

---

# **pyglotaran Documentation**

***Release v0.7.2***

**Joern Weissenborn, Joris Snellenburg, Ivo van Stokkum**

**2023-12-06**



## CONTENTS:

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
<b>3</b>	<b>Quickstart/Cheat-Sheet</b>	<b>5</b>
<b>4</b>	<b>Import the data into your project</b>	<b>9</b>
<b>5</b>	<b>Changelog</b>	<b>21</b>
<b>6</b>	<b>Authors</b>	<b>33</b>
<b>7</b>	<b>Overview</b>	<b>35</b>
<b>8</b>	<b>Data IO</b>	<b>37</b>
<b>9</b>	<b>Plotting</b>	<b>39</b>
<b>10</b>	<b>Modelling</b>	<b>41</b>
<b>11</b>	<b>Parameter</b>	<b>43</b>
<b>12</b>	<b>Optimizing</b>	<b>45</b>
<b>13</b>	<b>Plugins</b>	<b>47</b>
<b>14</b>	<b>Command-line Interface</b>	<b>49</b>
<b>15</b>	<b>Contributing</b>	<b>53</b>
<b>16</b>	<b>API Documentation</b>	<b>61</b>
<b>17</b>	<b>Plugin development</b>	<b>579</b>
<b>18</b>	<b>Indices and tables</b>	<b>587</b>
	<b>Bibliography</b>	<b>589</b>
	<b>Python Module Index</b>	<b>591</b>
	<b>Index</b>	<b>593</b>



## INTRODUCTION

Pyglotaran is a python library for global analysis of time-resolved spectroscopy data. It is designed to provide a state of the art modeling toolbox to researchers, in a user-friendly manner.

Its features are:

- user-friendly modeling with a custom YAML (\*.yaml) based modeling language
- parameter optimization using variable projection and non-negative least-squares algorithms
- easy to extend modeling framework
- battle-hardened model and algorithms for fluorescence dynamics
- build upon and fully integrated in the standard Python science stack (NumPy, SciPy, Jupyter)

### 1.1 A Note To Glotaran Users

Although closely related and developed in the same lab, pyglotaran is not a replacement for Glotaran - A GUI For TIMP. Pyglotaran only aims to provide the modeling and optimization framework and algorithms. It is of course possible to develop a new GUI which leverages the power of pyglotaran (contributions welcome).

The current ‘user-interface’ for pyglotaran is Jupyter Notebook. It is designed to seamlessly integrate in this environment and be compatible with all major visualization and data analysis tools in the scientific python environment.

If you are a non-technical user, you should give these tools a try, there are numerous tutorials how to use them. You don’t need to really learn to program. If you can use e.g. Matlab or Mathematica, you can use Jupyter and Python.



## INSTALLATION

### 2.1 Prerequisites

- Python 3.10 or 3.11

#### 2.1.1 Windows

The easiest way of getting Python (and some basic tools to work with it) in Windows is to use [Anaconda](#), which provides python.

You will need a terminal for the installation. One is provided by *Anaconda* and is called *Anaconda Console*. You can find it in the start menu.

---

**Note:** If you use a Windows Shell like cmd.exe or PowerShell, you might have to prefix ‘\$PATH\_TO\_ANACONDA/’ to all commands (e.g. *C:/Anaconda/pip.exe* instead of *pip*)

---

### 2.2 Stable release

**Warning:** pyglotaran is early development, so for the moment stable releases are sparse and outdated. We try to keep the master code stable, so please install from source for now.

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

To install pyglotaran, run this command in your terminal:

```
$ pip install pyglotaran
```

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

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

```
$ conda install -c conda-forge pyglotaran
```

## 2.3 From sources

First you have to install or update some dependencies.

Within a terminal:

```
$ pip install -U numpy scipy Cython
```

Alternatively, for Anaconda users:

```
$ conda install numpy scipy Cython
```

Afterwards you can simply use `pip` to install it directly from [Github](#).

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

For updating pyglotaran, just re-run the command above.

If you prefer to manually download the source files, you can find them on [Github](#). Alternatively you can clone them with `git` (preferred):

```
$ git clone https://github.com/glotaran/pyglotaran.git
```

Within a terminal, navigate to directory where you have unpacked or cloned the code and enter

```
$ pip install -e .
```

For updating, simply download and unpack the newest version (or run `$ git pull` in pyglotaran directory if you used `git`) and re-run the command above.

The following section was generated from docs/source/notebooks/quickstart/quickstart.ipynb .....



## QUICKSTART/CHEAT-SHEET

To start using pyglotaran in your analysis, you only have to import the Project class and open a project.

```
[1]: from glotaran.project import Project

quickstart_project = Project.open("quickstart_project")
quickstart_project
```

[1]: **3.1 Project (*quickstart\_project*)**

pyglotaran version: 0.7.2

### 3.1.1 Data

*None*

### 3.1.2 Model

- my\_model

### 3.1.3 Parameters

- my\_parameters

### 3.1.4 Results

*None*

If the project does not already exist this will create a new project and its folder structure for you. In our case we had only the models + parameters folders and the data + results folder were created when opening the project.

```
[2]: %ls quickstart_project

data/  models/  parameters/  project.gta  results/
```

Let us get some example data to analyze:

```
[3]: from glotaran.testing.simulated_data.sequential_spectral_decay import DATASET as my_
      ↪ dataset
```

(continues on next page)

(continued from previous page)

my\_dataset

```
[3]: <xarray.Dataset>
Dimensions:   (time: 2100, spectral: 72)
Coordinates:
  * time      (time) float64 -1.0 -0.99 -0.98 -0.97 ... 19.96 19.97 19.98 19.99
  * spectral  (spectral) float64 600.0 601.4 602.8 604.2 ... 696.6 698.0 699.4
Data variables:
  data       (time, spectral) float64 -0.0003386 0.002708 ... 2.549 2.312
Attributes:
  source_path: dataset_1.nc
```

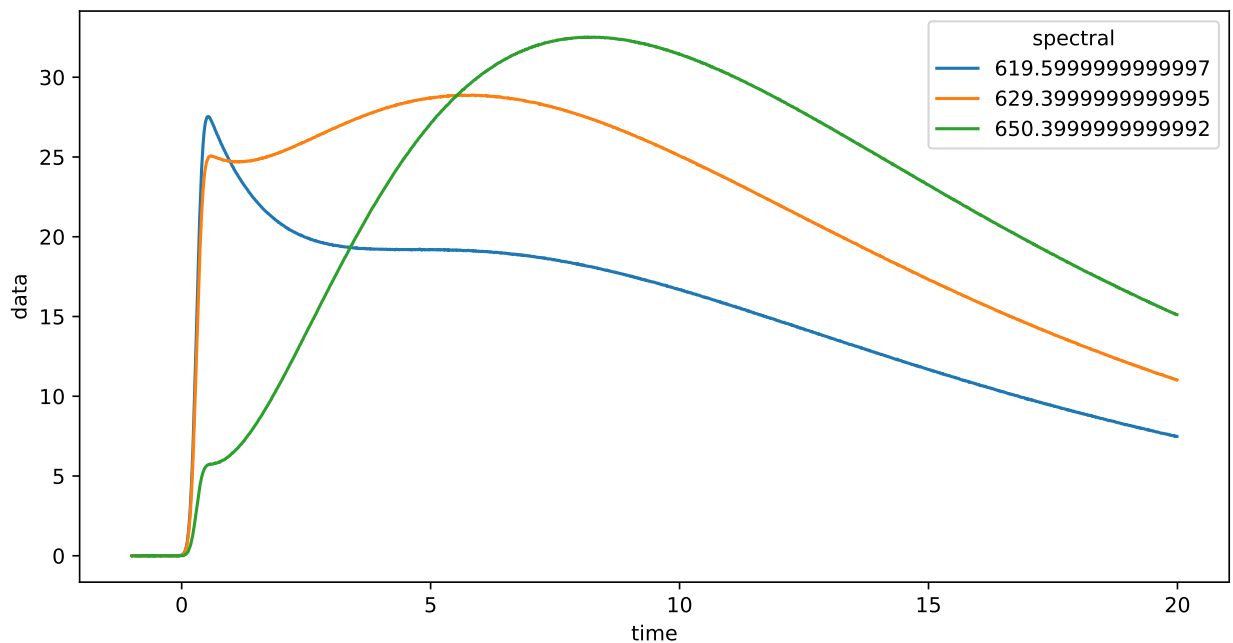
Like all data in pyglotaran, the dataset is a `xarray.Dataset`. You can find more information about the `xarray` library the [xarray homepage](#).

The loaded dataset is a simulated sequential model.

## 3.2 Plotting raw data

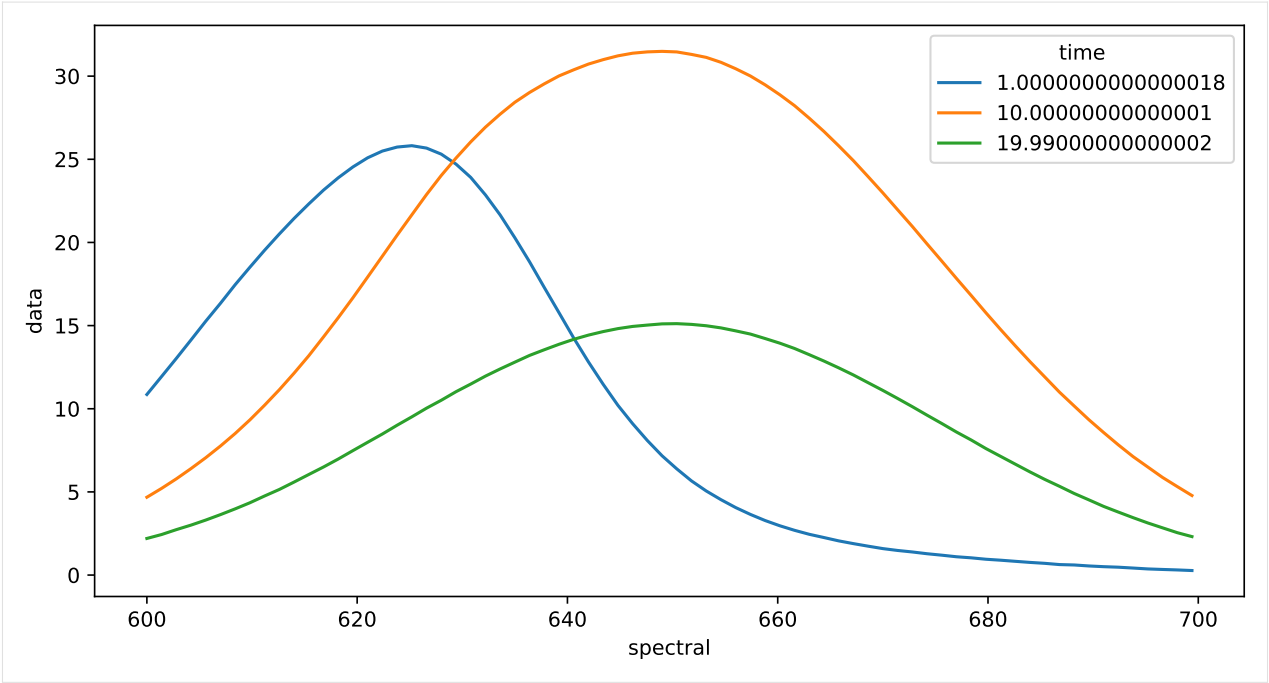
Now lets plot some time traces.

```
[4]: plot_data = my_dataset.data.sel(spectral=[620, 630, 650], method="nearest")
plot_data.plot.line(x="time", aspect=2, size=5);
```



We can also plot spectra at different times.

```
[5]: plot_data = my_dataset.data.sel(time=[1, 10, 20], method="nearest")
plot_data.plot.line(x="spectral", aspect=2, size=5);
```





## IMPORT THE DATA INTO YOUR PROJECT

As long as you can read your data into a `xarray.Dataset` or `xarray.DataArray` you can directly import it in to your project.

This will save your data as NetCDF (.nc) file into the data folder inside of your project with the name that you gave it (here `quickstart_project/data/my_data.nc`).

If the data format you are using is supported by a plugin you can simply copy the file to the data folder of the project (here `quickstart_project/data`).

```
[6]: quickstart_project.import_data(my_dataset, dataset_name="my_data")
      quickstart_project
```

[6]: **4.1 Project (*quickstart\_project*)**

pyglotaran version: 0.7.2

### 4.1.1 Data

- `my_data`

### 4.1.2 Model

- `my_model`

### 4.1.3 Parameters

- `my_parameters`

### 4.1.4 Results

*None*

After importing our `quickstart_project` is aware of the data that we named `my_data` when importing.

## 4.2 Preparing data

To get an idea about how to model your data, you should inspect the singular value decomposition. As a convenience the `load_data` method has the option to add svd data on the fly.

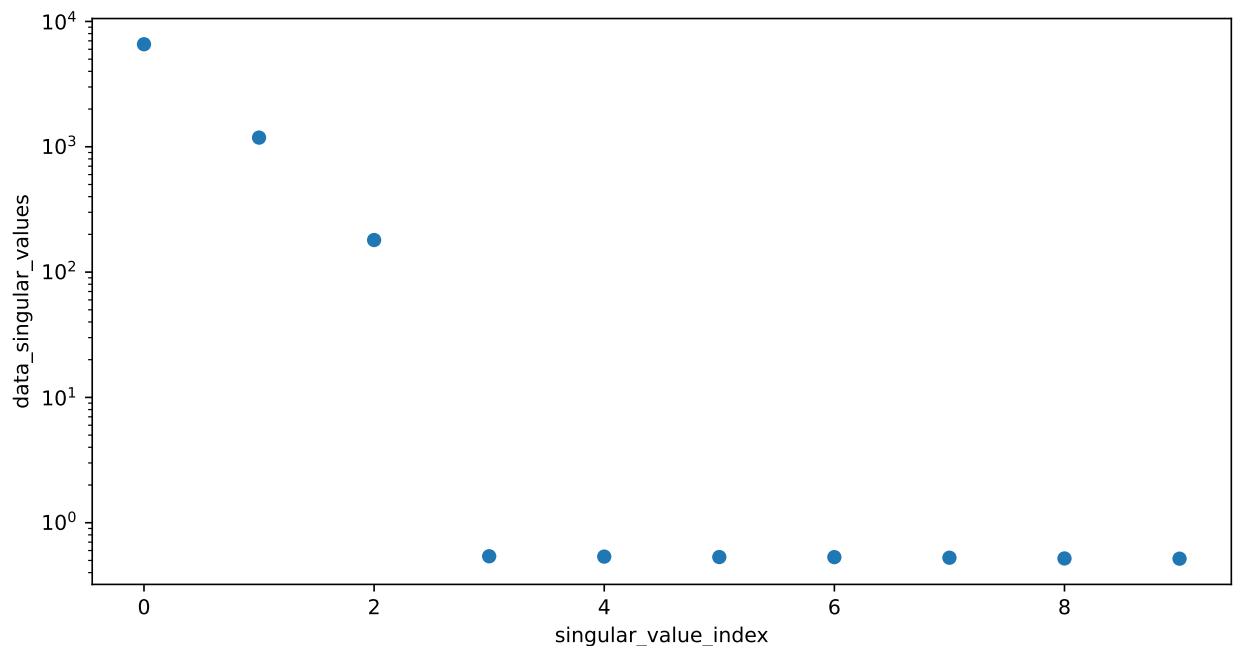
```
[7]: dataset_with_svd = quickstart_project.load_data("my_data", add_svd=True)
dataset_with_svd

[7]: <xarray.Dataset>
Dimensions:
      (time: 2100, spectral: 72,
      left_singular_value_index: 72,
      singular_value_index: 72,
      right_singular_value_index: 72)

Coordinates:
  * time      (time) float64 -1.0 -0.99 -0.98 ... 19.98 19.99
  * spectral  (spectral) float64 600.0 601.4 ... 698.0 699.4
Dimensions without coordinates: left_singular_value_index,
                                singular_value_index, right_singular_value_index
Data variables:
  data      (time, spectral) float64 -0.0003386 ... 2.312
  data_left_singular_vectors (time, left_singular_value_index) float64 -2...
  data_singular_values      (singular_value_index) float64 6.577e+03 ...
  data_right_singular_vectors (spectral, right_singular_value_index) float64 ...
Attributes:
  source_path: /home/docs/checkouts/readthedocs.org/user_builds/pyglotaran...
  loader:      <function load_dataset at 0x7fb231d616c0>
```

First, take a look at the first 10 singular values:

```
[8]: plot_data = dataset_with_svd.data_singular_values.sel(singular_value_index=range(10))
plot_data.plot(yscale="log", marker="o", linewidth=0, aspect=2, size=5);
```



This tells us that our data have at least three components which we need to model.

## 4.3 Working with models

To analyze our data, we need to create a model.

Create a file called `my_model.yaml` in your projects models directory and fill it with the following content.

```
[9]: quickstart_project.show_model_definition("my_model")
```

```
[9]: default_megacomplex: decay

initial_concentration:
  input:
    compartments: [s1, s2, s3]
    parameters: [input.1, input.0, input.0]

k_matrix:
  k1:
    matrix:
      (s2, s1): kinetic.1
      (s3, s2): kinetic.2
      (s3, s3): kinetic.3

megacomplex:
  m1:
    k_matrix: [k1]

irf:
  irf1:
    type: gaussian
    center: irf.center
    width: irf.width

dataset:
  my_data:
    initial_concentration: input
    megacomplex: [m1]
    irf: irf1
```

You can check your model for problems with the `validate` method.

```
[10]: quickstart_project.validate("my_model")
```

```
[10]: Your model is valid.
```

## 4.4 Working with parameters

Now define some starting parameters. Create a file called `parameters.yaml` in your projects `parameters` directory with the following content.

```
[11]: quickstart_project.show_parameters_definition("my_parameters")
```

```
[11]: input:
  - ["1", 1, { "vary": False }]
  - ["0", 0, { "vary": False }]

kinetic: [0.51, 0.31, 0.11]

irf:
  - ["center", 0.31]
  - ["width", 0.11]
```

Note the `{ "vary": False }` which tells pyglotaran that those parameters should not be changed.

You can use `validate` method also to check for missing parameters.

```
[12]: quickstart_project.validate("my_model", "my_parameters")
```

```
[12]: Your model is valid.
```

Since not all problems in the model can be detected automatically it is wise to visually inspect the model. For this purpose, you can just load the model and inspect its markdown rendered version.

```
[13]: quickstart_project.load_model("my_model")
```

```
[13]: 4.4.1 Model
```

### Dataset Groups

- **default**
  - *Label*: default
  - *Residual Function*: variable\_projection

### K Matrix

- **k1**
  - *Label*: k1
  - *Matrix*: `{('s2', 's1'): 'kinetic.1', ('s3', 's2'): 'kinetic.2', ('s3', 's3'): 'kinetic.3'}`

### Megacomplex

- **m1**
  - *Label*: m1

(continues on next page)



(continued from previous page)

- *Dimension*: time
- *Type*: decay
- *K Matrix*: ['k1']

Initial Concentration

- **input**
  - *Label*: input
  - *Compartments*: ['s1', 's2', 's3']
  - *Parameters*: ['input.1', 'input.0', 'input.0']
  - *Exclude From Normalize*: []

Irf

- **irf1**
  - *Label*: irf1
  - *Normalize*: True
  - *Backsweep*: False
  - *Type*: gaussian
  - *Center*: irf.center
  - *Width*: irf.width

Dataset

- **my\_data**
  - *Label*: my\_data
  - *Group*: default
  - *Force Index Dependent*: False
  - *Megacomplex*: ['m1']
  - *Initial Concentration*: input
  - *Irf*: irf1

The same way you should inspect your parameters.

```
[14]: quickstart_project.load_parameters("my_parameters")
```

```
[14]: • input:
```

La- bel	Value	Standard Er- ror	t- value	Mini- mum	Maxi- mum	Vary	Non- Negative	Expres- sion
1	1.000e+00	nan	nan	-inf	inf	False	False	None

(continued on next page)

4.4. Working with parameters13

(continued from previous page)

• **irf:**

<i>La- bel</i>	<i>Value</i>	<i>Standard Er- ror</i>	<i>t- value</i>	<i>Mini- mum</i>	<i>Maxi- mum</i>	<i>Vary</i>	<i>Non- Negative</i>	<i>Expres- sion</i>
cen- ter	3.100e- 01	nan	nan	-inf	inf	True	False	None
width	1.100e- 01	nan	nan	-inf	inf	True	False	None

• **kinetic:**

<i>La- bel</i>	<i>Value</i>	<i>Standard Er- ror</i>	<i>t- value</i>	<i>Mini- mum</i>	<i>Maxi- mum</i>	<i>Vary</i>	<i>Non- Negative</i>	<i>Expres- sion</i>
1	5.100e- 01	nan	nan	-inf	inf	True	False	None
2	3.100e- 01	nan	nan	-inf	inf	True	False	None
3	1.100e- 01	nan	nan	-inf	inf	True	False	None

## 4.5 Optimizing data

Now we have everything together to optimize our parameters.

```
[15]: result = quickstart_project.optimize("my_model", "my_parameters")
result
```

Iteration	Total nfev	Cost	Cost reduction	Step norm	Optimality
0	1	1.1175e+04			1.73e+06
1	2	1.5004e+01	1.12e+04	1.96e-02	1.26e+04
2	3	7.5686e+00	7.44e+00	5.85e-03	1.01e+03
3	4	7.5647e+00	3.86e-03	1.79e-05	6.91e-02
4	5	7.5647e+00	1.80e-11	4.63e-09	9.21e-06

Both `ftol` and `xtol` termination conditions are satisfied.

Function evaluations 5, initial cost 1.1175e+04, final cost 7.5647e+00, first-order ↪ optimality 9.21e-06.

```
[15]:
```

Optimization Result	
Number of residual evaluation	5
Number of residuals	151200
Number of free parameters	5
Number of conditionally linear parameters	216
Degrees of freedom	150979
Chi Square	1.51e+01
Reduced Chi Square	1.00e-04
Root Mean Square Error (RMSE)	1.00e-02

(continues on next page)

(continued from previous page)

### 4.5.1 Model

#### Dataset Groups

- **default**
  - *Label*: default
  - *Residual Function*: variable\_projection

#### K Matrix

- **k1**
  - *Label*: k1
  - *Matrix*: `{('s2', 's1'): 'kinetic.1(5.00e-01±6.79e-05, t-value: 7359, initial: 5.10e-01)', ('s3', 's2'):`  
`'kinetic.2(3.00e-01±3.94e-05, t-value: 7615, initial: 3.10e-01)', ('s3', 's3'):`  
`'kinetic.3(1.00e-01±4.23e-06, t-value: 23647, initial: 1.10e-01)'}`

#### Megacomplex

- **m1**
  - *Label*: m1
  - *Dimension*: time
  - *Type*: decay
  - *K Matrix*: `['k1']`

#### Initial Concentration

- **input**
  - *Label*: input
  - *Compartments*: `['s1', 's2', 's3']`
  - *Parameters*: `['input.1(1.00e+00, fixed)', 'input.0(0.00e+00, fixed)', 'input.0(0.00e+00, fixed)']`
  - *Exclude From Normalize*: `[]`

#### Irf

- **irf1**
  - *Label*: irf1
  - *Normalize*: True
  - *Backsweep*: False

(continues on next page)

(continued from previous page)

- *Type*: gaussian
- *Center*: irf.center( $3.00\text{e-}01 \pm 5.04\text{e-}06$ , t-value: 59499, initial:  $3.10\text{e-}01$ )
- *Width*: irf.width( $1.00\text{e-}01 \pm 6.73\text{e-}06$ , t-value: 14863, initial:  $1.10\text{e-}01$ )

## Dataset

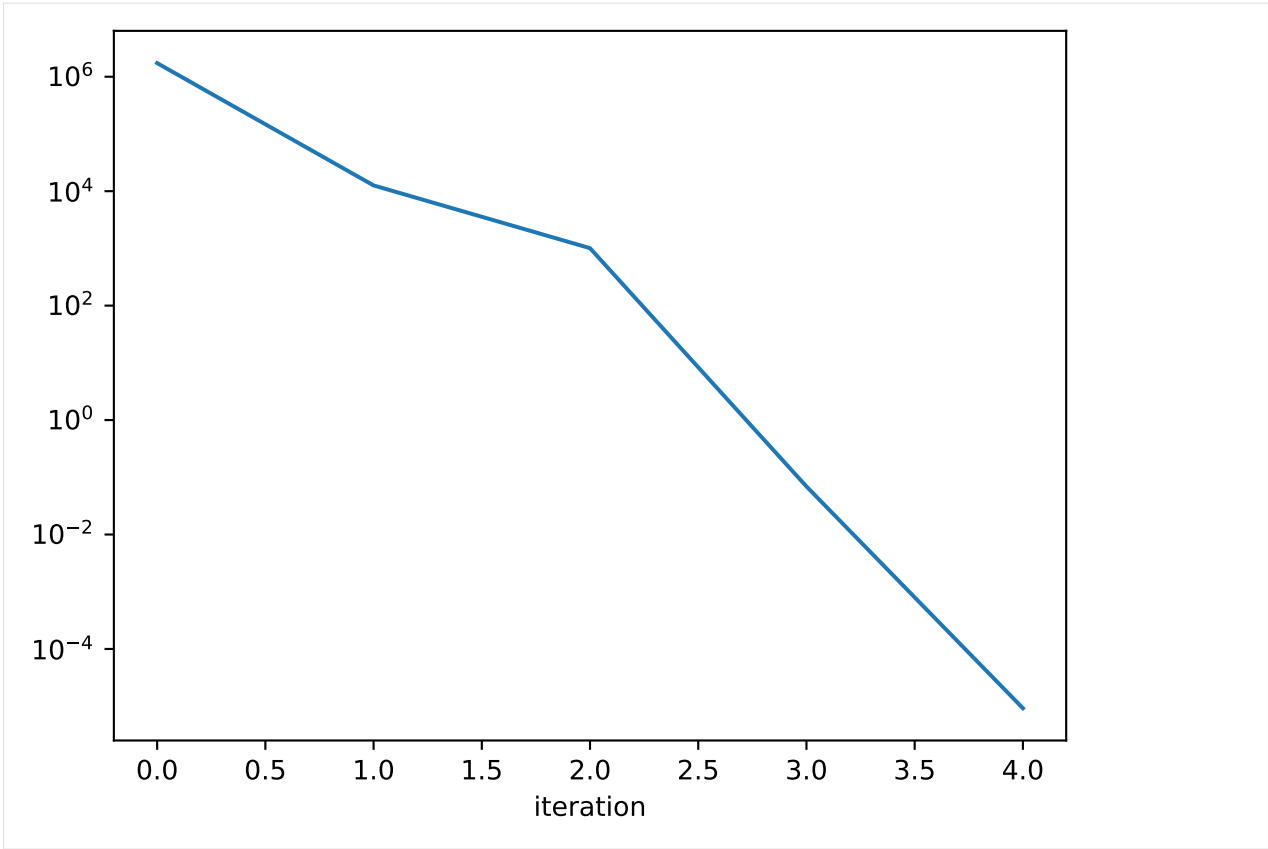
- **my\_data**
  - *Label*: my\_data
  - *Group*: default
  - *Force Index Dependent*: False
  - *Megacomplex*: ['m1']
  - *Initial Concentration*: input
  - *Irf*: irf1

Each time you run an optimization the result will be saved in the projects results folder.

```
[16]: %ls "quickstart_project/results"
      my_model_run_0000/
```

To visualize how quickly the optimization converged we can plot the optimality of the `optimization_history`.

```
[17]: result.optimization_history.data["optimality"].plot(logy=True)
[17]: <Axes: xlabel='iteration'>
```



[18]: result.optimized\_parameters

[18]: • input:

La- bel	Value	Standard Er- ror	t- value	Mini- mum	Maxi- mum	Vary	Non- Negative	Expres- sion
1	1.000e+00	nan	nan	-inf	inf	False	False	None
0	0.000e+00	nan	nan	-inf	inf	False	False	None

• irf:

La- bel	Value	Standard Er- ror	t- value	Mini- mum	Maxi- mum	Vary	Non- Negative	Expres- sion
center	3.000e-01	5.042e-06	59499	-inf	inf	True	False	None
width	1.000e-01	6.728e-06	14863	-inf	inf	True	False	None

• kinetic:

La- bel	Value	Standard Er- ror	t- value	Mini- mum	Maxi- mum	Vary	Non- Negative	Expres- sion
1	4.999e-01	6.794e-05	7359	-inf	inf	True	False	None
2	3.000e-01	3.940e-05	7615	-inf	inf	True	False	None
3	1.000e-01	4.229e-06	23647	-inf	inf	True	False	None

(continued from previous page)

You can inspect the data of your `result` by accessing `data` attribute. In our example it only contains our single `my_data` dataset, but it can contain as many datasets as your analysis needs.

```
[19]: result.data
```

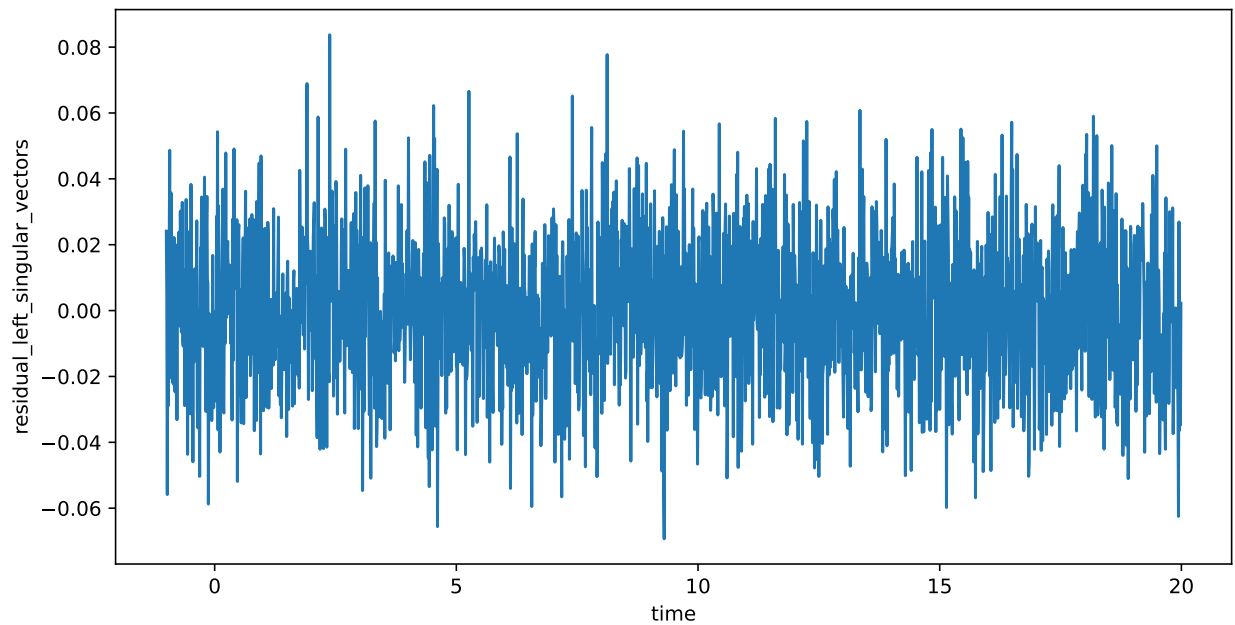
```
[19]: {'my_data': <xarray.Dataset>}
```

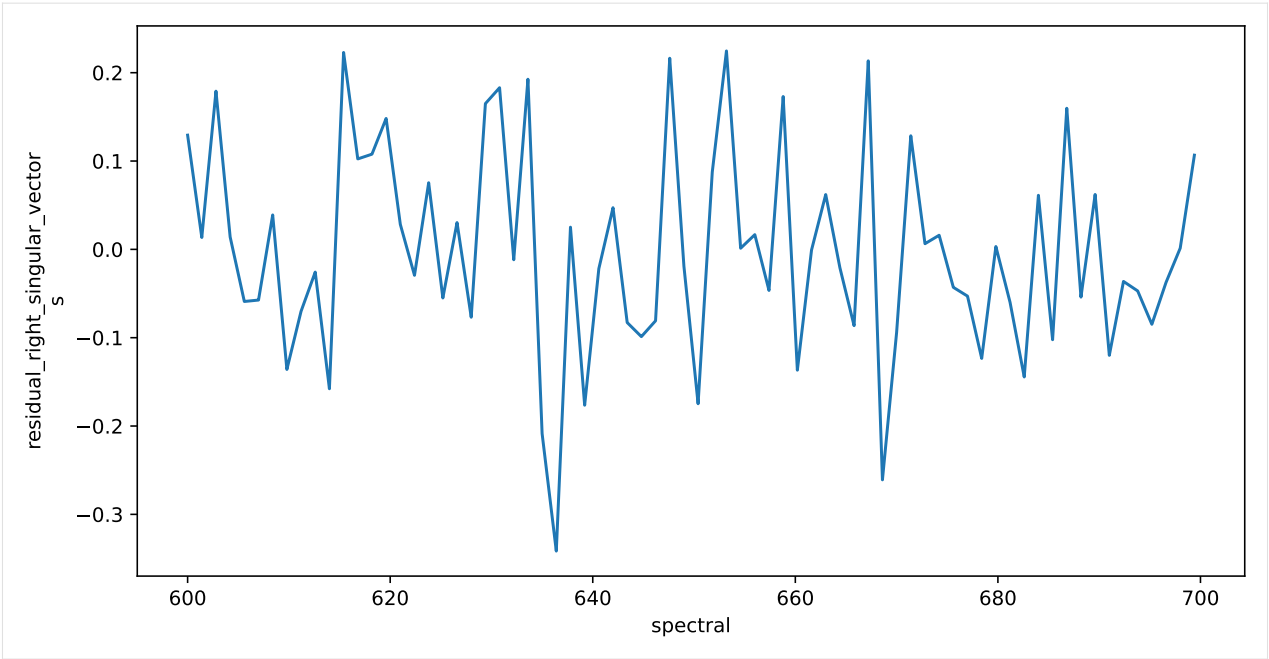
## 4.6 Visualize the Result

The resulting data can be visualized the same way as the dataset. To judge the quality of the fit, you should look at first left and right singular vectors of the residual.

```
[20]: result_dataset = result.data["my_data"]
```

```
residual_left = result_dataset.residual_left_singular_vectors.sel(left_singular_value_
↪ index=0)
residual_right = result_dataset.residual_right_singular_vectors.sel(right_singular_value_
↪ index=0)
residual_left.plot.line(x="time", aspect=2, size=5)
residual_right.plot.line(x="spectral", aspect=2, size=5);
```









## CHANGELOG

### 5.1 0.7.2 (2023-12-06)

#### 5.1.1 Features

- Official numpy 1.26 support (#1374)

#### 5.1.2 Maintenance

- Remove unused dependency: 'rich' (#1345)

### 5.2 0.7.1 (2023-07-28)

#### 5.2.1 Features

- Python 3.11 support (#1161)

#### 5.2.2 Bug fixes

- Fix coherent artifact clp label duplication (#1292)

### 5.3 0.7.0 (Unreleased)

#### 5.3.1 BREAKING CHANGE

- Dropped support for Python 3.8 and 3.9 and only support 3.10 (#1135)

### 5.3.2 Features

- Add optimization history to result and iteration column to parameter history (#1134)
- Complete refactor of model and parameter packages using attrs (#1135)
- Move index dependent calculation to megacomplexes for speed-up (#1175)
- Add PreProcessingPipeline (#1256, #1263)

### 5.3.3 Minor Improvements:

- Wrap model section in result markdown in details tag for notebooks (#1098)
- Allow more natural column names in pandas parameters file reading (#1174)
- Integrate plugin system into Project (#1229)
- Make yaml the default plugin when passing a folder to save\_result and load\_result (#1230)
- Allow usage of subfolders in project API for parameters, models and data (#1232)
- Allow import of xarray objects in project API import\_data (#1235)
- Add number\_of\_clps to result and correct degrees\_of\_freedom calculation (#1249)
- Improve Project API data handling (#1257)
- Deprecate Result.number\_of\_parameters in favor of Result.number\_of\_free\_parameters (#1262)
- Improve reporting of standard error in case of non\_negative constraint in the parameter (#1320)

### 5.3.4 Bug fixes

- Fix result data overwritten when using multiple dataset\_groups (#1147)
- Fix for normalization issue described in #1157 (multi-gaussian irfs and multiple time ranges (streak))
- Fix for crash described in #1183 when doing an optimization using more than 30 datasets (#1184)
- Fix pretty\_format\_numerical for negative values (#1192)
- Fix yaml result saving with relative paths (#1199)
- Fix model markdown render for items without label (#1213)
- Fix wrong file loading due to partial filename matching in Project (#1212)
- Fix Project.import\_data path resolving for different script and cwd (#1214)
- Refine project API (#1240)
- Fix search in docs (#1268)

### 5.3.5 Documentation

- Update quickstart guide to use Project API (#1241)

### 5.3.6 Deprecations (due in 0.8.0)

- `<model_file>.clp_area_penalties` -> `<model_file>.clp_penalties`
- `glotaran.ParameterGroup` -> `glotaran.Parameters`
- Command Line Interface (removed without replacement) (#1228)
- `Project.generate_model` (removed without replacement)
- `Project.generate_parameters` (removed without replacement)
- `glotaran.project.Result.number_of_data_points` -> `glotaran.project.Result.number_of_residuals`
- `glotaran.project.Result.number_of_parameters` -> `glotaran.project.Result.number_of_free_parameters`

### 5.3.7 Deprecated functionality removed in this release

- `glotaran.project.Scheme(..., non_negative_least_squares=...)`
- `glotaran.project.Scheme(..., group=...)`
- `glotaran.project.Scheme(..., group_tolerance=...)`
- `<model_file>.non-negative-least-squares: true`
- `<model_file>.non-negative-least-squares: false`
- `glotaran.parameter.ParameterGroup.to_csv(file_name=parameters.csv)`

### 5.3.8 Maintenance

- Fix wrong comparison in `pr_benchmark` workflow (#1097)
- Set sourcery-ai target python version to 3.8 (#1095)
- Fix manifest check (#1099)
- Refactor: optimization (#1060)
- Use `GITHUB_OUTPUT` instead of `set-output` in github actions (#1166, #1177)
- Add pinned version of `odfpy` to `requirements_dev.txt` (#1164)
- Use validation action and validation as a git submodule (#1165)
- Upgrade syntax to py310 using `pyupgrade` (#1162)
- Remove unused `'type: ignore'` (#1168)
- Raise minimum dependency version to releases that support py310 (#1170)
- Make `mypy` and `doc string` linters opt out instead of opt in (#1173)

## 5.4 0.6.0 (2022-06-06)

### 5.4.1 Features

- Python 3.10 support (#977)
- Add simple decay megacomplexes (#860)
- Feature: Generators (#866)
- Project Class (#869)
- Add clp guidance megacomplex (#1029)

### 5.4.2 Minor Improvements:

- Add proper repr for DatasetMapping (#957)
- Add SavingOptions to save\_result API (#966)
- Add parameter IO support for more formats supported by pandas (#896)
- Apply IRF shift in coherent artifact megacomplex (#992)
- Added IRF shift to result dataset (#994)
- Improve Result, Parameter and ParameterGroup markdown (#1012)
- Add suffix to rate and lifetime and guard for missing datasets (#1022)
- Move simulation to own module (#1041)
- Move optimization to new module glotaran.optimization (#1047)
- Fix missing installation of clp-guide megacomplex as plugin (#1066)
- Add 'extras' and 'full' extras\_require installation options (#1089)

### 5.4.3 Bug fixes

- Fix Crash in optimization\_group\_calculator\_linked when using guidance spectra (#950)
- ParameterGroup.get degrades full\_label of nested Parameters with nesting over 2 (#1043)
- Show validation problem if parameters are missing values (default: NaN) (#1076)

### 5.4.4 Documentation

- Add new logo (#1083, #1087)

### 5.4.5 Deprecations (due in 0.8.0)

- `glotaran.io.save_result(result, result_path, format_name='legacy')` -> `glotaran.io.save_result(result, Path(result_path) / 'result.yml')`
- `glotaran.analysis.simulation` -> `glotaran.simulation.simulation`
- `glotaran.analysis.optimize` -> `glotaran.optimization.optimize`

### 5.4.6 Deprecated functionality removed in this release

- `glotaran.ParameterGroup` -> `glotaran.parameter.ParameterGroup`
- `glotaran.read_model_from_yaml` -> `glotaran.io.load_model(..., format_name="yaml_str")`
- `glotaran.read_model_from_yaml_file` -> `glotaran.io.load_model(..., format_name="yaml")`
- `glotaran.read_parameters_from_csv_file` -> `glotaran.io.load_parameters(..., format_name="csv")`
- `glotaran.read_parameters_from_yaml` -> `glotaran.io.load_parameters(..., format_name="yaml_str")`
- `glotaran.read_parameters_from_yaml_file` -> `glotaran.io.load_parameters(..., format_name="yaml")`
- `glotaran.io.read_data_file` -> `glotaran.io.load_dataset`
- `result.get_dataset("<dataset_name>")` -> `result.data["<dataset_name>"]`
- `glotaran.analysis.result` -> `glotaran.project.result`
- `glotaran.analysis.scheme` -> `glotaran.project.scheme`

### 5.4.7 Maintenance

- Improve packaging tooling (#923)
- Exclude test files from duplication checks on sonarcloud (#959)
- Only run check-manifest on the CI (#967)
- Exclude dependabot push CI runs (#978)
- Exclude sourcery AI push CI runs (#1014)
- Auto remove notebook written data when building docs (#1019)
- Change integration tests to use self managed examples action (#1034)
- Exclude pre-commit bot branch from CI runs on push (#1085)

## 5.5 0.5.1 (2021-12-31)

### 5.5.1 Bug fixes

- Bugfix Use normalized initial\_concentrations in result creation for decay megacomplex (#927)
- Fix save\_result crashes on Windows if input data are on a different drive than result (#931)

### 5.5.2 Maintenance

- Forward port Improve result comparison workflow and v0.4 changelog (#938)
- Forward port of #936 test\_result\_consistency

## 5.6 0.5.0 (2021-12-01)

### 5.6.1 Features

- Feature: Megacomplex Models (#736)
- Feature: Full Models (#747)
- Damped Oscillation Megacomplex (a.k.a. DOAS) (#764)
- Add Dataset Groups (#851)
- Performance improvements (in some cases up to 5x) (#740)

### 5.6.2 Minor Improvements:

- Add dimensions to megacomplex and dataset\_descriptor (#702)
- Improve ordering in k\_matrix involved\_compartments function (#788)
- Improvements to application of clp\_penalties (equal area) (#801)
- Refactor model.from\_dict to parse megacomplex\_type from dict and add simple\_generator for testing (#807)
- Refactor model spec (#836)
- Refactor Result Saving (#841)
- Use ruaml.yaml parser for roundtrip support (#893)
- Refactor Result and Scheme loading/initializing from files (#903)
- Several refactoring in glotaran.Parameter (#910)
- Improved Reporting of Parameters (#910, #914, #918)
- Scheme now excepts paths to model, parameter and data file without initializing them first (#912)

### 5.6.3 Bug fixes

- Fix/cli0.5 (#765)
- Fix compartment ordering randomization due to use of set (#799)
- Fix check\_deprecations not showing deprecation warnings (#775)
- Fix and re-enable IRF Dispersion Test (#786)
- Fix coherent artifact crash for index dependent models #808
- False positive model validation fail when combining multiple default megacomplexes (#797)
- Fix ParameterGroup repr when created with 'from\_list' (#827)
- Fix for DOAS with reversed oscillations (negative rates) (#839)
- Fix parameter expression parsing (#843)
- Use a context manager when opening a nc dataset (#848)
- Disallow xarray versions breaking plotting in integration tests (#900)
- Fix 'dataset\_groups' not shown in model markdown (#906)

### 5.6.4 Documentation

- Moved API documentation from User to Developer Docs (#776)
- Add docs for the CLI (#784)
- Fix deprecation in model used in quickstart notebook (#834)

### 5.6.5 Deprecations (due in 0.7.0)

- `glotaran.model.Model.model_dimension` -> `glotaran.project.Scheme.model_dimension`
- `glotaran.model.Model.global_dimension` -> `glotaran.project.Scheme.global_dimension`
- `<model_file>.type.kinetic-spectrum` -> `<model_file>.default_megacomplex.decay`
- `<model_file>.type.spectral-model` -> `<model_file>.default_megacomplex.spectral`
- `<model_file>.spectral_relations` -> `<model_file>.clp_relations`
- `<model_file>.spectral_relations.compartment` -> `<model_file>.clp_relations.source`
- `<model_file>.spectral_constraints` -> `<model_file>.clp_constraints`
- `<model_file>.spectral_constraints.compartment` -> `<model_file>.clp_constraints.target`
- `<model_file>.equal_area_penalties` -> `<model_file>.clp_area_penalties`
- `<model_file>.irf.center_dispersion` -> `<model_file>.irf.center_dispersion_coefficients`
- `<model_file>.irf.width_dispersion` -> `<model_file>.irf.width_dispersion_coefficients`
- `glotaran.project.Scheme(..., non_negative_least_squares=...)` -> `<model_file>dataset_groups.default.residual_function`
- `glotaran.project.Scheme(..., group=...)` -> `<model_file>dataset_groups.default.link_clp`
- `glotaran.project.Scheme(..., group_tolerance=...)` -> `glotaran.project.Scheme(..., clp_link_tolerance=...)`

- `<scheme_file>.maximum-number-function-evaluations` -> `<scheme_file>.maximum_number_function_evaluations`
- `<model_file>.non-negative-least-squares: true` -> `<model_file>dataset_groups.default.residual_function: non_negative_least_squares`
- `<model_file>.non-negative-least-squares: false` -> `<model_file>dataset_groups.default.residual_function: variable_projection`
- `glotaran.parameter.ParameterGroup.to_csv(file_name=parameters.csv)` -> `glotaran.io.save_parameters(parameters, file_name=parameters.csv)`

## 5.6.6 Maintenance

- Fix Performance Regressions (between version) (#740)
- Add integration test result validation (#754)
- Add more QA tools for parts of glotaran (#739)
- Fix interrogate usage (#781)
- Speedup PR benchmark (#785)
- Use pinned versions of dependencies to run integration CI tests (#892)
- Move megacomplex integration tests from root level to megacomplexes (#894)
- Fix artifact download in `pr_benchmark_reaction` workflow (#907)

## 5.7 0.4.2 (2021-12-31)

### 5.7.1 Bug fixes

- Backport of bugfix #927 discovered in PR #860 related to `initial_concentration` normalization when saving results (#935).

