sv_data

plot_sv_data(*res: xr.Dataset*, *ax: Axis*, *indices: Sequence[int] = range(0, 10)*, *cycler: Cycler | None | UnsetType = Unset*, *use_svd_number: bool = False*) → None

Plot singular values of the data matrix.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **indices** (*Sequence[int]*) – Indices of the singular vector to plot. Defaults to range(10).
- **cycler** (*Cycler | None | UnsetType*) – Deprecated since it has no effect. Defaults to Unset.
- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

plot_sv_residual

plot_sv_residual(*res: xr.Dataset*, *ax: Axis*, *indices: Sequence[int] = range(0, 10)*, *cycler: Cycler | None | UnsetType = Unset*, *use_svd_number: bool = False*) → None

Plot singular values of the residual matrix.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **ax** (*Axis*) – Axis to plot on.
- **indices** (*Sequence[int]*) – Indices of the singular vector to plot. Defaults to range(10).
- **cycler** (*Cycler | None | UnsetType*) – Deprecated since it has no effect. Defaults to Unset.
- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

plot_svd

plot_svd(*res: xr.Dataset, axes: Axes, linlog: bool = False, linthresh: float = 1, cycler: Cycler | None = cycler('color', [<ColorCode.black: '#000000'>, <ColorCode.red: '#ff0000'>, <ColorCode.blue: '#0000ff'>, <ColorCode.green: '#00ff00'>, <ColorCode.magenta: '#ff00ff'>, <ColorCode.cyan: '#00ffff'>, <ColorCode.yellow: '#ffff00'>, <ColorCode.green4: '#008b00'>, <ColorCode.orange: '#ff8c00'>, <ColorCode.brown: '#964b00'>, <ColorCode.grey: '#808080'>, <ColorCode.violet: '#9400d3'>, <ColorCode.turquoise: '#40e0d0'>, <ColorCode.maroon: '#800000'>, <ColorCode.indigo: '#4b0082'>]), nr_of_data_svd_vectors: int = 4, nr_of_residual_svd_vectors: int = 2, show_data_svd_legend: bool = True, show_residual_svd_legend: bool = True, irf_location: float | None = None, use_svd_number: bool = False*) → None

Plot SVD (Singular Value Decomposition) of data and residual.

Parameters

- **res** (*xr.Dataset*) – Result dataset
- **axes** (*Axes*) – Axes to plot the SVDs on (needs to be at least 2x3).
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().cycler.
- **nr_of_data_svd_vectors** (*int*) – Number of data SVD vector to plot. Defaults to 4.
- **nr_of_residual_svd_vectors** (*int*) – Number of residual SVD vector to plot. Defaults to 2.
- **show_data_svd_legend** (*bool*) – Whether or not to show the data SVD legend. Defaults to True.
- **show_residual_svd_legend** (*bool*) – Whether or not to show the residual SVD legend. Defaults to True.
- **irf_location** (*float | None*) – Location of the `irf` by which the time axis will get shifted. If it is None the time axis will not be shifted. Defaults to None.
- **use_svd_number** (*bool*) – Whether to use singular value number (starts at 1) instead of singular value index (starts at 0) for labeling in plot. Defaults to False.

plot_traces

Module containing functionality to plot fitted traces.

Functions

Summary

plot_data_and_fits	Plot data and fits for a given `wavelength` on a given `axis`.
plot_fitted_traces	Plot data and their fit in per wavelength plot grid.

plot_data_and_fits

plot_data_and_fits(*result: ResultLike, wavelength: float, axis: Axis, center_: float | None = None, main_irf_nr: int = 0, linlog: bool = False, linthresh: float = 1, divide_by_scale: bool = True, per_axis_legend: bool = False, y_label: str = 'a.u.', cycler: Cycler | None = cycler('color', [<DataColorCode.grey: '#808080'>, <DataColorCode.black: '#000000'>, <DataColorCode.orange: '#ff8c00'>, <DataColorCode.red: '#ff0000'>, <DataColorCode.cyan: '#00ffff'>, <DataColorCode.blue: '#0000ff'>, <DataColorCode.green: '#00ff00'>, <DataColorCode.green4: '#008b00'>, <DataColorCode.magenta: '#ff00ff'>, <DataColorCode.indigo: '#4b0082'>, <DataColorCode.brown: '#964b00'>, <DataColorCode.maroon: '#800000'>, <DataColorCode.yellow: '#ffff00'>, <DataColorCode.orange: '#ff8c00'>]), show_zero_line: bool = True*) → None

Plot data and fits for a given `wavelength` on a given `axis`.

If the wavelength isn't part of a dataset, that dataset will be skipped.

Parameters

- **result** (*ResultLike*) – Data structure which can be converted to a mapping.
- **wavelength** (*float*) – Wavelength to plot data and fits for.
- **axis** (*Axis*) – Axis to plot the data and fits on.
- **center_** (*float | None*) – Center wavelength (in nm)
- **main_irf_nr** (*int*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks. Defaults to 0.
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **divide_by_scale** (*bool*) – Whether or not to divide the data by the dataset scale used for optimization. Defaults to True.
- **per_axis_legend** (*bool*) – Whether to use a legend per plot or for the whole figure. Defaults to False.
- **y_label** (*str*) – Label used for the y-axis of each subplot.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().data_cycler_solid.
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True.

See also:

`plot_fit_overview`

plot_fitted_traces

plot_fitted_traces(*result: ResultLike, wavelengths: Iterable[float], axes_shape: tuple[int, int] = (4, 4), center_: float | None = None, main_irf_nr: int = 0, linlog: bool = False, linthresh: float = 1, divide_by_scale: bool = True, per_axis_legend: bool = False, figsize: tuple[float, float] = (30, 15), title: str = 'Fit overview', y_label: str = 'a.u.', cycler: Cycler | None = cycler('color', [<DataColorCode.grey: '#808080'>, <DataColorCode.black: '#000000'>, <DataColorCode.orange: '#ff8c00'>, <DataColorCode.red: '#ff0000'>, <DataColorCode.cyan: '#00ffff'>, <DataColorCode.blue: '#0000ff'>, <DataColorCode.green: '#00ff00'>, <DataColorCode.green4: '#008b00'>, <DataColorCode.magenta: '#ff00ff'>, <DataColorCode.indigo: '#4b0082'>, <DataColorCode.brown: '#964b00'>, <DataColorCode.maroon: '#800000'>, <DataColorCode.yellow: '#ffff00'>, <DataColorCode.orange: '#ff8c00'>]), show_zero_line: bool = True*) → tuple[Figure, Axes]

Plot data and their fit in per wavelength plot grid.

Parameters

- **result** (*ResultLike*) – Data structure which can be converted to a mapping of datasets.
- **wavelengths** (*Iterable[float]*) – Wavelength which should be used for each subplot, should to be of length N*M with `axes_shape` being of shape (N, M), else it will result in missing plots.
- **axes_shape** (*tuple[int, int]*) – Shape of the plot grid (N, M). Defaults to (4, 4).
- **center_** (*float | None*) – Center wavelength of the IRF (in nm).
- **main_irf_nr** (*int*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks. Defaults to 0.
- **linlog** (*bool*) – Whether to use 'symlog' scale or not. Defaults to False.
- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear. This avoids having the plot go to infinity around zero. Defaults to 1.
- **divide_by_scale** (*bool*) – Whether or not to divide the data by the dataset scale used for optimization. Defaults to True.
- **per_axis_legend** (*bool*) – Whether to use a legend per plot or for the whole figure. Defaults to False.
- **figsize** (*tuple[float, float]*) – Size of the figure (N, M) in inches. Defaults to (30, 15).
- **title** (*str*) – Title to add to the figure. Defaults to "Fit overview".
- **y_label** (*str*) – Label used for the y-axis of each subplot.
- **cycler** (*Cycler | None*) – Plot style cycler to use. Defaults to PlotStyle().data_cycler_solid.
- **show_zero_line** (*bool*) – Whether or not to add a horizontal line at zero. Defaults to True.

Returns
Figure and axes which can then be refined by the user.

Return type
tuple[Figure, Axes]

See also:

`maximum_coordinate_range`, `add_unique_figure_legend`, *plot_data_and_fits*, `calculate_wavelengths`

style

Module containing predefined plot styles.

For reference see: https://glotaran.github.io/legacy/plot_styles

Classes

Summary

ColorCode	Color definitions from legacy glotaran.
DataColorCode	Colors used to plot data and fits.
DataLineStyle	Data plots can alternate between solid lines for data and dashed lines for fits.
LineStyle	Subset of line styles supported by matplotlib.
PlotStyle	Wrapper class to hold predefined Plot styles.

ColorCode

class ColorCode(*value*)

Color definitions from legacy glotaran.