### 5.7.2 Maintenance

- Updated ‘gold standard’ result comparison reference (`old` -> `new`)
- Refine `test_result_consistency` (#936).

## 5.8 0.4.1 (2021-09-07)

### 5.8.1 Features

- Integration test result validation (#760)



### 5.8.2 Bug fixes

- Fix unintended saving of sub-optimal parameters (0ece818, backport from #747)
- Improve ordering in `k_matrix` `involved_compartments` function (#791)

## 5.9 0.4.0 (2021-06-25)

### 5.9.1 Features

- Add basic spectral model (#672)
- Add Channel/Wavelength dependent shift parameter to `irf`. (#673)
- Refactored `Problem` class into `GroupedProblem` and `UngroupedProblem` (#681)
- Plugin system was rewritten (#600, #665)
- Deprecation framework (#631)
- Better notebook integration (#689)

### 5.9.2 Bug fixes

- Fix excessive memory usage in `_create_svd` (#576)
- Fix several issues with `KineticImage` model (#612)
- Fix exception in `sdt` reader index calculation (#647)
- Avoid crash in result markdown printing when optimization fails (#630)
- `ParameterNotFoundException` doesn't prepend `.'` if path is empty (#688)
- Ensure `Parameter.label` is str or None (#678)
- Properly scale `StdError` of estimated parameters with RMSE (#704)
- More robust `covariance_matrix` calculation (#706)
- `ParameterGroup.markdown()` independent parametergroups of order (#592)

### 5.9.3 Plugins

- `ProjectIo` `'folder'/'legacy'` plugin to save results (#620)
- `Model` `'spectral-model'` (#672)

### 5.9.4 Documentation

- User documentation is written in notebooks (#568)
- Documentation on how to write a DataIo plugin (#600)

### 5.9.5 Deprecations (due in 0.6.0)

- `glotaran.ParameterGroup` -> `glotaran.parameterParameterGroup`
- `glotaran.read_model_from_yaml` -> `glotaran.io.load_model(..., format_name="yaml_str")`
- `glotaran.read_model_from_yaml_file` -> `glotaran.io.load_model(..., format_name="yaml")`
- `glotaran.read_parameters_from_csv_file` -> `glotaran.io.load_parameters(..., format_name="csv")`
- `glotaran.read_parameters_from_yaml` -> `glotaran.io.load_parameters(..., format_name="yaml_str")`
- `glotaran.read_parameters_from_yaml_file` -> `glotaran.io.load_parameters(..., format_name="yaml")`
- `glotaran.io.read_data_file` -> `glotaran.io.load_dataset`
- `result.save` -> `glotaran.io.save_result(result, ..., format_name="legacy")`
- `result.get_dataset("<dataset_name>")` -> `result.data["<dataset_name>"]`
- `glotaran.analysis.result` -> `glotaran.project.result`
- `glotaran.analysis.scheme` -> `glotaran.project.scheme`
- `model.simulate` -> `glotaran.analysis.simulation.simulate(model, ...)`

## 5.10 0.3.3 (2021-03-18)

- Force recalculation of SVD attributes in `scheme._prepare_data` (#597)
- Remove unneeded check in `spectral_penalties._get_area` Fixes (#598)
- Added python 3.9 support (#450)

## 5.11 0.3.2 (2021-02-28)

- Re-release of version 0.3.1 due to packaging issue

## 5.12 0.3.1 (2021-02-28)

- Added compatibility for numpy 1.20 and raised minimum required numpy version to 1.20 (#555)
- Fixed excessive memory consumption in result creation due to full SVD computation (#574)
- Added feature parameter history (#557)
- Moved setup logic to `setup.cfg` (#560)

## 5.13 0.3.0 (2021-02-11)

- Significant code refactor with small API changes to parameter relation specification (see docs)
- Replaced `lmfit` with `scipy.optimize`

## 5.14 0.2.0 (2020-12-02)

- Large refactor with significant improvements but also small API changes (see docs)
- Removed `doas` plugin

## 5.15 0.1.0 (2020-07-14)

- Package was renamed to `pyglotaran` on PyPi

## 5.16 0.0.8 (2018-08-07)

- Changed `nan_policy` to `omit`

## 5.17 0.0.7 (2018-08-07)

- Added support for multiple shapes per compartment.

## 5.18 0.0.6 (2018-08-07)

- First release on PyPI, support for Windows installs added.
- Pre-Alpha Development



## AUTHORS

### 6.1 Development Lead

- Joern Weissenborn <[joern.weissenborn@gmail.com](mailto:joern.weissenborn@gmail.com)>
- Joris Snellenburg <[j.snellenburg@gmail.com](mailto:j.snellenburg@gmail.com)>

### 6.2 Contributors

- Sebastian Weigand <[s.weigand.phy@gmail.com](mailto:s.weigand.phy@gmail.com)>

### 6.3 Special Thanks

- Stefan Schuetz
- Sergey P. Laptanok

### 6.4 Supervision

- **dr. Ivo H.M. van Stokkum** <[i.h.m.van.stokkum@vu.nl](mailto:i.h.m.van.stokkum@vu.nl)> (University profile)

### 6.5 Original publications

1. Joris J. Snellenburg, Sergey Laptanok, Ralf Seger, Katharine M. Mullen, Ivo H. M. van Stokkum. “Glotaran: A Java-Based Graphical User Interface for the R Package TIMP”. *Journal of Statistical Software* (2012), Volume 49, Number 3, Pages: 1–22. URL <https://dx.doi.org/10.18637/jss.v049.i03>
2. Katharine M. Mullen, Ivo H. M. van Stokkum. “TIMP: An R Package for Modeling Multi-way Spectroscopic Measurements”. *Journal of Statistical Software* (2007), Volume 18, Number 3, Pages 1-46, ISSN 1548-7660. URL <https://dx.doi.org/10.18637/jss.v018.i03>
3. Ivo H. M. van Stokkum, Delmar S. Larsen, Rienk van Grondelle, “Global and target analysis of time-resolved spectra”. *Biochimica et Biophysica Acta (BBA) - Bioenergetics* (2004), Volume 1657, Issues 2–3, Pages 82-104, ISSN 0005-2728. URL <https://doi.org/10.1016/j.bbabi.2004.04.011>



**OVERVIEW**





---

**CHAPTER**  
**EIGHT**

---

**DATA IO**



**PLOTTING**



**MODELLING**



**PARAMETER**









## PLUGINS

To be as flexible as possible pyglotaran uses a plugin system to handle new `Models`, `DataIo` and `ProjectIo`. Those plugins can be defined by pyglotaran itself, the user or a 3rd party plugin package.

### 13.1 Builtin plugins

#### 13.1.1 Models

- `KineticSpectrumModel`
- `KineticImageModel`

#### 13.1.2 Data Io

Plugins reading and writing data to and from `xarray.Dataset` or `xarray.DataArray`.

- `AsciiDataIo`
- `NetCDFDataIo`
- `SdtDataIo`

#### 13.1.3 Project Io

Plugins reading and writing, `Model`, `class:Schema`, `class:ParameterGroup` or `Result`.

- `YmlProjectIo`
- `CsvProjectIo`
- `FolderProjectIo`

## 13.2 Reproducibility and plugins

With a plugin ecosystem there always is the possibility that multiple plugins try register under the same format/name. This is why plugins are registered at least twice. Once under the name the developer intended and secondly under their full name (full import path). This allows to ensure that a specific plugin is used by manually specifying the plugin, so if someone wants to run your analysis the results will be reproducible even if they have conflicting plugins installed. You can gain all information about the installed plugins by calling the corresponding `*_plugin_table` function with both options (`plugin_names` and `full_names`) set to true. To pin a used plugin use the corresponding `set_*_plugin` function with the intended name (`format_name/model_name`) and the full name (`full_plugin_name`) of the plugin to use.

If you wanted to ensure that the pyglotaran builtin plugin is used for sdt files you could add the following lines to the beginning of your analysis code.

```
from glotaran.io import set_data_plugin
set_data_plugin("sdt", "glotaran.builtin.io.sdt.sdt_file_reader.SdtDataIo_sdt")
```

### 13.2.1 Models

The functions for model plugins are located in `glotaran.model` and called `model_plugin_table` and `set_model_plugin`.

### 13.2.2 Data Io

The functions for data io plugins are located in `glotaran.io` and called `data_io_plugin_table` and `set_data_plugin`.

### 13.2.3 Project Io

The functions for project io plugins are located in `glotaran.io` and called `project_io_plugin_table` and `set_project_plugin`.

## 13.3 3rd party plugins

Plugins not part of pyglotaran itself.

- Not yet, why not be the first? Tell us about your plugin and we will feature it here.

## COMMAND-LINE INTERFACE

### 14.1 glotaran

The glotaran CLI main function.

```
glotaran [OPTIONS] COMMAND [ARGS] ...
```

#### Options

**--version**

Show the version and exit.

#### 14.1.1 optimize

Optimizes a model. e.g.: glotaran optimize –

```
glotaran optimize [OPTIONS] [SCHEME_FILE]
```

#### Options

**-dfmt, --dataformat <dataformat>**

The input format of the data. Will be inferred from extension if not set.

##### Options

ascii | nc | sdt

**-d, --data <data>**

Path to a dataset in the form ‘–data DATASET\_LABEL PATH\_TO\_DATA’

**-o, --out <out>**

Path to an output directory.

**-ofmt, --outformat <outformat>**

The format of the output.

##### Default

folder

##### Options

folder | legacy | yaml

**-n, --nfev** <nfev>  
Maximum number of function evaluations.

**--nnls**  
Use non-negative least squares.

**-y, --yes**  
Don't ask for confirmation.

**-p, --parameters\_file** <parameters\_file>  
(optional) Path to parameter file.

**-m, --model\_file** <model\_file>  
Path to model file.

## Arguments

**SCHEME\_FILE**  
Optional argument

### 14.1.2 pluginlist

Prints a list of installed plugins.

```
glotaran pluginlist [OPTIONS]
```

### 14.1.3 print

Parses scheme, a model or a parameter file and prints the result as a Markdown formatted string.

```
glotaran print [OPTIONS] [SCHEME_FILE]
```

## Options

**-p, --parameters\_file** <parameters\_file>  
(optional) Path to parameter file.

**-m, --model\_file** <model\_file>  
Path to model file.

## Arguments

**SCHEME\_FILE**  
Optional argument

### 14.1.4 validate

Validates a model file and optionally a parameter file.

```
glotaran validate [OPTIONS] [SCHEME_FILE]
```

#### Options

**-p, --parameters\_file** <parameters\_file>

(optional) Path to parameter file.

**-m, --model\_file** <model\_file>

Path to model file.

#### Arguments

**SCHEME\_FILE**

Optional argument





## 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:

### 15.1 Types of Contributions

#### 15.1.1 Report Bugs

Report bugs at <https://github.com/glottaran/pyglottaran/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.

#### 15.1.2 Fix Bugs

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

#### 15.1.3 Implement Features

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

#### 15.1.4 Write Documentation

pyglottaran could always use more documentation, whether as part of the official pyglottaran docs, in docstrings, or even on the web in blog posts, articles, and such. If you are writing docstrings please use the [NumPyDoc](#) style to write them.

### 15.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/glotaran/pyglotaran/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 :)

## 15.2 Get Started!

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

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

```
$ git clone https://github.com/<your_name_here>/pyglotaran.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
(pyglotaran)$ cd pyglotaran
(pyglotaran)$ python -m pip install -r requirements_dev.txt
(pyglotaran)$ pip install -e . --process-dependency-links
```

4. Install the pre-commit hooks, to automatically format and check your code:

```
$ pre-commit install
```

5. Create a branch for local development:

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

Now you can make your changes locally.

6. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ pre-commit run -a
$ py.test
```

Or to run all at once:

```
$ tox
```

7. 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
```

8. Submit a pull request through the GitHub website.

9. Add the change referring the pull request ((#<PR\_nr>)) to `changelog.md`. If you are in doubt in which section your pull request belongs, just ask a maintainer what they think where it belongs.

---

**Note:** By default pull requests will use the template located at `.github/PULL_REQUEST_TEMPLATE.md`. But we also provide custom tailored templates located inside of `.github/PULL_REQUEST_TEMPLATE`. Sadly the GitHub Web Interface doesn't provide an easy way to select them as it does for issue templates (see [this comment for more details](#)).

To use them you need to add the following query parameters to the url when creating the pull request and hit enter:

- Feature PR: `?expand=1&template=feature_PR.md`
  - Bug Fix PR: `?expand=1&template=bug_fix_PR`
  - Documentation PR: `?expand=1&template=docs_PR.md`
- 

## 15.3 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*.
3. The pull request should work for Python 3.10 and 3.11 Check your Github Actions [https://github.com/<your\\_name\\_here>/pyglotaran/actions](https://github.com/<your_name_here>/pyglotaran/actions) and make sure that the tests pass for all supported Python versions.

## 15.4 Docstrings

We use [numpy style docstrings](#), which can also be autogenerated from function/method signatures by extensions for your editor.

Some extensions for popular editors are:

- [autodocstring](#) (VS-Code)
- [vim-python-docstring](#) (Vim)

---

**Note:** If your pull request improves the docstring coverage (check `pre-commit run -a interrogate`), please raise the value of the interrogate setting `fail-under` in [pyproject.toml](#). That way the next person will improve the docstring coverage as well and everyone can enjoy a better documentation.

---

**Warning:** As soon as all our docstrings are in proper shape we will enforce that it stays that way. If you want to check if your docstrings are fine you can use [pydocstyle](#) and [darglint](#).

## 15.5 Tips

To run a subset of tests:

```
$ py.test tests.test_pyglotaran
```

## 15.6 Deprecations

Only maintainers are allowed to decide about deprecations, thus you should first open an issue and check back with them if they are ok with deprecating something.

To make deprecations as robust as possible and give users all needed information to adjust their code, we provide helper functions inside the module `glotaran.deprecation`.

The functions you most likely want to use are

- `deprecate()` for functions, methods and classes
- `warn_deprecated()` for call arguments
- `deprecate_module_attribute()` for module attributes
- `deprecate_submodule()` for modules
- `deprecate_dict_entry()` for dict entries
- `raise_deprecation_error()` if the original behavior cannot be maintained

Those functions not only make it easier to deprecate something, but they also check that that deprecations will be removed when they are due and that at least the imports in the warning work. Thus all deprecations need to be tested.

Tests for deprecations should be placed in `glotaran/deprecation/modules/test` which also provides the test helper functions `deprecation_warning_on_call_test_helper` and `changed_import_test_warn`. Since the tests for deprecation are mainly for maintainability and not to test the functionality (those tests should be in the appropriate place) `deprecation_warning_on_call_test_helper` will by default just test that a `GlottaranApiDeprecationWarning` was raised and ignore all `raise Exception`s. An exception to this rule is when adding back removed functionality (which shouldn't happen in the first place but might), which should be implemented in a file under `glotaran/deprecation/modules` and filenames should be like the relative import path from `glotaran` root, but with `_` instead of `..`.

E.g. `glotaran.analysis.scheme` would map to `analysis_scheme.py`

The only exceptions to this rule are the root `__init__.py` which is named `glotaran_root.py` and testing changed imports which should be placed in `test_changed_imports.py`.

### 15.6.1 Deprecating a Function, method or class

Deprecating a function, method or class is as easy as adding the `deprecate` decorator to it. Other decorators (e.g. `@staticmethod` or `@classmethod`) should be placed both `deprecate` in order to work.

Listing 1: `glotaran/some_module.py`

```
from glotaran.deprecation import deprecate

@deprecate(
    deprecated_qual_name_usage="glotaran.some_module.function_to_deprecate(filename)",
```

(continues on next page)

(continued from previous page)

```

    new_qual_name_usage='glotaran.some_module.new_function(filename, format_name="legacy
↪")',
    to_be_removed_in_version="0.6.0",
)
def function_to_deprecate(*args, **kwargs):
    ...

```

## 15.6.2 Deprecating a call argument

When deprecating a call argument you should use `warn_deprecated` and set the argument to deprecate to a default value (e.g. "deprecated") to check against. Note that for this use case we need to set `check_qual_names=(False, False)` which will deactivate the import testing. This might not always be possible, e.g. if the argument is positional only, so it might make more sense to deprecate the whole callable, just discuss what to do with our trusted maintainers.

Listing 2: `glotaran/some_module.py`

```

from glotaran.deprecation import deprecate

def function_to_deprecate(args1, new_arg="new_default_behavior", deprecated_arg=
↪"deprecated", **kwargs):
    if deprecated_arg != "deprecated":
        warn_deprecated(
            deprecated_qual_name_usage="deprecated_arg",
            new_qual_name_usage='new_arg="legacy"',
            to_be_removed_in_version="0.6.0",
            check_qual_names=(False, False)
        )
        new_arg = "legacy"
    ...

```

## 15.6.3 Deprecating a module attribute

Sometimes it might be necessary to remove an attribute (function, class, or constant) from a module to prevent circular imports or just to streamline the API. In those cases you would use `deprecate_module_attribute` inside a module `__getattr__` function definition. This will import the attribute from the new location and return it when an import or use is requested.

Listing 3: `glotaran/old_package/__init__.py`

```

def __getattr__(attribute_name: str):
    from glotaran.deprecation import deprecate_module_attribute

    if attribute_name == "deprecated_attribute":
        return deprecate_module_attribute(
            deprecated_qual_name="glotaran.old_package.deprecated_attribute",
            new_qual_name="glotaran.new_package.new_attribute_name",
            to_be_removed_in_version="0.6.0",
        )

    raise AttributeError(f"module {__name__} has no attribute {attribute_name}")

```

### 15.6.4 Deprecating a submodule

For a better logical structure, it might be needed to move modules to a different location in the project. In those cases, you would use `deprecate_submodule`, which imports the module from the new location, add it to `sys.modules` and as an attribute to the parent package.

Listing 4: `glotaran/old_package/__init__.py`

```
from glotaran.deprecation import deprecate_submodule

module_name = deprecate_submodule(
    deprecated_module_name="glotaran.old_package.module_name",
    new_module_name="glotaran.new_package.new_module_name",
    to_be_removed_in_version="0.6.0",
)
```

### 15.6.5 Deprecating dict entries

The possible dict deprecation actions are:

- Swapping of keys `{"foo": 1} -> {"bar": 1}` (done via `swap_keys=("foo", "bar")`)
- Replacing of matching values `{"foo": 1} -> {"foo": 2}` (done via `replace_rules={"foo": 1}, {"foo": 2}`)
- Replacing of matching values and swapping of keys `{"foo": 1} -> {"bar": 2}` (done via `replace_rules={"foo": 1}, {"bar": 2}`)

For full examples have a look at the examples from the docstring (`deprecate_dict_entry()`).

### 15.6.6 Deprecation Errors

In some cases deprecations cannot have a replacement with the original behavior maintained. This will be mostly the case when at this point in time and in the object hierarchy there isn't enough information available to calculate the appropriate values. Rather than using a 'dummy' value not to break the API, which could cause undefined behavior down the line, those cases should throw an error which informs the users about the new usage. In general this should only be used if it is unavoidable due to massive refactoring of the internal structure and tried to avoid by any means in a reasonable context.

If you have one of those rare cases you can use `raise_deprecation_error()`.

## 15.7 Testing Result consistency

To test the consistency of results locally you need to clone the `pyglotaran-examples` and run them:

```
$ git clone https://github.com/glotaran/pyglotaran-examples
$ cd pyglotaran-examples
$ python scripts/run_examples.py run-all --headless
```

---

**Note:** Make sure you got the the latest version (`git pull`) and are on the correct branch for both `pyglotaran` and `pyglotaran-examples`.

---

The results from the examples will be saved in your home folder under `pyglotaran_examples_results`. Those results then will be compared to the ‘gold standard’ defined by the maintainers.

To test the result consistency run:

```
$ pytest validation/pyglotaran-examples/test_result_consistency.py
```

If needed this will clone the ‘gold standard’ results to the folder `comparison-results`, update them and test your current results against them.

## 15.8 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in `changelog.md`), the version number only needs to be changed in `glotaran/__init__.py`.

Then make a [new release on GitHub](#) and give the tag a proper name, e.g. `v0.3.0` since it might be included in a citation.

GitHub Actions will then deploy to PyPI if the tests pass.





## API DOCUMENTATION

The API Documentation for pyglotaran is automatically created from its docstrings.

<i>glotaran</i>	Glotaran package root.
-----------------	------------------------

### 16.1 glotaran

Glotaran package root.

#### Modules

<i>glotaran.analysis</i>	This package contains functions for model simulation and fitting.
<i>glotaran.builtin</i> <i>glotaran.cli</i>	This package contains builtin plugins.
<i>glotaran.deprecation</i>	Deprecation helpers and place to put deprecated implementations till removing.
<i>glotaran.io</i>	Functions for data IO.
<i>glotaran.model</i>	The glotaran model package.
<i>glotaran.optimization</i>	This package contains functions for optimization.
<i>glotaran.parameter</i>	The glotaran parameter package.
<i>glotaran.plugin_system</i>	Plugin system package containing all plugin related implementations.
<i>glotaran.project</i>	The glotaran project package.
<i>glotaran.simulation</i>	Package containing code for simulation of dataset models.
<i>glotaran.testing</i>	Testing framework package for glotaran itself and plugins.
<i>glotaran.typing</i>	Glotaran specific typing module.
<i>glotaran.utils</i>	Glotaran utility function/class package.

### 16.1.1 analysis

This package contains functions for model simulation and fitting.

### 16.1.2 builtin

This package contains builtin plugins.

#### Modules

<i>glotaran.builtin.io</i>	Package containing the builtin IO plugins.
<i>glotaran.builtin.megacomplexes</i>	

#### io

Package containing the builtin IO plugins.

#### Modules

<i>glotaran.builtin.io.ascii</i>	
<i>glotaran.builtin.io.folder</i>	Plugin to dump pyglotaran object as files in a folder.
<i>glotaran.builtin.io.netCDF</i>	Package containing the NetCDF4 Data IO plugin.
<i>glotaran.builtin.io.pandas</i>	Pandas io package.
<i>glotaran.builtin.io.sdt</i>	Package containing the SDT Data IO plugin.
<i>glotaran.builtin.io.yml</i>	Package containing the YAML Data and Project IO plugins.

#### ascii

#### Modules

<i>glotaran.builtin.io.ascii.wavelength_time_explicit_file</i>	
--	--

## wavelength\_time\_explicit\_file

### Functions

#### Summary

*get\_data\_file\_format*

*get\_interval\_number*

#### **get\_data\_file\_format**

`glotaran.builtin.io.ascii.wavelength_time_explicit_file.get_data_file_format(line)`

#### **get\_interval\_number**

`glotaran.builtin.io.ascii.wavelength_time_explicit_file.get_interval_number(line)`

### Classes

#### Summary

<i>AsciiDataIo</i>	Initialize a Data IO plugin with the name of the format.
<i>DataFileType</i>	An enumeration.
<i>ExplicitFile</i>	Abstract class representing either a time- or wavelength-explicit file.
<i>TimeExplicitFile</i>	Represents a time explicit file
<i>WavelengthExplicitFile</i>	Represents a wavelength explicit file

#### **AsciiDataIo**

**class** `glotaran.builtin.io.ascii.wavelength_time_explicit_file.AsciiDataIo(format_name: str)`

Bases: *DataIoInterface*

Initialize a Data IO plugin with the name of the format.

#### **Parameters**

**format\_name** (*str*) – Name of the supported format an instance uses.

## Methods Summary

<code>load_dataset</code>	Reads an ascii file in wavelength- or time-explicit format.
<code>save_dataset</code>	Save data from <code>xarray.Dataset</code> to a file.

### load\_dataset

`AsciiDataIo.load_dataset(file_name: str, *, prepare: bool = True) → Dataset | DataArray`

Reads an ascii file in wavelength- or time-explicit format.

See [1] for documentation of this format.

#### Parameters

**fname** (*str*) – Name of the ascii file.

#### Returns

**dataset**

#### Return type

`xr.Dataset`

### Notes

### save\_dataset

`AsciiDataIo.save_dataset(dataset: DataArray | Dataset, file_name: str, *, comment: str = "", file_format: DataFileType = DataFileType.time_explicit, number_format: str = '%.10e')`

Save data from `xarray.Dataset` to a file.

#### NOT IMPLEMENTED

#### Parameters

- **dataset** (`xr.Dataset`) – Dataset to be saved to file.
- **file\_name** (*str*) – File to write the data to.

## Methods Documentation

`load_dataset(file_name: str, *, prepare: bool = True) → Dataset | DataArray`

Reads an ascii file in wavelength- or time-explicit format.

See [1] for documentation of this format.

#### Parameters

**fname** (*str*) – Name of the ascii file.

#### Returns

**dataset**

#### Return type

`xr.Dataset`

## Notes

**save\_dataset**(*dataset*: *DataArray* | *Dataset*, *file\_name*: *str*, \*, *comment*: *str* = "", *file\_format*: *DataFileType* = *DataFileType.time\_explicit*, *number\_format*: *str* = '%.10e')

Save data from `xarray.Dataset` to a file.

## NOT IMPLEMENTED

### Parameters

- **dataset** (*xr.Dataset*) – Dataset to be saved to file.
- **file\_name** (*str*) – File to write the data to.

## DataFileType

**class** `glotaran.builtin.io.ascii.wavelength_time_explicit_file.DataFileType`(*value*)

Bases: `Enum`

An enumeration.

## Attributes Summary

`time_explicit`

`wavelength_explicit`

### time\_explicit

`DataFileType.time_explicit = 'Time explicit'`

### wavelength\_explicit

`DataFileType.wavelength_explicit = 'Wavelength explicit'`

`time_explicit = 'Time explicit'`

`wavelength_explicit = 'Wavelength explicit'`

## ExplicitFile

**class** `glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`(*filepath*:  
*str* |  
*None*  
= *None*,  
*dataset*:  
*DataAr-*  
*ray* |  
*None*  
= *None*)

Bases: `object`

Abstract class representing either a time- or wavelength-explicit file.

### Methods Summary

<code>dataset</code>
<code>get_data_row</code>
<code>get_explicit_axis</code>
<code>get_format_name</code>
<code>get_observations</code>
<code>get_secondary_axis</code>
<code>read</code>
<code>set_explicit_axis</code>
<code>write</code>

### `dataset`

`ExplicitFile.dataset(prepare: bool = True) → Dataset | DataArray`

### `get_data_row`

`ExplicitFile.get_data_row(index)`

### `get_explicit_axis`

`ExplicitFile.get_explicit_axis()`

### `get_format_name`

`ExplicitFile.get_format_name()`

**get\_observations**

`ExplicitFile.get_observations(index)`

**get\_secondary\_axis**

`ExplicitFile.get_secondary_axis()`

**read**

`ExplicitFile.read(prepare: bool = True)`

**set\_explicit\_axis**

`ExplicitFile.set_explicit_axis(axis)`

**write**

`ExplicitFile.write(overwrite=False, comment="", file_format=DataFileType.time_explicit,  
number_format='%.10e')`

**Methods Documentation**

`dataset(prepare: bool = True) → Dataset | DataArray`

`get_data_row(index)`

`get_explicit_axis()`

`get_format_name()`

`get_observations(index)`

`get_secondary_axis()`

`read(prepare: bool = True)`

`set_explicit_axis(axis)`

`write(overwrite=False, comment="", file_format=DataFileType.time_explicit,  
number_format='%.10e')`

## TimeExplicitFile

```
class glotaran.builtin.io.ascii.wavelength_time_explicit_file.TimeExplicitFile(filepath:
    str
    |
    None
    =
    None,
    dataset:
    DataAr-
    ray
    |
    None
    =
    None)
```

Bases: *ExplicitFile*

Represents a time explicit file

### Methods Summary

<i>add_data_row</i>
<i>dataset</i>
<i>get_data_row</i>
<i>get_explicit_axis</i>
<i>get_format_name</i>
<i>get_observations</i>
<i>get_secondary_axis</i>
<i>read</i>
<i>set_explicit_axis</i>
<i>write</i>



**add\_data\_row**

`TimeExplicitFile.add_data_row(row)`

**dataset**

`TimeExplicitFile.dataset(prepare: bool = True) → Dataset | DataArray`

**get\_data\_row**

`TimeExplicitFile.get_data_row(index)`

**get\_explicit\_axis**

`TimeExplicitFile.get_explicit_axis()`

**get\_format\_name**

`TimeExplicitFile.get_format_name()`

**get\_observations**

`TimeExplicitFile.get_observations(index)`

**get\_secondary\_axis**

`TimeExplicitFile.get_secondary_axis()`

**read**

`TimeExplicitFile.read(prepare: bool = True)`

**set\_explicit\_axis**

`TimeExplicitFile.set_explicit_axis(axes)`

**write**

```
TimeExplicitFile.write(overwrite=False, comment="",  
                       file_format=DataFileType.time_explicit, number_format='%.10e')
```

**Methods Documentation**

```
add_data_row(row)
```

```
dataset(prepare: bool = True) → Dataset | DataArray
```

```
get_data_row(index)
```

```
get_explicit_axis()
```

```
get_format_name()
```

```
get_observations(index)
```

```
get_secondary_axis()
```

```
read(prepare: bool = True)
```

```
set_explicit_axis(axes)
```

```
write(overwrite=False, comment="", file_format=DataFileType.time_explicit,  
      number_format='%.10e')
```

**WavelengthExplicitFile**

```
class glotaran.builtin.io.ascii.wavelength_time_explicit_file.WavelengthExplicitFile(filepath:  
                                             str  
                                             |  
                                             None  
                                             =  
                                             None,  
                                             dataset:  
                                             DataAr-  
                                             ray  
                                             |  
                                             None  
                                             =  
                                             None)
```

Bases: [ExplicitFile](#)

Represents a wavelength explicit file

## Methods Summary

<code>add_data_row</code>
<code>dataset</code>
<code>get_data_row</code>
<code>get_explicit_axis</code>
<code>get_format_name</code>
<code>get_observations</code>
<code>get_secondary_axis</code>
<code>read</code>
<code>set_explicit_axis</code>
<code>times</code>
<code>wavelengths</code>
<code>write</code>

### **add\_data\_row**

`WavelengthExplicitFile.add_data_row(row)`

### **dataset**

`WavelengthExplicitFile.dataset(prepare: bool = True) → Dataset | DataArray`

### **get\_data\_row**

`WavelengthExplicitFile.get_data_row(index)`

**get\_explicit\_axis**

WavelengthExplicitFile.**get\_explicit\_axis**()

**get\_format\_name**

WavelengthExplicitFile.**get\_format\_name**()

**get\_observations**

WavelengthExplicitFile.**get\_observations**(*index*)

**get\_secondary\_axis**

WavelengthExplicitFile.**get\_secondary\_axis**()

**read**

WavelengthExplicitFile.**read**(*prepare: bool = True*)

**set\_explicit\_axis**

WavelengthExplicitFile.**set\_explicit\_axis**(*axis*)

**times**

WavelengthExplicitFile.**times**()

**wavelengths**

WavelengthExplicitFile.**wavelengths**()

**write**

WavelengthExplicitFile.**write**(*overwrite=False, comment="",  
file\_format=DataFileType.time\_explicit,  
number\_format='%.10e'*)

## Methods Documentation

**add\_data\_row**(*row*)

**dataset**(*prepare*: *bool* = *True*) → Dataset | DataArray

**get\_data\_row**(*index*)

**get\_explicit\_axis**()

**get\_format\_name**()

**get\_observations**(*index*)

**get\_secondary\_axis**()

**read**(*prepare*: *bool* = *True*)

**set\_explicit\_axis**(*axis*)

**times**()

**wavelengths**()

**write**(*overwrite*=*False*, *comment*="", *file\_format*=*DataFileType.time\_explicit*,  
*number\_format*='%.10e')

## folder

Plugin to dump pyglotaran object as files in a folder.

## Modules

<i>glotaran.builtin.io.folder.folder_plugin</i>	Implementation of the folder Io plugin.
---	---

## folder\_plugin

Implementation of the folder Io plugin.

The current implementation is an exact copy of how `Result.save(path)` worked in glotaran 0.3.x and meant as an compatibility function.

## Classes

### Summary

<i>FolderProjectIo</i>	Project Io plugin to save result data to a folder.
<i>LegacyProjectIo</i>	Project Io plugin to save result data in a backward compatible manner.

## FolderProjectIo

**class** glotaran.builtin.io.folder.folder\_plugin.**FolderProjectIo**(*format\_name: str*)

Bases: *ProjectIoInterface*

Project Io plugin to save result data to a folder.

There won't be a serialization of the Result object, but simply a markdown summary output and the important data saved to files.

Initialize a Project IO plugin with the name of the format.

### Parameters

**format\_name** (*str*) – Name of the supported format an instance uses.

## Methods Summary

<i>load_model</i>	Create a Model instance from the specs defined in a file.
<i>load_parameters</i>	Create a Parameters instance from the specs defined in a file.
<i>load_result</i>	Create a Result instance from the specs defined in a file.
<i>load_scheme</i>	Create a Scheme instance from the specs defined in a file.
<i>save_model</i>	Save a Model instance to a spec file.
<i>save_parameters</i>	Save a Parameters instance to a spec file.
<i>save_result</i>	Save the result to a given folder.
<i>save_scheme</i>	Save a Scheme instance to a spec file.

## load\_model

**FolderProjectIo.load\_model**(*file\_name: str*) → *Model*

Create a Model instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

**file\_name** (*str*) – File containing the model specs.

#### Returns

Model instance created from the file.

#### Return type

*Model*

### load\_parameters

FolderProjectIo.**load\_parameters**(*file\_name: str*) → *Parameters*

Create a Parameters instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

Parameters instance created from the file.

**Return type**

*Parameters*

### load\_result

FolderProjectIo.**load\_result**(*result\_path: str*) → *Result*

Create a Result instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**result\_path** (*str*) – Path containing the result data.

**Returns**

Result instance created from the file.

**Return type**

*Result*

### load\_scheme

FolderProjectIo.**load\_scheme**(*file\_name: str*) → *Scheme*

Create a Scheme instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

- *Scheme* – Scheme instance created from the file.
- **.. # noqa** (DAR202)
- **.. # noqa** (DAR401)

### save\_model

FolderProjectIo.**save\_model**(*model: Model, file\_name: str*)

Save a Model instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

### save\_parameters

FolderProjectIo.**save\_parameters**(parameters: Parameters, file\_name: str)

Save a Parameters instance to a spec file.

#### NOT IMPLEMENTED

##### Parameters

- **parameters** (Parameters) – Parameters instance to save to specs file.
- **file\_name** (str) – File to write the parameter specs to.

### save\_result

FolderProjectIo.**save\_result**(result: Result, result\_path: str, \*, saving\_options: SavingOptions = SavingOptions(data\_filter=None, data\_format='nc', parameter\_format='csv', report=True), used\_inside\_of\_plugin: bool = False) → list[str]

Save the result to a given folder.

Returns a list with paths of all saved items. The following files are saved if not configured otherwise: \* **result.md**: The result with the model formatted as markdown text. \* **initial\_parameters.csv**: Initially used parameters. \* **optimized\_parameters.csv**: The optimized parameter as csv file. \* **parameter\_history.csv**: Parameter changes over the optimization \* **{dataset\_label}.nc**: The result data for each dataset as NetCDF file.

---

**Note:** As a side effect it populates the file path properties of **result** which can be used in other plugins (e.g. the yml **save\_result**).

---

##### Parameters

- **result** (Result) – Result instance to be saved.
- **result\_path** (str) – The path to the folder in which to save the result.
- **saving\_options** (SavingOptions) – Options for saving the the result.
- **used\_inside\_of\_plugin** (bool) – Denote that this plugin is used from inside another plugin, if false a user warning will be thrown. , by default False

##### Returns

List of file paths which were created.

##### Return type

list[str]

##### Raises

**ValueError** – If **result\_path** is a file.

### save\_scheme

FolderProjectIo.**save\_scheme**(scheme: Scheme, file\_name: str)

Save a Scheme instance to a spec file.

#### NOT IMPLEMENTED

##### Parameters

- **scheme** (Scheme) – Scheme instance to save to specs file.
- **file\_name** (str) – File to write the scheme specs to.



## Methods Documentation

**load\_model**(*file\_name*: *str*) → *Model*

Create a Model instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the model specs.

**Returns**

Model instance created from the file.

**Return type**

*Model*

**load\_parameters**(*file\_name*: *str*) → *Parameters*

Create a Parameters instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

Parameters instance created from the file.

**Return type**

*Parameters*

**load\_result**(*result\_path*: *str*) → *Result*

Create a Result instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**result\_path** (*str*) – Path containing the result data.

**Returns**

Result instance created from the file.

**Return type**

*Result*

**load\_scheme**(*file\_name*: *str*) → *Scheme*

Create a Scheme instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**save\_model**(*model*: *Model*, *file\_name*: *str*)

Save a Model instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

**save\_parameters**(*parameters*: *Parameters*, *file\_name*: *str*)

Save a Parameters instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **parameters** ([Parameters](#)) – Parameters instance to save to specs file.
- **file\_name** (*str*) – File to write the parameter specs to.