See: https://glotaran.github.io/legacy/plot_styles

Attributes Summary

black
red
blue
green
magenta
cyan
yellow
green4
orange
brown
grey
violet
turquoise
maroon
indigo

Methods Summary

hex_to_rgb	Convert hex code to rgb or rgba tuple.
rgb_to_hex	Convert rgb value tuple to hex code.

hex_to_rgb

static ColorCode.**hex_to_rgb**(*hex_string: str*) → tuple[int, ...]

Convert hex code to rgb or rgba tuple.

Parameters
hex_string (*str*) – Hey code representation of a color.

Returns
rgb or rgba tuple representing the same color as `hex_string`..

Return type
tuple[int, . . .]

rgb_to_hex

static ColorCode.**rgb_to_hex**(*rgb_tuple: tuple[float, ...]*) → str

Convert rgb value tuple to hex code.

Parameters
rgb_tuple (*tuple[float, ...]*) – Tuple rgb or rgba values

Returns
Hex code representing the same color as `rgb_tuple`.

Return type
str

Methods Documentation

static **hex_to_rgb**(*hex_string: str*) → tuple[int, ...]

Convert hex code to rgb or rgba tuple.

Parameters
hex_string (*str*) – Hey code representation of a color.

Returns
rgb or rgba tuple representing the same color as `hex_string`..

Return type
tuple[int, . . .]

static **rgb_to_hex**(*rgb_tuple: tuple[float, ...]*) → str

Convert rgb value tuple to hex code.

Parameters
rgb_tuple (*tuple[float, ...]*) – Tuple rgb or rgba values

Returns
Hex code representing the same color as `rgb_tuple`.

Return type
str

DataColorCode

class DataColorCode(*value*)

Colors used to plot data and fits.

Pairs of visually similar looking colors whereby the first (lighter) color is used to plot the data, and the second (darker) color is used to represent the fitted trace (that goes 'through' the data).

Attributes Summary

grey
black
orange
red
cyan
blue
green
green4
magenta
indigo
brown
maroon
yellow

DataLineStyle

class DataLineStyle(*value*)

Data plots can alternate between solid lines for data and dashed lines for fits.

This is mostly useful for data with very low noise (e.g. simulated data), since data and fit often overlap.

Attributes Summary

solid	
dashed	

LineStyle

class LineStyle(*value*)

Subset of line styles supported by matplotlib.

Attributes Summary

solid	
dashed	
dotted	
dashdot	

PlotStyle

class PlotStyle

Wrapper class to hold predefined Plot styles.

Initialize Stiles from Enums.

Attributes Summary

cycler	Cycle for default legacy like plots using color defined in *ColorCode*.
data_cycler_solid	Color cycler using *DataColorCode*.
data_cycler_solid_dashed	Cycle that alternates between solid and dashes and uses *DataColorCode*.
line_cycler	Cycle iterating over line styles defined in *LineStyle*.

Methods Summary

set_default_colors	Set default color cycler for matplotlib.
set_default_fontsize	Set global plot settings for matplotlib.

set_default_colors

PlotStyle.**set_default_colors**() → None

Set default color cycler for matplotlib.

set_default_fontsize

PlotStyle.**set_default_fontsize**() → None

Set global plot settings for matplotlib.

Methods Documentation

set_default_colors() → None

Set default color cycler for matplotlib.

set_default_fontsize() → None

Set global plot settings for matplotlib.

utils

Module containing plotting utility functionality.

Functions

Summary

abs_max	Calculate the absolute maximum values of data along all dims except result_dims.
add_cycler_if_not_none	Add cycler to and axis if it is not None.
add_subplot_labels	Add labels to all subplots in axes in a consistent manner.
add_unique_figure_legend	Add a legend with unique elements sorted by label to a figure.
calculate_ticks_in_units_of_pi	Calculate tick values and labels in units of Pi.
ensure_axes_array	Ensure that axes have flatten method even if it is a single axis.
extract_dataset_scale	Extract 'dataset_scale' attribute from optimization result dataset.
extract_irf_dispersion_center	Extract the IRF dispersion center data from a result dataset where irf_nr==irf_nr_index.
extract_irf_location	Determine location of the irf, which can be used to shift plots.
format_sub_plot_number_upper_case_lette	Format sub_plot_number into an upper case letter, that can be used as label.
get_shifted_traces	Shift traces by the position of the main irf.
get_subplot_label_format_function	Get subplot label function from BuiltinSubPlotLabelFormatFunctions if it is a key.
maximum_coordinate_range	Calculate the minimal and maximal values of a coordinate across datasets.
not_single_element_dims	Names of dimensions in data which don't have a size equal to one.
select_irf_dispersion_center_by_index	Select a subset of the IRF dispersion data where irf_nr==main_irf_nr.
select_plot_wavelengths	Select wavelengths to be used in plot_fit_overview from a result.
shift_time_axis_by_irf_location	Shift plot_data 'time' axis by the position of the main irf.

abs_max

abs_max(*data: xr.DataArray*, *, *result_dims: Hashable | Iterable[Hashable] = ()*) → xr.DataArray

Calculate the absolute maximum values of data along all dims except result_dims.

Parameters

- **data** (*xr.DataArray*) – Data for which the absolute maximum should be calculated.
- **result_dims** (*Hashable | Iterable[Hashable]*) – Dimensions of data which should be preserved and part of the resulting DataArray. Defaults to () which results in using the absolute maximum of all values.

Returns

Absolute maximum values of data with dimensions result_dims.

Return type
xr.DataArray

add_cycler_if_not_none

add_cycler_if_not_none(*axis: Axis | Axes*, *cycler: Cycler | None*) → None

Add cycler to and axis if it is not None.

This is a convenience function that allow to opt out of using a cycler, which is needed to run a plotting function in a loop where the cycler is controlled from the outside.

Parameters

- **axis** (*Axis | Axes*) – Axis to plot on.
- **cycler** (*Cycler | None*) – Plot style cycler to use.

add_subplot_labels

add_subplot_labels(*axes: Axis | Axes*, ***, *label_position: tuple[float, float] = (-0.05, 1.05)*, *label_coords: SubPlotLabelCoord = 'axes fraction'*, *direction: Literal['row', 'column'] = 'row'*, *label_format_template: str = '{}'*, *label_format_function: BuiltinSubPlotLabelFormatFunctionKey | Callable[[int, int | None], str] = 'number'*, *fontsize: int = 16*) → None

Add labels to all subplots in `axes` in a consistent manner.

Parameters

- **axes** (*Axis | Axes*) – Axes (subplots) on which the labels should be added.
- **label_position** (*tuple[float, float]*) – Position of the label in `label_coords` coordinates.
- **label_coords** (*SubPlotLabelCoord*) – Coordinate system used for `label_position`. Defaults to "axes fraction"
- **direction** (*Literal["row", "column"]*) – Direct in which the axes should be iterated in. Defaults to "row"
- **label_format_template** (*str*) – Template string to inject the return value of `label_format_function` into. Defaults to "{}"
- **label_format_function** (*BuiltinSubPlotLabelFormatFunctionKey | Callable[[int, int | None], str]*) – Function to calculate the label for the axis index and `axes` size. Defaults to "number"
- **fontsize** (*int*) – Font size used for the label. Defaults to 16

add_unique_figure_legend

add_unique_figure_legend(*fig: Figure*, *axes: Axes*) → None

Add a legend with unique elements sorted by label to a figure.

The handles and labels are extracted from the `axes`

Parameters

- **fig** (*Figure*) – Figure to add the legend to.
- **axes** (*Axes*) – Axes plotted on the figure.

See also:

`plot_fit_overview`

calculate_ticks_in_units_of_pi

calculate_ticks_in_units_of_pi(*values: ndarray | DataArray*, ***, *step_size: float = 0.5*) → tuple[Iterable[float], Iterable[str]]

Calculate tick values and labels in units of Pi.

Parameters

- **values** (*np.ndarray | xr.DataArray*) – Values which the ticks should be calculated for.
- **step_size** (*float*) – Step size of the ticks in units of pi. Defaults to 0.5

Returns
Tick values and tick labels

Return type
tuple[Iterable[float], Iterable[str]]

See also:

pyglotaran_extras.plotting.plot_doas.plot_doas

Examples

If you have a case study that uses a `damped-oscillation` megacomplex you can plot the `damped_oscillation_phase` with y-tick in units of Pi by the following code given that the dataset is saved under `dataset.nc`.

```
import matplotlib.pyplot as plt

from glotaran.io import load_dataset
from pyglotaran_extras.plotting.utils import calculate_ticks_in_units_of_pi

dataset = load_dataset("dataset.nc")

fig, ax = plt.subplots(1, 1)

damped_oscillation_phase = dataset["damped_oscillation_phase"].sel(
    damped_oscillation=["osc1"]
```

(continues on next page)

(continued from previous page)

```
)
damped_oscillation_phase.plot.line(x="spectral", ax=ax)

ax.set_yticks(
    *calculate_ticks_in_units_of_pi(damped_oscillation_phase), rotation=
↪"horizontal"
)
```

ensure_axes_array

ensure_axes_array(*axes: Axis | Axes*) → Axes

Ensure that axes have flatten method even if it is a single axis.