**save\_result**(*result*: [Result](#), *result\_path*: *str*, \*, *saving\_options*: [SavingOptions](#) = [SavingOptions](#)(*data\_filter*=None, *data\_format*='nc', *parameter\_format*='csv', *report*=True), *used\_inside\_of\_plugin*: *bool* = False) → *list*[*str*]

Save the result to a given folder.

Returns a list with paths of all saved items. The following files are saved if not configured otherwise: \* *result.md*: The result with the model formatted as markdown text. \* *initial\_parameters.csv*: Initially used parameters. \* *optimized\_parameters.csv*: The optimized parameter as csv file. \* *parameter\_history.csv*: Parameter changes over the optimization \* *{dataset\_label}.nc*: The result data for each dataset as NetCDF file.

---

**Note:** As a side effect it populates the file path properties of *result* which can be used in other plugins (e.g. the *yml* *save\_result*).

---

**Parameters**

- **result** ([Result](#)) – Result instance to be saved.
- **result\_path** (*str*) – The path to the folder in which to save the result.
- **saving\_options** ([SavingOptions](#)) – Options for saving the the result.
- **used\_inside\_of\_plugin** (*bool*) – Denote that this plugin is used from inside another plugin, if false a user warning will be thrown. , by default False

**Returns**

List of file paths which were created.

**Return type**

*list*[*str*]

**Raises**

**ValueError** – If *result\_path* is a file.

**save\_scheme**(*scheme*: [Scheme](#), *file\_name*: *str*)

Save a Scheme instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## LegacyProjectIo

**class** `glotaran.builtin.io.folder.folder_plugin.LegacyProjectIo`(*format\_name*: *str*)

Bases: [ProjectIoInterface](#)

Project Io plugin to save result data in a backward compatible manner.

Initialize a Project IO plugin with the name of the format.

**Parameters**

**format\_name** (*str*) – Name of the supported format an instance uses.

## Methods Summary

<code>load_model</code>	Create a Model instance from the specs defined in a file.
<code>load_parameters</code>	Create a Parameters instance from the specs defined in a file.
<code>load_result</code>	Create a Result instance from the specs defined in a file.
<code>load_scheme</code>	Create a Scheme instance from the specs defined in a file.
<code>save_model</code>	Save a Model instance to a spec file.
<code>save_parameters</code>	Save a Parameters instance to a spec file.
<code>save_result</code>	Save the result to a given folder.
<code>save_scheme</code>	Save a Scheme instance to a spec file.

### load\_model

`LegacyProjectIo.load_model(file_name: str) → Model`

Create a Model instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the model specs.

##### Returns

Model instance created from the file.

##### Return type

*Model*

### load\_parameters

`LegacyProjectIo.load_parameters(file_name: str) → Parameters`

Create a Parameters instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the parameter specs.

##### Returns

Parameters instance created from the file.

##### Return type

*Parameters*

### load\_result

LegacyProjectIo.**load\_result**(*result\_path: str*) → *Result*

Create a Result instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**result\_path** (*str*) – Path containing the result data.

##### Returns

Result instance created from the file.

##### Return type

*Result*

### load\_scheme

LegacyProjectIo.**load\_scheme**(*file\_name: str*) → *Scheme*

Create a Scheme instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the parameter specs.

##### Returns

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

### save\_model

LegacyProjectIo.**save\_model**(*model: Model, file\_name: str*)

Save a Model instance to a spec file.

#### NOT IMPLEMENTED

##### Parameters

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

### save\_parameters

LegacyProjectIo.**save\_parameters**(*parameters: Parameters, file\_name: str*)

Save a Parameters instance to a spec file.

#### NOT IMPLEMENTED

##### Parameters

- **parameters** (*Parameters*) – Parameters instance to save to specs file.
- **file\_name** (*str*) – File to write the parameter specs to.

## save\_result

LegacyProjectIo.**save\_result**(*result*: [Result](#), *result\_path*: *str*, \*, *saving\_options*: [SavingOptions](#) = [SavingOptions](#)(*data\_filter*=None, *data\_format*='nc', *parameter\_format*='csv', *report*=True)) → [list](#)[*str*]

Save the result to a given folder.

**Warning:** Deprecated use `glotaran.io.save_result(result, Path(result_path) / 'result.yml')` instead.

Returns a list with paths of all saved items. The following files are saved if not configured otherwise: \* `result.md`: The result with the model formatted as mark-down text. \* `result.yml`: Yaml spec file of the result \* `model.yml`: Model spec file. \* `scheme.yml`: Scheme spec file. \* `initial_parameters.csv`: Initially used parameters. \* `optimized_parameters.csv`: The optimized parameter as csv file. \* `parameter_history.csv`: Parameter changes over the optimization \* `{dataset_label}.nc`: The result data for each dataset as NetCDF file.

### Parameters

- **result** ([Result](#)) – Result instance to be saved.
- **result\_path** (*str*) – The path to the folder in which to save the result.
- **saving\_options** ([SavingOptions](#)) – Options for saving the the result.

### Returns

List of file paths which were created.

### Return type

[list](#)[*str*]

## save\_scheme

LegacyProjectIo.**save\_scheme**(*scheme*: [Scheme](#), *file\_name*: *str*)

Save a Scheme instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## Methods Documentation

**load\_model**(*file\_name*: *str*) → [Model](#)

Create a Model instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

**file\_name** (*str*) – File containing the model specs.

#### Returns

Model instance created from the file.

#### Return type

[Model](#)

**load\_parameters**(*file\_name*: *str*) → *Parameters*

Create a Parameters instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

Parameters instance created from the file.

**Return type**

*Parameters*

**load\_result**(*result\_path*: *str*) → *Result*

Create a Result instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**result\_path** (*str*) – Path containing the result data.

**Returns**

Result instance created from the file.

**Return type**

*Result*

**load\_scheme**(*file\_name*: *str*) → *Scheme*

Create a Scheme instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**save\_model**(*model*: *Model*, *file\_name*: *str*)

Save a Model instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

**save\_parameters**(*parameters*: *Parameters*, *file\_name*: *str*)

Save a Parameters instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **parameters** (*Parameters*) – Parameters instance to save to specs file.
- **file\_name** (*str*) – File to write the parameter specs to.

**save\_result**(*result*: *Result*, *result\_path*: *str*, \*, *saving\_options*: *SavingOptions* = *SavingOptions*(*data\_filter*=None, *data\_format*='nc', *parameter\_format*='csv', *report*=True)) → list[*str*]

Save the result to a given folder.

<p><b>Warning:</b>                      Deprecated      use      <code>glotaran.io.save_result(result, Path(result_path) / 'result.yml')</code> instead.</p>
--

Returns a list with paths of all saved items. The following files are saved if not configured otherwise: \* `result.md`: The result with the model formatted as mark-down text. \* `result.yml`: Yaml spec file of the result \* `model.yml`: Model spec file. \* `scheme.yml`: Scheme spec file. \* `initial_parameters.csv`: Initially used parameters. \* `optimized_parameters.csv`: The optimized parameter as csv file. \* `parameter_history.csv`: Parameter changes over the optimization \* `{dataset_label}.nc`: The result data for each dataset as NetCDF file.

**Parameters**

- **result** ([Result](#)) – Result instance to be saved.
- **result\_path** (*str*) – The path to the folder in which to save the result.
- **saving\_options** ([SavingOptions](#)) – Options for saving the the result.

**Returns**

List of file paths which were created.

**Return type**

`list[str]`

**save\_scheme**(*scheme*: [Scheme](#), *file\_name*: *str*)

Save a Scheme instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

**netCDF**

Package containing the NetCDF4 Data IO plugin.

**Modules**

<i>glotaran.builtin.io.netCDF.netCDF</i>
--

Module containing the NetCDF4 Data IO plugin.
---

**netCDF**

Module containing the NetCDF4 Data IO plugin.

**Classes****Summary**

<i>NetCDFDataIo</i>
---------------------

Plugin for NetCDF4 data io.
-----------------------------

## NetCDFDataIo

**class** `glotaran.builtin.io.netCDF.netCDF.NetCDFDataIo(format_name: str)`

Bases: [DataIoInterface](#)

Plugin for NetCDF4 data io.

Initialize a Data IO plugin with the name of the format.

### Parameters

**format\_name** (str) – Name of the supported format an instance uses.

## Methods Summary

<a href="#">load_dataset</a>	Load a *.nc file into a <a href="#">xarray.Dataset</a> or <a href="#">xarray.DataArray</a> .
<a href="#">save_dataset</a>	Write a <a href="#">xarray.Dataset</a> to the *.nc at path <code>file_name</code> .

## load\_dataset

`NetCDFDataIo.load_dataset(file_name: str) → Dataset | DataArray`

Load a \*.nc file into a [xarray.Dataset](#) or [xarray.DataArray](#).

### Parameters

**file\_name** (str) – Path to the \*.nc file that should be loaded.

### Return type

[xr.Dataset](#) | [xr.DataArray](#)

## save\_dataset

`NetCDFDataIo.save_dataset(dataset: Dataset, file_name: str, *, data_filters: list[str] | None = None)`

Write a [xarray.Dataset](#) to the \*.nc at path `file_name`.

### Parameters

- **dataset** ([xr.Dataset](#)) – [xarray.Dataset](#) that should be written to file.
- **file\_name** (str) – Path of the file to write dataset to.
- **data\_filters** ([list\[str\]](#) | None) – List of data variable names that should be written to file. Defaults to None.

## Methods Documentation

`load_dataset(file_name: str) → Dataset | DataArray`

Load a \*.nc file into a [xarray.Dataset](#) or [xarray.DataArray](#).

### Parameters

**file\_name** (str) – Path to the \*.nc file that should be loaded.

### Return type

[xr.Dataset](#) | [xr.DataArray](#)



**save\_dataset**(*dataset*: *Dataset*, *file\_name*: *str*, \*, *data\_filters*: *list[str] | None = None*)

Write a `xarray.Dataset` to the \*.nc at path *file\_name*.

**Parameters**

- **dataset** (*xr.Dataset*) – `xarray.Dataset` that should be written to file.
- **file\_name** (*str*) – Path of the file to write dataset to.
- **data\_filters** (*list[str] | None*) – List of data variable names that should be written to file. Defaults to None.

## pandas

Pandas io package.

## Modules

<code>glotaran.builtin.io.pandas.csv</code>	Module containing CSV io support.
<code>glotaran.builtin.io.pandas.tsv</code>	Module containing TSV io support.
<code>glotaran.builtin.io.pandas.xlsx</code>	Module containing Excel like io support.

## csv

Module containing CSV io support.

## Classes

### Summary

<code>CsvProjectIo</code>	Plugin for CSV data io.
---------------------------	-------------------------

### CsvProjectIo

**class** `glotaran.builtin.io.pandas.csv.CsvProjectIo`(*format\_name*: *str*)

Bases: `ProjectIoInterface`

Plugin for CSV data io.

Initialize a Project IO plugin with the name of the format.

**Parameters**

- **format\_name** (*str*) – Name of the supported format an instance uses.

## Methods Summary

<code>load_model</code>	Create a Model instance from the specs defined in a file.
<code>load_parameters</code>	Load parameters from CSV file.
<code>load_result</code>	Create a Result instance from the specs defined in a file.
<code>load_scheme</code>	Create a Scheme instance from the specs defined in a file.
<code>save_model</code>	Save a Model instance to a spec file.
<code>save_parameters</code>	Save a Parameters to a CSV file.
<code>save_result</code>	Save a Result instance to a spec file.
<code>save_scheme</code>	Save a Scheme instance to a spec file.

### load\_model

`CsvProjectIo.load_model(file_name: str) → Model`

Create a Model instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the model specs.

##### Returns

Model instance created from the file.

##### Return type

*Model*

### load\_parameters

`CsvProjectIo.load_parameters(file_name: str, sep: str = ',') → Parameters`

Load parameters from CSV file.

##### Parameters

- **file\_name** (*str*) – Name of file to be loaded.
- **sep** (*str*) – Other separators can be used optionally., by default ‘,’

##### Return type

class: Parameters

### load\_result

`CsvProjectIo.load_result(result_path: str) → Result`

Create a Result instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**result\_path** (*str*) – Path containing the result data.

##### Returns

Result instance created from the file.

##### Return type

*Result*

## load\_scheme

CsvProjectIo.load\_scheme(file\_name: str) → Scheme

Create a Scheme instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

**file\_name** (str) – File containing the parameter specs.

#### Returns

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

## save\_model

CsvProjectIo.save\_model(model: Model, file\_name: str)

Save a Model instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **model** (Model) – Model instance to save to specs file.
- **file\_name** (str) – File to write the model specs to.

## save\_parameters

CsvProjectIo.save\_parameters(parameters: Parameters, file\_name: str, \*, sep: str = ',',  
as\_optimized: bool = True, replace\_infinity: bool = True)  
→ None

Save a Parameters to a CSV file.

#### Parameters

- **parameters** (Parameters) – Parameters to be saved to file.
- **file\_name** (str) – File to write the parameters to.
- **sep** (str) – Other separators can be used optionally., by default ','
- **as\_optimized** (bool) – Weather to include properties which are the result of optimization.
- **replace\_infinity** (bool) – Weather to replace infinity values with empty strings.

## save\_result

CsvProjectIo.save\_result(result: Result, result\_path: str, \*, saving\_options: SavingOptions  
= SavingOptions(data\_filter=None, data\_format='nc',  
parameter\_format='csv', report=True)) → list[str]

Save a Result instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **result** (Result) – Result instance to save to specs file.
- **result\_path** (str) – Path to write the result data to.
- **saving\_options** (SavingOptions) – Options for the saved result.

## save\_scheme

CsvProjectIo.**save\_scheme**(*scheme*: [Scheme](#), *file\_name*: *str*)

Save a Scheme instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## Methods Documentation

**load\_model**(*file\_name*: *str*) → [Model](#)

Create a Model instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

- **file\_name** (*str*) – File containing the model specs.

#### Returns

Model instance created from the file.

#### Return type

[Model](#)

**load\_parameters**(*file\_name*: *str*, *sep*: *str* = ',') → [Parameters](#)

Load parameters from CSV file.

#### Parameters

- **file\_name** (*str*) – Name of file to be loaded.
- **sep** (*str*) – Other separators can be used optionally., by default ','

#### Return type

class: Parameters

**load\_result**(*result\_path*: *str*) → [Result](#)

Create a Result instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

- **result\_path** (*str*) – Path containing the result data.

#### Returns

Result instance created from the file.

#### Return type

[Result](#)

**load\_scheme**(*file\_name*: *str*) → [Scheme](#)

Create a Scheme instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

- **file\_name** (*str*) – File containing the parameter specs.

#### Returns

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**save\_model**(*model*: [Model](#), *file\_name*: *str*)

Save a Model instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **model** ([Model](#)) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

**save\_parameters**(*parameters*: [Parameters](#), *file\_name*: *str*, \*, *sep*: *str* = ',', *as\_optimized*: *bool* = True, *replace\_infinity*: *bool* = True) → None

Save a Parameters to a CSV file.

**Parameters**

- **parameters** ([Parameters](#)) – Parameters to be saved to file.
- **file\_name** (*str*) – File to write the parameters to.
- **sep** (*str*) – Other separators can be used optionally., by default ','
- **as\_optimized** (*bool*) – Weather to include properties which are the result of optimization.
- **replace\_infinity** (*bool*) – Weather to replace infinity values with empty strings.

**save\_result**(*result*: [Result](#), *result\_path*: *str*, \*, *saving\_options*: [SavingOptions](#) = [SavingOptions](#)(*data\_filter*=None, *data\_format*='nc', *parameter\_format*='csv', *report*=True)) → list[*str*]

Save a Result instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **result** ([Result](#)) – Result instance to save to specs file.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** ([SavingOptions](#)) – Options for the saved result.

**save\_scheme**(*scheme*: [Scheme](#), *file\_name*: *str*)

Save a Scheme instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

**tsv**

Module containing TSV io support.

**Classes****Summary**

<i>TsvProjectIo</i>	Plugin for TSV data io.
---------------------	-------------------------

## TsvProjectIo

**class** `glotaran.builtin.io.pandas.tsv.TsvProjectIo`(*format\_name: str*)

Bases: *ProjectIoInterface*

Plugin for TSV data io.

Initialize a Project IO plugin with the name of the format.

### Parameters

**format\_name** (*str*) – Name of the supported format an instance uses.

## Methods Summary

<i>load_model</i>	Create a Model instance from the specs defined in a file.
<i>load_parameters</i>	Load parameters from TSV file.
<i>load_result</i>	Create a Result instance from the specs defined in a file.
<i>load_scheme</i>	Create a Scheme instance from the specs defined in a file.
<i>save_model</i>	Save a Model instance to a spec file.
<i>save_parameters</i>	Save a Parameters to a TSV file.
<i>save_result</i>	Save a Result instance to a spec file.
<i>save_scheme</i>	Save a Scheme instance to a spec file.

## load\_model

`TsvProjectIo.load_model`(*file\_name: str*) → *Model*

Create a Model instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

**file\_name** (*str*) – File containing the model specs.

#### Returns

Model instance created from the file.

#### Return type

*Model*

## load\_parameters

`TsvProjectIo.load_parameters`(*file\_name: str*) → *Parameters*

Load parameters from TSV file.

#### Parameters

**file\_name** (*str*) – Name of file to be loaded.

#### Return type

*Parameters*

## load\_result

`TsvProjectIo.load_result(result_path: str) → Result`

Create a Result instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

**result\_path** (*str*) – Path containing the result data.

#### Returns

Result instance created from the file.

#### Return type

*Result*

## load\_scheme

`TsvProjectIo.load_scheme(file_name: str) → Scheme`

Create a Scheme instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

**file\_name** (*str*) – File containing the parameter specs.

#### Returns

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

## save\_model

`TsvProjectIo.save_model(model: Model, file_name: str)`

Save a Model instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

## save\_parameters

`TsvProjectIo.save_parameters(parameters: Parameters, file_name: str, *, replace_infinity: bool = True) → None`

Save a Parameters to a TSV file.

#### Parameters

- **parameters** (*Parameters*) – Parameters to be saved to file.
- **file\_name** (*str*) – File to write the parameters to.
- **replace\_infinity** (*bool*) – Weather to replace infinity values with empty strings.

## save\_result

`TsvProjectIo.save_result(result: Result, result_path: str, *, saving_options: SavingOptions = SavingOptions(data_filter=None, data_format='nc', parameter_format='csv', report=True)) → list[str]`

Save a Result instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **result** ([Result](#)) – Result instance to save to specs file.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** ([SavingOptions](#)) – Options for the saved result.

## save\_scheme

`TsvProjectIo.save_scheme(scheme: Scheme, file_name: str)`

Save a Scheme instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## Methods Documentation

`load_model(file_name: str)` → [Model](#)

Create a Model instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

- **file\_name** (*str*) – File containing the model specs.

#### Returns

Model instance created from the file.

#### Return type

[Model](#)

`load_parameters(file_name: str)` → [Parameters](#)

Load parameters from TSV file.

#### Parameters

- **file\_name** (*str*) – Name of file to be loaded.

#### Return type

[Parameters](#)

`load_result(result_path: str)` → [Result](#)

Create a Result instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

- **result\_path** (*str*) – Path containing the result data.

#### Returns

Result instance created from the file.

#### Return type

[Result](#)



**load\_scheme**(*file\_name*: *str*) → *Scheme*

Create a Scheme instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**save\_model**(*model*: *Model*, *file\_name*: *str*)

Save a Model instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

**save\_parameters**(*parameters*: *Parameters*, *file\_name*: *str*, \*, *replace\_infinity*: *bool* = *True*)  
→ *None*

Save a Parameters to a TSV file.

**Parameters**

- **parameters** (*Parameters*) – Parameters to be saved to file.
- **file\_name** (*str*) – File to write the parameters to.
- **replace\_infinity** (*bool*) – Weather to replace infinity values with empty strings.

**save\_result**(*result*: *Result*, *result\_path*: *str*, \*, *saving\_options*: *SavingOptions* =  
*SavingOptions*(*data\_filter*=*None*, *data\_format*='nc', *parameter\_format*='csv',  
*report*=*True*)) → *list*[*str*]

Save a Result instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **result** (*Result*) – Result instance to save to specs file.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** (*SavingOptions*) – Options for the saved result.

**save\_scheme**(*scheme*: *Scheme*, *file\_name*: *str*)

Save a Scheme instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **scheme** (*Scheme*) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## xlsx

Module containing Excel like io support.

## Classes

### Summary

<i>ExcelProjectIo</i>	Plugin for Excel like data io.
-----------------------	--------------------------------

### ExcelProjectIo

**class** `glotaran.builtin.io.pandas.xlsx.ExcelProjectIo(format_name: str)`

Bases: *ProjectIoInterface*

Plugin for Excel like data io.

Initialize a Project IO plugin with the name of the format.

#### Parameters

**format\_name** (*str*) – Name of the supported format an instance uses.

### Methods Summary

<i>load_model</i>	Create a Model instance from the specs defined in a file.
<i>load_parameters</i>	Load parameters from XLSX file.
<i>load_result</i>	Create a Result instance from the specs defined in a file.
<i>load_scheme</i>	Create a Scheme instance from the specs defined in a file.
<i>save_model</i>	Save a Model instance to a spec file.
<i>save_parameters</i>	Save a Parameters to a Excel file.
<i>save_result</i>	Save a Result instance to a spec file.
<i>save_scheme</i>	Save a Scheme instance to a spec file.

### load\_model

`ExcelProjectIo.load_model(file_name: str) → Model`

Create a Model instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the model specs.

##### Returns

Model instance created from the file.

##### Return type

*Model*

### load\_parameters

ExcelProjectIo.load\_parameters(file\_name: str) → Parameters

Load parameters from XLSX file.

**Parameters**

**file\_name** (str) – Name of file to be loaded.

**Return type**

Parameters

### load\_result

ExcelProjectIo.load\_result(result\_path: str) → Result

Create a Result instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**result\_path** (str) – Path containing the result data.

**Returns**

Result instance created from the file.

**Return type**

Result

### load\_scheme

ExcelProjectIo.load\_scheme(file\_name: str) → Scheme

Create a Scheme instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (str) – File containing the parameter specs.

**Returns**

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

### save\_model

ExcelProjectIo.save\_model(model: Model, file\_name: str)

Save a Model instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **model** (Model) – Model instance to save to specs file.
- **file\_name** (str) – File to write the model specs to.

### save\_parameters

ExcelProjectIo.**save\_parameters**(parameters: [Parameters](#), file\_name: *str*)

Save a Parameters to a Excel file.

#### Parameters

- **parameters** ([Parameters](#)) – Parameters to be saved to file.
- **file\_name** (*str*) – File to write the parameters to.

### save\_result

ExcelProjectIo.**save\_result**(result: [Result](#), result\_path: *str*, \*, saving\_options: [SavingOptions](#) = [SavingOptions](#)(data\_filter=None, data\_format='nc', parameter\_format='csv', report=True)) → [list](#)[*str*]

Save a Result instance to a spec file.

#### NOT IMPLEMENTED

##### Parameters

- **result** ([Result](#)) – Result instance to save to specs file.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** ([SavingOptions](#)) – Options for the saved result.

### save\_scheme

ExcelProjectIo.**save\_scheme**(scheme: [Scheme](#), file\_name: *str*)

Save a Scheme instance to a spec file.

#### NOT IMPLEMENTED

##### Parameters

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## Methods Documentation

**load\_model**(file\_name: *str*) → [Model](#)

Create a Model instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

- **file\_name** (*str*) – File containing the model specs.

##### Returns

Model instance created from the file.

##### Return type

[Model](#)

**load\_parameters**(file\_name: *str*) → [Parameters](#)

Load parameters from XLSX file.

##### Parameters

- **file\_name** (*str*) – Name of file to be loaded.

##### Return type

[Parameters](#)

**load\_result**(*result\_path*: *str*) → *Result*

Create a Result instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**result\_path** (*str*) – Path containing the result data.

**Returns**

Result instance created from the file.

**Return type**

*Result*

**load\_scheme**(*file\_name*: *str*) → *Scheme*

Create a Scheme instance from the specs defined in a file.

**NOT IMPLEMENTED**

**Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**save\_model**(*model*: *Model*, *file\_name*: *str*)

Save a Model instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

**save\_parameters**(*parameters*: *Parameters*, *file\_name*: *str*)

Save a Parameters to a Excel file.

**Parameters**

- **parameters** (*Parameters*) – Parameters to be saved to file.
- **file\_name** (*str*) – File to write the parameters to.

**save\_result**(*result*: *Result*, *result\_path*: *str*, \*, *saving\_options*: *SavingOptions* = *SavingOptions*(*data\_filter*=None, *data\_format*='nc', *parameter\_format*='csv', *report*=True)) → list[*str*]

Save a Result instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **result** (*Result*) – Result instance to save to specs file.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** (*SavingOptions*) – Options for the saved result.

**save\_scheme**(*scheme*: *Scheme*, *file\_name*: *str*)

Save a Scheme instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **scheme** (*Scheme*) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## sdt

Package containing the SDT Data IO plugin.

## Modules

<code>glotaran.builtin.io.sdt.sdt_file_reader</code>	Module containing the SDT Data IO plugin.
--	---

## sdt\_file\_reader

Module containing the SDT Data IO plugin.

## Classes

### Summary

<code>SdtDataIo</code>	Plugin for SDT data io.
------------------------	-------------------------

### SdtDataIo

**class** `glotaran.builtin.io.sdt.sdt_file_reader.SdtDataIo`(*format\_name*: *str*)

Bases: `DataIoInterface`

Plugin for SDT data io.

Initialize a Data IO plugin with the name of the format.

#### Parameters

**format\_name** (*str*) – Name of the supported format an instance uses.

### Methods Summary

<code>load_dataset</code>	Read a <i>*.sdt</i> file and returns a <code>xarray.Dataset</code> containing its data.
<code>save_dataset</code>	Save data from <code>xarray.Dataset</code> to a file.

## load\_dataset

SdtDataIo.**load\_dataset**(*file\_name*: *str*, \*, *index*: *ndarray* | *None* = *None*, *flim*: *bool* = *False*,  
*dataset\_index*: *int* | *None* = *None*, *swap\_axis*: *bool* = *False*,  
*orig\_time\_axis\_index*: *int* = 2) → *Dataset*

Read a \*.sdt file and returns a *xarray.Dataset* containing its data.

### Parameters

- **file\_name** (*str*) – Path to the sdt file which should be read.
- **index** (*np.ndarray* | *None*) – This is only needed if *type\_of\_data*=="st", since \*.sdt files, which only contain spectral temporal data, lack the spectral information. Thus for the spectral axis data need to be given by the user.
- **flim** (*bool*) – Set true if reading a result from a FLIM measurement. Defaults to *False*.
- **dataset\_index** (*int*) – If the \*.sdt file contains multiple datasets the index will be used to select the wanted one. Defaults to 0.
- **swap\_axis** (*bool*) – Flag to switch a wavelength explicit *input\_df* to time explicit *input\_df*, before generating the SpectralTemporalDataset. Defaults to *False*.
- **orig\_time\_axis\_index** (*int*) – Index of the axis which corresponds to the time axis. I.e. for data of shape (64, 64, 256), which are a 64x64 pixel map with 256 time steps, *orig\_time\_axis\_index*=2.

### Return type

*xr.Dataset*

### Raises

**IndexError** – If the length of the index array is incompatible with the data.

## save\_dataset

SdtDataIo.**save\_dataset**(*dataset*: *xr.Dataset* | *xr.DataArray*, *file\_name*: *str*)

Save data from *xarray.Dataset* to a file.

### NOT IMPLEMENTED

### Parameters

- **dataset** (*xr.Dataset*) – Dataset to be saved to file.
- **file\_name** (*str*) – File to write the data to.

## Methods Documentation

**load\_dataset**(*file\_name*: *str*, \*, *index*: *ndarray* | *None* = *None*, *flim*: *bool* = *False*,  
*dataset\_index*: *int* | *None* = *None*, *swap\_axis*: *bool* = *False*,  
*orig\_time\_axis\_index*: *int* = 2) → *Dataset*

Read a \*.sdt file and returns a *xarray.Dataset* containing its data.

### Parameters

- **file\_name** (*str*) – Path to the sdt file which should be read.
- **index** (*np.ndarray* | *None*) – This is only needed if *type\_of\_data*=="st", since \*.sdt files, which only contain spectral temporal data, lack the spectral information. Thus for the spectral axis data need to be given by the user.
- **flim** (*bool*) – Set true if reading a result from a FLIM measurement. Defaults to *False*.
- **dataset\_index** (*int*) – If the \*.sdt file contains multiple datasets the index will be used to select the wanted one. Defaults to 0.
- **swap\_axis** (*bool*) – Flag to switch a wavelength explicit *input\_df* to time explicit *input\_df*, before generating the SpectralTemporalDataset. Defaults to *False*.

- **orig\_time\_axis\_index** (*int*) – Index of the axis which corresponds to the time axis. I.e. for data of shape (64, 64, 256), which are a 64x64 pixel map with 256 time steps, orig\_time\_axis\_index=2.

**Return type**

xr.Dataset

**Raises**

**IndexError** – If the length of the index array is incompatible with the data.

**save\_dataset** (*dataset: xr.Dataset | xr.DataArray, file\_name: str*)

Save data from `xarray.Dataset` to a file.

**NOT IMPLEMENTED****Parameters**

- **dataset** (*xr.Dataset*) – Dataset to be saved to file.
- **file\_name** (*str*) – File to write the data to.

## yaml

Package containing the YAML Data and Project IO plugins.

## Modules

<code>glotaran.builtin.io.yaml.utils</code>	Utility module for <code>glotaran.builtin.io.yaml.yaml</code> .
<code>glotaran.builtin.io.yaml.yaml</code>	Module containing the YAML Data and Project IO plugins.

## utils

Utility module for `glotaran.builtin.io.yaml.yaml`.

## Functions

### Summary

<code>load_dict</code>	Load yaml code from a file or string and returns the dict interpretation.
<code>write_dict</code>	Write a mapping (e.g. <code>dict</code> ) or sequence (e.g. <code>list</code> ) as yaml to file or str.



## load\_dict

glotaran.builtin.io.yml.utils.**load\_dict**(source: *str* | *Path*, is\_file: *bool*) → dict[*str*, Any]

Load yaml code from a file or string and returns the dict interpretation.

### Parameters

- **source** (*str* | *Path*) – Path to a file or string containing the yaml code.
- **is\_file** (*bool*) – Whether or not source is a file.

### Return type

dict[*str*, Any]

## write\_dict

glotaran.builtin.io.yml.utils.**write\_dict**(data: Mapping[*str*, Any] | Sequence[*Any*],  
file\_name: *str* | *Path* | *None* = *None*, offset: *int* =  
0) → *str* | *None*

Write a mapping (e.g. dict) or sequence (e.g. list) as yaml to file or str.

### Parameters

- **data** (Mapping[*str*, Any] | Sequence[*Any*]) – Data that should be converted to yaml.
- **file\_name** (*str* | *Path* | *None*) – Path of the file to write the yaml code to. Defaults to *None* which makes this function return a string.
- **offset** (*int*) – Block indentation level. Defaults to 0 See <https://yaml.readthedocs.io/en/latest/detail/#indentation-of-block-sequences>

### Returns

String if file\_name is *None* or *None* if file\_name is a valid path.

### Return type

*str* | *None*

## yml

Module containing the YAML Data and Project IO plugins.

## Classes

### Summary

*YmlProjectIo*

Plugin for YAML project io.

## YmlProjectIo

**class** `glotaran.builtin.io.yml.yml.YmlProjectIo`(*format\_name: str*)

Bases: *ProjectIoInterface*

Plugin for YAML project io.

Initialize a Project IO plugin with the name of the format.

### Parameters

**format\_name** (*str*) – Name of the supported format an instance uses.

## Methods Summary

<i>load_model</i>	Load a <i>Model</i> from a model specification in a yaml file.
<i>load_parameters</i>	Load <i>Parameters</i> instance from the specification defined in <i>file_name</i> .
<i>load_result</i>	Create a <i>Result</i> instance from the specs defined in a file.
<i>load_scheme</i>	Load <i>Scheme</i> instance from the specification defined in <i>file_name</i> .
<i>save_model</i>	Save a <i>Model</i> instance to a specification file.
<i>save_parameters</i>	Save a <i>Parameters</i> instance to a spec file.
<i>save_result</i>	Write a <i>Result</i> instance to a specification file and data files.
<i>save_scheme</i>	Write a <i>Scheme</i> instance to a specification file <i>file_name</i> .

## load\_model

`YmlProjectIo.load_model`(*file\_name: str*) → *Model*

Load a *Model* from a model specification in a yaml file.

### Parameters

**file\_name** (*str*) – Path to the model file to read.

### Raises

- **ValueError** – If *megacomplex* was not provided in the model specification.
- **ValueError** – If *default\_megacomplex* was not provided and any *megacomplex* is missing the *type* attribute.

### Return type

*Model*

## load\_parameters

`YmlProjectIo.load_parameters(file_name: str) → Parameters`

Load Parameters instance from the specification defined in `file_name`.

**Parameters**

**file\_name** (*str*) – File containing the parameter specification.

**Return type**

*Parameters*

## load\_result

`YmlProjectIo.load_result(result_path: str) → Result`

Create a Result instance from the specs defined in a file.

**Parameters**

**result\_path** (*str*) – Path containing the result data.

**Returns**

Result instance created from the saved format.

**Return type**

*Result*

## load\_scheme

`YmlProjectIo.load_scheme(file_name: str) → Scheme`

Load Scheme instance from the specification defined in `file_name`.

**Parameters**

**file\_name** (*str*) – File containing the scheme specification.

**Return type**

*Scheme*

## save\_model

`YmlProjectIo.save_model(model: Model, file_name: str)`

Save a Model instance to a specification file.

**Parameters**

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

## save\_parameters

`YmlProjectIo.save_parameters(parameters: Parameters, file_name: str)`

Save a Parameters instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **parameters** (*Parameters*) – Parameters instance to save to specs file.
- **file\_name** (*str*) – File to write the parameter specs to.

## save\_result

`YmlProjectIo.save_result(result: Result, result_path: str, saving_options: SavingOptions = SavingOptions(data_filter=None, data_format='nc', parameter_format='csv', report=True)) → list[str]`

Write a `Result` instance to a specification file and data files.

Returns a list with paths of all saved items. The following files are saved if not configured otherwise: \* `result.md`: The result with the model formatted as markdown text. \* `result.yml`: Yaml spec file of the result \* `model.yml`: Model spec file. \* `scheme.yml`: Scheme spec file. \* `initial_parameters.csv`: Initially used parameters. \* `optimized_parameters.csv`: The optimized parameter as csv file. \* `parameter_history.csv`: Parameter changes over the optimization \* `optimization_history.csv`: Parsed table printed by the SciPy optimizer \* `{dataset_label}.nc`: The result data for each dataset as NetCDF file.

### Parameters

- **result** ([Result](#)) – `Result` instance to write.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** ([SavingOptions](#)) – Options for saving the the result.

### Returns

List of file paths which were created.

### Return type

*list*[*str*]

## save\_scheme

`YmlProjectIo.save_scheme(scheme: Scheme, file_name: str)`

Write a `Scheme` instance to a specification file `file_name`.

### Parameters

- **scheme** ([Scheme](#)) – `Scheme` instance to save to file.
- **file\_name** (*str*) – Path to the file to write the scheme specification to.

## Methods Documentation

`load_model(file_name: str)` → [Model](#)

Load a `Model` from a model specification in a yaml file.

### Parameters

**file\_name** (*str*) – Path to the model file to read.

### Raises

- **ValueError** – If megacomplex was not provided in the model specification.
- **ValueError** – If `default_megacomplex` was not provided and any megacomplex is missing the type attribute.

### Return type

[Model](#)

`load_parameters(file_name: str)` → [Parameters](#)

Load `Parameters` instance from the specification defined in `file_name`.

### Parameters

**file\_name** (*str*) – File containing the parameter specification.

### Return type

[Parameters](#)

**load\_result**(*result\_path*: *str*) → *Result*

Create a *Result* instance from the specs defined in a file.

**Parameters**

**result\_path** (*str*) – Path containing the result data.

**Returns**

*Result* instance created from the saved format.

**Return type**

*Result*

**load\_scheme**(*file\_name*: *str*) → *Scheme*

Load *Scheme* instance from the specification defined in *file\_name*.

**Parameters**

**file\_name** (*str*) – File containing the scheme specification.

**Return type**

*Scheme*

**save\_model**(*model*: *Model*, *file\_name*: *str*)

Save a *Model* instance to a specification file.

**Parameters**

- **model** (*Model*) – *Model* instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

**save\_parameters**(*parameters*: *Parameters*, *file\_name*: *str*)

Save a *Parameters* instance to a spec file.

**NOT IMPLEMENTED**

**Parameters**

- **parameters** (*Parameters*) – *Parameters* instance to save to specs file.
- **file\_name** (*str*) – File to write the parameter specs to.

**save\_result**(*result*: *Result*, *result\_path*: *str*, *saving\_options*: *SavingOptions* = *SavingOptions*(*data\_filter*=None, *data\_format*='nc', *parameter\_format*='csv', *report*=True)) → *list*[*str*]

Write a *Result* instance to a specification file and data files.

Returns a list with paths of all saved items. The following files are saved if not configured otherwise: \* *result.md*: The result with the model formatted as markdown text. \* *result.yml*: Yaml spec file of the result \* *model.yml*: Model spec file. \* *scheme.yml*: Scheme spec file. \* *initial\_parameters.csv*: Initially used parameters. \* *optimized\_parameters.csv*: The optimized parameter as csv file. \* *parameter\_history.csv*: Parameter changes over the optimization \* *optimization\_history.csv*: Parsed table printed by the SciPy optimizer \* {*dataset\_label*}.nc: The result data for each dataset as NetCDF file.

**Parameters**

- **result** (*Result*) – *Result* instance to write.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** (*SavingOptions*) – Options for saving the the result.

**Returns**

List of file paths which were created.

**Return type**

*list*[*str*]

**save\_scheme**(*scheme*: *Scheme*, *file\_name*: *str*)

Write a *Scheme* instance to a specification file *file\_name*.

**Parameters**

- **scheme** (*Scheme*) – *Scheme* instance to save to file.
- **file\_name** (*str*) – Path to the file to write the scheme specification to.

## megacomplexes

### Modules

<code>glotaran.builtin.megacomplexes.baseline</code>
<code>glotaran.builtin.megacomplexes.clp_guide</code>
<code>glotaran.builtin.megacomplexes.coherent_artifact</code>
<code>glotaran.builtin.megacomplexes.damped_oscillation</code>
<code>glotaran.builtin.megacomplexes.decay</code>
<code>glotaran.builtin.megacomplexes.spectral</code>

## baseline

### Modules

<code>glotaran.builtin.megacomplexes.baseline.baseline_megacomplex</code>
---

## baseline\_megacomplex

### Classes

#### Summary

<code>BaselineMegacomplex</code>	Method generated by attrs for class BaselineMegacomplex.
----------------------------------	--

#### BaselineMegacomplex

```
class glotaran.builtin.megacomplexes.baseline.baseline_megacomplex.BaselineMegacomplex(*,
label:
str,
di-
men-
sion:
str
|
None
=
None,
type:
str
=
'base-
line')
```

Bases: Megacomplex  
Method generated by attrs for class BaselineMegacomplex.

Attributes Summary

type
dimension
label

type

BaselineMegacomplex.type: str

dimension

BaselineMegacomplex.dimension: str | None

label

BaselineMegacomplex.label: str

## Methods Summary

<code>calculate_matrix</code>	Calculate the megacomplex matrix.
<code>finalize_data</code>	Finalize a dataset.
<code>get_dataset_model_type</code>	Get the dataset model type.
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

### `calculate_matrix`

`BaselineMegacomplex.calculate_matrix(dataset_model: DatasetModel, global_axis: ArrayLike, model_axis: ArrayLike, **kwargs)`

Calculate the megacomplex matrix.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **\*\*kwargs** – Additional arguments.

#### Returns

- `tuple[list[str], ArrayLike]` – The clp labels and the matrix.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

### `finalize_data`

`BaselineMegacomplex.finalize_data(dataset_model: DatasetModel, dataset: Dataset, is_full_model: bool = False, as_global: bool = False)`

Finalize a dataset.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **dataset** (`xr.Dataset`) – The dataset.
- **is\_full\_model** (`bool`) – Whether the model is a full model.
- **as\_global** (`bool`) – Whether megacomplex is calculated as global megacomplex.

### `get_dataset_model_type`

`classmethod BaselineMegacomplex.get_dataset_model_type() → type | None`

Get the dataset model type.

#### Return type

`type | None`



### get\_item\_type

**classmethod** BaselineMegacomplex.get\_item\_type() → str

Get the type string.

**Return type**

str

### get\_item\_type\_class

**classmethod** BaselineMegacomplex.get\_item\_type\_class(item\_type: str) → Type

Get the type for a type string.

**Parameters**

**item\_type** (str) – The type string.

**Return type**

Type

### get\_item\_types

**classmethod** BaselineMegacomplex.get\_item\_types() → list[str]

Get all type strings.

**Return type**

list[str]

## Methods Documentation

**calculate\_matrix**(dataset\_model: DatasetModel, global\_axis: ArrayLike, model\_axis: ArrayLike, \*\*kwargs)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **global\_axis** (ArrayLike) – The global axis.
- **model\_axis** (ArrayLike) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple*[list[str], ArrayLike] – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**dimension:** str | None

**finalize\_data**(dataset\_model: DatasetModel, dataset: Dataset, is\_full\_model: bool = False, as\_global: bool = False)

Finalize a dataset.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **dataset** (xr.Dataset) – The dataset.
- **is\_full\_model** (bool) – Whether the model is a full model.
- **as\_global** (bool) – Whether megacomplex is calculated as global megacomplex.