Parameters

axes (*Axis | Axes*) – Axis or Axes to convert for API consistency.

Returns

Numpy ndarray of axes.

Return type

Axes

extract_dataset_scale

extract_dataset_scale(*res: Dataset*, *divide_by_scale: bool = True*) → float

Extract 'dataset_scale' attribute from optimization result dataset.

Parameters

- **res** (*xr.Dataset*) – Result dataset from a pyglotaran optimization.
- **divide_by_scale** (*bool*) – Whether or not to divide the data by the dataset scale used for optimization. Defaults to True.

Returns

Dataset scale extracted from `res` falls back to 1 if not present.

Return type

float

extract_irf_dispersion_center

extract_irf_dispersion_center(*res: Dataset*, *main_irf_nr: int = 0*, ***, *as_dataarray: bool = True*) → DataArray | float

Extract the IRF dispersion center data from a result dataset where `irf_nr==irf_nr_index`.

Parameters

- **res** (*xr.Dataset*) – Result dataset from a pyglotaran optimization.
- **main_irf_nr** (*int*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks. Defaults to 0.
- **as_dataarray** (*bool*) – Ensure that the returned data are xr.DataArray instead of a float, even if the dispersion is none existent or constant. Defaults to True

Returns
IRF dispersion data as float or xr.DataArray

Return type
xr.DataArray | float

extract_irf_location

extract_irf_location(*res: Dataset*, *center_: float | None = None*, *main_irf_nr: int | None = 0*) → float

Determine location of the `irf`, which can be used to shift plots.

Parameters

- **res** (*xr.Dataset*) – Result dataset from a pyglotaran optimization.
- **center_** (*float | None*) – Center wavelength (in nm)
- **main_irf_nr** (*int | None*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks. If it is none `None` the location will be 0. Defaults to 0.

Returns
Location of the `irf`

Return type
float

format_sub_plot_number_upper_case_letter

format_sub_plot_number_upper_case_letter(*sub_plot_number: int*, *size: int | None = None*) → str

Format `sub_plot_number` into an upper case letter, that can be used as label.

Parameters

- **sub_plot_number** (*int*) – Number of the subplot starting at One.
- **size** (*None | int*) – Size of the axes array (number of plots). Defaults to None

Returns
Upper case label for a sub plot.

Return type
str

Examples

```
>>> print(format_sub_plot_number_upper_case_letter(1))
A
```

```
>>> print(format_sub_plot_number_upper_case_letter(26))
Z
```

```
>>> print(format_sub_plot_number_upper_case_letter(27))
AA
```

```
>>> print(format_sub_plot_number_upper_case_letter(1, 26))
AA
```

```
>>> print(format_sub_plot_number_upper_case_letter(2, 26))
AB
```

```
>>> print(format_sub_plot_number_upper_case_letter(26, 26))
AZ
```

```
>>> print(format_sub_plot_number_upper_case_letter(27, 50))
BA
```

See also:

BuiltinLabelFormatFunctions, *add_subplot_labels*

get_shifted_traces

get_shifted_traces(*res: Dataset*, *center_: float | None = None*, *main_irf_nr: int = 0*) → DataArray

Shift traces by the position of the main `irf`.

Parameters

- **res** (*xr.Dataset*) – Result dataset from a pyglotaran optimization.
- **center_** (*float | None*) – Center wavelength (in nm). Defaults to None.
- **main_irf_nr** (*int*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks. Defaults to 0.

Returns

Traces shifted by the ``irf``s location, to align the at 0.

Return type

xr.DataArray

Raises

ValueError – If no known concentration was found in the result.

get_subplot_label_format_function

get_subplot_label_format_function(*format_function: BuiltinSubPlotLabelFormatFunctionKey | Callable[[int, int | None], str]*) → Callable[[int, int | None], str]

Get subplot label function from `BuiltinSubPlotLabelFormatFunctions` if it is a key.

This function is mainly needed for typing reasons.

Parameters

format_function (*BuiltinSubPlotLabelFormatFunctionKey | Callable[[int, int | None], str]*) – Key

`BuiltinSubPlotLabelFormatFunctions` to retrieve builtin function or user defined format function.

Returns
Function to format subplot label.

Return type
Callable[[int, int | None], str]

maximum_coordinate_range

maximum_coordinate_range(*result: ResultLike*, *coord_name: str = 'spectral'*) → tuple[float, float]

Calculate the minimal and maximal values of a coordinate across datasets.

Parameters

- **result** (*ResultLike*) – Data structure which can be converted to a mapping.
- **coord_name** (*str*) – Name of the coordinate to calculate the maximal range for.

Returns
Minimal and maximal values across datasets

Return type
tuple[float, float]

See also:

`plot_fit_overview`

not_single_element_dims

not_single_element_dims(*data_array: xr.DataArray*) → list[Hashable]

Names of dimensions in `data` which don't have a size equal to one.

This helper function is for example used to determine if a data only have a single trace, since this requires different plotting code (e.g. `data_array.plot.line(x="time")`).

Parameters
data_array (*xr.DataArray*) – DataArray to check if it has only a single dimension.

Returns
Names of dimensions in `data` which don't have a size equal to one.

Return type
list[Hashable]

select_irf_dispersion_center_by_index

select_irf_dispersion_center_by_index(*irf_dispersion: DataArray*, *main_irf_nr: int = 0*) → DataArray | float

Select a subset of the IRF dispersion data where `irf_nr==main_irf_nr`.

Parameters

- **irf_dispersion** (*xr.DataArray*) – Data Variable from a result dataset which contains the IRF dispersion data.

- **main_irf_nr** (*int*) – Index of the main `irf` component when using an `irf` parametrized with multiple peaks. Defaults to 0.

Returns
DataArray only containing the IRF dispersion data for the main IRF.

Return type
xr.DataArray | float

Raises
ValueError – If `irf_nr` is not in the coordinates

select_plot_wavelengths

select_plot_wavelengths(*result: ResultLike*, *axes_shape: tuple[int, int] = (4, 4)*, *wavelength_range: tuple[float, float] | None = None*, *equidistant_wavelengths: bool = True*) → Iterable[float]

Select wavelengths to be used in `plot_fit_overview` from a result.

Parameters

- **result** (*ResultLike*) – Data structure which can be converted to a mapping of datasets.
- **axes_shape** (*tuple[int, int]*) – Shape of the plot grid (N, M). Defaults to (4, 4).
- **wavelength_range** (*tuple[float, float] | None*) – Tuple of minimum and maximum values to calculate the the wavelengths used for plotting. If not provided the values will be tetermined over all datasets. Defaults to None.
- **equidistant_wavelengths** (*bool*) – Whether or not wavelengths should be selected based on equidistant values or equidistant indices (only supported for a single dataset). Since in general multiple datasets will have. Defaults to True.

Returns
Wavelength which should be used for each subplot by `plot_fit_overview`.

Return type
Iterable[float]

See also:

maximum_coordinate_range

shift_time_axis_by_irf_location

shift_time_axis_by_irf_location(*plot_data: DataArray*, *irf_location: float | None*, *, *_internal_call: bool = False*) → DataArray

Shift `plot_data` 'time' axis by the position of the main `irf`.

Parameters

- **plot_data** (*xr.DataArray*) – Data to plot.
- **irf_location** (*float | None*) – Location of the `irf`, if the value is None the original `plot_data` will be returned.

- **_internal_call** (*bool*) – This indicates internal use stripping away user help and silently skipping execution. Defaults to False.

Returns
plot_data with the time axis shifted by the position of the main irf.

Return type
xr.DataArray

Raises
ValueError – If plot_data does not have a time axis.

See also:

extract_irf_location

Classes

Summary

MinorSymLogLocator	Dynamically find minor tick positions based on major ticks for a symlog scaling.

MinorSymLogLocator

class MinorSymLogLocator(*linthresh: float*, *nints: int = 10*)

Dynamically find minor tick positions based on major ticks for a symlog scaling.

Ref.: https://stackoverflow.com/a/45696768

Ticks will be placed between the major ticks.

The placement is linear for x between -linthresh and linthresh, otherwise its logarithmically. nints gives the number of intervals that will be bounded by the minor ticks.