**classmethod** `get_dataset_model_type()` → `type` | `None`

Get the dataset model type.

**Return type**

`type` | `None`

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**label:** `str`

**type:** `str`

## clp\_guide

### Modules

---

```
glotaran.builtin.megacomplexes.clp_guide.  
clp_guide_megacomplex
```

---

## clp\_guide\_megacomplex

### Classes

#### Summary

---

*ClpGuideMegacomplex*

Method generated by attrs for class ClpGuide-Megacomplex.

---

ClpGuideMegacomplex

```
class glotaran.builtin.megacomplexes.clp_guide.clp_guide_megacomplex.ClpGuideMegacomplex(*,
label:
str,
di-
men-
sion:
str
|
None
=
None,
type:
str
=
'clp-
guide',
tar-
get:
str)
```

Bases: Megacomplex  
Method generated by attrs for class ClpGuideMegacomplex.

Attributes Summary

<i>type</i>
<i>target</i>
<i>dimension</i>
<i>label</i>

**type**

ClpGuideMegacomplex.type: `str`

**target**

ClpGuideMegacomplex.target: `str`

**dimension**

ClpGuideMegacomplex.dimension: `str` | `None`

**label**

ClpGuideMegacomplex.label: `str`

**Methods Summary**

<code>calculate_matrix</code>	Calculate the megacomplex matrix.
<code>finalize_data</code>	Finalize a dataset.
<code>get_dataset_model_type</code>	Get the dataset model type.
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

**calculate\_matrix**

ClpGuideMegacomplex.calculate\_matrix(*dataset\_model*: [DatasetModel](#), *global\_axis*: *ArrayLike*, *model\_axis*: *ArrayLike*, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** ([DatasetModel](#)) – The dataset model.
- **global\_axis** (*ArrayLike*) – The global axis.
- **model\_axis** (*ArrayLike*) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple*[*list*[*str*], *ArrayLike*] – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

## finalize\_data

`ClpGuideMegacomplex.finalize_data(dataset_model: DatasetModel, dataset: Dataset, is_full_model: bool = False, as_global: bool = False)`

Finalize a dataset.

### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **dataset** (`xr.Dataset`) – The dataset.
- **is\_full\_model** (`bool`) – Whether the model is a full model.
- **as\_global** (`bool`) – Whether megacomplex is calculated as global megacomplex.

## get\_dataset\_model\_type

`classmethod ClpGuideMegacomplex.get_dataset_model_type() → type | None`

Get the dataset model type.

### Return type

`type | None`

## get\_item\_type

`classmethod ClpGuideMegacomplex.get_item_type() → str`

Get the type string.

### Return type

`str`

## get\_item\_type\_class

`classmethod ClpGuideMegacomplex.get_item_type_class(item_type: str) → Type`

Get the type for a type string.

### Parameters

- **item\_type** (`str`) – The type string.

### Return type

`Type`

## get\_item\_types

`classmethod ClpGuideMegacomplex.get_item_types() → list[str]`

Get all type strings.

### Return type

`list[str]`

## Methods Documentation

**calculate\_matrix**(*dataset\_model*: DatasetModel, *global\_axis*: ArrayLike, *model\_axis*: ArrayLike, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **global\_axis** (ArrayLike) – The global axis.
- **model\_axis** (ArrayLike) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple*[*list*[*str*], ArrayLike] – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**dimension**: *str* | *None*

**finalize\_data**(*dataset\_model*: DatasetModel, *dataset*: Dataset, *is\_full\_model*: *bool* = *False*, *as\_global*: *bool* = *False*)

Finalize a dataset.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **dataset** (*xr.Dataset*) – The dataset.
- **is\_full\_model** (*bool*) – Whether the model is a full model.
- **as\_global** (*bool*) – Whether megacomplex is calculated as global megacomplex.

**classmethod get\_dataset\_model\_type**() → *type* | *None*

Get the dataset model type.

**Return type**

*type* | *None*

**classmethod get\_item\_type**() → *str*

Get the type string.

**Return type**

*str*

**classmethod get\_item\_type\_class**(*item\_type*: *str*) → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

**classmethod get\_item\_types**() → *list*[*str*]

Get all type strings.

**Return type**

*list*[*str*]

**label**: *str*

**target**: *str*

**type**: *str*

## coherent\_artifact

### Modules

<i>glotaran.builtin.megacomplexes. coherent_artifact. coherent_artifact_megacomplex</i>	This package contains the kinetic megacomplex item.
---	---

## coherent\_artifact\_megacomplex

This package contains the kinetic megacomplex item.

### Classes

#### Summary

<i>CoherentArtifactMegacomplex</i>	Method generated by attrs for class CoherentArtifactMegacomplex.
------------------------------------	--

## CoherentArtifactMegacomplex

`class` `glotaran.builtin.megacomplexes.coherent_artifact.coherent_artifact_megacomplex.CoherentArtifa`

Bases: `Megacomplex`

Method generated by attrs for class `CoherentArtifactMegacomplex`.

**Attributes Summary**

<i>dimension</i>
<i>type</i>
<i>order</i>
<i>width</i>
<i>label</i>



**dimension**

`CoherentArtifactMegacomplex.dimension: str`

**type**

`CoherentArtifactMegacomplex.type: str`

**order**

`CoherentArtifactMegacomplex.order: int`

**width**

`CoherentArtifactMegacomplex.width: ParameterType | None`

**label**

`CoherentArtifactMegacomplex.label: str`

**Methods Summary**

<code>calculate_matrix</code> <code>compartments</code>	Calculate the megacomplex matrix.
<code>finalize_data</code>	Finalize a dataset.
<code>get_dataset_model_type</code>	Get the dataset model type.
<code>get_irf_parameter</code>	
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

**calculate\_matrix**

`CoherentArtifactMegacomplex.calculate_matrix(dataset_model: DatasetModel, global_axis: ArrayLike, model_axis: ArrayLike, **kwargs)`

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- `tuple[list[str], ArrayLike]` – The clp labels and the matrix.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

## compartments

`CoherentArtifactMegacomplex.compartments()`

## finalize\_data

`CoherentArtifactMegacomplex.finalize_data(dataset_model: DatasetModel, dataset: Dataset, is_full_model: bool = False, as_global: bool = False)`

Finalize a dataset.

### Parameters

- `dataset_model` (`DatasetModel`) – The dataset model.
- `dataset` (`xr.Dataset`) – The dataset.
- `is_full_model` (`bool`) – Whether the model is a full model.
- `as_global` (`bool`) – Whether megacomplex is calculated as global megacomplex.

## get\_dataset\_model\_type

**classmethod** `CoherentArtifactMegacomplex.get_dataset_model_type() → type | None`

Get the dataset model type.

### Return type

`type | None`

## get\_irf\_parameter

`CoherentArtifactMegacomplex.get_irf_parameter(irf: IrfMultiGaussian, global_index: int | None, global_axis: ArrayLike) → tuple[float, float]`

## get\_item\_type

**classmethod** `CoherentArtifactMegacomplex.get_item_type() → str`

Get the type string.

### Return type

`str`

## get\_item\_type\_class

**classmethod** `CoherentArtifactMegacomplex.get_item_type_class(item_type: str) → Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

Type

## get\_item\_types

**classmethod** `CoherentArtifactMegacomplex.get_item_types() → list[str]`

Get all type strings.

**Return type**

list[str]

## Methods Documentation

**calculate\_matrix**(*dataset\_model*: DatasetModel, *global\_axis*: ArrayLike, *model\_axis*: ArrayLike, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **global\_axis** (ArrayLike) – The global axis.
- **model\_axis** (ArrayLike) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple*[list[str], ArrayLike] – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**compartments**()

**dimension**: str

**finalize\_data**(*dataset\_model*: DatasetModel, *dataset*: Dataset, *is\_full\_model*: bool = False, *as\_global*: bool = False)

Finalize a dataset.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **dataset** (xr.Dataset) – The dataset.
- **is\_full\_model** (bool) – Whether the model is a full model.
- **as\_global** (bool) – Whether megacomplex is calculated as global megacomplex.

**classmethod** `get_dataset_model_type() → type | None`

Get the dataset model type.

**Return type**

type | None

**get\_irf\_parameter**(*irf*: IrfMultiGaussian, *global\_index*: int | None, *global\_axis*: ArrayLike) → tuple[float, float]

```
classmethod get_item_type() → str
    Get the type string.
    Return type
        str

classmethod get_item_type_class(item_type: str) → Type
    Get the type for a type string.
    Parameters
        item_type (str) – The type string.
    Return type
        Type

classmethod get_item_types() → list[str]
    Get all type strings.
    Return type
        list[str]

label: str
order: int
type: str
width: ParameterType | None
```

## damped\_oscillation

### Modules

---

```
glotaran.builtin.megacomplexes.  
damped_oscillation.  
damped_oscillation_megacomplex
```

---

## damped\_oscillation\_megacomplex

### Functions

#### Summary

<code>calculate_damped_oscillation_matrix_gau</code>	Calculate the damped oscillation matrix taking into account a gaussian irf
<code>calculate_damped_oscillation_matrix_gau</code>	
<code>calculate_damped_oscillation_matrix_no_</code>	
<code>validate_oscillation_parameter</code>	

---

### calculate\_damped\_oscillation\_matrix\_gaussian\_irf

glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation\_megacomplex.calculate\_damped\_oscillation\_matrix\_gaussian\_irf

Calculate the damped oscillation matrix taking into account a gaussian irf

#### Parameters

- **frequencies** (*np.ndarray*) – an array of frequencies in THz, one per oscillation
- **rates** (*np.ndarray*) – an array of rates, one per oscillation
- **model\_axis** (*np.ndarray*) – the model axis (time)
- **center** (*float*) – the center of the gaussian IRF
- **width** (*float*) – the width () parameter of the the IRF
- **shift** (*float*) – a shift parameter per item on the global axis
- **scale** (*float*) – the scale parameter to scale the matrix by

#### Returns

An array of the real and imaginary part of the oscillation matrix, the shape being (len(model\_axis), 2\*len(frequencies)), with the first half of the second dimension representing the real part, and the other the imagine part of the oscillation

#### Return type

*np.ndarray*

### calculate\_damped\_oscillation\_matrix\_gaussian\_irf\_on\_index

`glotaran.builtin.megacomplexes.damped_oscillation.damped_oscillation_megacomplex.calculate_damped_o`

**`calculate_damped_oscillation_matrix_no_irf`**

`glotaran.builtin.megacomplexes.damped_oscillation.damped_oscillation_megacomplex.calculate_damped_o`

**validate\_oscillation\_parameter**

glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation\_megacomplex.validate\_oscillation\_parameter

**Classes**

**Summary**

<i>DampedOscillationMegacomplex</i>	Method generated by attrs for class DampedOscillationMegacomplex.
<i>OscillationParameterIssue</i>	

### DampedOscillationMegacomplex

```
class glotaran.builtin.megacomplexes.damped_oscillation.damped_oscillation_megacomplex.DampedOscillationMegacomplex
```

Bases: `Megacomplex`

Method generated by attrs for class `DampedOscillationMegacomplex`.

#### Attributes Summary

<i>dimension</i>
<i>type</i>
<i>labels</i>
<i>frequencies</i>
<i>rates</i>
<i>label</i>



## dimension

DampedOscillationMegacomplex.**dimension**: `str`

## type

DampedOscillationMegacomplex.**type**: `str`

## labels

DampedOscillationMegacomplex.**labels**: `list[str]`

## frequencies

DampedOscillationMegacomplex.**frequencies**: `list[ParameterType]`

## rates

DampedOscillationMegacomplex.**rates**: `list[ParameterType]`

## label

DampedOscillationMegacomplex.**label**: `str`

## Methods Summary

<code>calculate_matrix</code>	Calculate the megacomplex matrix.
<code>finalize_data</code>	Finalize a dataset.
<code>get_dataset_model_type</code>	Get the dataset model type.
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

## calculate\_matrix

DampedOscillationMegacomplex.**calculate\_matrix**(*dataset\_model*: [DatasetModel](#),  
*global\_axis*: *ArrayLike*, *model\_axis*:  
*ArrayLike*, *\*\*kwargs*)

Calculate the megacomplex matrix.

### Parameters

- **dataset\_model** ([DatasetModel](#)) – The dataset model.
- **global\_axis** (*ArrayLike*) – The global axis.
- **model\_axis** (*ArrayLike*) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple*[*list*[*str*], *ArrayLike*] – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**finalize\_data**

DampedOscillationMegacomplex.**finalize\_data**(*dataset\_model*: DatasetModel, *dataset*: Dataset, *is\_full\_model*: bool = False, *as\_global*: bool = False)

Finalize a dataset.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **dataset** (xr.Dataset) – The dataset.
- **is\_full\_model** (bool) – Whether the model is a full model.
- **as\_global** (bool) – Whether megacomplex is calculated as global megacomplex.

**get\_dataset\_model\_type**

**classmethod** DampedOscillationMegacomplex.**get\_dataset\_model\_type**() → type | None

Get the dataset model type.

**Return type**

type | None

**get\_item\_type**

**classmethod** DampedOscillationMegacomplex.**get\_item\_type**() → str

Get the type string.

**Return type**

str

**get\_item\_type\_class**

**classmethod** DampedOscillationMegacomplex.**get\_item\_type\_class**(*item\_type*: str) → Type

Get the type for a type string.

**Parameters**

**item\_type** (str) – The type string.

**Return type**

Type

## get\_item\_types

**classmethod** `DampedOscillationMegacomplex.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

## Methods Documentation

**calculate\_matrix**(*dataset\_model*: `DatasetModel`, *global\_axis*: `ArrayLike`, *model\_axis*: `ArrayLike`, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- `tuple[list[str], ArrayLike]` – The clp labels and the matrix.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

**dimension:** `str`

**finalize\_data**(*dataset\_model*: `DatasetModel`, *dataset*: `Dataset`, *is\_full\_model*: `bool = False`, *as\_global*: `bool = False`)

Finalize a dataset.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **dataset** (`xr.Dataset`) – The dataset.
- **is\_full\_model** (`bool`) – Whether the model is a full model.
- **as\_global** (`bool`) – Whether megacomplex is calculated as global megacomplex.

**frequencies:** `list[ParameterType]`

**classmethod** `get_dataset_model_type()` → `type | None`

Get the dataset model type.

**Return type**

`type | None`

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

- **item\_type** (`str`) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**  
`list[str]`

**label:** `str`

**labels:** `list[str]`

**rates:** `list[ParameterType]`

**type:** `str`

### OscillationParameterIssue

`class glotaran.builtin.megacomplexes.damped_oscillation.damped_oscillation_megacomplex.OscillationParameterIssue`

Bases: `ItemIssue`

### Methods Summary

<code>to_string</code>	Get the issue as string.
------------------------	--------------------------

### `to_string`

`OscillationParameterIssue.to_string()` → `str`

Get the issue as string.

**Returns**

- `str`
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

### Methods Documentation

`to_string()` → `str`

Get the issue as string.

**Returns**

- `str`
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

## decay

### Modules

<code>glotaran.builtin.megacomplexes.decay.</code>	
<code>decay_matrix_gaussian_irf</code>	
<code>glotaran.builtin.megacomplexes.decay.</code>	
<code>decay_megacomplex</code>	
<code>glotaran.builtin.megacomplexes.decay.</code>	This package contains the decay megacomplex item.
<code>decay_parallel_megacomplex</code>	
<code>glotaran.builtin.megacomplexes.decay.</code>	This package contains the decay megacomplex item.
<code>decay_sequential_megacomplex</code>	
<code>glotaran.builtin.megacomplexes.decay.</code>	This package contains the initial concentration item.
<code>initial_concentration</code>	
<code>glotaran.builtin.megacomplexes.decay.irf</code>	This package contains irf items.
<code>glotaran.builtin.megacomplexes.decay.</code>	K-Matrix
<code>k_matrix</code>	
<code>glotaran.builtin.megacomplexes.decay.util</code>	

### decay\_matrix\_gaussian\_irf

#### Functions

##### Summary

<code>calculate_decay_matrix_gaussian_irf</code>	
<code>calculate_decay_matrix_gaussian_irf_on_</code>	Calculates a decay matrix with a gaussian irf.

### calculate\_decay\_matrix\_gaussian\_irf

`glotaran.builtin.megacomplexes.decay.decay_matrix_gaussian_irf.calculate_decay_matrix_gaussian_irf`

`calculate_decay_matrix_gaussian_irf_on_index`

glotaran.builtin.megacomplexes.decay.decay\_matrix\_gaussian\_irf.calculate\_decay\_matrix\_gaussian\_irf

Calculates a decay matrix with a gaussian irf.

decay\_megacomplex

Classes

Summary

<i>DecayDatasetModel</i>	Method generated by attrs for class DecayDataset-Model.
<i>DecayMegacomplex</i>	Method generated by attrs for class DecayMega-complex.

## DecayDatasetModel



```

class glotaran.builtin.megacomplexes.decay.decay_megacomplex.DecayDatasetModel(*,
    la-
    bel:
    str,
    group:
    str
    =
    'de-
    fault',
    force_index_dependent:
    bool
    =
    False,
    mega-
    com-
    plex:
    list[Megacomplex
    |
    str],
    mega-
    com-
    plex_scale:
    list[Parameter
    |
    str]
    |
    None
    =
    None,
    global_megacomplex:
    list[Megacomplex
    |
    str]
    |
    None
    =
    None,
    global_megacomplex_sca-
    le:
    list[Parameter
    |
    str]
    |
    None
    =
    None,
    scale:
    Pa-
    ram-
    e-
    ter
    |
    str
    |
    None
    =
    None,
    ini-
    tial_concentration:
    Ini-
    tial-

```

Bases: [DatasetModel](#)

Method generated by attrs for class DecayDatasetModel.

### Attributes Summary

<i>initial_concentration</i>
<i>irf</i>
<i>group</i>
<i>force_index_dependent</i>
<i>megacomplex</i>
<i>megacomplex_scale</i>
<i>global_megacomplex</i>
<i>global_megacomplex_scale</i>
<i>scale</i>
<i>label</i>

#### **initial\_concentration**

DecayDatasetModel.**initial\_concentration**:  
**ModelItemType**[[InitialConcentration](#)] | **None**

#### **irf**

DecayDatasetModel.**irf**: **ModelItemType**[[Irf](#)] | **None**

#### **group**

DecayDatasetModel.**group**: **str**

**force\_index\_dependent**

DecayDatasetModel.force\_index\_dependent: `bool`

**megacomplex**

DecayDatasetModel.megacomplex: `list[ModelItemType[Megacomplex]]`

**megacomplex\_scale**

DecayDatasetModel.megacomplex\_scale: `list[ParameterType] | None`

**global\_megacomplex**

DecayDatasetModel.global\_megacomplex: `list[ModelItemType[Megacomplex]] | None`

**global\_megacomplex\_scale**

DecayDatasetModel.global\_megacomplex\_scale: `list[ParameterType] | None`

**scale**

DecayDatasetModel.scale: `ParameterType | None`

**label**

DecayDatasetModel.label: `str`

**Methods Summary****Methods Documentation**

**force\_index\_dependent:** `bool`

**global\_megacomplex:** `list[ModelItemType[Megacomplex]] | None`

**global\_megacomplex\_scale:** `list[ParameterType] | None`

**group:** `str`

**initial\_concentration:** `ModelItemType[InitialConcentration] | None`

```
irf: ModelItemType[Irf] | None
label: str
megacomplex: list[ModelItemType[Megacomplex]]
megacomplex_scale: list[ParameterType] | None
scale: ParameterType | None
```

### DecayMegacomplex

```
class glotaran.builtin.megacomplexes.decay.decay_megacomplex.DecayMegacomplex(*,
                                                                              la-
                                                                              bel:
                                                                              str,
                                                                              di-
                                                                              men-
                                                                              sion:
                                                                              str
                                                                              =
                                                                              'time',
                                                                              type:
                                                                              str
                                                                              =
                                                                              'de-
                                                                              cay',
                                                                              k_matrix:
                                                                              list[KMatrix
                                                                              |
                                                                              str])
```

Bases: Megacomplex

Method generated by attrs for class DecayMegacomplex.

### Attributes Summary

<i>dimension</i>
<i>type</i>
<i>k_matrix</i>
<i>label</i>

**dimension**

DecayMegacomplex.**dimension**: `str`

**type**

DecayMegacomplex.**type**: `str`

**k\_matrix**

DecayMegacomplex.**k\_matrix**: `list[ModelItemType[KMatrix]]`

**label**

DecayMegacomplex.**label**: `str`

**Methods Summary**

<code>calculate_matrix</code>	Calculate the megacomplex matrix.
<code>finalize_data</code>	Finalize a dataset.
<code>get_a_matrix</code>	
<code>get_compartments</code>	
<code>get_dataset_model_type</code>	Get the dataset model type.
<code>get_initial_concentration</code>	
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.
<code>get_k_matrix</code>	

**calculate\_matrix**

DecayMegacomplex.**calculate\_matrix**(*dataset\_model*: `DatasetModel`, *global\_axis*: `ArrayLike`, *model\_axis*: `ArrayLike`, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- `tuple[list[str], ArrayLike]` – The clp labels and the matrix.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

### finalize\_data

DecayMegacomplex.**finalize\_data**(dataset\_model: DatasetModel, dataset: Dataset,  
is\_full\_model: bool = False, as\_global: bool = False)

Finalize a dataset.

#### Parameters

- **dataset\_model** (DatasetModel) – The dataset model.
- **dataset** (xr.Dataset) – The dataset.
- **is\_full\_model** (bool) – Whether the model is a full model.
- **as\_global** (bool) – Whether megacomplex is calculated as global megacomplex.

### get\_a\_matrix

DecayMegacomplex.**get\_a\_matrix**(dataset\_model: DatasetModel) → ndarray

### get\_compartments

DecayMegacomplex.**get\_compartments**(dataset\_model: DatasetModel) → list[str]

### get\_dataset\_model\_type

**classmethod** DecayMegacomplex.**get\_dataset\_model\_type**() → type | None

Get the dataset model type.

#### Return type

type | None

### get\_initial\_concentration

DecayMegacomplex.**get\_initial\_concentration**(dataset\_model: DatasetModel,  
normalized: bool = True) → ndarray

### get\_item\_type

**classmethod** DecayMegacomplex.**get\_item\_type**() → str

Get the type string.

#### Return type

str

### get\_item\_type\_class

**classmethod** DecayMegacomplex.get\_item\_type\_class(*item\_type*: *str*) → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

### get\_item\_types

**classmethod** DecayMegacomplex.get\_item\_types() → *list[str]*

Get all type strings.

**Return type**

*list[str]*

### get\_k\_matrix

DecayMegacomplex.get\_k\_matrix() → *KMatrix*

## Methods Documentation

**calculate\_matrix**(*dataset\_model*: *DatasetModel*, *global\_axis*: *ArrayLike*, *model\_axis*: *ArrayLike*, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (*DatasetModel*) – The dataset model.
- **global\_axis** (*ArrayLike*) – The global axis.
- **model\_axis** (*ArrayLike*) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple[list[str], ArrayLike]* – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**dimension**: *str*

**finalize\_data**(*dataset\_model*: *DatasetModel*, *dataset*: *Dataset*, *is\_full\_model*: *bool* = *False*, *as\_global*: *bool* = *False*)

Finalize a dataset.

**Parameters**

- **dataset\_model** (*DatasetModel*) – The dataset model.
- **dataset** (*xr.Dataset*) – The dataset.
- **is\_full\_model** (*bool*) – Whether the model is a full model.
- **as\_global** (*bool*) – Whether megacomplex is calculated as global megacomplex.

**get\_a\_matrix**(*dataset\_model*: *DatasetModel*) → *ndarray*

**get\_compartments**(*dataset\_model*: *DatasetModel*) → *list[str]*

```
classmethod get_dataset_model_type() → type | None
    Get the dataset model type.
    Return type
        type | None

get_initial_concentration(dataset_model: DatasetModel, normalized: bool = True) →
    ndarray

classmethod get_item_type() → str
    Get the type string.
    Return type
        str

classmethod get_item_type_class(item_type: str) → Type
    Get the type for a type string.
    Parameters
        item_type (str) – The type string.
    Return type
        Type

classmethod get_item_types() → list[str]
    Get all type strings.
    Return type
        list[str]

get_k_matrix() → KMatrix

k_matrix: list[ModelItemType[KMatrix]]

label: str

type: str
```

## decay\_parallel\_megacomplex

This package contains the decay megacomplex item.

### Classes

#### Summary

<i>DecayDatasetModel</i>	Method generated by attrs for class DecayDatasetModel.
<i>DecayParallelMegacomplex</i>	Method generated by attrs for class DecayParallelMegacomplex.



## DecayDatasetModel

```
class glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex.DecayDatasetModel(*,
label:
str,
group:
str
=
'de-
fault',
force_index_a:
bool
=
False,
mega-
com-
plex:
list[Megacomplex]
|
str],
mega-
com-
plex_scale:
list[Parameter]
|
str]
|
None
=
None,
global_megacomplex:
list[Megacomplex]
|
str]
|
None
=
None,
global_megacomplex_scale:
list[Parameter]
|
str]
|
None
=
None,
scale:
Parameter
|
str
|
None
=
None,
irf:
Irf
|
str
```

Bases: *DatasetModel*

Method generated by attrs for class DecayDatasetModel.

### Attributes Summary

<i>irf</i>
<i>group</i>
<i>force_index_dependent</i>
<i>megacomplex</i>
<i>megacomplex_scale</i>
<i>global_megacomplex</i>
<i>global_megacomplex_scale</i>
<i>scale</i>
<i>label</i>

#### **irf**

DecayDatasetModel.**irf**: *Irf* | **str** | **None**

#### **group**

DecayDatasetModel.**group**: **str**

#### **force\_index\_dependent**

DecayDatasetModel.**force\_index\_dependent**: **bool**

#### **megacomplex**

DecayDatasetModel.**megacomplex**: **list**[*Megacomplex* | **str**]

**megacomplex\_scale**

DecayDatasetModel.megacomplex\_scale: `list[Parameter | str] | None`

**global\_megacomplex**

DecayDatasetModel.global\_megacomplex: `list[Megacomplex | str] | None`

**global\_megacomplex\_scale**

DecayDatasetModel.global\_megacomplex\_scale: `list[Parameter | str] | None`

**scale**

DecayDatasetModel.scale: `Parameter | str | None`

**label**

DecayDatasetModel.label: `str`

**Methods Summary****Methods Documentation**

`force_index_dependent: bool`

`global_megacomplex: list[Megacomplex | str] | None`

`global_megacomplex_scale: list[Parameter | str] | None`

`group: str`

`irf: Irf | str | None`

`label: str`

`megacomplex: list[Megacomplex | str]`

`megacomplex_scale: list[Parameter | str] | None`

`scale: Parameter | str | None`

DecayParallelMegacomplex

```
class glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex.DecayParallelMegacomplex(*,
    label:
    str,
    dimension:
    str
    =
    'time'
    type:
    str
    =
    'decay_parallel_megacomplex'
    compartments:
    list[str]
    rates:
    list[float]
    |
    str)
```

Bases: Megacomplex  
Method generated by attrs for class DecayParallelMegacomplex.

Attributes Summary

<i>dimension</i>
<i>type</i>
<i>compartments</i>
<i>rates</i>
<i>label</i>

**dimension**

DecayParallelMegacomplex.**dimension**: `str`

**type**

DecayParallelMegacomplex.**type**: `str`

**compartments**

DecayParallelMegacomplex.**compartments**: `list[str]`

**rates**

DecayParallelMegacomplex.**rates**: `list[ParameterType]`

**label**

DecayParallelMegacomplex.**label**: `str`

**Methods Summary**

<i>calculate_matrix</i>	Calculate the megacomplex matrix.
<i>finalize_data</i>	Finalize a dataset.
<i>get_a_matrix</i>	
<i>get_compartments</i>	
<i>get_dataset_model_type</i>	Get the dataset model type.
<i>get_initial_concentration</i>	
<i>get_item_type</i>	Get the type string.
<i>get_item_type_class</i>	Get the type for a type string.
<i>get_item_types</i>	Get all type strings.
<i>get_k_matrix</i>	

### calculate\_matrix

DecayParallelMegacomplex.**calculate\_matrix**(*dataset\_model*: [DecayDatasetModel](#),  
*global\_axis*: *ArrayLike*, *model\_axis*:  
*ArrayLike*, *\*\*kwargs*)

Calculate the megacomplex matrix.

#### Parameters

- **dataset\_model** ([DatasetModel](#)) – The dataset model.
- **global\_axis** (*ArrayLike*) – The global axis.
- **model\_axis** (*ArrayLike*) – The model axis.
- **\*\*kwargs** – Additional arguments.

#### Returns

- *tuple*[*list*[*str*], *ArrayLike*] – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

### finalize\_data

DecayParallelMegacomplex.**finalize\_data**(*dataset\_model*: [DatasetModel](#), *dataset*:  
*Dataset*, *is\_full\_model*: *bool* = *False*,  
*as\_global*: *bool* = *False*)

Finalize a dataset.

#### Parameters

- **dataset\_model** ([DatasetModel](#)) – The dataset model.
- **dataset** (*xr.Dataset*) – The dataset.
- **is\_full\_model** (*bool*) – Whether the model is a full model.
- **as\_global** (*bool*) – Whether megacomplex is calculated as global megacomplex.

### get\_a\_matrix

DecayParallelMegacomplex.**get\_a\_matrix**(*dataset\_model*: [DatasetModel](#)) → *ndarray*

### get\_compartments

DecayParallelMegacomplex.**get\_compartments**(*dataset\_model*: [DatasetModel](#)) → *list*[*str*]

### get\_dataset\_model\_type

**classmethod** DecayParallelMegacomplex.**get\_dataset\_model\_type**() → *type* | *None*

Get the dataset model type.

#### Return type

*type* | *None*

### get\_initial\_concentration

`DecayParallelMegacomplex.get_initial_concentration(dataset_model: DatasetModel, normalized: bool = True) → ndarray`

### get\_item\_type

**classmethod** `DecayParallelMegacomplex.get_item_type() → str`

Get the type string.

**Return type**

`str`

### get\_item\_type\_class

**classmethod** `DecayParallelMegacomplex.get_item_type_class(item_type: str) → Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

`Type`

### get\_item\_types

**classmethod** `DecayParallelMegacomplex.get_item_types() → list[str]`

Get all type strings.

**Return type**

`list[str]`

### get\_k\_matrix

`DecayParallelMegacomplex.get_k_matrix() → KMatrix`

## Methods Documentation

**calculate\_matrix**(*dataset\_model: DecayDatasetModel, global\_axis: ArrayLike, model\_axis: ArrayLike, \*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple[list[str], ArrayLike]* – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)



**compartments:** `list[str]`

**dimension:** `str`

**finalize\_data**(*dataset\_model*: `DatasetModel`, *dataset*: `Dataset`, *is\_full\_model*: `bool = False`,  
*as\_global*: `bool = False`)

Finalize a dataset.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **dataset** (`xr.Dataset`) – The dataset.
- **is\_full\_model** (`bool`) – Whether the model is a full model.
- **as\_global** (`bool`) – Whether megacomplex is calculated as global megacomplex.

**get\_a\_matrix**(*dataset\_model*: `DatasetModel`) → `ndarray`

**get\_compartments**(*dataset\_model*: `DatasetModel`) → `list[str]`

**classmethod get\_dataset\_model\_type**() → `type | None`

Get the dataset model type.

**Return type**

`type | None`

**get\_initial\_concentration**(*dataset\_model*: `DatasetModel`, *normalized*: `bool = True`) →  
`ndarray`

**classmethod get\_item\_type**() → `str`

Get the type string.

**Return type**

`str`

**classmethod get\_item\_type\_class**(*item\_type*: `str`) → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

**classmethod get\_item\_types**() → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**get\_k\_matrix**() → `KMatrix`

**label:** `str`

**rates:** `list[ParameterType]`

**type:** `str`

## decay\_sequential\_megacomplex

This package contains the decay megacomplex item.

### Classes

#### Summary

<i>DecaySequentialMegacomplex</i>	A Megacomplex with one or more K-Matrices.
-----------------------------------	--

#### DecaySequentialMegacomplex

**class** glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex.**DecaySequentialMegacomplex**

Bases: *DecayParallelMegacomplex*

A Megacomplex with one or more K-Matrices.

Method generated by attrs for class DecaySequentialMegacomplex.

Attributes Summary

<i>type</i>
<i>dimension</i>
<i>compartments</i>
<i>rates</i>
<i>label</i>

**type**

DecaySequentialMegacomplex.**type**: `str`

**dimension**

DecaySequentialMegacomplex.**dimension**: `str`

**compartments**

DecaySequentialMegacomplex.**compartments**: `list[str]`

**rates**

DecaySequentialMegacomplex.**rates**: `list[ParameterType]`

**label**

DecaySequentialMegacomplex.**label**: `str`

## Methods Summary

<code>calculate_matrix</code>	Calculate the megacomplex matrix.
<code>finalize_data</code>	Finalize a dataset.
<code>get_a_matrix</code>	
<code>get_compartments</code>	
<code>get_dataset_model_type</code>	Get the dataset model type.
<code>get_initial_concentration</code>	
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.
<code>get_k_matrix</code>	

### calculate\_matrix

`DecaySequentialMegacomplex.calculate_matrix(dataset_model: DatasetModel, global_axis: ArrayLike, model_axis: ArrayLike, **kwargs)`

Calculate the megacomplex matrix.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **\*\*kwargs** – Additional arguments.

#### Returns

- `tuple[list[str], ArrayLike]` – The clp labels and the matrix.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

### finalize\_data

`DecaySequentialMegacomplex.finalize_data(dataset_model: DatasetModel, dataset: Dataset, is_full_model: bool = False, as_global: bool = False)`

Finalize a dataset.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **dataset** (`xr.Dataset`) – The dataset.
- **is\_full\_model** (`bool`) – Whether the model is a full model.
- **as\_global** (`bool`) – Whether megacomplex is calculated as global megacomplex.

**get\_a\_matrix**

DecaySequentialMegacomplex.**get\_a\_matrix**(*dataset\_model*: DatasetModel) → ndarray

**get\_compartments**

DecaySequentialMegacomplex.**get\_compartments**(*dataset\_model*: DatasetModel) →  
list[str]

**get\_dataset\_model\_type**

**classmethod** DecaySequentialMegacomplex.**get\_dataset\_model\_type**() → type | None

Get the dataset model type.

**Return type**  
type | None

**get\_initial\_concentration**

DecaySequentialMegacomplex.**get\_initial\_concentration**(*dataset\_model*:  
DatasetModel, *normalized*:  
bool = True) → ndarray

**get\_item\_type**

**classmethod** DecaySequentialMegacomplex.**get\_item\_type**() → str

Get the type string.

**Return type**  
str

**get\_item\_type\_class**

**classmethod** DecaySequentialMegacomplex.**get\_item\_type\_class**(*item\_type*: str) →  
Type

Get the type for a type string.

**Parameters**  
**item\_type** (str) – The type string.  
**Return type**  
Type

### get\_item\_types

**classmethod** `DecaySequentialMegacomplex.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

### get\_k\_matrix

`DecaySequentialMegacomplex.get_k_matrix()` → *KMatrix*

## Methods Documentation

**calculate\_matrix**(*dataset\_model*: `DatasetModel`, *global\_axis*: *ArrayLike*, *model\_axis*: *ArrayLike*, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (*ArrayLike*) – The global axis.
- **model\_axis** (*ArrayLike*) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple*[*list*[*str*], *ArrayLike*] – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**compartments**: `list[str]`

**dimension**: `str`

**finalize\_data**(*dataset\_model*: `DatasetModel`, *dataset*: *Dataset*, *is\_full\_model*: *bool* = *False*, *as\_global*: *bool* = *False*)

Finalize a dataset.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **dataset** (*xr.Dataset*) – The dataset.
- **is\_full\_model** (*bool*) – Whether the model is a full model.
- **as\_global** (*bool*) – Whether megacomplex is calculated as global megacomplex.

**get\_a\_matrix**(*dataset\_model*: `DatasetModel`) → `ndarray`

**get\_compartments**(*dataset\_model*: `DatasetModel`) → `list[str]`

**classmethod** `get_dataset_model_type()` → *type* | *None*

Get the dataset model type.

**Return type**

*type* | *None*

**get\_initial\_concentration**(*dataset\_model*: `DatasetModel`, *normalized*: *bool* = *True*) → `ndarray`

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**get\_k\_matrix()** → *KMatrix*

**label:** `str`

**rates:** `list[ParameterType]`

**type:** `str`

## initial\_concentration

This package contains the initial concentration item.

## Classes

### Summary

*InitialConcentration*

An initial concentration describes the population of the compartments at the beginning of an experiment.

### InitialConcentration

```
class glotaran.builtin.megacomplexes.decay.initial_concentration.InitialConcentration(*,
label:
str,
com-
part-
ments:
list[str],
pa-
ram-
e-
ters:
list[Parameter
|
str],
ex-
clude_from_normalize:
list[str]
=
[])
```

Bases: `ModelItem`

An initial concentration describes the population of the compartments at the beginning of an experiment.

Method generated by attrs for class `InitialConcentration`.

Attributes Summary

<i>compartments</i>
<i>parameters</i>
<i>exclude_from_normalize</i>
<i>label</i>



**compartments**

InitialConcentration.compartments: `list[str]`

**parameters**

InitialConcentration.parameters: `list[Parameter | str]`

**exclude\_from\_normalize**

InitialConcentration.exclude\_from\_normalize: `list[str]`

**label**

InitialConcentration.label: `str`

**Methods Summary**

*normalized*

**normalized**

InitialConcentration.normalized() → `ndarray`

**Methods Documentation**

compartments: `list[str]`

exclude\_from\_normalize: `list[str]`

label: `str`

normalized() → `ndarray`

parameters: `list[Parameter | str]`

## irf

This package contains irf items.

## Classes

### Summary

<i>Irf</i>	Represents an IRF.
<i>IrfGaussian</i>	Method generated by attrs for class IrfGaussian.
<i>IrfMultiGaussian</i>	Represents a gaussian IRF.
<i>IrfSpectralGaussian</i>	Method generated by attrs for class IrfSpectral-Gaussian.
<i>IrfSpectralMultiGaussian</i>	Represents a gaussian IRF.

## Irf

**class** `glotaran.builtin.megacomplexes.decay.irf.Irf(*, label: str, type: str)`

Bases: `ModelItemTyped`

Represents an IRF.

Method generated by attrs for class Irf.

### Attributes Summary

<i>type</i>
<i>label</i>

### type

`Irf.type: str`

### label

`Irf.label: str`

## Methods Summary

<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

### get\_item\_type

**classmethod** `Irf.get_item_type()` → `str`

Get the type string.

**Return type**

`str`

### get\_item\_type\_class

**classmethod** `Irf.get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

`Type`

### get\_item\_types

**classmethod** `Irf.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

## Methods Documentation

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**label:** `str`

**type:** `str`

### IrfGaussian

```
class glotaran.builtin.megacomplexes.decay.irf.IrfGaussian(*, label: str, scale:
list[Parameter | str] | None
= None, shift:
list[Parameter | str] | None
= None, normalize: bool =
True, backsweep: bool =
False, backsweep_period:
Parameter | str | None =
None, type: str =
'gaussian', center:
Parameter | str, width:
Parameter | str)
```

Bases: *IrfMultiGaussian*

Method generated by attrs for class IrfGaussian.

### Attributes Summary

<code>type</code>
<code>center</code>
<code>width</code>
<code>scale</code>
<code>shift</code>
<code>normalize</code>
<code>backsweep</code>
<code>backsweep_period</code>
<code>label</code>

**type**

IrfGaussian.type: `str`

**center**

IrfGaussian.center: `Parameter` | `str`

**width**

IrfGaussian.width: `Parameter` | `str`

**scale**

IrfGaussian.scale: `list[ParameterType]` | `None`

**shift**

IrfGaussian.shift: `list[ParameterType]` | `None`

**normalize**

IrfGaussian.normalize: `bool`

**backsweep**

IrfGaussian.backsweep: `bool`

**backsweep\_period**

IrfGaussian.backsweep\_period: `ParameterType` | `None`

**label**

IrfGaussian.label: `str`

## Methods Summary

<code>calculate</code>	
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.
<code>is_index_dependent</code>	
<code>parameter</code>	Returns the properties of the irf with shift applied.

### `calculate`

`IrfGaussian.calculate(index: int, global_axis: ndarray, model_axis: ndarray) → ndarray`

### `get_item_type`

**classmethod** `IrfGaussian.get_item_type()` → *str*

Get the type string.

**Return type**

*str*

### `get_item_type_class`

**classmethod** `IrfGaussian.get_item_type_class(item_type: str) → Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

### `get_item_types`

**classmethod** `IrfGaussian.get_item_types()` → *list[str]*

Get all type strings.

**Return type**

*list[str]*

**is\_index\_dependent**

IrfGaussian.**is\_index\_dependent**()

**parameter**

IrfGaussian.**parameter**(*global\_index*: *int*, *global\_axis*: *ndarray*) → tuple[*ndarray*, *ndarray*,  
*ndarray*, *float*, *bool*, *float*]

Returns the properties of the irf with shift applied.

**Methods Documentation**

**backsweep**: *bool*

**backsweep\_period**: *ParameterType* | *None*

**calculate**(*index*: *int*, *global\_axis*: *ndarray*, *model\_axis*: *ndarray*) → *ndarray*

**center**: *Parameter* | *str*

**classmethod get\_item\_type**() → *str*

Get the type string.