Parameters

- **linthresh** (*float*) – A single float which defines the range (-x, x), within which the plot is linear.
- **nints** (*int*) – Number of minor tick between major ticks. Defaults to 10

Attributes Summary

MAXTICKS
axis

Methods Summary

create_dummy_axis	
nonsingular	Adjust a range as needed to avoid singularities.
raise_if_exceeds	Log at WARNING level if *locs* is longer than *Locator.MAXTICKS*.
set_axis	
set_params	Do nothing, and raise a warning.
tick_values	Return the values of the located ticks given **_vmin** and **_vmax** (not implemented).
view_limits	Select a scale for the range from vmin to vmax.

create_dummy_axis

MinorSymLogLocator.**create_dummy_axis**(***kwargs*)

nonsingular

MinorSymLogLocator.**nonsingular**(*v0*, *v1*)

Adjust a range as needed to avoid singularities.

This method gets called during autoscaling, with `(v0, v1)` set to the data limits on the axes if the axes contains any data, or `(-inf, +inf)` if not.

- If `v0 == v1` (possibly up to some floating point slop), this method returns an expanded interval around this value.
- If `(v0, v1) == (-inf, +inf)`, this method returns appropriate default view limits.
- Otherwise, `(v0, v1)` is returned without modification.

raise_if_exceeds

MinorSymLogLocator.**raise_if_exceeds**(*locs*)

Log at WARNING level if *locs* is longer than *Locator.MAXTICKS*.

This is intended to be called immediately before returning *locs* from `__call__` to inform users in case their Locator returns a huge number of ticks, causing Matplotlib to run out of memory.

The "strange" name of this method dates back to when it would raise an exception instead of emitting a log.

set_axis

MinorSymLogLocator.**set_axis**(*axis*)

set_params

MinorSymLogLocator.**set_params**(***kwargs*)

Do nothing, and raise a warning. Any locator class not supporting the set_params() function will call this.

tick_values

MinorSymLogLocator.**tick_values**(*_vmin: float*, *_vmax: float*) → None

Return the values of the located ticks given **_vmin** and **_vmax** (not implemented).

Parameters

- **_vmin** (*float*) – Minimum value.
- **_vmax** (*float*) – Maximum value.

Raises

NotImplementedError – Not used

view_limits

MinorSymLogLocator.**view_limits**(*vmin*, *vmax*)

Select a scale for the range from vmin to vmax.

Subclasses should override this method to change locator behaviour.

Methods Documentation

create_dummy_axis(***kwargs*)

nonsingular(*v0*, *v1*)

Adjust a range as needed to avoid singularities.

This method gets called during autoscaling, with `(v0, v1)` set to the data limits on the axes if the axes contains any data, or `(-inf, +inf)` if not.

- If `v0 == v1` (possibly up to some floating point slop), this method returns an expanded interval around this value.
- If `(v0, v1) == (-inf, +inf)`, this method returns appropriate default view limits.
- Otherwise, `(v0, v1)` is returned without modification.

raise_if_exceeds(*locs*)

Log at WARNING level if *locs* is longer than *Locator.MAXTICKS*.

This is intended to be called immediately before returning *locs* from `__call__` to inform users in case their Locator returns a huge number of ticks, causing Matplotlib to run out of memory.

The "strange" name of this method dates back to when it would raise an exception instead of emitting a log.

set_axis(*axis*)

set_params(***kwargs*)

Do nothing, and raise a warning. Any locator class not supporting the set_params() function will call this.

tick_values(*_vmin: float*, *_vmax: float*) → None

Return the values of the located ticks given **_vmin** and **_vmax** (not implemented).

Parameters

- **_vmin** (*float*) – Minimum value.
- **_vmax** (*float*) – Maximum value.

Raises

NotImplementedError – Not used

view_limits(*vmin*, *vmax*)

Select a scale for the range from vmin to vmax.

Subclasses should override this method to change locator behaviour.

Exceptions

Exception Summary

PlotDuplicationWarning	Warning given when there are more subplots than datapoints.

PlotDuplicationWarning

exception PlotDuplicationWarning

Warning given when there are more subplots than datapoints.

4.1.5 types

Module containing type definitions.

Module Attributes

Unset	Value to use as default for an arguments where None is a meaningful value.
DatasetConvertible	Types of data which can be converted to a dataset.
ResultLike	Result like data which can be converted to a per dataset mapping.
BuiltinSubPlotLabelFormatFunctionKey	Key supported by `BuiltinLabelFormatFunctions`.

pyglotaran_extras.types.Unset

Unset = Unset

Value to use as default for an arguments where None is a meaningful value.