**Return type**

*str*

**classmethod get\_item\_type\_class**(*item\_type*: *str*) → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

**classmethod get\_item\_types**() → *list*[*str*]

Get all type strings.

**Return type**

*list*[*str*]

**is\_index\_dependent**()

**label**: *str*

**normalize**: *bool*

**parameter**(*global\_index*: *int*, *global\_axis*: *ndarray*) → tuple[*ndarray*, *ndarray*, *ndarray*, *float*,  
*bool*, *float*]

Returns the properties of the irf with shift applied.

**scale**: *list*[*ParameterType*] | *None*

**shift**: *list*[*ParameterType*] | *None*

**type**: *str*

**width**: *Parameter* | *str*

## IrfMultiGaussian

```
class glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian(*, label: str, type:
                                                                    str =
                                                                    'multi-gaussian',
                                                                    center:
                                                                    list[Parameter | str],
                                                                    width:
                                                                    list[Parameter | str],
                                                                    scale: list[Parameter
                                                                    | str] | None = None,
                                                                    shift: list[Parameter
                                                                    | str] | None = None,
                                                                    normalize: bool =
                                                                    True, backsweep:
                                                                    bool = False,
                                                                    backsweep_period:
                                                                    Parameter | str |
                                                                    None = None)
```

Bases: [Irf](#)

Represents a gaussian IRF.

One width and one center is a single gauss.

One center and multiple widths is a multiple gaussian.

Multiple center and multiple widths is Double-, Triple-, etc. Gaussian.

### Parameters

- **label** – label of the irf
- **center** ([list](#)[[glotaran.parameter.parameter.Parameter](#) | [str](#)]) – one or more center of the irf as parameter indices
- **width** ([list](#)[[glotaran.parameter.parameter.Parameter](#) | [str](#)]) – one or more widths of the gaussian as parameter index
- **center\_dispersion\_coefficients** – polynomial coefficients for the dispersion of the center as list of parameter indices. None for no dispersion.
- **width\_dispersion\_coefficients** – polynomial coefficients for the dispersion of the width as parameter indices. None for no dispersion.

Method generated by attrs for class IrfMultiGaussian.



## Attributes Summary

<i>type</i>
<i>center</i>
<i>width</i>
<i>scale</i>
<i>shift</i>
<i>normalize</i>
<i>backsweep</i>
<i>backsweep_period</i>
<i>label</i>

### type

IrfMultiGaussian.**type**: `str`

### center

IrfMultiGaussian.**center**: `list[Parameter | str]`

### width

IrfMultiGaussian.**width**: `list[Parameter | str]`

### scale

IrfMultiGaussian.**scale**: `list[Parameter | str] | None`

### shift

IrfMultiGaussian.**shift**: `list[Parameter | str] | None`

**normalize**

IrfMultiGaussian.**normalize**: `bool`

**backsweep**

IrfMultiGaussian.**backsweep**: `bool`

**backsweep\_period**

IrfMultiGaussian.**backsweep\_period**: `Parameter | str | None`

**label**

IrfMultiGaussian.**label**: `str`

**Methods Summary**

<i>calculate</i>	
<i>get_item_type</i>	Get the type string.
<i>get_item_type_class</i>	Get the type for a type string.
<i>get_item_types</i>	Get all type strings.
<i>is_index_dependent</i>	
<i>parameter</i>	Returns the properties of the irf with shift applied.

**calculate**

IrfMultiGaussian.**calculate**(*index*: `int`, *global\_axis*: `ndarray`, *model\_axis*: `ndarray`) → `ndarray`

**get\_item\_type**

**classmethod** IrfMultiGaussian.**get\_item\_type**() → `str`

Get the type string.

**Return type**

`str`

### get\_item\_type\_class

**classmethod** `IrfMultiGaussian.get_item_type_class(item_type: str) → Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

Type

### get\_item\_types

**classmethod** `IrfMultiGaussian.get_item_types() → list[str]`

Get all type strings.

**Return type**

list[str]

### is\_index\_dependent

`IrfMultiGaussian.is_index_dependent()`

### parameter

`IrfMultiGaussian.parameter(global_index: int, global_axis: ndarray) → tuple[ndarray, ndarray, ndarray, float, bool, float]`

Returns the properties of the irf with shift applied.

## Methods Documentation

**backsweep:** `bool`

**backsweep\_period:** `Parameter | str | None`

**calculate**(*index: int, global\_axis: ndarray, model\_axis: ndarray*) → *ndarray*

**center:** `list[Parameter | str]`

**classmethod** `get_item_type() → str`

Get the type string.

**Return type**

str

**classmethod** `get_item_type_class(item_type: str) → Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

Type

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**is\_index\_dependent()**

**label:** `str`

**normalize:** `bool`

**parameter**(*global\_index: int, global\_axis: ndarray*) → `tuple[ndarray, ndarray, ndarray, float, bool, float]`

Returns the properties of the irf with shift applied.

**scale:** `list[Parameter | str] | None`

**shift:** `list[Parameter | str] | None`

**type:** `str`

**width:** `list[Parameter | str]`

### **IrfSpectralGaussian**

```
class glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian(*, label: str,
                                                                    scale:
                                                                    list[Parameter |
                                                                    str] | None =
                                                                    None, shift:
                                                                    list[Parameter |
                                                                    str] | None =
                                                                    None,
                                                                    normalize: bool
                                                                    = True,
                                                                    backsweep: bool
                                                                    = False, back-
                                                                    sweep_period:
                                                                    Parameter | str |
                                                                    None = None,
                                                                    disper-
                                                                    sion_center:
                                                                    Parameter | str,
                                                                    cen-
                                                                    ter_dispersion_coefficients:
                                                                    list[Parameter |
                                                                    str],
                                                                    width_dispersion_coefficients:
                                                                    list[Parameter |
                                                                    str] = _Noth-
                                                                    ing.NOTHING,
                                                                    model_dispersion_with_wavenumber:
                                                                    bool = False,
                                                                    type: str =
                                                                    'spectral-
                                                                    gaussian',
                                                                    center:
                                                                    Parameter | str,
                                                                    width:
                                                                    Parameter | str)
```

Bases: [\*IrfSpectralMultiGaussian\*](#)

Method generated by attrs for class IrfSpectralGaussian.

### Attributes Summary

<code>type</code>
<code>center</code>
<code>width</code>
<code>dispersion_center</code>
<code>center_dispersion_coefficients</code>
<code>width_dispersion_coefficients</code>
<code>model_dispersion_with_wavenumber</code>
<code>scale</code>
<code>shift</code>
<code>normalize</code>
<code>backsweep</code>
<code>backsweep_period</code>
<code>label</code>

#### type

`IrfSpectralGaussian.type`: `str`

#### center

`IrfSpectralGaussian.center`: `Parameter` | `str`

#### width

`IrfSpectralGaussian.width`: `Parameter` | `str`

**dispersion\_center**

IrfSpectralGaussian.dispersion\_center: ParameterType

**center\_dispersion\_coefficients**

IrfSpectralGaussian.center\_dispersion\_coefficients: list[ParameterType]

**width\_dispersion\_coefficients**

IrfSpectralGaussian.width\_dispersion\_coefficients: list[ParameterType]

**model\_dispersion\_with\_wavenumber**

IrfSpectralGaussian.model\_dispersion\_with\_wavenumber: bool

**scale**

IrfSpectralGaussian.scale: list[ParameterType] | None

**shift**

IrfSpectralGaussian.shift: list[ParameterType] | None

**normalize**

IrfSpectralGaussian.normalize: bool

**backsweep**

IrfSpectralGaussian.backsweep: bool

**backsweep\_period**

IrfSpectralGaussian.backsweep\_period: ParameterType | None

## label

IrfSpectralGaussian.**label**: `str`

## Methods Summary

<code>calculate</code>	
<code>calculate_dispersion</code>	
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.
<code>is_index_dependent</code>	
<code>parameter</code>	Returns the properties of the irf with shift and dispersion applied.

## calculate

IrfSpectralGaussian.**calculate**(*index*: `int`, *global\_axis*: `ndarray`, *model\_axis*: `ndarray`) → `ndarray`

## calculate\_dispersion

IrfSpectralGaussian.**calculate\_dispersion**(*axis*)

## get\_item\_type

**classmethod** IrfSpectralGaussian.**get\_item\_type**() → `str`

Get the type string.

**Return type**

`str`

## get\_item\_type\_class

**classmethod** IrfSpectralGaussian.**get\_item\_type\_class**(*item\_type*: `str`) → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`



## get\_item\_types

**classmethod** `IrfSpectralGaussian.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

## is\_index\_dependent

`IrfSpectralGaussian.is_index_dependent()`

## parameter

`IrfSpectralGaussian.parameter(global_index: int, global_axis: ndarray)`

Returns the properties of the irf with shift and dispersion applied.

## Methods Documentation

**backsweep:** `bool`

**backsweep\_period:** `ParameterType | None`

**calculate**(*index: int, global\_axis: ndarray, model\_axis: ndarray*) → `ndarray`

**calculate\_dispersion**(*axis*)

**center:** `Parameter | str`

**center\_dispersion\_coefficients:** `list[ParameterType]`

**dispersion\_center:** `ParameterType`

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**is\_index\_dependent()**

**label:** `str`

`model_dispersion_with_wavenumber: bool`

`normalize: bool`

`parameter(global_index: int, global_axis: ndarray)`

Returns the properties of the irf with shift and dispersion applied.

`scale: list[ParameterType] | None`

`shift: list[ParameterType] | None`

`type: str`

`width: Parameter | str`

`width_dispersion_coefficients: list[ParameterType]`

### **IrfSpectralMultiGaussian**

```

class glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian(*, label:
    str,
    center:
    list[Parameter
    | str],
    width:
    list[Parameter
    | str],
    scale:
    list[Parameter
    | str] |
    None =
    None,
    shift:
    list[Parameter
    | str] |
    None =
    None,
    normal-
    ize: bool
    = True,
    back-
    sweep:
    bool =
    False,
    back-
    sweep_period:
    Parameter
    | str |
    None =
    None,
    type: str
    =
    'spectral-
    multi-
    gaussian',
    disper-
    sion_center:
    Parameter
    | str, cen-
    ter_dispersion_coefficients:
    list[Parameter
    | str],
    width_dispersion_coefficients:
    list[Parameter
    | str] =
    _Noth-
    ing.NOTHING,
    model_dispersion_with_wavenumb
    bool =
    False)

```

Bases: [IrfMultiGaussian](#)

Represents a gaussian IRF.

One width and one center is a single gauss.

One center and multiple widths is a multiple gaussian.

Multiple center and multiple widths is Double-, Triple-, etc. Gaussian.

#### Parameters

- **label** – label of the irf
- **center** – one or more center of the irf as parameter indices
- **width** – one or more widths of the gaussian as parameter index
- **center\_dispersion\_coefficients** (*list*[*glotaran.parameter.parameter.Parameter* / *str*]) – list of parameters with polynomial coefficients describing the dispersion of the irf center location. None for no dispersion.
- **width\_dispersion\_coefficients** (*list*[*glotaran.parameter.parameter.Parameter* / *str*]) – list of parameters with polynomial coefficients describing the dispersion of the width of the irf. None for no dispersion.

Method generated by attrs for class `IrfSpectralMultiGaussian`.

#### Attributes Summary

<i>type</i>
<i>dispersion_center</i>
<i>center_dispersion_coefficients</i>
<i>width_dispersion_coefficients</i>
<i>model_dispersion_with_wavenumber</i>
<i>center</i>
<i>width</i>
<i>scale</i>
<i>shift</i>
<i>normalize</i>
<i>backsweep</i>
<i>backsweep_period</i>
<i>label</i>

**type**

IrfSpectralMultiGaussian.type: `str`

**dispersion\_center**

IrfSpectralMultiGaussian.dispersion\_center: `Parameter` | `str`

**center\_dispersion\_coefficients**

IrfSpectralMultiGaussian.center\_dispersion\_coefficients: `list[Parameter | str]`

**width\_dispersion\_coefficients**

IrfSpectralMultiGaussian.width\_dispersion\_coefficients: `list[Parameter | str]`

**model\_dispersion\_with\_wavenumber**

IrfSpectralMultiGaussian.model\_dispersion\_with\_wavenumber: `bool`

**center**

IrfSpectralMultiGaussian.center: `list[ParameterType]`

**width**

IrfSpectralMultiGaussian.width: `list[ParameterType]`

**scale**

IrfSpectralMultiGaussian.scale: `list[ParameterType]` | `None`

**shift**

IrfSpectralMultiGaussian.shift: `list[ParameterType]` | `None`

**normalize**

IrfSpectralMultiGaussian.**normalize**: `bool`

**backsweep**

IrfSpectralMultiGaussian.**backsweep**: `bool`

**backsweep\_period**

IrfSpectralMultiGaussian.**backsweep\_period**: `ParameterType` | `None`

**label**

IrfSpectralMultiGaussian.**label**: `str`

**Methods Summary**

<i>calculate</i>	
<i>calculate_dispersion</i>	
<i>get_item_type</i>	Get the type string.
<i>get_item_type_class</i>	Get the type for a type string.
<i>get_item_types</i>	Get all type strings.
<i>is_index_dependent</i>	
<i>parameter</i>	Returns the properties of the irf with shift and dispersion applied.

**calculate**

IrfSpectralMultiGaussian.**calculate**(*index*: `int`, *global\_axis*: `ndarray`, *model\_axis*: `ndarray`) → `ndarray`

**calculate\_dispersion**

IrfSpectralMultiGaussian.**calculate\_dispersion**(*axis*)

### get\_item\_type

**classmethod** `IrfSpectralMultiGaussian.get_item_type()` → `str`

Get the type string.

**Return type**

`str`

### get\_item\_type\_class

**classmethod** `IrfSpectralMultiGaussian.get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

### get\_item\_types

**classmethod** `IrfSpectralMultiGaussian.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

### is\_index\_dependent

`IrfSpectralMultiGaussian.is_index_dependent()`

### parameter

`IrfSpectralMultiGaussian.parameter(global_index: int, global_axis: ndarray)`

Returns the properties of the irf with shift and dispersion applied.

## Methods Documentation

**backsweep:** `bool`

**backsweep\_period:** `ParameterType` | `None`

**calculate**(*index: int, global\_axis: ndarray, model\_axis: ndarray*) → `ndarray`

**calculate\_dispersion**(*axis*)

**center:** `list[ParameterType]`

**center\_dispersion\_coefficients:** `list[Parameter | str]`

**dispersion\_center:** `Parameter` | `str`

**classmethod** `get_item_type()`  $\rightarrow$  `str`  
Get the type string.  
**Return type**  
`str`

**classmethod** `get_item_type_class(item_type: str)`  $\rightarrow$  `Type`  
Get the type for a type string.  
**Parameters**  
`item_type` (*`str`*) – The type string.  
**Return type**  
`Type`

**classmethod** `get_item_types()`  $\rightarrow$  `list[str]`  
Get all type strings.  
**Return type**  
`list[str]`

**is\_index\_dependent()**

**label:** `str`

**model\_dispersion\_with\_wavenumber:** `bool`

**normalize:** `bool`

**parameter**(*global\_index: `int`, global\_axis: `ndarray`*)  
Returns the properties of the irf with shift and dispersion applied.

**scale:** `list[ParameterType] | None`

**shift:** `list[ParameterType] | None`

**type:** `str`

**width:** `list[ParameterType]`

**width\_dispersion\_coefficients:** `list[Parameter | str]`

## `k_matrix`

K-Matrix

## Functions

### Summary

`calculate_gamma`



## calculate\_gamma

glotaran.builtin.megacomplexes.decay.k\_matrix.calculate\_gamma(*eigenvectors*: *ndarray*,  
*initial\_concentration*:  
*ndarray*) → *ndarray*

## Classes

### Summary

<i>KMatrix</i>	A K-Matrix represents a first order differential system.
----------------	--

### KMatrix

**class** glotaran.builtin.megacomplexes.decay.k\_matrix.**KMatrix**(\*, *label*: *str*, *matrix*:  
*dict[tuple[str, str],*  
*Parameter | str]*)

Bases: `ModelItem`

A K-Matrix represents a first order differential system.

Method generated by attrs for class KMatrix.

### Attributes Summary

<i>matrix</i>
<i>label</i>

### matrix

**KMatrix.matrix:** *dict[tuple[str, str], Parameter | str]*

### label

**KMatrix.label:** *str*

## Methods Summary

<code>a_matrix</code>	The A matrix of the KMatrix.
<code>a_matrix_as_markdown</code>	Returns the A Matrix as markdown formatted table.
<code>a_matrix_general</code>	The A matrix of the KMatrix for a general model.
<code>a_matrix_sequential</code>	The A matrix of the KMatrix for a sequential model.
<code>combine</code>	Creates a combined matrix.
<code>eigen</code>	Returns the eigenvalues and eigenvectors of the k matrix.
<code>empty</code>	Creates an empty K-Matrix.
<code>full</code>	The full representation of the KMatrix as numpy array.
<code>involved_compartments</code>	A list of all compartments in the Matrix.
<code>is_sequential</code>	Returns true in the KMatrix represents an uni-branched model.
<code>matrix_as_markdown</code>	Returns the KMatrix as markdown formatted table.
<code>rates</code>	The resulting rates of the matrix.
<code>reduced</code>	The reduced representation of the KMatrix as numpy array.

### `a_matrix`

`KMatrix.a_matrix(compartments: list[str], initial_concentration: ndarray) → ndarray`

The A matrix of the KMatrix.

#### Parameters

**initial\_concentration** – The initial concentration.

### `a_matrix_as_markdown`

`KMatrix.a_matrix_as_markdown(compartments: list[str], initial_concentration: ndarray) → MarkdownStr`

Returns the A Matrix as markdown formatted table.

#### Parameters

**initial\_concentration** – The initial concentration.

### `a_matrix_general`

`KMatrix.a_matrix_general(compartments: list[str], initial_concentration: ndarray) → ndarray`

The A matrix of the KMatrix for a general model.

#### Parameters

**initial\_concentration** – The initial concentration.

### `a_matrix_sequential`

`KMatrix.a_matrix_sequential(compartments: list[str]) → ndarray`

The A matrix of the KMatrix for a sequential model.

**Parameters**

**initial\_concentration** – The initial concentration.

### `combine`

`KMatrix.combine(k_matrix: KMatrix) → KMatrix`

Creates a combined matrix.

When combining k-matrices km1 and km2 (km1.combine(km2)), entries in km1 will be overwritten by corresponding entries in km2.

**Parameters**

**k\_matrix** – KMatrix to combine with.

**Returns**

The combined KMatrix.

**Return type**

combined

### `eigen`

`KMatrix.eigen(compartments: list[str]) → tuple[ndarray, ndarray]`

Returns the eigenvalues and eigenvectors of the k matrix.

**Parameters**

**compartments** – The compartment order.

### `empty`

**classmethod** `KMatrix.empty(label: str, compartments: list[str]) → KMatrix`

Creates an empty K-Matrix. Useful for combining.

**Parameters**

- **label** – Label of the K-Matrix
- **compartments** – A list of all compartments in the model.

### `full`

`KMatrix.full(compartments: list[str]) → ndarray`

The full representation of the KMatrix as numpy array.

**Parameters**

**compartments** – The compartment order.

### involved\_compartments

`KMatrix.involved_compartments()` → `list[str]`

A list of all compartments in the Matrix.

### is\_sequential

`KMatrix.is_sequential(compartments: list[str], initial_concentration: ndarray)` → `bool`

Returns true in the KMatrix represents an unbranched model.

**Parameters**

**initial\_concentration** – The initial concentration.

### matrix\_as\_markdown

`KMatrix.matrix_as_markdown(compartments: list[str] | None = None, fill_parameters: bool = False)` → `MarkdownStr`

Returns the KMatrix as markdown formatted table.

**Parameters**

- **compartments** – (default = None) An optional list defining the desired order of compartments.
- **fill\_parameters** (`bool`) – (default = False) If true, the entries will be filled with the actual parameter values instead of labels.

### rates

`KMatrix.rates(compartments: list[str], initial_concentration: ndarray)` → `ndarray`

The resulting rates of the matrix.

By definition, the eigenvalues of the compartmental model are negative and the rates are the negatives of the eigenvalues, thus the eigenvalues need to be multiplied with -1 to get rates with the correct sign.

**Parameters**

- **compartments** (`list[str]`) – Names of compartment used to order the matrix.
- **initial\_concentration** (`np.ndarray`) – The initial concentration.

### reduced

`KMatrix.reduced(compartments: list[str])` → `ndarray`

The reduced representation of the KMatrix as numpy array.

**Parameters**

**compartments** – The compartment order.

## Methods Documentation

**a\_matrix**(*compartments*: *list[str]*, *initial\_concentration*: *ndarray*) → *ndarray*

The A matrix of the KMatrix.

**Parameters**

**initial\_concentration** – The initial concentration.

**a\_matrix\_as\_markdown**(*compartments*: *list[str]*, *initial\_concentration*: *ndarray*) → *MarkdownStr*

Returns the A Matrix as markdown formatted table.

**Parameters**

**initial\_concentration** – The initial concentration.

**a\_matrix\_general**(*compartments*: *list[str]*, *initial\_concentration*: *ndarray*) → *ndarray*

The A matrix of the KMatrix for a general model.

**Parameters**

**initial\_concentration** – The initial concentration.

**a\_matrix\_sequential**(*compartments*: *list[str]*) → *ndarray*

The A matrix of the KMatrix for a sequential model.

**Parameters**

**initial\_concentration** – The initial concentration.

**combine**(*k\_matrix*: *KMatrix*) → *KMatrix*

Creates a combined matrix.

When combining k-matrices km1 and km2 (km1.combine(km2)), entries in km1 will be overwritten by corresponding entries in km2.

**Parameters**

**k\_matrix** – KMatrix to combine with.

**Returns**

The combined KMatrix.

**Return type**

combined

**eigen**(*compartments*: *list[str]*) → *tuple[ndarray, ndarray]*

Returns the eigenvalues and eigenvectors of the k matrix.

**Parameters**

**compartments** – The compartment order.

**classmethod empty**(*label*: *str*, *compartments*: *list[str]*) → *KMatrix*

Creates an empty K-Matrix. Useful for combining.

**Parameters**

- **label** – Label of the K-Matrix
- **compartments** – A list of all compartments in the model.

**full**(*compartments*: *list[str]*) → *ndarray*

The full representation of the KMatrix as numpy array.

**Parameters**

**compartments** – The compartment order.

**involved\_compartments**() → *list[str]*

A list of all compartments in the Matrix.

**is\_sequential**(*compartments*: *list[str]*, *initial\_concentration*: *ndarray*) → *bool*

Returns true in the KMatrix represents an unbranched model.

**Parameters**

**initial\_concentration** – The initial concentration.

**label:** `str`

**matrix:** `dict[tuple[str, str], Parameter | str]`

**matrix\_as\_markdown**(*compartments: list[str] | None = None, fill\_parameters: bool = False*) → *MarkdownStr*

Returns the KMatrix as markdown formatted table.

**Parameters**

- **compartments** – (default = None) An optional list defining the desired order of compartments.
- **fill\_parameters** (*bool*) – (default = False) If true, the entries will be filled with the actual parameter values instead of labels.

**rates**(*compartments: list[str], initial\_concentration: ndarray*) → *ndarray*

The resulting rates of the matrix.

By definition, the eigenvalues of the compartmental model are negative and the rates are the negatives of the eigenvalues, thus the eigenvalues need to be multiplied with -1 to get rates with the correct sign.

**Parameters**

- **compartments** (*list[str]*) – Names of compartment used to order the matrix.
- **initial\_concentration** (*np.ndarray*) – The initial concentration.

**reduced**(*compartments: list[str]*) → *ndarray*

The reduced representation of the KMatrix as numpy array.

**Parameters**

**compartments** – The compartment order.

**util****Functions**

## Summary

<code>calculate_decay_matrix_no_irf</code>	
<code>calculate_matrix</code>	
<code>collect_megacomplexes</code>	
<code>decay_matrix_implementation_index_deper</code>	
<code>decay_matrix_implementation_index_inde</code>	
<code>finalize_data</code>	
<code>index_dependent</code>	Determine if a dataset_model is index dependent.
<code>retrieve_decay_associated_data</code>	
<code>retrieve_initial_concentration</code>	
<code>retrieve_irf</code>	
<code>retrieve_species_associated_data</code>	

### calculate\_decay\_matrix\_no\_irf

`glotaran.builtin.megacomplexes.decay.util.calculate_decay_matrix_no_irf`(*matrix*,  
*rates*,  
*times*)

### calculate\_matrix

`glotaran.builtin.megacomplexes.decay.util.calculate_matrix`(*megacomplex*:  
*Megacomplex*,  
*dataset\_model*:  
[DatasetModel](#),  
*global\_axis*: *ArrayLike*,  
*model\_axis*: *ArrayLike*,  
*\*\*kwargs*)

### collect\_megacomplexes

```
glotaran.builtin.megacomplexes.decay.util.collect_megacomplexes(dataset_model:
                                                                    DatasetModel,
                                                                    as_global: bool) →
                                                                    list[Megacomplex]
```

### decay\_matrix\_implementation\_index\_dependent

```
glotaran.builtin.megacomplexes.decay.util.decay_matrix_implementation_index_dependent(matrix:
                                                                                          ndarray,
                                                                                          rates:
                                                                                          ndarray,
                                                                                          global_axis:
                                                                                          ndarray,
                                                                                          model_axis:
                                                                                          ndarray,
                                                                                          dataset_model:
                                                                                          Dataset-
                                                                                          Model)
```

### decay\_matrix\_implementation\_index\_independent

```
glotaran.builtin.megacomplexes.decay.util.decay_matrix_implementation_index_independent(matrix:
                                                                                          ndarray,
                                                                                          rates:
                                                                                          ndarray,
                                                                                          global_axis:
                                                                                          ndarray,
                                                                                          model_axis:
                                                                                          ndarray,
                                                                                          dataset_model:
                                                                                          Dataset-
                                                                                          Model)
```



## finalize\_data

```
glotaran.builtin.megacomplexes.decay.util.finalize_data(dataset_model: DatasetModel,
                                                         dataset: Dataset,
                                                         is_full_model: bool = False,
                                                         as_global: bool = False)
```

## index\_dependent

```
glotaran.builtin.megacomplexes.decay.util.index_dependent(dataset_model:
                                                           DatasetModel) → bool
```

Determine if a dataset\_model is index dependent.

### Parameters

**dataset\_model** ([DatasetModel](#)) – A dataset model instance.

### Returns

Returns True if the dataset\_model has an IRF that is index dependent (e.g. has dispersion).

### Return type

[bool](#)

## retrieve\_decay\_associated\_data

```
glotaran.builtin.megacomplexes.decay.util.retrieve_decay_associated_data(megacomplex:
                                                                            Mega-
                                                                            complex,
                                                                            dataset_model:
                                                                            Dataset-
                                                                            Model,
                                                                            dataset:
                                                                            Dataset,
                                                                            global_dimension:
                                                                            str,
                                                                            name:
                                                                            str)
```

## retrieve\_initial\_concentration

```
glotaran.builtin.megacomplexes.decay.util.retrieve_initial_concentration(dataset_model:
                                                                            Dataset-
                                                                            Model,
                                                                            dataset:
                                                                            Dataset,
                                                                            species_dimension:
                                                                            str)
```

### retrieve\_irf

```
glotaran.builtin.megacomplexes.decay.util.retrieve_irf(dataset_model: DatasetModel,  
                                                       dataset: Dataset,  
                                                       global_dimension: str)
```

### retrieve\_species\_associated\_data

```
glotaran.builtin.megacomplexes.decay.util.retrieve_species_associated_data(dataset_model:  
                                                                              Dataset-  
                                                                              Model,  
                                                                              dataset:  
                                                                              Dataset,  
                                                                              species:  
                                                                              list[str],  
                                                                              species_dimension:  
                                                                              str,  
                                                                              global_dimension:  
                                                                              str,  
                                                                              name:  
                                                                              str,  
                                                                              is_full_model:  
                                                                              bool,  
                                                                              as_global:  
                                                                              bool)
```

## spectral

### Modules

<code>glotaran.builtin.megacomplexes.spectral. shape</code>	This package contains the spectral shape item.
<code>glotaran.builtin.megacomplexes.spectral. spectral_megacomplex</code>	

### shape

This package contains the spectral shape item.

## Classes

### Summary

<i>SpectralShape</i>	Method generated by attrs for class SpectralShape.
<i>SpectralShapeGaussian</i>	A Gaussian spectral shape
<i>SpectralShapeOne</i>	A constant spectral shape with value 1
<i>SpectralShapeSkewedGaussian</i>	A skewed Gaussian spectral shape
<i>SpectralShapeZero</i>	A constant spectral shape with value 0

### SpectralShape

**class** glotaran.builtin.megacomplexes.spectral.shape.**SpectralShape**(\*, *label*: *str*, *type*: *str*)

Bases: `ModelItemTyped`

Method generated by attrs for class SpectralShape.

### Attributes Summary

<i>type</i>
<i>label</i>

### type

`SpectralShape.type: str`

### label

`SpectralShape.label: str`

### Methods Summary

<i>get_item_type</i>	Get the type string.
<i>get_item_type_class</i>	Get the type for a type string.
<i>get_item_types</i>	Get all type strings.

### get\_item\_type

**classmethod** SpectralShape.get\_item\_type() → str

Get the type string.

**Return type**

str

### get\_item\_type\_class

**classmethod** SpectralShape.get\_item\_type\_class(item\_type: str) → Type

Get the type for a type string.

**Parameters**

**item\_type** (str) – The type string.

**Return type**

Type

### get\_item\_types

**classmethod** SpectralShape.get\_item\_types() → list[str]

Get all type strings.

**Return type**

list[str]

## Methods Documentation

**classmethod** get\_item\_type() → str

Get the type string.

**Return type**

str

**classmethod** get\_item\_type\_class(item\_type: str) → Type

Get the type for a type string.

**Parameters**

**item\_type** (str) – The type string.

**Return type**

Type

**classmethod** get\_item\_types() → list[str]

Get all type strings.

**Return type**

list[str]

**label:** str

**type:** str

### SpectralShapeGaussian

```
class glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian(*,
                                                                            label:
                                                                              str,
                                                                            type:
                                                                              str =
                                                                              'gaus-
                                                                              sian',
                                                                            ampli-
                                                                              tude:
                                                                              Param-
                                                                              eter |
                                                                              str |
                                                                              None =
                                                                              None,
                                                                            loca-
                                                                              tion:
                                                                              Param-
                                                                              eter |
                                                                              str,
                                                                            width:
                                                                              Param-
                                                                              eter |
                                                                              str)
```

Bases: *SpectralShape*

A Gaussian spectral shape

Method generated by attrs for class SpectralShapeGaussian.

### Attributes Summary

<i>type</i>
<i>amplitude</i>
<i>location</i>
<i>width</i>
<i>label</i>

**type**

SpectralShapeGaussian.**type**: `str`

**amplitude**

SpectralShapeGaussian.**amplitude**: `Parameter` | `str` | `None`

**location**

SpectralShapeGaussian.**location**: `Parameter` | `str`

**width**

SpectralShapeGaussian.**width**: `Parameter` | `str`

**label**

SpectralShapeGaussian.**label**: `str`

**Methods Summary**

<code>calculate</code>	Calculate a normal Gaussian shape for a given axis.
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

**calculate**

SpectralShapeGaussian.**calculate**(axis: `ndarray`) → `ndarray`

Calculate a normal Gaussian shape for a given axis.

The following equation is used for the calculation:

$$f(x, A, x_0, \Delta) = A \exp \left( -\frac{\log(2) (2(x - x_0))^2}{\Delta^2} \right)$$

The parameters of the equation represent the following attributes of the shape:

- `x`: axis
- `A`: amplitude
- `x0`: location
- `Δ`: width

In this formalism,  $\Delta$  represents the full width at half maximum (FWHM). Compared to the more common definition  $\exp(-(x - \mu)^2/(2\sigma^2))$  we have  $\sigma = \Delta/(2\sqrt{2 \ln(2)}) = \Delta/2.35482$

**Parameters**

**axis** (*np.ndarray*) – The axis to calculate the shape for.

**Returns**

An array representing a Gaussian shape.

**Return type**

*np.ndarray*

**get\_item\_type**

**classmethod** *SpectralShapeGaussian.get\_item\_type()* → *str*

Get the type string.

**Return type**

*str*

**get\_item\_type\_class**

**classmethod** *SpectralShapeGaussian.get\_item\_type\_class(item\_type: str)* → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

**get\_item\_types**

**classmethod** *SpectralShapeGaussian.get\_item\_types()* → *list[str]*

Get all type strings.

**Return type**

*list[str]*

**Methods Documentation**

**amplitude:** *Parameter* | *str* | *None*

**calculate**(*axis: ndarray*) → *ndarray*

Calculate a normal Gaussian shape for a given axis.

The following equation is used for the calculation:

$$f(x, A, x_0, \Delta) = A \exp \left( -\frac{\log(2) (2(x - x_0))^2}{\Delta^2} \right)$$

The parameters of the equation represent the following attributes of the shape:

- *x* : axis
- *A* : amplitude
- *x*<sub>0</sub> : location
- $\Delta$  : width

In this formalism,  $\Delta$  represents the full width at half maximum (FWHM). Compared to the more common definition  $\exp(-(x - \mu)^2/(2\sigma^2))$  we have  $\sigma = \Delta/(2\sqrt{2 \ln(2)}) = \Delta/2.35482$

**Parameters**

**axis** (*np.ndarray*) – The axis to calculate the shape for.

**Returns**

An array representing a Gaussian shape.

**Return type**

*np.ndarray*

**classmethod** `get_item_type()` → *str*

Get the type string.

**Return type**

*str*

**classmethod** `get_item_type_class(item_type: str)` → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

**classmethod** `get_item_types()` → *list[str]*

Get all type strings.

**Return type**

*list[str]*

**label:** *str*

**location:** *Parameter* | *str*

**type:** *str*

**width:** *Parameter* | *str*

## SpectralShapeOne

```
class glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeOne(*, label: str,  
                                                                    type: str =  
                                                                    'one')
```

Bases: *SpectralShape*

A constant spectral shape with value 1

Method generated by attrs for class SpectralShapeOne.

### Attributes Summary

<i>type</i>
<i>label</i>



**type**

SpectralShapeOne.**type**: `str`

**label**

SpectralShapeOne.**label**: `str`

**Methods Summary**

<code>calculate</code>	calculate calculates the shape.
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

**calculate**

SpectralShapeOne.**calculate**(*axis*: `ndarray`) → `ndarray`

calculate calculates the shape.

**Parameters**

**axis** (`np.ndarray`) – The axis to calculate the shape on.

**Returns**

shape

**Return type**

`numpy.ndarray`

**get\_item\_type**

**classmethod** SpectralShapeOne.**get\_item\_type**() → `str`

Get the type string.

**Return type**

`str`

**get\_item\_type\_class**

**classmethod** SpectralShapeOne.**get\_item\_type\_class**(*item\_type*: `str`) → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

### `get_item_types`

**classmethod** `SpectralShapeOne.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

## Methods Documentation

**calculate**(*axis*: `ndarray`) → `ndarray`

calculate calculates the shape.

**Parameters**

**axis** (`np.ndarray`) – The axis to calculate the shape on.

**Returns**

shape

**Return type**

`numpy.ndarray`

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**label**: `str`

**type**: `str`

## SpectralShapeSkewedGaussian

```
class glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeSkewedGaussian(*,
label:
str,
am-
pli-
tude:
Pa-
ram-
e-
ter
|
str
|
None
=
None,
lo-
ca-
tion:
Pa-
ram-
e-
ter
|
str,
width:
Pa-
ram-
e-
ter
|
str,
type:
str
=
'skewed-
gaussian',
skew-
ness:
Pa-
ram-
e-
ter
|
str)
```

Bases: *SpectralShapeGaussian*

A skewed Gaussian spectral shape

Method generated by attrs for class SpectralShapeSkewedGaussian.

### Attributes Summary

<i>type</i>
<i>skewness</i>
<i>amplitude</i>
<i>location</i>
<i>width</i>
<i>label</i>

#### type

SpectralShapeSkewedGaussian.**type**: `str`

#### skewness

SpectralShapeSkewedGaussian.**skewness**: `Parameter` | `str`

#### amplitude

SpectralShapeSkewedGaussian.**amplitude**: `ParameterType` | `None`

#### location

SpectralShapeSkewedGaussian.**location**: `ParameterType`

#### width

SpectralShapeSkewedGaussian.**width**: `ParameterType`

#### label

SpectralShapeSkewedGaussian.**label**: `str`

## Methods Summary

<code>calculate</code>	Calculate the skewed Gaussian shape for <code>axis</code> .
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

### calculate

`SpectralShapeSkewedGaussian.calculate(axis: ndarray) → ndarray`

Calculate the skewed Gaussian shape for `axis`.

The following equation is used for the calculation:

$$f(x, x_0, A, \Delta, b) = \begin{cases} 0 & \text{if } \theta \leq 0 \\ A \exp\left(-\frac{\log(2) \log(\theta(x, x_0, \Delta, b))^2}{b^2}\right) & \text{if } \theta > 0 \end{cases}$$

With:

$$\theta(x, x_0, \Delta, b) = \frac{2b(x - x_0) + \Delta}{\Delta}$$

The parameters of the equation represent the following attributes of the shape:

- `x`: axis
- `A`: amplitude
- `x0`: location
- `Δ`: width
- `b`: skewness

Where `Δ` represents the full width at half maximum (FWHM), see `calculate_gaussian()`.

Note that in the limit of skewness parameter `b` equal to zero  $f(x, x_0, A, \Delta, b)$  simplifies to a normal gaussian (since  $\lim_{b \rightarrow 0} \frac{\ln(1+bx)}{b} = x$ ), see the definition in [SpectralShapeGaussian.calculate\(\)](#).

#### Parameters

**axis** (`np.ndarray`) – The axis to calculate the shape for.

#### Returns

An array representing a skewed Gaussian shape.

#### Return type

`np.ndarray`

### get\_item\_type

`classmethod SpectralShapeSkewedGaussian.get_item_type() → str`

Get the type string.

#### Return type

`str`

### get\_item\_type\_class

**classmethod** SpectralShapeSkewedGaussian.**get\_item\_type\_class**(*item\_type*: *str*) → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

### get\_item\_types

**classmethod** SpectralShapeSkewedGaussian.**get\_item\_types**() → *list[str]*

Get all type strings.

**Return type**

*list[str]*

## Methods Documentation

**amplitude**: *ParameterType* | **None**

**calculate**(*axis*: *ndarray*) → *ndarray*

Calculate the skewed Gaussian shape for axis.

The following equation is used for the calculation:

$$f(x, x_0, A, \Delta, b) = \begin{cases} 0 & \text{if } \theta \leq 0 \\ A \exp\left(-\frac{\log(2) \log(\theta(x, x_0, \Delta, b))^2}{b^2}\right) & \text{if } \theta > 0 \end{cases}$$

With:

$$\theta(x, x_0, \Delta, b) = \frac{2b(x - x_0) + \Delta}{\Delta}$$

The parameters of the equation represent the following attributes of the shape:

- *x*: axis
- *A*: amplitude
- *x*<sub>0</sub>: location
- $\Delta$ : width
- *b*: skewness

Where  $\Delta$  represents the full width at half maximum (FWHM), see `calculate_gaussian()`.

Note that in the limit of skewness parameter *b* equal to zero  $f(x, x_0, A, \Delta, b)$  simplifies to a normal gaussian (since  $\lim_{b \rightarrow 0} \frac{\ln(1+bx)}{b} = x$ ), see the definition in [SpectralShapeGaussian.calculate\(\)](#).

**Parameters**

**axis** (*np.ndarray*) – The axis to calculate the shape for.

**Returns**

An array representing a skewed Gaussian shape.

**Return type**

*np.ndarray*

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**label:** `str`

**location:** `ParameterType`

**skewness:** `Parameter` | `str`

**type:** `str`

**width:** `ParameterType`

## SpectralShapeZero

```
class glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeZero(*, label: str,
                                                                    type: str =
                                                                    'zero')
```

Bases: `SpectralShape`

A constant spectral shape with value 0

Method generated by attrs for class SpectralShapeZero.

## Attributes Summary

*type*

*label*

**type**

SpectralShapeZero.type: `str`

**label**

SpectralShapeZero.label: `str`

**Methods Summary**

<code>calculate</code>	calculate calculates the shape.
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

**calculate**

SpectralShapeZero.calculate(*axis*: `ndarray`) → `ndarray`

calculate calculates the shape.

Only works after calling fill.

**Parameters**

**axis** (`np.ndarray`) – The axis to calculate the shape on.

**Returns**

**shape**

**Return type**

`numpy.ndarray`

**get\_item\_type**

classmethod SpectralShapeZero.get\_item\_type() → `str`

Get the type string.

**Return type**

`str`

**get\_item\_type\_class**

classmethod SpectralShapeZero.get\_item\_type\_class(*item\_type*: `str`) → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`



## get\_item\_types

**classmethod** `SpectralShapeZero.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

## Methods Documentation

**calculate**(*axis*: `ndarray`) → `ndarray`

calculate calculates the shape.

Only works after calling fill.

**Parameters**

**axis** (`np.ndarray`) – The axis to calculate the shape on.

**Returns**

`shape`

**Return type**

`numpy.ndarray`

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**label**: `str`

**type**: `str`

## spectral\_megacomplex

### Classes

#### Summary

<code>SpectralDatasetModel</code>	Method generated by attrs for class <code>SpectralDatasetModel</code> .
<code>SpectralMegacomplex</code>	Method generated by attrs for class <code>SpectralMegacomplex</code> .