This way we can prevent regressions.

pyglotaran_extras.types.DatasetConvertible

DatasetConvertible: TypeAlias = xarray.core.dataset.Dataset | xarray.core.dataarray.DataArray | str | pathlib.Path

Types of data which can be converted to a dataset.

pyglotaran_extras.types.ResultLike

ResultLike: TypeAlias = glotaran.project.result.Result | xarray.core.dataset.Dataset | xarray.core.dataarray.DataArray | str | pathlib.Path | collections.abc.Mapping[str, xarray.core.dataset.Dataset | xarray.core.dataarray.DataArray | str | pathlib.Path] | collections.abc.Sequence[xarray.core.dataset.Dataset | xarray.core.dataarray.DataArray | str | pathlib.Path]

Result like data which can be converted to a per dataset mapping.

pyglotaran_extras.types.BuiltinSubPlotLabelFormatFunctionKey

BuiltinSubPlotLabelFormatFunctionKey

Key supported by `BuiltinLabelFormatFunctions`.

alias of `Literal`['number', 'upper_case_letter', 'lower_case_letter']

Classes

Summary

UnsetType	Type for the `Unset` singleton.

UnsetType

class UnsetType

Type for the `Unset` singleton.

Methods Summary

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/glotaran/pyglotaran-extras/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

pyglotaran_extras could always use more documentation, whether as part of the official pyglotaran_extras docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/glotaran/pyglotaran-extras/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up `pyglotaran_extras` for local development.

1. Fork the `pyglotaran-extras` repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pyglotaran_extras.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv pyglotaran_extras
$ cd pyglotaran-extras/
$ pip install -e .
```

4. install the `pre-commit` and `pre-push` hooks:

```
$ pre-commit install && pre-commit install -t pre-push
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.md.
3. The pull request should work for Python 3.10 and 3.11. Check https://github.com/glotaran/pyglotaran-extras/actions and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_pyglotaran_extras
```

CHAPTER

SIX

CHANGELOG

6.1 0.7.2 (2023-12-07)

- Switch tooling to ruff (#197)
- Fix crash when plotting spectral model result (#200)
- Use hatch as build backend (#204)
- Use black-pre-commit-mirror for 2x speedup (#205)
- Use ruff for formatting (#214)
- Use weighted residual instead of residual plots if present (#216)
- Add color map arguments to plot_data_overview (#217)
- Add das_cycler and svd_cycler to plot collection functions (#218)
- Add use_svd_number switch to use SV number instead of index as label (#219)
- Fix use_svd_number not being passed on to plot_sv_data in plot_data_overview (#221)
- Use trusted publisher workflow for publishing to PyPI (#222)

6.2 0.7.1 (2023-07-27)

- Fix crashes of plot_doas and plot_coherent_artifact for non dispersive IRF (#173, #182)
- Add minor ticks to linlog plots (#183)
- Remove upper python version limit (#174)
- Add add_subplot_labels function (#181)

6.3 0.7.0 (2023-04-15)

- Fix typo in internal function name (#94)
- Add IRF dispersion center plotting (#95)
- Improve typing (#96)
- Shift time axis by IRF location (#99)
- Fix ‘Test pyglotaran dev’ CI step (#117)

- Add option to deactivate data/residual plotting in overview plots (#118)
- Add coherent artifact plot functionality (#123)
- Add a-matrix inspection function (#124)
- Add show_zero_line option to plot_overview and plot_fitted_traces (#128)
- Fix SVD vector labels always starting from zero (#133)
- Make result input for plot_coherent_artifact more generic (#134)
- Add plot_doas function that only plots DOAS related information (#135)
- Shift irf derivatives in plot_coherent_artifact by irf position (#136)
- Convert plot style Enums to StrEnums for ease of use (#144)
- Fix heading in show_a_matrixes for multiple a-matrixes per dataset (#148)
- Add legend_format_string argument to plot_doas (#150)
- Make plot_data_overview able to plot single trace data (#137)
- Improve plot_doas default legend_format_string (#151)
- Fix figsize typing (#152)
- Fix search in docs (#157)

6.4 0.6.0 (2022-06-07)

- Made adding a cycler to and axis opt out by using None by @s-weigand in #58
- Fix autogenerated title for concentration plot by @s-weigand in #63
- Improve SVD plotting by @s-weigand in #64
- Improve readme by @s-weigand in #69
- Add sourcery-ai config by @s-weigand in #71
- Remove github section in sourcery config due to bug in sourcery-ai by @s-weigand in #72
- Improve legend placement in plot_data_overview by @s-weigand in #77

6.5 0.5.0 (2022-02-05)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

N

O

P

R

S

T

U

V

W