## SpectralDatasetModel

```
class glotaran.builtin.megacomplexes.spectral.spectral_megacomplex.SpectralDatasetModel(*,
label:
str,
group:
str
=
'de-
fault',
force_index_a
bool
=
False,
mega-
com-
plex:
list[Mega-com-
plex]
|
str],
mega-
com-
plex_scale:
list[Parameter]
|
str]
|
None
=
None,
global_mega-com-
plex_scale:
list[Parameter]
|
str]
|
None
=
None,
global_mega-com-
plex_scale:
list[Parameter]
|
str]
|
None
=
None,
scale:
Pa-
ram-
eter
|
str
|
None
=
None,
spec-
tral_axis_inve
bool
=
```

Bases: *DatasetModel*

Method generated by attrs for class SpectralDatasetModel.

### Attributes Summary

<i>spectral_axis_inverted</i>
<i>spectral_axis_scale</i>
<i>group</i>
<i>force_index_dependent</i>
<i>megacomplex</i>
<i>megacomplex_scale</i>
<i>global_megacomplex</i>
<i>global_megacomplex_scale</i>
<i>scale</i>
<i>label</i>

#### **spectral\_axis\_inverted**

`SpectralDatasetModel.spectral_axis_inverted: bool`

#### **spectral\_axis\_scale**

`SpectralDatasetModel.spectral_axis_scale: float`

#### **group**

`SpectralDatasetModel.group: str`

**force\_index\_dependent**

SpectralDatasetModel.force\_index\_dependent: `bool`

**megacomplex**

SpectralDatasetModel.megacomplex: `list[ModelItemType[Megacomplex]]`

**megacomplex\_scale**

SpectralDatasetModel.megacomplex\_scale: `list[ParameterType] | None`

**global\_megacomplex**

SpectralDatasetModel.global\_megacomplex: `list[ModelItemType[Megacomplex]] | None`

**global\_megacomplex\_scale**

SpectralDatasetModel.global\_megacomplex\_scale: `list[ParameterType] | None`

**scale**

SpectralDatasetModel.scale: `ParameterType | None`

**label**

SpectralDatasetModel.label: `str`

**Methods Summary****Methods Documentation**

**force\_index\_dependent:** `bool`

**global\_megacomplex:** `list[ModelItemType[Megacomplex]] | None`

**global\_megacomplex\_scale:** `list[ParameterType] | None`

**group:** `str`

**label:** `str`

```
megacomplex: list[ModelItemType[Megacomplex]]
megacomplex_scale: list[ParameterType] | None
scale: ParameterType | None
spectral_axis_inverted: bool
spectral_axis_scale: float
```

### SpectralMegacomplex

```
class glotaran.builtin.megacomplexes.spectral.spectral_megacomplex.SpectralMegacomplex(*,
                                                                                          la-
                                                                                          bel:
                                                                                          str,
                                                                                          di-
                                                                                          men-
                                                                                          sion:
                                                                                          str
                                                                                          =
                                                                                          'spec-
                                                                                          tral',
                                                                                          type:
                                                                                          str
                                                                                          =
                                                                                          'spec-
                                                                                          tral',
                                                                                          shape:
                                                                                          dict[str,
                                                                                          Spec-
                                                                                          tral-
                                                                                          Shape
                                                                                          |
                                                                                          str])
```

Bases: `Megacomplex`

Method generated by attrs for class `SpectralMegacomplex`.

### Attributes Summary

<i>dimension</i>
<i>type</i>
<i>shape</i>
<i>label</i>

**dimension**

`SpectralMegacomplex.dimension`: `str`

**type**

`SpectralMegacomplex.type`: `str`

**shape**

`SpectralMegacomplex.shape`: `dict[str, ModelItemType[SpectralShape]]`

**label**

`SpectralMegacomplex.label`: `str`

**Methods Summary**

<code>calculate_matrix</code>	Calculate the megacomplex matrix.
<code>finalize_data</code>	Finalize a dataset.
<code>get_dataset_model_type</code>	Get the dataset model type.
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

**calculate\_matrix**

`SpectralMegacomplex.calculate_matrix`(*dataset\_model*: [DatasetModel](#), *global\_axis*: *ArrayLike*, *model\_axis*: *ArrayLike*, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** ([DatasetModel](#)) – The dataset model.
- **global\_axis** (*ArrayLike*) – The global axis.
- **model\_axis** (*ArrayLike*) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple*[*list*[*str*], *ArrayLike*] – The clp labels and the matrix.
- .. # noqa (*DAR202*)
- .. # noqa (*DAR401*)

### finalize\_data

`SpectralMegacomplex.finalize_data(dataset_model: DatasetModel, dataset: Dataset, is_full_model: bool = False, as_global: bool = False)`

Finalize a dataset.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **dataset** (`xr.Dataset`) – The dataset.
- **is\_full\_model** (`bool`) – Whether the model is a full model.
- **as\_global** (`bool`) – Whether megacomplex is calculated as global megacomplex.

### get\_dataset\_model\_type

`classmethod SpectralMegacomplex.get_dataset_model_type() → type | None`

Get the dataset model type.

#### Return type

`type | None`

### get\_item\_type

`classmethod SpectralMegacomplex.get_item_type() → str`

Get the type string.

#### Return type

`str`

### get\_item\_type\_class

`classmethod SpectralMegacomplex.get_item_type_class(item_type: str) → Type`

Get the type for a type string.

#### Parameters

- **item\_type** (`str`) – The type string.

#### Return type

`Type`

### get\_item\_types

`classmethod SpectralMegacomplex.get_item_types() → list[str]`

Get all type strings.

#### Return type

`list[str]`



## Methods Documentation

**calculate\_matrix**(*dataset\_model*: DatasetModel, *global\_axis*: ArrayLike, *model\_axis*: ArrayLike, *\*\*kwargs*)

Calculate the megacomplex matrix.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **global\_axis** (ArrayLike) – The global axis.
- **model\_axis** (ArrayLike) – The model axis.
- **\*\*kwargs** – Additional arguments.

**Returns**

- *tuple*[*list*[*str*], ArrayLike] – The clp labels and the matrix.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**dimension**: str

**finalize\_data**(*dataset\_model*: DatasetModel, *dataset*: Dataset, *is\_full\_model*: bool = False, *as\_global*: bool = False)

Finalize a dataset.

**Parameters**

- **dataset\_model** (DatasetModel) – The dataset model.
- **dataset** (xr.Dataset) – The dataset.
- **is\_full\_model** (bool) – Whether the model is a full model.
- **as\_global** (bool) – Whether megacomplex is calculated as global megacomplex.

**classmethod get\_dataset\_model\_type**() → type | None

Get the dataset model type.

**Return type**

type | None

**classmethod get\_item\_type**() → str

Get the type string.

**Return type**

str

**classmethod get\_item\_type\_class**(*item\_type*: str) → Type

Get the type for a type string.

**Parameters**

**item\_type** (str) – The type string.

**Return type**

Type

**classmethod get\_item\_types**() → list[str]

Get all type strings.

**Return type**

list[str]

**label**: str

**shape**: dict[str, ModelItemType[SpectralShape]]

**type**: str

### 16.1.3 cli

#### Modules

<code>glotaran.cli.commands</code>	
<code>glotaran.cli.main</code>	The glotaran CLI main function.

#### commands

#### Modules

<code>glotaran.cli.commands.explore</code>
<code>glotaran.cli.commands.export</code>
<code>glotaran.cli.commands.optimize</code>
<code>glotaran.cli.commands.pluginlist</code>
<code>glotaran.cli.commands.print</code>
<code>glotaran.cli.commands.util</code>
<code>glotaran.cli.commands.validate</code>

#### explore

#### Functions

##### Summary

<code>export</code>	Exports data from netCDF4 to ascii.
---------------------	-------------------------------------

##### export

`glotaran.cli.commands.explore.export(filename: str, select, out: str, name: str)`

Exports data from netCDF4 to ascii.

## export

## optimize

### Functions

#### Summary

<code>optimize_cmd</code>	Optimizes a model.
---------------------------	--------------------

#### optimize\_cmd

```
glotaran.cli.commands.optimize.optimize_cmd(dataformat: str, data: list[str], out: str,
                                             outformat: str, nfev: int, nnls: bool, yes: bool,
                                             parameters_file: str, model_file: str,
                                             scheme_file: str)
```

Optimizes a model. e.g.: `glotaran optimize -`

## pluginlist

### Functions

#### Summary

<code>plugin_list_cmd</code>	Prints a list of installed plugins.
------------------------------	-------------------------------------

#### plugin\_list\_cmd

```
glotaran.cli.commands.pluginlist.plugin_list_cmd()
```

Prints a list of installed plugins.

## print

### Functions

#### Summary

<code>print_cmd</code>	Parses scheme, a model or a parameter file and prints the result as a Markdown formatted string.
------------------------	--

## print\_cmd

`glotaran.cli.commands.print.print_cmd(parameters_file: str, model_file: str, scheme_file: str)`

Parses scheme, a model or a parameter file and prints the result as a Markdown formatted string.

## util

### Functions

#### Summary

<code>load_dataset_file</code>	
<code>load_model_file</code>	
<code>load_parameter_file</code>	
<code>load_scheme_file</code>	
<code>project_io_list_supporting_plugins</code>	List all project-io plugin that implement method_name.
<code>select_data</code>	
<code>select_name</code>	
<code>signature_analysis</code>	
<code>write_data</code>	

#### load\_dataset\_file

`glotaran.cli.commands.util.load_dataset_file(filename, fmt=None, verbose=False)`

#### load\_model\_file

`glotaran.cli.commands.util.load_model_file(filename, verbose=False)`

### load\_parameter\_file

glotaran.cli.commands.util.**load\_parameter\_file**(filename, fmt=None, verbose=False)

### load\_scheme\_file

glotaran.cli.commands.util.**load\_scheme\_file**(filename, verbose=False)

### project\_io\_list\_supporting\_plugins

glotaran.cli.commands.util.**project\_io\_list\_supporting\_plugins**(method\_name: *str*,  
block\_list:  
*Iterable[str] | None = None*) → *Iterable[str]*

List all project-io plugin that implement method\_name.

#### Parameters

- **method\_name** (*str*) – Name of the method which should be supported.
- **block\_list** (*Iterable[str]*) – Iterable of plugin names which should be omitted.

### select\_data

glotaran.cli.commands.util.**select\_data**(data, dim, selection)

### select\_name

glotaran.cli.commands.util.**select\_name**(filename, dataset)

### signature\_analysis

glotaran.cli.commands.util.**signature\_analysis**(cmd)

### write\_data

glotaran.cli.commands.util.**write\_data**(data, out)

## Classes

### Summary

<i>ValOrRangeOrList</i>
-------------------------

### ValOrRangeOrList

**class** glotaran.cli.commands.util.ValOrRangeOrList

Bases: ParamType

### Attributes Summary

<i>arity</i>	
<i>envvar_list_splitter</i>	if a list of this type is expected and the value is pulled from a string environment variable, this is what splits it up.
<i>is_composite</i>	
<i>name</i>	the descriptive name of this type

### arity

ValOrRangeOrList.arity: **t.ClassVar[int]** = 1

### envvar\_list\_splitter

ValOrRangeOrList.envvar\_list\_splitter: **t.ClassVar[t.Optional[str]]** = None

if a list of this type is expected and the value is pulled from a string environment variable, this is what splits it up. *None* means any whitespace. For all parameters the general rule is that whitespace splits them up. The exception are paths and files which are split by `os.path.pathsep` by default (":" on Unix and ";" on Windows).

### is\_composite

ValOrRangeOrList.is\_composite: **t.ClassVar[bool]** = False

**name**

`ValOrRangeOrList.name: str = 'number or range or list'`

the descriptive name of this type

**Methods Summary**

<code>convert</code>	Convert the value to the correct type.
<code>fail</code>	Helper method to fail with an invalid value message.
<code>get_metavar</code>	Returns the metavar default for this param if it provides one.
<code>get_missing_message</code>	Optionally might return extra information about a missing parameter.
<code>shell_complete</code>	Return a list of <code>CompletionItem</code> objects for the incomplete value.
<code>split_envvar_value</code>	Given a value from an environment variable this splits it up into small chunks depending on the defined envvar list splitter.
<code>to_info_dict</code>	Gather information that could be useful for a tool generating user-facing documentation.

**convert**

`ValOrRangeOrList.convert(value, param, ctx)`

Convert the value to the correct type. This is not called if the value is `None` (the missing value).

This must accept string values from the command line, as well as values that are already the correct type. It may also convert other compatible types.

The `param` and `ctx` arguments may be `None` in certain situations, such as when converting prompt input.

If the value cannot be converted, call `fail()` with a descriptive message.

**Parameters**

- **value** – The value to convert.
- **param** – The parameter that is using this type to convert its value. May be `None`.
- **ctx** – The current context that arrived at this value. May be `None`.

**fail**

`ValOrRangeOrList.fail(message: str, param: Parameter | None = None, ctx: Context | None = None) → t.NoReturn`

Helper method to fail with an invalid value message.

### `get_metavar`

`ValOrRangeOrList.get_metavar(param: Parameter) → str | None`

Returns the metavar default for this param if it provides one.

### `get_missing_message`

`ValOrRangeOrList.get_missing_message(param: Parameter) → str | None`

Optionally might return extra information about a missing parameter.

New in version 2.0.

### `shell_complete`

`ValOrRangeOrList.shell_complete(ctx: Context, param: Parameter, incomplete: str) → List[CompletionItem]`

Return a list of `CompletionItem` objects for the incomplete value. Most types do not provide completions, but some do, and this allows custom types to provide custom completions as well.

#### Parameters

- **ctx** – Invocation context for this command.
- **param** – The parameter that is requesting completion.
- **incomplete** – Value being completed. May be empty.

New in version 8.0.

### `split_envvar_value`

`ValOrRangeOrList.split_envvar_value(rv: str) → Sequence[str]`

Given a value from an environment variable this splits it up into small chunks depending on the defined envvar list splitter.

If the splitter is set to *None*, which means that whitespace splits, then leading and trailing whitespace is ignored. Otherwise, leading and trailing splitters usually lead to empty items being included.

### `to_info_dict`

`ValOrRangeOrList.to_info_dict() → Dict[str, Any]`

Gather information that could be useful for a tool generating user-facing documentation.

Use `click.Context.to_info_dict()` to traverse the entire CLI structure.

New in version 8.0.



## Methods Documentation

**arity:** `t.ClassVar[int] = 1`

**convert**(*value*, *param*, *ctx*)

Convert the value to the correct type. This is not called if the value is `None` (the missing value).

This must accept string values from the command line, as well as values that are already the correct type. It may also convert other compatible types.

The `param` and `ctx` arguments may be `None` in certain situations, such as when converting prompt input.

If the value cannot be converted, call `fail()` with a descriptive message.

### Parameters

- **value** – The value to convert.
- **param** – The parameter that is using this type to convert its value. May be `None`.
- **ctx** – The current context that arrived at this value. May be `None`.

**envvar\_list\_splitter:** `t.ClassVar[t.Optional[str]] = None`

if a list of this type is expected and the value is pulled from a string environment variable, this is what splits it up. `None` means any whitespace. For all parameters the general rule is that whitespace splits them up. The exception are paths and files which are split by `os.path.pathsep` by default (":" on Unix and ";" on Windows).

**fail**(*message: str*, *param: Parameter | None = None*, *ctx: Context | None = None*) → `t.NoReturn`

Helper method to fail with an invalid value message.

**get\_metavar**(*param: Parameter*) → `str | None`

Returns the metavar default for this param if it provides one.

**get\_missing\_message**(*param: Parameter*) → `str | None`

Optionally might return extra information about a missing parameter.

New in version 2.0.

**is\_composite:** `t.ClassVar[bool] = False`

**name:** `str = 'number or range or list'`

the descriptive name of this type

**shell\_complete**(*ctx: Context*, *param: Parameter*, *incomplete: str*) → `List[CompletionItem]`

Return a list of `CompletionItem` objects for the incomplete value. Most types do not provide completions, but some do, and this allows custom types to provide custom completions as well.

### Parameters

- **ctx** – Invocation context for this command.
- **param** – The parameter that is requesting completion.
- **incomplete** – Value being completed. May be empty.

New in version 8.0.

**split\_envvar\_value**(*rv: str*) → `Sequence[str]`

Given a value from an environment variable this splits it up into small chunks depending on the defined envvar list splitter.

If the splitter is set to `None`, which means that whitespace splits, then leading and trailing whitespace is ignored. Otherwise, leading and trailing splitters usually lead to empty items being included.

`to_info_dict()` → `Dict[str, Any]`

Gather information that could be useful for a tool generating user-facing documentation.

Use `click.Context.to_info_dict()` to traverse the entire CLI structure.

New in version 8.0.

## validate

### Functions

#### Summary

<code>validate_cmd</code>	Validates a model file and optionally a parameter file.
---------------------------	---

#### validate\_cmd

`glotaran.cli.commands.validate.validate_cmd(parameters_file: str, model_file: str, scheme_file: str)`

Validates a model file and optionally a parameter file.

## main

`glotaran.cli.main = <Cli main>`

The glotaran CLI main function.

## 16.1.4 deprecation

Deprecation helpers and place to put deprecated implementations till removing.

### Modules

<code>glotaran.deprecation.deprecation_utils</code>	Helper functions to give deprecation warnings.
<code>glotaran.deprecation.modules</code>	Package containing deprecated implementations which were removed.

## deprecation\_utils

Helper functions to give deprecation warnings.

### Functions

#### Summary

<code>check_overdue</code>	Check if a deprecation is overdue for removal.
<code>check_qualnames_in_tests</code>	Test that qualnames import path exists when running tests.
<code>deprecate</code>	Decorate a function, method or class to deprecate it.
<code>deprecate_dict_entry</code>	Replace dict entry inplace and warn about usage change, if present in the dict.
<code>deprecate_module_attribute</code>	Import and return and attribute from the new location.
<code>deprecate_submodule</code>	Create a module at runtime which retrieves attributes from new module.
<code>glotaran_version</code>	Version of the distribution.
<code>module_attribute</code>	Import and return the attribute (e.g. function or class) of a module.
<code>parse_version</code>	Parse version string to tuple of three ints for comparison.
<code>raise_deprecation_error</code>	Raise <code>GlottaranDeprectedApiError</code> error, with formatted message.
<code>warn_deprecated</code>	Raise deprecation warning with change information.

#### check\_overdue

```
glotaran.deprecation.deprecation_utils.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_yaml_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`.

## check\_qualnames\_in\_tests

```
glotaran.deprecation.deprecation_utils.check_qualnames_in_tests(qual_names:  
    Sequence[str],  
    importable_indices:  
    Sequence[int])
```

Test that qualnames import path exists when running tests.

All deprecations should be tested anyway in order to get the proper errors when a deprecation is overdue. This helperfunction also helps to ensure that at least the import paths (`qual_names`) of the old and new usage exist.

### Parameters

- **qual\_names** (*Sequence[str]*) – Sequence of fully qualified module attribute names, optionally with call arguments.
- **importable\_indices** (*Sequence[int]*) – Indices of corresponding to `qual_names` indicating how to slice each `qual_name` split at `.`, for the import and attribute checking.

### See also:

[`warn\_deprecated`](#), [`deprecate`](#)

## deprecate

```
glotaran.deprecation.deprecation_utils.deprecate(*, deprecated_qual_name_usage: str,  
    new_qual_name_usage: str,  
    to_be_removed_in_version: str,  
    has_glotaran_replacement: bool = True,  
    importable_indices: tuple[int, int] = (1,  
    1)) → Callable[[DecoratedCallable],  
    DecoratedCallable]
```

Decorate a function, method or class to deprecate it.

This raises deprecation warning with 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_yaml_str)'`
- **new\_qual\_name\_usage** (*str*) – New usage as fully qualified name e.g.: `'glotaran.io.load_model(model_yaml_str, format_name="yaml_str")'`
- **to\_be\_removed\_in\_version** (*str*) – Version the support for this usage will be removed.
- **has\_glotaran\_replacement** (*bool*) – Whether or not this functionality has a replacement in core pyglotaran. This will be mapped to the second entry of `check_qualnames` in [`warn\_deprecated\(\)`](#).
- **importable\_indices** (*Sequence[int]*) – Indices from right for most nested item which is importable for `deprecated_qual_name_usage` and `new_qual_name_usage` after splitting at `..` This is used when the old or new usage is a method or mapping access. E.g. let `deprecated_qual_name_usage` be `package.module.class.mapping["key"]`, then you would use

importable\_indices=(2, 1), this way func:*check\_qualnames\_in\_tests* will import package.module.class and check if class has an attribute mapping. Default

#### Returns

Original function or class throwing a Deprecation warning when used.

#### Return type

DecoratedCallable

#### Raises

**OverDueDeprecation** – If the current version is greater or equal to *to\_be\_removed\_in\_version*.

#### See also:

*warn\_deprecated*, *deprecate\_module\_attribute*, *deprecate\_submodule*,  
*check\_qualnames\_in\_tests*

### Examples

This is the way the old *read\_parameters\_from\_yaml\_file* was deprecated and the usage of *load\_model* was promoted instead.

Listing 1: glotaran/deprecation/modules/glotaran\_root.py

```
@deprecate(
    deprecated_qualname_usage="glotaran.read_parameters_from_yaml_
    ↪file(model_path)",
    new_qualname_usage="glotaran.io.load_model(model_path)",
    to_be_removed_in_version="0.6.0",
)
def read_parameters_from_yaml_file(model_path: str):
    return load_model(model_path)
```

### deprecate\_dict\_entry

glotaran.deprecation.deprecation\_utils.**deprecate\_dict\_entry**(\*, dict\_to\_check: *MutableMapping*[*Hashable*, *Any*], deprecated\_usage: *str*, new\_usage: *str*, to\_be\_removed\_in\_version: *str*, swap\_keys: *tuple*[*Hashable*, *Hashable*] | *None* = *None*, replace\_rules: *tuple*[*Mapping*[*Hashable*, *Any*], *Mapping*[*Hashable*, *Any*]] | *None* = *None*, stacklevel: *int* = 3) → *None*

Replace dict entry inplace and warn about usage change, if present in the dict.

#### Parameters

- **dict\_to\_check** (*MutableMapping*[*Hashable*, *Any*]) – Dict which should be checked.
- **deprecated\_usage** (*str*) – Old usage to inform user (only used in warning).
- **new\_usage** (*str*) – New usage to inform user (only used in warning).
- **to\_be\_removed\_in\_version** (*str*) – Version the support for this usage will be removed.
- **swap\_keys** (*tuple*[*Hashable*, *Hashable*]) – (old\_key, new\_key), dict\_to\_check[new\_key] will be assigned the value dict\_to\_check[old\_key] and old\_key will be removed from the dict. by default None
- **replace\_rules** (*Mapping*[*Hashable*, *tuple*[*Any*, *Any*]]) – ({old\_key: old\_value}, {new\_key: new\_value}), If dict\_to\_check[old\_key] has the value old\_value, dict\_to\_check[new\_key] it will be set to new\_value. old\_key will be removed from the dict if old\_key and new\_key aren't equal. by default None
- **stacklevel** (*int*) – Stack at which the warning should be shown as raise. , by default 3

#### Raises

- **ValueError** – If both swap\_keys and replace\_rules are None (default) or not None.
- **OverDueDeprecation** – If the current version is greater or equal to to\_be\_removed\_in\_version.

#### See also:

[\*warn\\_deprecated\*](#)

#### Notes

To prevent confusion exactly one of `replace_rules` and `swap_keys` needs to be passed.

#### Examples

For readability sake the warnings won't be shown in the examples.

Swapping key names:

```
>>> dict_to_check = {"foo": 123}
>>> deprecate_dict_entry(
    dict_to_check=dict_to_check,
    deprecated_usage="foo",
    new_usage="bar",
    to_be_removed_in_version="0.6.0",
    swap_keys=("foo", "bar")
)
>>> dict_to_check
{"bar": 123}
```

Changing values:

```

>>> dict_to_check = {"foo": 123}
>>> deprecate_dict_entry(
    dict_to_check=dict_to_check,
    deprecated_usage="foo: 123",
    new_usage="foo: 123.0",
    to_be_removed_in_version="0.6.0",
    replace_rules=({"foo": 123}, {"foo": 123.0})
)
>>> dict_to_check
{"foo": 123.0}

```

Swapping key names AND changing values:

```

>>> dict_to_check = {"type": "kinetic-spectrum"}
>>> deprecate_dict_entry(
    dict_to_check=dict_to_check,
    deprecated_usage="type: kinetic-spectrum",
    new_usage="default_megacomplex: decay",
    to_be_removed_in_version="0.6.0",
    replace_rules=({"type": "kinetic-spectrum"}, {"default_megacomplex": "decay"}))
>>> dict_to_check
{"default_megacomplex": "decay"}

```

## deprecate\_module\_attribute

`glotaran.deprecation.deprecation_utils.deprecate_module_attribute(*, deprecated_qual_name: str, new_qual_name: str, to_be_removed_in_version: str, module_load_overwrite: str = "") → Any`

Import and return and attribute from the new location.

This needs to be wrapped in the definition of a module wide `__getattr__` function so it won't throw warnings all the time (see example).

### Parameters

- **deprecated\_qual\_name** (*str*) – Fully qualified name of the deprecated attribute e.g.: `glotaran.ParameterGroup`
- **new\_qual\_name** (*str*) – Fully qualified name of the new attribute e.g.: `glotaran.parameter.ParameterGroup`
- **to\_be\_removed\_in\_version** (*str*) – Version the support for this usage will be removed.
- **module\_load\_overwrite** (*str*) – Overwrite the location the functionality will be set from. This allows preserving functionality without polluting a new module with code just for the sake of it. By default ""

**Returns**

Module attribute from its new location.

**Return type**

Any

**Raises**

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

**See also:**

*deprecate*, *warn\_deprecated*, *deprecate\_submodule*

**Examples**

When deprecating the usage of `ParameterGroup` the root of `glotaran` and promoting to import it from `glotaran.parameter` the following code was added to the root `__init__.py`.

Listing 2: `glotaran/__init__.py`

```
def __getattr__(attribute_name: str):
    from glotaran.deprecation import deprecate_module_attribute

    if attribute_name == "ParameterGroup":
        return deprecate_module_attribute(
            deprecated_qual_name="glotaran.ParameterGroup",
            new_qual_name="glotaran.parameter.ParameterGroup",
            to_be_removed_in_version="0.6.0",
        )

    raise AttributeError(f"module {__name__} has no attribute {attribute_
↪ name}")
```

**`deprecate_submodule`**

```
glotaran.deprecation.deprecation_utils.deprecate_submodule(*,
                                                             deprecated_module_name:
                                                             str, new_module_name:
                                                             str,
                                                             to_be_removed_in_version:
                                                             str,
                                                             module_load_overwrite:
                                                             str = "") → module
```

Create a module at runtime which retrieves attributes from new module.

When moving a module, create a variable with the modules name in the parent packages `__init__.py`, so imports will be redirected to the new module location and a deprecation warning will be given, to help the user adjust the outdated code. Each time an attribute is retrieved there will be a deprecation warning.

**Parameters**

- **deprecated\_module\_name** (*str*) – Fully qualified name of the deprecated module  
e.g.: `'glotaran.analysis.result'`



- **new\_module\_name** (*str*) – Fully qualified name of the new module e.g.: 'glotaran.project.result'
- **to\_be\_removed\_in\_version** (*str*) – Version the support for this usage will be removed.
- **module\_load\_overwrite** (*str*) – Overwrite the location for the new module the deprecated functionality is loaded from. This allows preserving functionality without polluting a new module with code just for the sake of it. By default ''

**Returns**

Module containing

**Return type**

ModuleType

**Raises**

**OverDueDeprecation** – If the current version is greater or equal to to\_be\_removed\_in\_version.

**See also:***deprecate, deprecate\_module\_attribute***Examples**

When moving the module `result` from `glotaran.analysis.result` to `glotaran.project.result` the following code was added to the old parent packages (`glotaran.analysis`) `__init__.py`.

Listing 3: `glotaran/analysis/__init__.py`

```
from glotaran.deprecation.deprecation_utils import deprecate_submodule

result = deprecate_submodule(
    deprecated_module_name="glotaran.analysis.result",
    new_module_name="glotaran.project.result",
    to_be_removed_in_version="0.6.0",
)
```

**glotaran\_version**

`glotaran.deprecation.deprecation_utils.glotaran_version()` → *str*

Version of the distribution.

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

**Returns**

The version string.

**Return type***str*

### module\_attribute

```
glotaran.deprecation.deprecation_utils.module_attribute(module_qual_name: str,  
                                                         attribute_name: str) → Any
```

Import and return the attribute (e.g. function or class) of a module.

This is basically the same as `from module_name import attribute_name as return_value` where this function returns `return_value`.

#### Parameters

- **module\_qual\_name** (*str*) – Fully qualified name for a module e.g. `glotaran.model.base_model`
- **attribute\_name** (*str*) – Name of the attribute e.g. `Model`

#### Returns

Attribute of the module, e.g. a function or class.

#### Return type

Any

### parse\_version

```
glotaran.deprecation.deprecation_utils.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 than three elements separated by `..`
- **ValueError** – If `version_str`’s first three elements can not be casted to int.

### raise\_deprecation\_error

```
glotaran.deprecation.deprecation_utils.raise_deprecation_error(*, depre-  
cated_qual_name_usage:  
    str,  
    new_qual_name_usage:  
    str,  
    to_be_removed_in_version:  
    str) → NoReturn
```

Raise `GlottaranDepreciatedApiError` error, with formatted message.

This should only be used if there is no reasonable way to keep the deprecated usage functional!

#### Parameters

- **deprecated\_qual\_name\_usage** (*str*) – Old usage with fully qualified name e.g.:  
'glotaran.read\_model\_from\_yaml(model\_yaml\_str)'
- **new\_qual\_name\_usage** (*str*) – New usage as fully qualified name e.g.:  
'glotaran.io.load\_model(model\_yaml\_str, format\_name="yaml\_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`.
- **GlottaranDeprecatedApiError** – If `OverDueDeprecation` wasn't raised before.

### warn\_deprecated

```
glotaran.deprecation.deprecation_utils.warn_deprecated(*,
    deprecated_qual_name_usage:
    str, new_qual_name_usage: str,
    to_be_removed_in_version: str,
    check_qual_names: tuple[bool,
    bool] = (True, True), stacklevel:
    int = 2, importable_indices:
    tuple[int, int] = (1, 1)) → 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\_yaml\_str)'
- **new\_qual\_name\_usage** (*str*) – New usage as fully qualified name e.g.:  
'glotaran.io.load\_model(model\_yaml\_str, format\_name="yaml\_str")'
- **to\_be\_removed\_in\_version** (*str*) – Version the support for this usage will be removed.
- **check\_qual\_names** (*tuple[bool, bool]*) – Whether or not to check for the existence `deprecated_qual_name_usage` and `deprecated_qual_name_usage`
  - Set the first value to False to prevent infinite recursion error when changing a module attribute import.
  - Set the second value to False if the new usage is in a different package or there is none.
- **stacklevel** (*int*) – Stack at which the warning should be shown as raise. Default: 2
- **importable\_indices** (*tuple[int, int]*) – Indices from right for most nested item which is importable for `deprecated_qual_name_usage` and `new_qual_name_usage` after splitting at `..`. This is used when the old or new usage is a method or mapping access. E.g. let `deprecated_qual_name_usage` be `package.module.class.mapping["key"]`, then you would use `importable_indices=(2, 1)`, this way `func:check_qualnames_in_tests` will import `package.module.class` and check if `class` has an attribute `mapping`.

**Raises**

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

**See also:**

*deprecate*, *deprecate\_module\_attribute*, *deprecate\_submodule*,  
*check\_qualnames\_in\_tests*

**Examples**

This is the way the old `read_parameters_from_yaml_file` could be deprecated and the usage of `load_model` being promoted instead.

Listing 4: `glotaran/deprecation/modules/glotaran_root.py`

```
def read_parameters_from_yaml_file(model_path: str):
    warn_deprecated(
        deprecated_qual_name_usage="glotaran.read_parameters_from_yaml_
↪file(model_path)",
        new_qual_name_usage="glotaran.io.load_model.load_model(model_path)
↪",
        to_be_removed_in_version="0.6.0",
    )
    return load_model(model_path)
```

**Exceptions****Exception Summary**

<code>GlottaranApiDeprecationWarning</code>	Warning to give users about API changes.
<code>GlottaranDeprecatedApiError</code>	Exception raised when a deprecation has no replacement.
<code>OverDueDeprecation</code>	Error thrown when a deprecation should have been removed.

**GlottaranApiDeprecationWarning**

**exception** `glotaran.deprecation.deprecation_utils.GlottaranApiDeprecationWarning`

Warning to give users about API changes.

**See also:**

*deprecate*, *warn\_deprecated*, *deprecate\_module\_attribute*, *deprecate\_submodule*,  
*deprecate\_dict\_entry*

## GlotaranDeprecatedApiError

**exception** `glotaran.deprecation.deprecation_utils.GlotaranDeprecatedApiError`

Exception raised when a deprecation has no replacement.

**See also:**

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

## OverDueDeprecation

**exception** `glotaran.deprecation.deprecation_utils.OverDueDeprecation`

Error thrown when a deprecation should have been removed.

**See also:**

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

## modules

Package containing deprecated implementations which were removed.

To keep things organized the filenames should be like the relative import path from glotaran root, but with `_` instead of `..`. E.g. `glotaran.analysis.scheme` would map to `analysis_scheme.py`

The only exceptions to this rule are the root `__init__.py` which is named `glotaran_root.py` and testing changed imports which should be placed in `test_changed_imports.py`.

## Modules

<code>glotaran.deprecation.modules.</code>	Deprecation functions for the yaml parser.
<code>builtin_io_yaml</code>	
<code>glotaran.deprecation.modules.examples</code>	Deprecation package for 'glotaran.examples'.

## builtin\_io\_yaml

Deprecation functions for the yaml parser.

## Functions

### Summary

<code>model_spec_deprecations</code>	Check deprecations in the model specification spec dict.
--------------------------------------	--

## model\_spec\_deprecations

`glotaran.deprecation.modules.builtin_io_yaml.model_spec_deprecations`(*spec*:  
*MutableMapping*[*Any*,  
*Any*]) → *None*

Check deprecations in the model specification `spec` dict.

### Parameters

**spec** (*MutableMapping*[*Any*, *Any*]) – Model specification dictionary

## examples

Deprecation package for ‘glotaran.examples’.

## Modules

<code>glotaran.deprecation.modules.examples.sequential</code>	Deprecated functionality export for 'glotaran.examples.sequential'.
---	---

## sequential

Deprecated functionality export for ‘glotaran.examples.sequential’.

## 16.1.5 io

Functions for data IO.

### Note:

Since Io functionality is purely plugin based this package mostly reexports functions from the `glotaran.plugin_system` from a common place.

## Modules

<code>glotaran.io.interface</code>	Baseclasses to create Data/Project IO plugins from.
<code>glotaran.io.prepare_dataset</code>	Module containing dataset preparation functionality.
<code>glotaran.io.preprocessor</code>	Tools for data pre-processing.

## interface

Baseclasses to create Data/Project IO plugins from.

The main purpose of those classes are to guarantee a consistent API via typechecker like mypy and demonstrate with methods are accessed by highlevel convenience functions for a given type of plugin.

To add additional options to a method, those options need to be keyword only arguments. See: <https://www.python.org/dev/peps/pep-3102/>

## Classes

### Summary

<i>DataIoInterface</i>	Baseclass for Data IO plugins.
<i>ProjectIoInterface</i>	Baseclass for Project IO plugins.
<i>SavingOptions</i>	A collection of options for result saving.

### DataIoInterface

**class** `glotaran.io.interface.DataIoInterface`(*format\_name*: *str*)

Bases: `object`

Baseclass for Data IO plugins.

Initialize a Data IO plugin with the name of the format.

#### Parameters

**format\_name** (*str*) – Name of the supported format an instance uses.

### Methods Summary

<i>load_dataset</i>	Read data from a file to <code>xarray.Dataset</code> or <code>xarray.DataArray</code> .
<i>save_dataset</i>	Save data from <code>xarray.Dataset</code> to a file.

### load\_dataset

`DataIoInterface.load_dataset`(*file\_name*: *str*) → `xr.Dataset` | `xr.DataArray`

Read data from a file to `xarray.Dataset` or `xarray.DataArray`.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the data.

##### Returns

Data loaded from the file.

##### Return type

`xr.Dataset`|`xr.DataArray`

### save\_dataset

DataIoInterface.**save\_dataset**(dataset: *xr.Dataset* | *xr.DataArray*, file\_name: *str*)

Save data from *xarray.Dataset* to a file.

#### NOT IMPLEMENTED

##### Parameters

- **dataset** (*xr.Dataset*) – Dataset to be saved to file.
- **file\_name** (*str*) – File to write the data to.

## Methods Documentation

**load\_dataset**(file\_name: *str*) → *xr.Dataset* | *xr.DataArray*

Read data from a file to *xarray.Dataset* or *xarray.DataArray*.

#### NOT IMPLEMENTED

##### Parameters

- **file\_name** (*str*) – File containing the data.

##### Returns

Data loaded from the file.

##### Return type

*xr.Dataset*|*xr.DataArray*

**save\_dataset**(dataset: *xr.Dataset* | *xr.DataArray*, file\_name: *str*)

Save data from *xarray.Dataset* to a file.

#### NOT IMPLEMENTED

##### Parameters

- **dataset** (*xr.Dataset*) – Dataset to be saved to file.
- **file\_name** (*str*) – File to write the data to.

## ProjectIoInterface

**class** glotaran.io.interface.**ProjectIoInterface**(format\_name: *str*)

Bases: *object*

Baseclass for Project IO plugins.

Initialize a Project IO plugin with the name of the format.

##### Parameters

- **format\_name** (*str*) – Name of the supported format an instance uses.



## Methods Summary

<code>load_model</code>	Create a Model instance from the specs defined in a file.
<code>load_parameters</code>	Create a Parameters instance from the specs defined in a file.
<code>load_result</code>	Create a Result instance from the specs defined in a file.
<code>load_scheme</code>	Create a Scheme instance from the specs defined in a file.
<code>save_model</code>	Save a Model instance to a spec file.
<code>save_parameters</code>	Save a Parameters instance to a spec file.
<code>save_result</code>	Save a Result instance to a spec file.
<code>save_scheme</code>	Save a Scheme instance to a spec file.

### load\_model

`ProjectIoInterface.load_model(file_name: str) → Model`

Create a Model instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the model specs.

##### Returns

Model instance created from the file.

##### Return type

*Model*

### load\_parameters

`ProjectIoInterface.load_parameters(file_name: str) → Parameters`

Create a Parameters instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the parameter specs.

##### Returns

Parameters instance created from the file.

##### Return type

*Parameters*

### load\_result

`ProjectIoInterface.load_result(result_path: str) → Result`

Create a Result instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**result\_path** (*str*) – Path containing the result data.

##### Returns

Result instance created from the file.

##### Return type

*Result*

### load\_scheme

`ProjectIoInterface.load_scheme(file_name: str) → Scheme`

Create a Scheme instance from the specs defined in a file.

#### NOT IMPLEMENTED

##### Parameters

**file\_name** (*str*) – File containing the parameter specs.

##### Returns

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

### save\_model

`ProjectIoInterface.save_model(model: Model, file_name: str)`

Save a Model instance to a spec file.

#### NOT IMPLEMENTED

##### Parameters

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

### save\_parameters

`ProjectIoInterface.save_parameters(parameters: Parameters, file_name: str)`

Save a Parameters instance to a spec file.

#### NOT IMPLEMENTED

##### Parameters

- **parameters** (*Parameters*) – Parameters instance to save to specs file.
- **file\_name** (*str*) – File to write the parameter specs to.

## save\_result

`ProjectIoInterface.save_result(result: Result, result_path: str, *, saving_options: SavingOptions = SavingOptions(data_filter=None, data_format='nc', parameter_format='csv', report=True))`  
 → `list[str]`

Save a Result instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **result** ([Result](#)) – Result instance to save to specs file.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** ([SavingOptions](#)) – Options for the saved result.

## save\_scheme

`ProjectIoInterface.save_scheme(scheme: Scheme, file_name: str)`

Save a Scheme instance to a spec file.

### NOT IMPLEMENTED

#### Parameters

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## Methods Documentation

`load_model(file_name: str)` → [Model](#)

Create a Model instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

- **file\_name** (*str*) – File containing the model specs.

#### Returns

Model instance created from the file.

#### Return type

[Model](#)

`load_parameters(file_name: str)` → [Parameters](#)

Create a Parameters instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

- **file\_name** (*str*) – File containing the parameter specs.

#### Returns

Parameters instance created from the file.

#### Return type

[Parameters](#)

`load_result(result_path: str)` → [Result](#)

Create a Result instance from the specs defined in a file.

### NOT IMPLEMENTED

#### Parameters

- **result\_path** (*str*) – Path containing the result data.

**Returns**

Result instance created from the file.

**Return type**

*Result*

**load\_scheme**(*file\_name*: *str*) → *Scheme*

Create a Scheme instance from the specs defined in a file.

**NOT IMPLEMENTED****Parameters**

**file\_name** (*str*) – File containing the parameter specs.

**Returns**

- *Scheme* – Scheme instance created from the file.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**save\_model**(*model*: *Model*, *file\_name*: *str*)

Save a Model instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **model** (*Model*) – Model instance to save to specs file.
- **file\_name** (*str*) – File to write the model specs to.

**save\_parameters**(*parameters*: *Parameters*, *file\_name*: *str*)

Save a Parameters instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **parameters** (*Parameters*) – Parameters instance to save to specs file.
- **file\_name** (*str*) – File to write the parameter specs to.

**save\_result**(*result*: *Result*, *result\_path*: *str*, \*, *saving\_options*: *SavingOptions* = *SavingOptions*(*data\_filter*=None, *data\_format*='nc', *parameter\_format*='csv', *report*=True)) → list[*str*]

Save a Result instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **result** (*Result*) – Result instance to save to specs file.
- **result\_path** (*str*) – Path to write the result data to.
- **saving\_options** (*SavingOptions*) – Options for the saved result.

**save\_scheme**(*scheme*: *Scheme*, *file\_name*: *str*)

Save a Scheme instance to a spec file.

**NOT IMPLEMENTED****Parameters**

- **scheme** (*Scheme*) – Scheme instance to save to specs file.
- **file\_name** (*str*) – File to write the scheme specs to.

## SavingOptions

```
class glotaran.io.interface.SavingOptions(data_filter: list[str] | None = None, data_format:
    Literal['nc'] = 'nc', parameter_format:
    Literal['csv'] = 'csv', report: bool = True)
```

Bases: `object`

A collection of options for result saving.

### Attributes Summary

<code>data_filter</code>
<code>data_format</code>
<code>parameter_format</code>
<code>report</code>

#### data\_filter

`SavingOptions.data_filter: list[str] | None = None`

#### data\_format

`SavingOptions.data_format: Literal['nc'] = 'nc'`

#### parameter\_format

`SavingOptions.parameter_format: Literal['csv'] = 'csv'`

#### report

`SavingOptions.report: bool = True`

### Methods Summary

## Methods Documentation

```
data_filter: list[str] | None = None
data_format: Literal['nc'] = 'nc'
parameter_format: Literal['csv'] = 'csv'
report: bool = True
```

## prepare\_dataset

Module containing dataset preparation functionality.

## Functions

### Summary

<code>add_svd_to_dataset</code>	Add the SVD of a dataset inplace as Data variables to the dataset.
<code>prepare_time_trace_dataset</code>	Prepare a time trace dataset for global analysis.

## add\_svd\_to\_dataset

```
glotaran.io.prepare_dataset.add_svd_to_dataset(dataset: xr.Dataset, name: str = 'data',
                                              lsv_dim: Hashable = 'time', rsv_dim:
                                              Hashable = 'spectral', data_array:
                                              xr.DataArray | None = None) → None
```

Add the SVD of a dataset inplace as Data variables to the dataset.

The SVD is only computed if it doesn't already exist on the dataset.

### Parameters

- **dataset** (*xr.Dataset*) – Dataset the SVD values should be added to.
- **name** (*str*) – Key to access the datarray inside of the dataset, by default “data”
- **lsv\_dim** (*Hashable*) – Name of the dimension for the left singular value, by default “time”
- **rsv\_dim** (*Hashable*) – Name of the dimension for the right singular value, by default “spectral”
- **data\_array** (*xr.DataArray* / *None*) – Dataarray to calculate the SVD for, when provided the data extraction from the dataset will be skipped, by default *None*

## prepare\_time\_trace\_dataset

`glotaran.io.prepare_dataset.prepare_time_trace_dataset`(*dataset: [DataArray](#) | [Dataset](#)*,  
*weight: [ndarray](#) | [None](#) = [None](#)*,  
*irf: [ndarray](#) | [DataArray](#) | [None](#) = [None](#)*) → [Dataset](#)

Prepare a time trace dataset for global analysis.

### Parameters

- **dataset** (*[xr.DataArray](#) | [xr.Dataset](#)*) – Dataset to add data to.
- **weight** (*[np.ndarray](#) | [None](#)*) – A weight for the dataset.
- **irf** (*[np.ndarray](#) | [xr.DataArray](#) | [None](#)*) – An IRF for the dataset.

### Returns

Original dataset with added SVD data and weight and/or irf data variables if provided.

### Return type

[xr.Dataset](#)

### Raises

**ValueError** – If an IRF with more than one dimensions was provided that isn't a [xarray.DataArray](#).

## preprocessor

Tools for data pre-processing.

### Modules

<a href="#"><code>glotaran.io.preprocessor.pipeline</code></a>	A pre-processor pipeline for data.
<a href="#"><code>glotaran.io.preprocessor.preprocessor</code></a>	A pre-processor pipeline for data.

## pipeline

A pre-processor pipeline for data.

### Classes

#### Summary

<a href="#"><code>PreProcessingPipeline</code></a>	A pipeline for pre-processors.
--	--------------------------------

## PreProcessingPipeline

```
class glotaran.io.preprocessor.pipeline.PreProcessingPipeline(*, actions:
    list[CorrectBaselineValue
    | CorrectBaselineAverage] = None)
```

Bases: BaseModel

A pipeline for pre-processors.

Create a new model by parsing and validating input data from keyword arguments.

Raises `[ValidationError][pydantic_core.ValidationError]` if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

### Attributes Summary

<code>model_computed_fields</code>	Get the computed fields of this model instance.
<code>model_config</code>	Configuration for the model, should be a dictionary conforming to <code>[ConfigDict][pydantic.config.ConfigDict]</code> .
<code>model_extra</code>	Get extra fields set during validation.
<code>model_fields</code>	Metadata about the fields defined on the model, mapping of field names to <code>[FieldInfo][pydantic.fields.FieldInfo]</code> .
<code>model_fields_set</code>	Returns the set of fields that have been explicitly set on this model instance.
<code>actions</code>	

### model\_computed\_fields

`PreProcessingPipeline.model_computed_fields`

Get the computed fields of this model instance.

**Returns:**

A dictionary of computed field names and their corresponding `ComputedFieldInfo` objects.

### model\_config

`PreProcessingPipeline.model_config: ClassVar[ConfigDict] = {}`

Configuration for the model, should be a dictionary conforming to `[ConfigDict][pydantic.config.ConfigDict]`.



## model\_extra

`PreProcessingPipeline.model_extra`

Get extra fields set during validation.

**Returns:**

A dictionary of extra fields, or *None* if *config.extra* is not set to “allow”.

## model\_fields

```
PreProcessingPipeline.model_fields: ClassVar[dict[str, FieldInfo]] =  
{'actions': FieldInfo(annotation=list[Annotated[Union[CorrectBaselineValue,  
CorrectBaselineAverage], FieldInfo(annotation=NoneType, required=True,  
discriminator='action')]], required=False, default_factory=list)}
```

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.\_\_fields\_\_* from Pydantic V1.

## model\_fields\_set

`PreProcessingPipeline.model_fields_set`

Returns the set of fields that have been explicitly set on this model instance.

**Returns:**

A set of strings representing the fields that have been set,  
i.e. that were not filled from defaults.

## actions

`PreProcessingPipeline.actions: list[PipelineAction]`

## Methods Summary

<code>apply</code>	Apply all pre-processors on data.
<code>construct</code>	
<code>copy</code>	Returns a copy of the model.
<code>correct_baseline_average</code>	Correct a dataset by subtracting the average over a part of the data.
<code>correct_baseline_value</code>	Correct a dataset by subtracting baseline value.
<code>dict</code>	
<code>from_orm</code>	
<code>json</code>	
<code>model_construct</code>	Creates a new instance of the <i>Model</i> class with validated data.
<code>model_copy</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy">https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy</a>
<code>model_dump</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump">https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump</a>
<code>model_dump_json</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json">https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json</a>
<code>model_json_schema</code>	Generates a JSON schema for a model class.
<code>model_parametrized_name</code>	Compute the class name for parametrizations of generic classes.
<code>model_post_init</code>	Override this method to perform additional initialization after <code>__init__</code> and <code>model_construct</code> .
<code>model_rebuild</code>	Try to rebuild the pydantic-core schema for the model.
<code>model_validate</code>	Validate a pydantic model instance.
<code>model_validate_json</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/json/#json-parsing">https://docs.pydantic.dev/2.5/concepts/json/#json-parsing</a>
<code>model_validate_strings</code>	Validate the given object contains string data against the Pydantic model.
<code>parse_file</code>	
<code>parse_obj</code>	
<code>parse_raw</code>	
<code>schema</code>	
<code>schema_json</code>	
<code>update_forward_refs</code>	
<code>validate</code>	

## apply

`PreProcessingPipeline.apply(original: dataArray) → dataArray`

Apply all pre-processors on data.

**Parameters**

**original** (*xr.DataArray*) – The data to process.

**Return type**

*xr.DataArray*

## construct

**classmethod** `PreProcessingPipeline.construct(_fields_set: set[str] | None = None,  
**values: Any) → Model`

## copy

`PreProcessingPipeline.copy(*, include: AbstractSetIntStr | MappingIntStrAny | None =  
None, exclude: AbstractSetIntStr | MappingIntStrAny | None =  
None, update: Dict[str, Any] | None = None, deep: bool =  
False) → Model`

Returns a copy of the model.

**!!! warning “Deprecated”**

This method is now deprecated; use *model\_copy* instead.

If you need *include* or *exclude*, use:

```
`py data = self.model_dump(include=include, exclude=exclude,  
round_trip=True) data = {**data, **(update or {})} copied = self.  
model_validate(data)`
```

**Args:**

**include: Optional set or mapping**

specifying which fields to include in the copied model.

**exclude: Optional set or mapping**

specifying which fields to exclude in the copied model.

**update: Optional dictionary of field-value pairs to override field values**

in the copied model.

**deep:** If True, the values of fields that are Pydantic models will be deep copied.

**Returns:**

A copy of the model with included, excluded and updated fields as specified.

## correct\_baseline\_average

`PreProcessingPipeline.correct_baseline_average(select: dict[str, slice | list[int] | int] |  
None = None, exclude: dict[str, slice  
| list[int] | int] | None = None) →  
PreProcessingPipeline`

Correct a dataset by subtracting the average over a part of the data.

**Parameters**

- **select** (*dict[str, slice | list[int] | int] | None*) – The selection to average as dictionary of dimension and indexer. The indexer can be a slice, a list or an integer value.

- **exclude** (*dict[str, slice | list[int] | int] | None*) – Excluded regions from the average as dictionary of dimension and indexer. The indexer can be a slice, a list or an integer value.

**Return type**

*PreProcessingPipeline*

### **correct\_baseline\_value**

`PreProcessingPipeline.correct_baseline_value(value: float) → PreProcessingPipeline`

Correct a dataset by subtracting baseline value.

**Parameters**

**value** (*float*) – The value to subtract.

**Return type**

*PreProcessingPipeline*

### **dict**

`PreProcessingPipeline.dict(*, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False) → Dict[str, Any]`

### **from\_orm**

**classmethod** `PreProcessingPipeline.from_orm(obj: Any) → Model`

### **json**

`PreProcessingPipeline.json(*, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, encoder: Callable[[Any], Any] | None = PydanticUndefined, models_as_dict: bool = PydanticUndefined, **kwargs: Any) → str`

### **model\_construct**

**classmethod** `PreProcessingPipeline.model_construct(_fields_set: set[str] | None = None, **values: Any) → Model`

Creates a new instance of the *Model* class with validated data.

Creates a new model setting `__dict__` and `__pydantic_fields_set__` from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if `Config.extra = 'allow'` was set since it adds all passed values

**Args:**

`_fields_set`: The set of field names accepted for the *Model* instance. `values`: Trusted or pre-validated data dictionary.

**Returns:**

A new instance of the *Model* class with validated data.

## model\_copy

`PreProcessingPipeline.model_copy(*, update: dict[str, Any] | None = None, deep: bool = False) → Model`

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#model\\_copy](https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy)

Returns a copy of the model.

### Args:

**update:** Values to change/add in the new model. **Note: the data is not validated** before creating the new model. You should trust this data.  
**deep:** Set to *True* to make a deep copy of the model.

### Returns:

New model instance.

## model\_dump

`PreProcessingPipeline.model_dump(*, mode: Literal['json', 'python'] | str = 'python', include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True) → dict[str, Any]`

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

### Args:

**mode:** The mode in which to *python* should run.

If mode is 'json', the dictionary will only contain JSON serializable types. If mode is 'python', the dictionary may contain any Python objects.

**include:** A list of fields to include in the output. **exclude:** A list of fields to exclude from the output. **by\_alias:** Whether to use the field's alias in the dictionary key if defined. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that are set to their default value from the output. **exclude\_none:** Whether to exclude fields that have a value of *None* from the output. **round\_trip:** Whether to enable serialization and deserialization round-trip support. **warnings:** Whether to log warnings when invalid fields are encountered.

### Returns:

A dictionary representation of the model.

## model\_dump\_json

`PreProcessingPipeline.model_dump_json(*, indent: int | None = None, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True) → str`

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump\\_json](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json)

Generates a JSON representation of the model using Pydantic's *to\_json* method.

**Args:**

indent: Indentation to use in the JSON output. If None is passed, the output will be compact. include: Field(s) to include in the JSON output. Can take either a string or set of strings. exclude: Field(s) to exclude from the JSON output. Can take either a string or set of strings. by\_alias: Whether to serialize using field aliases. exclude\_unset: Whether to exclude fields that have not been explicitly set. exclude\_defaults: Whether to exclude fields that have the default value. exclude\_none: Whether to exclude fields that have a value of *None*. round\_trip: Whether to use serialization/deserialization between JSON and class instance. warnings: Whether to show any warnings that occurred during serialization.

**Returns:**

A JSON string representation of the model.

## model\_json\_schema

```
classmethod PreProcessingPipeline.model_json_schema(by_alias: bool = True,  
                                                    ref_template: str =  
                                                    '#/$defs/{model}',  
                                                    schema_generator:  
                                                    type[~pydantic.json_schema.GenerateJsonSchema]  
                                                    = <class 'pydan-  
                                                    tic.json_schema.GenerateJsonSchema'>,  
                                                    mode:  
                                                    ~typing.Literal['validation',  
                                                    'serialization'] = 'validation')  
                                                    → dict[str, Any]
```

Generates a JSON schema for a model class.

**Args:**

by\_alias: Whether to use attribute aliases or not. ref\_template: The reference template. schema\_generator: To override the logic used to generate the JSON schema, as a subclass of *GenerateJsonSchema* with your desired modifications. mode: The mode in which to generate the schema.

**Returns:**

The JSON schema for the given model class.

## model\_parametrized\_name

```
classmethod PreProcessingPipeline.model_parametrized_name(params:  
                                                         tuple[type[Any], ...]  
                                                         → str
```

Compute the class name for parametrizations of generic classes.

This method can be overridden to achieve a custom naming scheme for generic BaseModels.

**Args:**

**params: Tuple of types of the class. Given a generic class**

*Model* with 2 type variables and a concrete model *Model[str, int]*, the value (*str, int*) would be passed to *params*.

**Returns:**

String representing the new class where *params* are passed to *cls* as type variables.

**Raises:**

*TypeError*: Raised when trying to generate concrete names for non-generic models.

## model\_post\_init

`PreProcessingPipeline.model_post_init(_BaseModel__context: Any) → None`

Override this method to perform additional initialization after `__init__` and `model_construct`. This is useful if you want to do some validation that requires the entire model to be initialized.

## model\_rebuild

**classmethod** `PreProcessingPipeline.model_rebuild(*, force: bool = False, raise_errors: bool = True, _parent_namespace_depth: int = 2, _types_namespace: dict[str, Any] | None = None) → bool | None`

Try to rebuild the pydantic-core schema for the model.

This may be necessary when one of the annotations is a `ForwardRef` which could not be resolved during the initial attempt to build the schema, and automatic rebuilding fails.

**Args:**

`force`: Whether to force the rebuilding of the model schema, defaults to *False*. `raise_errors`: Whether to raise errors, defaults to *True*. `_parent_namespace_depth`: The depth level of the parent namespace, defaults to 2. `_types_namespace`: The types namespace, defaults to *None*.

**Returns:**

Returns *None* if the schema is already “complete” and rebuilding was not required. If rebuilding `_was_required`, returns *True* if rebuilding was successful, otherwise *False*.

## model\_validate

**classmethod** `PreProcessingPipeline.model_validate(obj: Any, *, strict: bool | None = None, from_attributes: bool | None = None, context: dict[str, Any] | None = None) → Model`

Validate a pydantic model instance.

**Args:**

`obj`: The object to validate. `strict`: Whether to raise an exception on invalid fields. `from_attributes`: Whether to extract data from object attributes. `context`: Additional context to pass to the validator.

**Raises:**

`ValidationError`: If the object could not be validated.

**Returns:**

The validated model instance.

### model\_validate\_json

```
classmethod PreProcessingPipeline.model_validate_json(json_data: str | bytes |  
                                                       bytearray, *, strict: bool |  
                                                       None = None, context:  
                                                       dict[str, Any] | None =  
                                                       None) → Model
```

Usage docs: <https://docs.pydantic.dev/2.5/concepts/json/#json-parsing>

Validate the given JSON data against the Pydantic model.

**Args:**

json\_data: The JSON data to validate. strict: Whether to enforce types strictly. context:  
Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**Raises:**

ValueError: If *json\_data* is not a JSON string.

### model\_validate\_strings

```
classmethod PreProcessingPipeline.model_validate_strings(obj: Any, *, strict: bool  
                                                         | None = None, context:  
                                                         dict[str, Any] | None =  
                                                         None) → Model
```

Validate the given object contains string data against the Pydantic model.

**Args:**

obj: The object contains string data to validate. strict: Whether to enforce types strictly.  
context: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

### parse\_file

```
classmethod PreProcessingPipeline.parse_file(path: str | Path, *, content_type: str |  
                                              None = None, encoding: str = 'utf8',  
                                              proto: DeprecatedParseProtocol | None  
                                              = None, allow_pickle: bool = False) →  
                                              Model
```

### parse\_obj

```
classmethod PreProcessingPipeline.parse_obj(obj: Any) → Model
```



**parse\_raw**

```
classmethod PreProcessingPipeline.parse_raw(b: str | bytes, *, content_type: str | None
= None, encoding: str = 'utf8', proto:
DeprecatedParseProtocol | None = None,
allow_pickle: bool = False) → Model
```

**schema**

```
classmethod PreProcessingPipeline.schema(by_alias: bool = True, ref_template: str =
'#{/$defs/{model}'} → Dict[str, Any]
```

**schema\_json**

```
classmethod PreProcessingPipeline.schema_json(*, by_alias: bool = True, ref_template:
str = '#{/$defs/{model}'}',
**dumps_kwargs: Any) → str
```

**update\_forward\_refs**

```
classmethod PreProcessingPipeline.update_forward_refs(**locals: Any) → None
```

**validate**

```
classmethod PreProcessingPipeline.validate(value: Any) → Model
```

**Methods Documentation**

```
actions: list[PipelineAction]
```

```
apply(original: DataArray) → DataArray
```

Apply all pre-processors on data.

**Parameters**

**original** (*xr.DataArray*) – The data to process.

**Return type**

*xr.DataArray*

```
classmethod construct(_fields_set: set[str] | None = None, **values: Any) → Model
```

```
copy(*, include: AbstractSet[IntStr] | Mapping[IntStr, Any] | None = None, exclude: AbstractSet[IntStr]
| Mapping[IntStr, Any] | None = None, update: Dict[str, Any] | None = None, deep: bool =
False) → Model
```

Returns a copy of the model.

!!! warning “Deprecated”

This method is now deprecated; use *model\_copy* instead.

If you need *include* or *exclude*, use:

```
`py data = self.model_dump(include=include, exclude=exclude,
round_trip=True) data = {**data, **(update or {})} copied = self.
model_validate(data) `
```

**Args:**

**include: Optional set or mapping**

specifying which fields to include in the copied model.

**exclude: Optional set or mapping**

specifying which fields to exclude in the copied model.

**update: Optional dictionary of field-value pairs to override field values**

in the copied model.

deep: If True, the values of fields that are Pydantic models will be deep copied.

**Returns:**

A copy of the model with included, excluded and updated fields as specified.

**correct\_baseline\_average**(*select: dict[str, slice | list[int] | int] | None = None, exclude: dict[str, slice | list[int] | int] | None = None*) → *PreProcessingPipeline*

Correct a dataset by subtracting the average over a part of the data.

**Parameters**

- **select** (*dict[str, slice | list[int] | int] | None*) – The selection to average as dictionary of dimension and indexer. The indexer can be a slice, a list or an integer value.
- **exclude** (*dict[str, slice | list[int] | int] | None*) – Excluded regions from the average as dictionary of dimension and indexer. The indexer can be a slice, a list or an integer value.

**Return type**

*PreProcessingPipeline*

**correct\_baseline\_value**(*value: float*) → *PreProcessingPipeline*

Correct a dataset by subtracting baseline value.

**Parameters**

**value** (*float*) – The value to subtract.

**Return type**

*PreProcessingPipeline*

**dict**(*\*, include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False*) → *Dict[str, Any]*

**classmethod from\_orm**(*obj: Any*) → *Model*

**json**(*\*, include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False, encoder: Callable[[Any], Any] | None = PydanticUndefined, models\_as\_dict: bool = PydanticUndefined, \*\*dumps\_kwargs: Any*) → *str*

**property model\_computed\_fields:** *dict[str, ComputedFieldInfo]*

Get the computed fields of this model instance.

**Returns:**

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

**model\_config:** *ClassVar[ConfigDict] = {}*

Configuration for the model, should be a dictionary conforming to *[ConfigDict][pydantic.config.ConfigDict]*.

**classmethod** `model_construct(_fields_set: set[str] | None = None, **values: Any) → Model`

Creates a new instance of the *Model* class with validated data.

Creates a new model setting `__dict__` and `__pydantic_fields_set__` from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if `Config.extra = 'allow'` was set since it adds all passed values

**Args:**

`_fields_set`: The set of field names accepted for the Model instance. `values`: Trusted or pre-validated data dictionary.

**Returns:**

A new instance of the *Model* class with validated data.

**model\_copy**(\*, `update: dict[str, Any] | None = None, deep: bool = False`) → *Model*

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#model\\_copy](https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy)

Returns a copy of the model.

**Args:**

**update**: Values to change/add in the new model. **Note: the data is not validated**

before creating the new model. You should trust this data.

**deep**: Set to *True* to make a deep copy of the model.

**Returns:**

New model instance.

**model\_dump**(\*, `mode: Literal['json', 'python'] | str = 'python', include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True`) → dict[str, Any]

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

**Args:**

**mode**: The mode in which to *python* should run.

If mode is 'json', the dictionary will only contain JSON serializable types. If mode is 'python', the dictionary may contain any Python objects.

**include**: A list of fields to include in the output. **exclude**: A list of fields to exclude from the output. **by\_alias**: Whether to use the field's alias in the dictionary key if defined. **exclude\_unset**: Whether to exclude fields that have not been explicitly set. **exclude\_defaults**:

Whether to exclude fields that are set to their default value from the output. **exclude\_none**: Whether to exclude fields that have a value of *None* from the output. **round\_trip**: Whether to enable serialization and deserialization round-trip support. **warnings**: Whether to log

warnings when invalid fields are encountered.

**Returns:**

A dictionary representation of the model.

**model\_dump\_json**(\*, `indent: int | None = None, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True`) → str

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump\\_json](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json)

Generates a JSON representation of the model using Pydantic's *to\_json* method.

**Args:**

**indent**: Indentation to use in the JSON output. If *None* is passed, the output will be compact. **include**: Field(s) to include in the JSON output. Can take either a string or set of strings. **exclude**: Field(s) to exclude from the JSON output. Can take either a string or

set of strings. `by_alias`: Whether to serialize using field aliases. `exclude_unset`: Whether to exclude fields that have not been explicitly set. `exclude_defaults`: Whether to exclude fields that have the default value. `exclude_none`: Whether to exclude fields that have a value of *None*. `round_trip`: Whether to use serialization/deserialization between JSON and class instance. `warnings`: Whether to show any warnings that occurred during serialization.

**Returns:**

A JSON string representation of the model.

**property** `model_extra`: `dict[str, Any] | None`

Get extra fields set during validation.

**Returns:**

A dictionary of extra fields, or *None* if `config.extra` is not set to “allow”.

**model\_fields**: `ClassVar[dict[str, FieldInfo]] = {'actions': FieldInfo(annotation=list[Annotated[Union[CorrectBaselineValue, CorrectBaselineAverage], FieldInfo(annotation=NoneType, required=True, discriminator='action')]], required=False, default_factory=list)}`

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces `Model.__fields__` from Pydantic V1.

**property** `model_fields_set`: `set[str]`

Returns the set of fields that have been explicitly set on this model instance.

**Returns:**

A set of strings representing the fields that have been set,  
i.e. that were not filled from defaults.

**classmethod** `model_json_schema`(`by_alias`: `bool = True`, `ref_template`: `str = '#/$defs/{model}'`, `schema_generator`: `type[~pydantic.json_schema.GenerateJsonSchema] = <class 'pydantic.json_schema.GenerateJsonSchema'>`, `mode`: `~typing.Literal['validation', 'serialization'] = 'validation'`)  $\rightarrow$  `dict[str, Any]`

Generates a JSON schema for a model class.

**Args:**

`by_alias`: Whether to use attribute aliases or not. `ref_template`: The reference template. `schema_generator`: To override the logic used to generate the JSON schema, as a subclass of

*GenerateJsonSchema* with your desired modifications

`mode`: The mode in which to generate the schema.

**Returns:**

The JSON schema for the given model class.

**classmethod** `model_parametrized_name`(`params`: `tuple[type[Any], ...]`)  $\rightarrow$  `str`

Compute the class name for parametrizations of generic classes.

This method can be overridden to achieve a custom naming scheme for generic BaseModels.

**Args:**

**params**: Tuple of types of the class. Given a generic class

*Model* with 2 type variables and a concrete model *Model*[*str*, *int*], the value (*str*, *int*) would be passed to *params*.

**Returns:**

String representing the new class where *params* are passed to *cls* as type variables.

**Raises:**

`TypeError`: Raised when trying to generate concrete names for non-generic models.

**model\_post\_init**(*\_BaseModel\_\_context: Any*) → *None*

Override this method to perform additional initialization after `__init__` and `model_construct`. This is useful if you want to do some validation that requires the entire model to be initialized.

**classmethod model\_rebuild**(\**, force: bool = False, raise\_errors: bool = True, \_parent\_namespace\_depth: int = 2, \_types\_namespace: dict[str, Any] | None = None*) → *bool | None*

Try to rebuild the pydantic-core schema for the model.

This may be necessary when one of the annotations is a ForwardRef which could not be resolved during the initial attempt to build the schema, and automatic rebuilding fails.

**Args:**

*force*: Whether to force the rebuilding of the model schema, defaults to *False*. *raise\_errors*: Whether to raise errors, defaults to *True*. *\_parent\_namespace\_depth*: The depth level of the parent namespace, defaults to 2. *\_types\_namespace*: The types namespace, defaults to *None*.

**Returns:**

Returns *None* if the schema is already “complete” and rebuilding was not required. If rebuilding *\_was\_* required, returns *True* if rebuilding was successful, otherwise *False*.

**classmethod model\_validate**(*obj: Any, \*, strict: bool | None = None, from\_attributes: bool | None = None, context: dict[str, Any] | None = None*) → *Model*

Validate a pydantic model instance.

**Args:**

*obj*: The object to validate. *strict*: Whether to raise an exception on invalid fields. *from\_attributes*: Whether to extract data from object attributes. *context*: Additional context to pass to the validator.

**Raises:**

*ValidationError*: If the object could not be validated.

**Returns:**

The validated model instance.

**classmethod model\_validate\_json**(*json\_data: str | bytes | bytearray, \*, strict: bool | None = None, context: dict[str, Any] | None = None*) → *Model*

Usage docs: <https://docs.pydantic.dev/2.5/concepts/json/#json-parsing>

Validate the given JSON data against the Pydantic model.

**Args:**

*json\_data*: The JSON data to validate. *strict*: Whether to enforce types strictly. *context*: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**Raises:**

*ValueError*: If *json\_data* is not a JSON string.

**classmethod model\_validate\_strings**(*obj: Any, \*, strict: bool | None = None, context: dict[str, Any] | None = None*) → *Model*

Validate the given object contains string data against the Pydantic model.

**Args:**

*obj*: The object contains string data to validate. *strict*: Whether to enforce types strictly. *context*: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**classmethod parse\_file**(*path: str | Path, \*, content\_type: str | None = None, encoding: str = 'utf8', proto: DeprecatedParseProtocol | None = None, allow\_pickle: bool = False*) → *Model*

```
classmethod parse_obj(obj: Any) → Model

classmethod parse_raw(b: str | bytes, *, content_type: str | None = None, encoding: str =
    'utf8', proto: DeprecatedParseProtocol | None = None, allow_pickle:
    bool = False) → Model

classmethod schema(by_alias: bool = True, ref_template: str = '#/$defs/{model}') → Dict[str,
    Any]

classmethod schema_json(*, by_alias: bool = True, ref_template: str = '#/$defs/{model}',
    **dumps_kwargs: Any) → str

classmethod update_forward_refs(**localns: Any) → None

classmethod validate(value: Any) → Model
```

## preprocessor

A pre-processor pipeline for data.

## Classes

### Summary

<i>CorrectBaselineAverage</i>	Corrects a dataset by subtracting the average over a part of the data.
<i>CorrectBaselineValue</i>	Corrects a dataset by subtracting baseline value.
<i>PreProcessor</i>	A base class for pre=processors.

### CorrectBaselineAverage

```
class glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage(*, action: Lit-
    eral['baseline-
    average'] =
    'baseline-
    average', select:
    dict[str, slice |
    list[int] | int] |
    None = None,
    exclude:
    dict[str, slice |
    list[int] | int] |
    None = None)
```

Bases: *PreProcessor*

Corrects a dataset by subtracting the average over a part of the data.

Create a new model by parsing and validating input data from keyword arguments.

Raises `[ValidationError][pydantic_core.ValidationError]` if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

## Attributes Summary

<code>model_computed_fields</code>	Get the computed fields of this model instance.
<code>model_config</code>	Configuration for the model, should be a dictionary conforming to <code>[ConfigDict][pydantic.config.ConfigDict]</code> .
<code>model_extra</code>	Get extra fields set during validation.
<code>model_fields</code>	Metadata about the fields defined on the model, mapping of field names to <code>[FieldInfo][pydantic.fields.FieldInfo]</code> .
<code>model_fields_set</code>	Returns the set of fields that have been explicitly set on this model instance.
<code>action</code>	
<code>select</code>	
<code>exclude</code>	

## model\_computed\_fields

`CorrectBaselineAverage.model_computed_fields`

Get the computed fields of this model instance.

**Returns:**

A dictionary of computed field names and their corresponding `ComputedFieldInfo` objects.

## model\_config

`CorrectBaselineAverage.model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True}`

Configuration for the model, should be a dictionary conforming to `[ConfigDict][pydantic.config.ConfigDict]`.

## model\_extra

`CorrectBaselineAverage.model_extra`

Get extra fields set during validation.

**Returns:**

A dictionary of extra fields, or `None` if `config.extra` is not set to “allow”.

### model\_fields

```
CorrectBaselineAverage.model_fields: ClassVar[dict[str, FieldInfo]] =  
{'action': FieldInfo(annotation=Literal['baseline-average'],  
required=False, default='baseline-average'), 'exclude':  
FieldInfo(annotation=Union[dict[str, Union[slice, list[int], int]],  
NoneType], required=False), 'select': FieldInfo(annotation=Union[dict[str,  
Union[slice, list[int], int]], NoneType], required=False)}
```

Metadata about the fields defined on the model, mapping of field names to [*Field-Info*][pydantic.fields.FieldInfo].

This replaces *Model.\_\_fields\_\_* from Pydantic V1.

### model\_fields\_set

CorrectBaselineAverage.model\_fields\_set

Returns the set of fields that have been explicitly set on this model instance.

**Returns:**

**A set of strings representing the fields that have been set,**  
i.e. that were not filled from defaults.

### action

CorrectBaselineAverage.action: Literal['baseline-average']

### select

CorrectBaselineAverage.select: dict[str, slice | list[int] | int] | None

### exclude

CorrectBaselineAverage.exclude: dict[str, slice | list[int] | int] | None



## Methods Summary

<code>apply</code>	Apply the pre-processor.
<code>construct</code>	
<code>copy</code>	Returns a copy of the model.
<code>dict</code>	
<code>from_orm</code>	
<code>json</code>	
<code>model_construct</code>	Creates a new instance of the <i>Model</i> class with validated data.
<code>model_copy</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy">https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy</a>
<code>model_dump</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump">https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump</a>
<code>model_dump_json</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json">https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json</a>
<code>model_json_schema</code>	Generates a JSON schema for a model class.
<code>model_parametrized_name</code>	Compute the class name for parametrizations of generic classes.
<code>model_post_init</code>	Override this method to perform additional initialization after <code>__init__</code> and <code>model_construct</code> .
<code>model_rebuild</code>	Try to rebuild the pydantic-core schema for the model.
<code>model_validate</code>	Validate a pydantic model instance.
<code>model_validate_json</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/json/#json-parsing">https://docs.pydantic.dev/2.5/concepts/json/#json-parsing</a>
<code>model_validate_strings</code>	Validate the given object contains string data against the Pydantic model.
<code>parse_file</code>	
<code>parse_obj</code>	
<code>parse_raw</code>	
<code>schema</code>	
<code>schema_json</code>	
<code>update_forward_refs</code>	
<code>validate</code>	

## apply

`CorrectBaselineAverage.apply(data: dataArray) → dataArray`

Apply the pre-processor.

**Parameters**

**data** (*xr.DataArray*) – The data to process.

**Return type**

*xr.DataArray*

## construct

**classmethod** `CorrectBaselineAverage.construct(_fields_set: set[str] | None = None,  
**values: Any) → Model`

## copy

`CorrectBaselineAverage.copy(*, include: AbstractSetIntStr | MappingIntStrAny | None =  
None, exclude: AbstractSetIntStr | MappingIntStrAny | None =  
None, update: Dict[str, Any] | None = None, deep: bool =  
False) → Model`

Returns a copy of the model.

**!!! warning “Deprecated”**

This method is now deprecated; use *model\_copy* instead.

If you need *include* or *exclude*, use:

```
`py data = self.model_dump(include=include, exclude=exclude,  
round_trip=True) data = {**data, **(update or {})} copied = self.  
model_validate(data) `
```

**Args:**

**include: Optional set or mapping**

specifying which fields to include in the copied model.

**exclude: Optional set or mapping**

specifying which fields to exclude in the copied model.

**update: Optional dictionary of field-value pairs to override field values**

in the copied model.

deep: If True, the values of fields that are Pydantic models will be deep copied.

**Returns:**

A copy of the model with included, excluded and updated fields as specified.

## dict

`CorrectBaselineAverage.dict(*, include: IncEx = None, exclude: IncEx = None, by_alias:  
bool = False, exclude_unset: bool = False, exclude_defaults:  
bool = False, exclude_none: bool = False) → Dict[str, Any]`

## from\_orm

**classmethod** `CorrectBaselineAverage.from_orm(obj: Any) → Model`

## json

`CorrectBaselineAverage.json(*, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, encoder: Callable[[Any], Any] | None = PydanticUndefined, models_as_dict: bool = PydanticUndefined, **dumps_kwargs: Any) → str`

## model\_construct

**classmethod** `CorrectBaselineAverage.model_construct(_fields_set: set[str] | None = None, **values: Any) → Model`

Creates a new instance of the *Model* class with validated data.

Creates a new model setting `__dict__` and `__pydantic_fields_set__` from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if `Config.extra = 'allow'` was set since it adds all passed values

**Args:**

`_fields_set`: The set of field names accepted for the Model instance. `values`: Trusted or pre-validated data dictionary.

**Returns:**

A new instance of the *Model* class with validated data.

## model\_copy

`CorrectBaselineAverage.model_copy(*, update: dict[str, Any] | None = None, deep: bool = False) → Model`

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#model\\_copy](https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy)

Returns a copy of the model.

**Args:**

**update**: Values to change/add in the new model. **Note: the data is not validated** before creating the new model. You should trust this data.  
**deep**: Set to *True* to make a deep copy of the model.

**Returns:**

New model instance.

## model\_dump

```
CorrectBaselineAverage.model_dump(*, mode: Literal['json', 'python'] | str = 'python',
                                  include: IncEx = None, exclude: IncEx = None,
                                  by_alias: bool = False, exclude_unset: bool = False,
                                  exclude_defaults: bool = False, exclude_none: bool =
                                  False, round_trip: bool = False, warnings: bool =
                                  True) → dict[str, Any]
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

### Args:

**mode:** The mode in which to *python* should run.

If mode is 'json', the dictionary will only contain JSON serializable types. If mode is 'python', the dictionary may contain any Python objects.

**include:** A list of fields to include in the output. **exclude:** A list of fields to exclude from the output. **by\_alias:** Whether to use the field's alias in the dictionary key if defined. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that are set to their default value from the output. **exclude\_none:** Whether to exclude fields that have a value of *None* from the output. **round\_trip:** Whether to enable serialization and deserialization round-trip support. **warnings:** Whether to log warnings when invalid fields are encountered.

### Returns:

A dictionary representation of the model.

## model\_dump\_json

```
CorrectBaselineAverage.model_dump_json(*, indent: int | None = None, include: IncEx =
None, exclude: IncEx = None, by_alias: bool =
False, exclude_unset: bool = False,
exclude_defaults: bool = False, exclude_none:
bool = False, round_trip: bool = False,
warnings: bool = True) → str
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump\\_json](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json)

Generates a JSON representation of the model using Pydantic's *to\_json* method.

### Args:

**indent:** Indentation to use in the JSON output. If *None* is passed, the output will be compact. **include:** Field(s) to include in the JSON output. Can take either a string or set of strings. **exclude:** Field(s) to exclude from the JSON output. Can take either a string or set of strings. **by\_alias:** Whether to serialize using field aliases. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that have the default value. **exclude\_none:** Whether to exclude fields that have a value of *None*. **round\_trip:** Whether to use serialization/deserialization between JSON and class instance. **warnings:** Whether to show any warnings that occurred during serialization.

### Returns:

A JSON string representation of the model.

## model\_json\_schema

```

classmethod CorrectBaselineAverage.model_json_schema(by_alias: bool = True,
                                                         ref_template: str =
                                                         '#/$defs/{model}',
                                                         schema_generator:
                                                         type[~pydantic.json_schema.GenerateJsonSchema]
                                                         = <class 'pydantic.json_schema.GenerateJsonSchema'>,
                                                         mode:
                                                         ~typing.Literal['validation',
                                                         'serialization'] =
                                                         'validation') → dict[str, Any]

```

Generates a JSON schema for a model class.

**Args:**

by\_alias: Whether to use attribute aliases or not. ref\_template: The reference template.  
 schema\_generator: To override the logic used to generate the JSON schema, as a subclass of  
     *GenerateJsonSchema* with your desired modifications  
 mode: The mode in which to generate the schema.

**Returns:**

The JSON schema for the given model class.

## model\_parametrized\_name

```

classmethod CorrectBaselineAverage.model_parametrized_name(params:
                                                             tuple[type[Any], ...])
                                                             → str

```

Compute the class name for parametrizations of generic classes.

This method can be overridden to achieve a custom naming scheme for generic BaseModels.

**Args:**

**params: Tuple of types of the class. Given a generic class**  
     *Model* with 2 type variables and a concrete model *Model[str, int]*, the value (*str*, *int*)  
     would be passed to *params*.

**Returns:**

String representing the new class where *params* are passed to *cls* as type variables.

**Raises:**

*TypeError*: Raised when trying to generate concrete names for non-generic models.

## model\_post\_init

```

CorrectBaselineAverage.model_post_init(_BaseModel__context: Any) → None

```

Override this method to perform additional initialization after `__init__` and `model_construct`.  
 This is useful if you want to do some validation that requires the entire model to be initialized.

## model\_rebuild

```
classmethod CorrectBaselineAverage.model_rebuild(*, force: bool = False,
                                                raise_errors: bool = True,
                                                _parent_namespace_depth: int =
                                                2, _types_namespace: dict[str,
                                                Any] | None = None) → bool |
                                                None
```

Try to rebuild the pydantic-core schema for the model.

This may be necessary when one of the annotations is a ForwardRef which could not be resolved during the initial attempt to build the schema, and automatic rebuilding fails.

**Args:**

force: Whether to force the rebuilding of the model schema, defaults to *False*. raise\_errors: Whether to raise errors, defaults to *True*. \_parent\_namespace\_depth: The depth level of the parent namespace, defaults to 2. \_types\_namespace: The types namespace, defaults to *None*.

**Returns:**

Returns *None* if the schema is already “complete” and rebuilding was not required. If rebuilding was required, returns *True* if rebuilding was successful, otherwise *False*.

## model\_validate

```
classmethod CorrectBaselineAverage.model_validate(obj: Any, *, strict: bool | None =
                                                None, from_attributes: bool |
                                                None = None, context: dict[str,
                                                Any] | None = None) → Model
```

Validate a pydantic model instance.

**Args:**

obj: The object to validate. strict: Whether to raise an exception on invalid fields. from\_attributes: Whether to extract data from object attributes. context: Additional context to pass to the validator.

**Raises:**

ValidationError: If the object could not be validated.

**Returns:**

The validated model instance.

## model\_validate\_json

```
classmethod CorrectBaselineAverage.model_validate_json(json_data: str | bytes |
                                                         bytearray, *, strict: bool |
                                                         None = None, context:
                                                         dict[str, Any] | None =
                                                         None) → Model
```

Usage docs: <https://docs.pydantic.dev/2.5/concepts/json/#json-parsing>

Validate the given JSON data against the Pydantic model.

**Args:**

json\_data: The JSON data to validate. strict: Whether to enforce types strictly. context: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**Raises:**

ValueError: If *json\_data* is not a JSON string.

**model\_validate\_strings**

**classmethod** CorrectBaselineAverage.**model\_validate\_strings**(*obj*: Any, \*, *strict*: bool | None = None, *context*: dict[str, Any] | None = None) → Model

Validate the given object contains string data against the Pydantic model.

**Args:**

*obj*: The object contains string data to validate. *strict*: Whether to enforce types strictly. *context*: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**parse\_file**

**classmethod** CorrectBaselineAverage.**parse\_file**(*path*: str | Path, \*, *content\_type*: str | None = None, *encoding*: str = 'utf8', *proto*: DeprecatedParseProtocol | None = None, *allow\_pickle*: bool = False) → Model

**parse\_obj**

**classmethod** CorrectBaselineAverage.**parse\_obj**(*obj*: Any) → Model

**parse\_raw**

**classmethod** CorrectBaselineAverage.**parse\_raw**(*b*: str | bytes, \*, *content\_type*: str | None = None, *encoding*: str = 'utf8', *proto*: DeprecatedParseProtocol | None = None, *allow\_pickle*: bool = False) → Model

**schema**

**classmethod** CorrectBaselineAverage.**schema**(*by\_alias*: bool = True, *ref\_template*: str = '#/\$defs/{model}') → Dict[str, Any]

### schema\_json

```
classmethod CorrectBaselineAverage.schema_json(*, by_alias: bool = True,
                                                ref_template: str = '#/$defs/{model}',
                                                **kwargs: Any) → str
```

### update\_forward\_refs

```
classmethod CorrectBaselineAverage.update_forward_refs(**localns: Any) → None
```

### validate

```
classmethod CorrectBaselineAverage.validate(value: Any) → Model
```

## Methods Documentation

### class Config

Bases: `object`

Config for BaseModel.

**arbitrary\_types\_allowed** = True

**action**: `Literal['baseline-average']`

**apply**(data: `DataArray`) → `DataArray`

Apply the pre-processor.

**Parameters**

**data** (`xr.DataArray`) – The data to process.

**Return type**

`xr.DataArray`

```
classmethod construct(_fields_set: set[str] | None = None, **values: Any) → Model
```

```
copy(*, include: AbstractSetIntStr | MappingIntStrAny | None = None, exclude: AbstractSetIntStr
    | MappingIntStrAny | None = None, update: Dict[str, Any] | None = None, deep: bool =
    False) → Model
```

Returns a copy of the model.

**!!! warning “Deprecated”**

This method is now deprecated; use `model_copy` instead.

If you need `include` or `exclude`, use:

```
`py data = self.model_dump(include=include, exclude=exclude,
    round_trip=True) data = {**data, **(update or {})} copied = self.
    model_validate(data) `
```

**Args:**

**include**: **Optional set or mapping**

specifying which fields to include in the copied model.

**exclude**: **Optional set or mapping**

specifying which fields to exclude in the copied model.

**update**: **Optional dictionary of field-value pairs to override field values**

in the copied model.



deep: If True, the values of fields that are Pydantic models will be deep copied.

**Returns:**

A copy of the model with included, excluded and updated fields as specified.

**dict**(\*, include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False) → Dict[str, Any]

**exclude:** dict[str, slice | list[int] | int] | None

**classmethod from\_orm**(obj: Any) → Model

**json**(\*, include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False, encoder: Callable[[Any], Any] | None = PydanticUndefined, models\_as\_dict: bool = PydanticUndefined, \*\*dumps\_kwargs: Any) → str

**property model\_computed\_fields:** dict[str, ComputedFieldInfo]

Get the computed fields of this model instance.

**Returns:**

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

**model\_config:** ClassVar[ConfigDict] = {'arbitrary\_types\_allowed': True}

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

**classmethod model\_construct**(\_fields\_set: set[str] | None = None, \*\*values: Any) → Model

Creates a new instance of the *Model* class with validated data.

Creates a new model setting `__dict__` and `__pydantic_fields_set__` from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if *Config.extra* = 'allow' was set since it adds all passed values

**Args:**

`_fields_set`: The set of field names accepted for the Model instance. `values`: Trusted or pre-validated data dictionary.

**Returns:**

A new instance of the *Model* class with validated data.

**model\_copy**(\*, update: dict[str, Any] | None = None, deep: bool = False) → Model

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#model\\_copy](https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy)

Returns a copy of the model.

**Args:**

**update:** Values to change/add in the new model. **Note: the data is not validated** before creating the new model. You should trust this data.  
**deep:** Set to *True* to make a deep copy of the model.

**Returns:**

New model instance.

**model\_dump**(\*, mode: Literal['json', 'python'] | str = 'python', include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False, round\_trip: bool = False, warnings: bool = True) → dict[str, Any]

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

**Args:**

**mode:** The mode in which *to\_python* should run.

If mode is 'json', the dictionary will only contain JSON serializable types. If mode is 'python', the dictionary may contain any Python objects.

**include:** A list of fields to include in the output. **exclude:** A list of fields to exclude from the output. **by\_alias:** Whether to use the field's alias in the dictionary key if defined. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that are set to their default value from the output. **exclude\_none:** Whether to exclude fields that have a value of *None* from the output. **round\_trip:** Whether to enable serialization and deserialization round-trip support. **warnings:** Whether to log warnings when invalid fields are encountered.

**Returns:**

A dictionary representation of the model.

```
model_dump_json(* , indent: int | None = None, include: IncEx = None, exclude: IncEx = None,  
by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool  
= False, exclude_none: bool = False, round_trip: bool = False, warnings:  
bool = True) → str
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump\\_json](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json)

Generates a JSON representation of the model using Pydantic's *to\_json* method.

**Args:**

**indent:** Indentation to use in the JSON output. If *None* is passed, the output will be compact. **include:** Field(s) to include in the JSON output. Can take either a string or set of strings. **exclude:** Field(s) to exclude from the JSON output. Can take either a string or set of strings. **by\_alias:** Whether to serialize using field aliases. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that have the default value. **exclude\_none:** Whether to exclude fields that have a value of *None*. **round\_trip:** Whether to use serialization/deserialization between JSON and class instance. **warnings:** Whether to show any warnings that occurred during serialization.

**Returns:**

A JSON string representation of the model.

**property model\_extra:** `dict[str, Any] | None`

Get extra fields set during validation.

**Returns:**

A dictionary of extra fields, or *None* if *config.extra* is not set to "allow".

```
model_fields: ClassVar[dict[str, FieldInfo]] = {'action':  
FieldInfo(annotation=Literal['baseline-average'], required=False,  
default='baseline-average'), 'exclude':  
FieldInfo(annotation=Union[dict[str, Union[slice, list[int], int]],  
NoneType], required=False), 'select': FieldInfo(annotation=Union[dict[str,  
Union[slice, list[int], int]], NoneType], required=False)}
```

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.\_\_fields\_\_* from Pydantic V1.

**property model\_fields\_set:** `set[str]`

Returns the set of fields that have been explicitly set on this model instance.

**Returns:**

A set of strings representing the fields that have been set,  
i.e. that were not filled from defaults.

```
classmethod model_json_schema(by_alias: bool = True, ref_template: str =
                              '#/$defs/{model}', schema_generator:
                              type[~pydantic.json_schema.GenerateJsonSchema] =
                              <class 'pydantic.json_schema.GenerateJsonSchema'>,
                              mode: ~typing.Literal['validation', 'serialization'] =
                              'validation') → dict[str, Any]
```

Generates a JSON schema for a model class.

**Args:**

by\_alias: Whether to use attribute aliases or not. ref\_template: The reference template.  
 schema\_generator: To override the logic used to generate the JSON schema, as a subclass  
 of

*GenerateJsonSchema* with your desired modifications

mode: The mode in which to generate the schema.

**Returns:**

The JSON schema for the given model class.

```
classmethod model_parametrized_name(params: tuple[type[Any], ...]) → str
```

Compute the class name for parametrizations of generic classes.

This method can be overridden to achieve a custom naming scheme for generic BaseModels.

**Args:**

**params: Tuple of types of the class. Given a generic class**

*Model* with 2 type variables and a concrete model *Model[str, int]*, the value (*str, int*)  
 would be passed to *params*.

**Returns:**

String representing the new class where *params* are passed to *cls* as type variables.

**Raises:**

*TypeError*: Raised when trying to generate concrete names for non-generic models.

```
model_post_init(_BaseModel__context: Any) → None
```

Override this method to perform additional initialization after `__init__` and `model_construct`.

This is useful if you want to do some validation that requires the entire model to be initialized.

```
classmethod model_rebuild(*, force: bool = False, raise_errors: bool = True,
                          _parent_namespace_depth: int = 2, _types_namespace: dict[str,
                          Any] | None = None) → bool | None
```

Try to rebuild the pydantic-core schema for the model.

This may be necessary when one of the annotations is a `ForwardRef` which could not be resolved during the initial attempt to build the schema, and automatic rebuilding fails.

**Args:**

force: Whether to force the rebuilding of the model schema, defaults to *False*. raise\_errors:  
 Whether to raise errors, defaults to *True*. \_parent\_namespace\_depth: The depth level of  
 the parent namespace, defaults to 2. \_types\_namespace: The types namespace, defaults to  
*None*.

**Returns:**

Returns *None* if the schema is already “complete” and rebuilding was not required. If re-  
 building *\_was\_* required, returns *True* if rebuilding was successful, otherwise *False*.

```
classmethod model_validate(obj: Any, *, strict: bool | None = None, from_attributes: bool |
                             None = None, context: dict[str, Any] | None = None) → Model
```

Validate a pydantic model instance.

**Args:**

obj: The object to validate. strict: Whether to raise an exception on invalid fields.  
 from\_attributes: Whether to extract data from object attributes. context: Additional context  
 to pass to the validator.

**Raises:**

ValidationError: If the object could not be validated.

**Returns:**

The validated model instance.

```
classmethod model_validate_json(json_data: str | bytes | bytearray, *, strict: bool | None = None, context: dict[str, Any] | None = None) → Model
```

Usage docs: <https://docs.pydantic.dev/2.5/concepts/json/#json-parsing>

Validate the given JSON data against the Pydantic model.

**Args:**

json\_data: The JSON data to validate. strict: Whether to enforce types strictly. context: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**Raises:**

ValueError: If *json\_data* is not a JSON string.

```
classmethod model_validate_strings(obj: Any, *, strict: bool | None = None, context: dict[str, Any] | None = None) → Model
```

Validate the given object contains string data against the Pydantic model.

**Args:**

obj: The object contains string data to validate. strict: Whether to enforce types strictly. context: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

```
classmethod parse_file(path: str | Path, *, content_type: str | None = None, encoding: str = 'utf8', proto: DeprecatedParseProtocol | None = None, allow_pickle: bool = False) → Model
```

```
classmethod parse_obj(obj: Any) → Model
```

```
classmethod parse_raw(b: str | bytes, *, content_type: str | None = None, encoding: str = 'utf8', proto: DeprecatedParseProtocol | None = None, allow_pickle: bool = False) → Model
```

```
classmethod schema(by_alias: bool = True, ref_template: str = '#/$defs/{model}') → Dict[str, Any]
```

```
classmethod schema_json(*, by_alias: bool = True, ref_template: str = '#/$defs/{model}', **kwargs: Any) → str
```

```
select: dict[str, slice | list[int] | int] | None
```

```
classmethod update_forward_refs(**localns: Any) → None
```

```
classmethod validate(value: Any) → Model
```

### CorrectBaselineValue

```
class glotaran.io.preprocessor.preprocessor.CorrectBaselineValue(*, action:
    Literal['baseline-
    value'] =
    'baseline-value',
    value: float)
```

Bases: *PreProcessor*

Corrects a dataset by subtracting baseline value.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

### Attributes Summary

<i>model_computed_fields</i>	Get the computed fields of this model instance.
<i>model_config</i>	Configuration for the model, should be a dictionary conforming to [ <i>ConfigDict</i> ][pydantic.config.ConfigDict].
<i>model_extra</i>	Get extra fields set during validation.
<i>model_fields</i>	Metadata about the fields defined on the model, mapping of field names to [ <i>FieldInfo</i> ][pydantic.fields.FieldInfo].
<i>model_fields_set</i>	Returns the set of fields that have been explicitly set on this model instance.
<i>action</i>	
<i>value</i>	

### model\_computed\_fields

CorrectBaselineValue.**model\_computed\_fields**

Get the computed fields of this model instance.

**Returns:**

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

### model\_config

```
CorrectBaselineValue.model_config: ClassVar[ConfigDict] =  
{'arbitrary_types_allowed': True}
```

Configuration for the model, should be a dictionary conforming to `[ConfigDict][pydantic.config.ConfigDict]`.

### model\_extra

```
CorrectBaselineValue.model_extra
```

Get extra fields set during validation.

**Returns:**

A dictionary of extra fields, or *None* if `config.extra` is not set to “allow”.

### model\_fields

```
CorrectBaselineValue.model_fields: ClassVar[dict[str, FieldInfo]] =  
{'action': FieldInfo(annotation=Literal['baseline-value'], required=False,  
default='baseline-value'), 'value': FieldInfo(annotation=float,  
required=True)}
```

Metadata about the fields defined on the model, mapping of field names to `[FieldInfo][pydantic.fields.FieldInfo]`.

This replaces `Model.__fields__` from Pydantic V1.

### model\_fields\_set

```
CorrectBaselineValue.model_fields_set
```

Returns the set of fields that have been explicitly set on this model instance.

**Returns:**

A set of strings representing the fields that have been set,  
i.e. that were not filled from defaults.

### action

```
CorrectBaselineValue.action: Literal['baseline-value']
```

### value

```
CorrectBaselineValue.value: float
```

## Methods Summary

<i>apply</i>	Apply the pre-processor.
<i>construct</i>	
<i>copy</i>	Returns a copy of the model.
<i>dict</i>	
<i>from_orm</i>	
<i>json</i>	
<i>model_construct</i>	Creates a new instance of the <i>Model</i> class with validated data.
<i>model_copy</i>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy">https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy</a>
<i>model_dump</i>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump">https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump</a>
<i>model_dump_json</i>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json">https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json</a>
<i>model_json_schema</i>	Generates a JSON schema for a model class.
<i>model_parametrized_name</i>	Compute the class name for parametrizations of generic classes.
<i>model_post_init</i>	Override this method to perform additional initialization after <i>__init__</i> and <i>model_construct</i> .
<i>model_rebuild</i>	Try to rebuild the pydantic-core schema for the model.
<i>model_validate</i>	Validate a pydantic model instance.
<i>model_validate_json</i>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/json/#json-parsing">https://docs.pydantic.dev/2.5/concepts/json/#json-parsing</a>
<i>model_validate_strings</i>	Validate the given object contains string data against the Pydantic model.
<i>parse_file</i>	
<i>parse_obj</i>	
<i>parse_raw</i>	
<i>schema</i>	
<i>schema_json</i>	
<i>update_forward_refs</i>	
<i>validate</i>	

## apply

`CorrectBaselineValue.apply(data: dataArray) → dataArray`

Apply the pre-processor.

### Parameters

**data** (*xr.DataArray*) – The data to process.

### Return type

*xr.DataArray*

## construct

**classmethod** `CorrectBaselineValue.construct(_fields_set: set[str] | None = None,  
**values: Any) → Model`

## copy

`CorrectBaselineValue.copy(*, include: AbstractSetIntStr | MappingIntStrAny | None = None,  
exclude: AbstractSetIntStr | MappingIntStrAny | None = None,  
update: Dict[str, Any] | None = None, deep: bool = False) →  
Model`

Returns a copy of the model.

### !!! warning “Deprecated”

This method is now deprecated; use *model\_copy* instead.

If you need *include* or *exclude*, use:

```
`py data = self.model_dump(include=include, exclude=exclude,  
round_trip=True) data = {**data, **(update or {})} copied = self.  
model_validate(data) `
```

### Args:

#### **include: Optional set or mapping**

specifying which fields to include in the copied model.

#### **exclude: Optional set or mapping**

specifying which fields to exclude in the copied model.

#### **update: Optional dictionary of field-value pairs to override field values**

in the copied model.

deep: If True, the values of fields that are Pydantic models will be deep copied.

### Returns:

A copy of the model with included, excluded and updated fields as specified.

## dict

`CorrectBaselineValue.dict(*, include: IncEx = None, exclude: IncEx = None, by_alias:  
bool = False, exclude_unset: bool = False, exclude_defaults:  
bool = False, exclude_none: bool = False) → Dict[str, Any]`



## from\_orm

**classmethod** `CorrectBaselineValue.from_orm(obj: Any) → Model`

## json

`CorrectBaselineValue.json(*, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, encoder: Callable[[Any], Any] | None = PydanticUndefined, models_as_dict: bool = PydanticUndefined, **kwargs: Any) → str`

## model\_construct

**classmethod** `CorrectBaselineValue.model_construct(_fields_set: set[str] | None = None, **values: Any) → Model`

Creates a new instance of the *Model* class with validated data.

Creates a new model setting `__dict__` and `__pydantic_fields_set__` from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if `Config.extra = 'allow'` was set since it adds all passed values

**Args:**

`_fields_set`: The set of field names accepted for the Model instance. `values`: Trusted or pre-validated data dictionary.

**Returns:**

A new instance of the *Model* class with validated data.

## model\_copy

`CorrectBaselineValue.model_copy(*, update: dict[str, Any] | None = None, deep: bool = False) → Model`

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#model\\_copy](https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy)

Returns a copy of the model.

**Args:**

**update**: Values to change/add in the new model. **Note: the data is not validated** before creating the new model. You should trust this data.  
**deep**: Set to *True* to make a deep copy of the model.

**Returns:**

New model instance.

## model\_dump

```
CorrectBaselineValue.model_dump(*, mode: Literal['json', 'python'] | str = 'python', include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True) → dict[str, Any]
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

### Args:

**mode: The mode in which to *python* should run.**

If mode is 'json', the dictionary will only contain JSON serializable types. If mode is 'python', the dictionary may contain any Python objects.

**include:** A list of fields to include in the output. **exclude:** A list of fields to exclude from the output. **by\_alias:** Whether to use the field's alias in the dictionary key if defined. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that are set to their default value from the output. **exclude\_none:** Whether to exclude fields that have a value of *None* from the output. **round\_trip:** Whether to enable serialization and deserialization round-trip support. **warnings:** Whether to log warnings when invalid fields are encountered.

### Returns:

A dictionary representation of the model.

## model\_dump\_json

```
CorrectBaselineValue.model_dump_json(*, indent: int | None = None, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True) → str
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump\\_json](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json)

Generates a JSON representation of the model using Pydantic's *to\_json* method.

### Args:

**indent:** Indentation to use in the JSON output. If *None* is passed, the output will be compact. **include:** Field(s) to include in the JSON output. Can take either a string or set of strings. **exclude:** Field(s) to exclude from the JSON output. Can take either a string or set of strings. **by\_alias:** Whether to serialize using field aliases. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that have the default value. **exclude\_none:** Whether to exclude fields that have a value of *None*. **round\_trip:** Whether to use serialization/deserialization between JSON and class instance. **warnings:** Whether to show any warnings that occurred during serialization.

### Returns:

A JSON string representation of the model.

## model\_json\_schema

```

classmethod CorrectBaselineValue.model_json_schema(by_alias: bool = True,
                                                    ref_template: str =
                                                    '#/$defs/{model}',
                                                    schema_generator:
                                                    type[~pydantic.json_schema.GenerateJsonSchema]
                                                    = <class 'pydantic.json_schema.GenerateJsonSchema'>,
                                                    mode:
                                                    ~typing.Literal['validation',
                                                    'serialization'] = 'validation')
                                                    → dict[str, Any]

```

Generates a JSON schema for a model class.

**Args:**

by\_alias: Whether to use attribute aliases or not. ref\_template: The reference template.  
 schema\_generator: To override the logic used to generate the JSON schema, as a subclass  
 of *GenerateJsonSchema* with your desired modifications  
 mode: The mode in which to generate the schema.

**Returns:**

The JSON schema for the given model class.

## model\_parametrized\_name

```

classmethod CorrectBaselineValue.model_parametrized_name(params:
                                                           tuple[type[Any], ...])
                                                           → str

```

Compute the class name for parametrizations of generic classes.

This method can be overridden to achieve a custom naming scheme for generic BaseModels.

**Args:**

**params: Tuple of types of the class. Given a generic class**  
*Model* with 2 type variables and a concrete model *Model*[*str*, *int*], the value (*str*, *int*)  
 would be passed to *params*.

**Returns:**

String representing the new class where *params* are passed to *cls* as type variables.

**Raises:**

*TypeError*: Raised when trying to generate concrete names for non-generic models.

## model\_post\_init

```

CorrectBaselineValue.model_post_init(_BaseModel__context: Any) → None

```

Override this method to perform additional initialization after `__init__` and `model_construct`.  
 This is useful if you want to do some validation that requires the entire model to be initialized.

## model\_rebuild

```
classmethod CorrectBaselineValue.model_rebuild(*force: bool = False, raise_errors: bool = True,  
_parent_namespace_depth: int = 2,  
_types_namespace: dict[str, Any] | None = None) → bool | None
```

Try to rebuild the pydantic-core schema for the model.

This may be necessary when one of the annotations is a ForwardRef which could not be resolved during the initial attempt to build the schema, and automatic rebuilding fails.

**Args:**

*force*: Whether to force the rebuilding of the model schema, defaults to *False*. *raise\_errors*: Whether to raise errors, defaults to *True*. *\_parent\_namespace\_depth*: The depth level of the parent namespace, defaults to 2. *\_types\_namespace*: The types namespace, defaults to *None*.

**Returns:**

Returns *None* if the schema is already “complete” and rebuilding was not required. If rebuilding *\_was\_ required*, returns *True* if rebuilding was successful, otherwise *False*.

## model\_validate

```
classmethod CorrectBaselineValue.model_validate(obj: Any, *, strict: bool | None = None, from_attributes: bool | None = None, context: dict[str, Any] | None = None) → Model
```

Validate a pydantic model instance.

**Args:**

*obj*: The object to validate. *strict*: Whether to raise an exception on invalid fields. *from\_attributes*: Whether to extract data from object attributes. *context*: Additional context to pass to the validator.

**Raises:**

ValidationError: If the object could not be validated.

**Returns:**

The validated model instance.

## model\_validate\_json

```
classmethod CorrectBaselineValue.model_validate_json(json_data: str | bytes | bytearray, *, strict: bool | None = None, context: dict[str, Any] | None = None) → Model
```

Usage docs: <https://docs.pydantic.dev/2.5/concepts/json/#json-parsing>

Validate the given JSON data against the Pydantic model.

**Args:**

*json\_data*: The JSON data to validate. *strict*: Whether to enforce types strictly. *context*: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**Raises:**

ValueError: If *json\_data* is not a JSON string.

**model\_validate\_strings**

**classmethod** CorrectBaselineValue.**model\_validate\_strings**(*obj*: Any, \*, *strict*: bool | None = None, *context*: dict[str, Any] | None = None) → Model

Validate the given object contains string data against the Pydantic model.

**Args:**

*obj*: The object contains string data to validate. *strict*: Whether to enforce types strictly. *context*: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**parse\_file**

**classmethod** CorrectBaselineValue.**parse\_file**(*path*: str | Path, \*, *content\_type*: str | None = None, *encoding*: str = 'utf8', *proto*: DeprecatedParseProtocol | None = None, *allow\_pickle*: bool = False) → Model

**parse\_obj**

**classmethod** CorrectBaselineValue.**parse\_obj**(*obj*: Any) → Model

**parse\_raw**

**classmethod** CorrectBaselineValue.**parse\_raw**(*b*: str | bytes, \*, *content\_type*: str | None = None, *encoding*: str = 'utf8', *proto*: DeprecatedParseProtocol | None = None, *allow\_pickle*: bool = False) → Model

**schema**

**classmethod** CorrectBaselineValue.**schema**(*by\_alias*: bool = True, *ref\_template*: str = '#/\$defs/{model}') → Dict[str, Any]

### schema\_json

```
classmethod CorrectBaselineValue.schema_json(*, by_alias: bool = True, ref_template:
    str = '#/$defs/{model}',
    **kwargs: Any) → str
```

### update\_forward\_refs

```
classmethod CorrectBaselineValue.update_forward_refs(**kwargs: Any) → None
```

### validate

```
classmethod CorrectBaselineValue.validate(value: Any) → Model
```

## Methods Documentation

### class Config

Bases: `object`

Config for BaseModel.

**arbitrary\_types\_allowed** = True

**action**: `Literal['baseline-value']`

**apply**(data: `DataArray`) → `DataArray`

Apply the pre-processor.

**Parameters**

**data** (`xr.DataArray`) – The data to process.

**Return type**

`xr.DataArray`

```
classmethod construct(_fields_set: set[str] | None = None, **values: Any) → Model
```

```
copy(*, include: AbstractSetIntStr | MappingIntStrAny | None = None, exclude: AbstractSetIntStr
    | MappingIntStrAny | None = None, update: Dict[str, Any] | None = None, deep: bool =
    False) → Model
```

Returns a copy of the model.

**!!! warning “Deprecated”**

This method is now deprecated; use `model_copy` instead.

If you need `include` or `exclude`, use:

```
`py data = self.model_dump(include=include, exclude=exclude,
    round_trip=True) data = {**data, **(update or {})} copied = self.
    model_validate(data) `
```

**Args:**

**include**: **Optional set or mapping**

specifying which fields to include in the copied model.

**exclude**: **Optional set or mapping**

specifying which fields to exclude in the copied model.

**update**: **Optional dictionary of field-value pairs to override field values**  
in the copied model.

deep: If True, the values of fields that are Pydantic models will be deep copied.

**Returns:**

A copy of the model with included, excluded and updated fields as specified.

**dict**(\*, include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False) → Dict[str, Any]

**classmethod from\_orm**(obj: Any) → Model

**json**(\*, include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False, encoder: Callable[[Any], Any] | None = PydanticUndefined, models\_as\_dict: bool = PydanticUndefined, \*\*dumps\_kwargs: Any) → str

**property model\_computed\_fields:** Dict[str, ComputedFieldInfo]

Get the computed fields of this model instance.

**Returns:**

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

**model\_config:** ClassVar[ConfigDict] = {'arbitrary\_types\_allowed': True}

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

**classmethod model\_construct**(\_fields\_set: set[str] | None = None, \*\*values: Any) → Model

Creates a new instance of the *Model* class with validated data.

Creates a new model setting `__dict__` and `__pydantic_fields_set__` from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if *Config.extra* = 'allow' was set since it adds all passed values

**Args:**

`_fields_set`: The set of field names accepted for the Model instance. `values`: Trusted or pre-validated data dictionary.

**Returns:**

A new instance of the *Model* class with validated data.

**model\_copy**(\*, update: dict[str, Any] | None = None, deep: bool = False) → Model

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#model\\_copy](https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy)

Returns a copy of the model.

**Args:**

**update:** Values to change/add in the new model. **Note: the data is not validated**

before creating the new model. You should trust this data.

`deep`: Set to *True* to make a deep copy of the model.

**Returns:**

New model instance.

**model\_dump**(\*, mode: Literal['json', 'python'] | str = 'python', include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False, round\_trip: bool = False, warnings: bool = True) → dict[str, Any]

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

**Args:**

**mode:** The mode in which to *python* should run.

If mode is 'json', the dictionary will only contain JSON serializable types. If mode is 'python', the dictionary may contain any Python objects.

include: A list of fields to include in the output. exclude: A list of fields to exclude from the output. by\_alias: Whether to use the field's alias in the dictionary key if defined. exclude\_unset: Whether to exclude fields that have not been explicitly set. exclude\_defaults: Whether to exclude fields that are set to their default value from the output. exclude\_none: Whether to exclude fields that have a value of *None* from the output. round\_trip: Whether to enable serialization and deserialization round-trip support. warnings: Whether to log warnings when invalid fields are encountered.

**Returns:**

A dictionary representation of the model.

```
model_dump_json(*, indent: int | None = None, include: IncEx = None, exclude: IncEx = None,
                 by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool
                 = False, exclude_none: bool = False, round_trip: bool = False, warnings:
                 bool = True) → str
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump\\_json](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json)

Generates a JSON representation of the model using Pydantic's *to\_json* method.

**Args:**

indent: Indentation to use in the JSON output. If *None* is passed, the output will be compact. include: Field(s) to include in the JSON output. Can take either a string or set of strings. exclude: Field(s) to exclude from the JSON output. Can take either a string or set of strings. by\_alias: Whether to serialize using field aliases. exclude\_unset: Whether to exclude fields that have not been explicitly set. exclude\_defaults: Whether to exclude fields that have the default value. exclude\_none: Whether to exclude fields that have a value of *None*. round\_trip: Whether to use serialization/deserialization between JSON and class instance. warnings: Whether to show any warnings that occurred during serialization.

**Returns:**

A JSON string representation of the model.

```
property model_extra: dict[str, Any] | None
```

Get extra fields set during validation.

**Returns:**

A dictionary of extra fields, or *None* if *config.extra* is not set to “allow”.

```
model_fields: ClassVar[dict[str, FieldInfo]] = {'action':
FieldInfo(annotation=Literal['baseline-value'], required=False,
default='baseline-value'), 'value': FieldInfo(annotation=float,
required=True)}
```

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.\_\_fields\_\_* from Pydantic V1.

```
property model_fields_set: set[str]
```

Returns the set of fields that have been explicitly set on this model instance.

**Returns:**

A set of strings representing the fields that have been set,  
i.e. that were not filled from defaults.

```
classmethod model_json_schema(by_alias: bool = True, ref_template: str =
                              '#/$defs/{model}', schema_generator:
                              type[~pydantic.json_schema.GenerateJsonSchema] =
                              <class 'pydantic.json_schema.GenerateJsonSchema'>,
                              mode: ~typing.Literal['validation', 'serialization'] =
                              'validation') → dict[str, Any]
```

Generates a JSON schema for a model class.



**Args:**

`by_alias`: Whether to use attribute aliases or not. `ref_template`: The reference template. `schema_generator`: To override the logic used to generate the JSON schema, as a subclass of

`GenerateJsonSchema` with your desired modifications

`mode`: The mode in which to generate the schema.

**Returns:**

The JSON schema for the given model class.

**classmethod `model_parametrized_name`**(*params*: *tuple*[*type*[*Any*], ...]) → *str*

Compute the class name for parametrizations of generic classes.

This method can be overridden to achieve a custom naming scheme for generic BaseModels.

**Args:**

**params**: Tuple of types of the class. Given a generic class

`Model` with 2 type variables and a concrete model `Model[str, int]`, the value `(str, int)` would be passed to `params`.

**Returns:**

String representing the new class where `params` are passed to `cls` as type variables.

**Raises:**

`TypeError`: Raised when trying to generate concrete names for non-generic models.

**method `model_post_init`**(*\_BaseModel\_\_context*: *Any*) → *None*

Override this method to perform additional initialization after `__init__` and `model_construct`.

This is useful if you want to do some validation that requires the entire model to be initialized.

**classmethod `model_rebuild`**(*\**, *force*: *bool* = *False*, *raise\_errors*: *bool* = *True*, *\_parent\_namespace\_depth*: *int* = 2, *\_types\_namespace*: *dict*[*str*, *Any*] | *None* = *None*) → *bool* | *None*

Try to rebuild the pydantic-core schema for the model.

This may be necessary when one of the annotations is a `ForwardRef` which could not be resolved during the initial attempt to build the schema, and automatic rebuilding fails.

**Args:**

`force`: Whether to force the rebuilding of the model schema, defaults to *False*. `raise_errors`: Whether to raise errors, defaults to *True*. `_parent_namespace_depth`: The depth level of the parent namespace, defaults to 2. `_types_namespace`: The types namespace, defaults to *None*.

**Returns:**

Returns *None* if the schema is already “complete” and rebuilding was not required. If rebuilding `_was_required`, returns *True* if rebuilding was successful, otherwise *False*.

**classmethod `model_validate`**(*obj*: *Any*, *\**, *strict*: *bool* | *None* = *None*, *from\_attributes*: *bool* | *None* = *None*, *context*: *dict*[*str*, *Any*] | *None* = *None*) → *Model*

Validate a pydantic model instance.

**Args:**

`obj`: The object to validate. `strict`: Whether to raise an exception on invalid fields. `from_attributes`: Whether to extract data from object attributes. `context`: Additional context to pass to the validator.

**Raises:**

`ValidationError`: If the object could not be validated.

**Returns:**

The validated model instance.

**classmethod `model_validate_json`**(*json\_data*: *str* | *bytes* | *bytearray*, *\**, *strict*: *bool* | *None* = *None*, *context*: *dict*[*str*, *Any*] | *None* = *None*) → *Model*

Usage docs: <https://docs.pydantic.dev/2.5/concepts/json/#json-parsing>

Validate the given JSON data against the Pydantic model.

**Args:**

json\_data: The JSON data to validate. strict: Whether to enforce types strictly. context: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**Raises:**

ValueError: If json\_data is not a JSON string.

**classmethod model\_validate\_strings**(obj: Any, \*, strict: bool | None = None, context: dict[str, Any] | None = None) → Model

Validate the given object contains string data against the Pydantic model.

**Args:**

obj: The object contains string data to validate. strict: Whether to enforce types strictly. context: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**classmethod parse\_file**(path: str | Path, \*, content\_type: str | None = None, encoding: str = 'utf8', proto: DeprecatedParseProtocol | None = None, allow\_pickle: bool = False) → Model

**classmethod parse\_obj**(obj: Any) → Model

**classmethod parse\_raw**(b: str | bytes, \*, content\_type: str | None = None, encoding: str = 'utf8', proto: DeprecatedParseProtocol | None = None, allow\_pickle: bool = False) → Model

**classmethod schema**(by\_alias: bool = True, ref\_template: str = '#/\$defs/{model}') → Dict[str, Any]

**classmethod schema\_json**(\*, by\_alias: bool = True, ref\_template: str = '#/\$defs/{model}', \*\*kwargs: Any) → str

**classmethod update\_forward\_refs**(\*\*localns: Any) → None

**classmethod validate**(value: Any) → Model

value: float

## PreProcessor

**class** glotaran.io.preprocessor.preprocessor.PreProcessor

Bases: BaseModel, ABC

A base class for pre=processors.

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

## Attributes Summary

<code>model_computed_fields</code>	Get the computed fields of this model instance.
<code>model_config</code>	Configuration for the model, should be a dictionary conforming to <code>[ConfigDict][pydantic.config.ConfigDict]</code> .
<code>model_extra</code>	Get extra fields set during validation.
<code>model_fields</code>	Metadata about the fields defined on the model, mapping of field names to <code>[FieldInfo][pydantic.fields.FieldInfo]</code> .
<code>model_fields_set</code>	Returns the set of fields that have been explicitly set on this model instance.

### model\_computed\_fields

`PreProcessor.model_computed_fields`

Get the computed fields of this model instance.

**Returns:**

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

### model\_config

`PreProcessor.model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True}`

Configuration for the model, should be a dictionary conforming to `[ConfigDict][pydantic.config.ConfigDict]`.

### model\_extra

`PreProcessor.model_extra`

Get extra fields set during validation.

**Returns:**

A dictionary of extra fields, or *None* if *config.extra* is not set to “allow”.

### model\_fields

`PreProcessor.model_fields: ClassVar[dict[str, FieldInfo]] = {}`

Metadata about the fields defined on the model, mapping of field names to `[FieldInfo][pydantic.fields.FieldInfo]`.

This replaces *Model.\_\_fields\_\_* from Pydantic V1.

## `model_fields_set`

### `PreProcessor.model_fields_set`

Returns the set of fields that have been explicitly set on this model instance.

**Returns:**

**A set of strings representing the fields that have been set,**  
i.e. that were not filled from defaults.

## Methods Summary

<code>apply</code>	Apply the pre-processor.
<code>construct</code>	
<code>copy</code>	Returns a copy of the model.
<code>dict</code>	
<code>from_orm</code>	
<code>json</code>	
<code>model_construct</code>	Creates a new instance of the <i>Model</i> class with validated data.
<code>model_copy</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy">https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy</a>
<code>model_dump</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump">https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump</a>
<code>model_dump_json</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json">https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json</a>
<code>model_json_schema</code>	Generates a JSON schema for a model class.
<code>model_parametrized_name</code>	Compute the class name for parametrizations of generic classes.
<code>model_post_init</code>	Override this method to perform additional initialization after <code>__init__</code> and <code>model_construct</code> .
<code>model_rebuild</code>	Try to rebuild the pydantic-core schema for the model.
<code>model_validate</code>	Validate a pydantic model instance.
<code>model_validate_json</code>	Usage docs: <a href="https://docs.pydantic.dev/2.5/concepts/json/#json-parsing">https://docs.pydantic.dev/2.5/concepts/json/#json-parsing</a>
<code>model_validate_strings</code>	Validate the given object contains string data against the Pydantic model.
<code>parse_file</code>	
<code>parse_obj</code>	
<code>parse_raw</code>	
<code>schema</code>	
<code>schema_json</code>	
<code>update_forward_refs</code>	
<code>validate</code>	

## apply

**abstract** `PreProcessor.apply(data: dataArray) → dataArray`

Apply the pre-processor.

**Parameters**

**data** (*xr.DataArray*) – The data to process.

**Returns**

- *xr.DataArray*
- .. # **noqa** (*DAR202*)

## construct

**classmethod** `PreProcessor.construct(_fields_set: set[str] | None = None, **values: Any)`  
→ *Model*

## copy

`PreProcessor.copy(*, include: AbstractSetIntStr | MappingIntStrAny | None = None, exclude:`  
*AbstractSetIntStr* | *MappingIntStrAny* | *None* = *None*, update: *Dict[str,*  
*Any]* | *None* = *None*, deep: *bool* = *False*) → *Model*

Returns a copy of the model.

**!!! warning “Deprecated”**

This method is now deprecated; use *model\_copy* instead.

If you need *include* or *exclude*, use:

```
`py data = self.model_dump(include=include, exclude=exclude,  
round_trip=True) data = {**data, **(update or {})} copied = self.  
model_validate(data) `
```

**Args:**

**include: Optional set or mapping**

specifying which fields to include in the copied model.

**exclude: Optional set or mapping**

specifying which fields to exclude in the copied model.

**update: Optional dictionary of field-value pairs to override field values**

in the copied model.

**deep:** If True, the values of fields that are Pydantic models will be deep copied.

**Returns:**

A copy of the model with included, excluded and updated fields as specified.

## dict

`PreProcessor.dict(*, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False,`  
*exclude\_unset: bool* = *False*, *exclude\_defaults: bool* = *False*,  
*exclude\_none: bool* = *False*) → *Dict[str, Any]*

## from\_orm

**classmethod** PreProcessor.**from\_orm**(obj: Any) → *Model*

## json

PreProcessor.**json**(\*, include: IncEx = None, exclude: IncEx = None, by\_alias: bool = False, exclude\_unset: bool = False, exclude\_defaults: bool = False, exclude\_none: bool = False, encoder: Callable[[Any], Any] | None = PydanticUndefined, models\_as\_dict: bool = PydanticUndefined, \*\*kwargs: Any) → str

## model\_construct

**classmethod** PreProcessor.**model\_construct**(\_fields\_set: set[str] | None = None, \*\*values: Any) → *Model*

Creates a new instance of the *Model* class with validated data.

Creates a new model setting `__dict__` and `__pydantic_fields_set__` from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if `Config.extra = 'allow'` was set since it adds all passed values

**Args:**

`_fields_set`: The set of field names accepted for the *Model* instance. `values`: Trusted or pre-validated data dictionary.

**Returns:**

A new instance of the *Model* class with validated data.

## model\_copy

PreProcessor.**model\_copy**(\*, update: dict[str, Any] | None = None, deep: bool = False) → *Model*

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#model\\_copy](https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy)

Returns a copy of the model.

**Args:**

**update**: Values to change/add in the new model. **Note: the data is not validated** before creating the new model. You should trust this data.  
**deep**: Set to *True* to make a deep copy of the model.

**Returns:**

New model instance.

## model\_dump

```
PreProcessor.model_dump(*, mode: Literal['json', 'python'] | str = 'python', include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True) → dict[str, Any]
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

### Args:

**mode: The mode in which to *python* should run.**

If mode is 'json', the dictionary will only contain JSON serializable types. If mode is 'python', the dictionary may contain any Python objects.

**include:** A list of fields to include in the output. **exclude:** A list of fields to exclude from the output. **by\_alias:** Whether to use the field's alias in the dictionary key if defined. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that are set to their default value from the output. **exclude\_none:** Whether to exclude fields that have a value of *None* from the output. **round\_trip:** Whether to enable serialization and deserialization round-trip support. **warnings:** Whether to log warnings when invalid fields are encountered.

### Returns:

A dictionary representation of the model.

## model\_dump\_json

```
PreProcessor.model_dump_json(*, indent: int | None = None, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True) → str
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump\\_json](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json)

Generates a JSON representation of the model using Pydantic's *to\_json* method.

### Args:

**indent:** Indentation to use in the JSON output. If *None* is passed, the output will be compact. **include:** Field(s) to include in the JSON output. Can take either a string or set of strings. **exclude:** Field(s) to exclude from the JSON output. Can take either a string or set of strings. **by\_alias:** Whether to serialize using field aliases. **exclude\_unset:** Whether to exclude fields that have not been explicitly set. **exclude\_defaults:** Whether to exclude fields that have the default value. **exclude\_none:** Whether to exclude fields that have a value of *None*. **round\_trip:** Whether to use serialization/deserialization between JSON and class instance. **warnings:** Whether to show any warnings that occurred during serialization.

### Returns:

A JSON string representation of the model.



## model\_json\_schema

```

classmethod PreProcessor.model_json_schema(by_alias: bool = True, ref_template: str =
                                             '#/$defs/{model}', schema_generator:
                                             type[~pydantic.json_schema.GenerateJsonSchema]
                                             = <class 'pydantic.json_schema.GenerateJsonSchema'>,
                                             mode: ~typing.Literal['validation',
                                             'serialization'] = 'validation') → dict[str,
                                             Any]

```

Generates a JSON schema for a model class.

**Args:**

by\_alias: Whether to use attribute aliases or not. ref\_template: The reference template.  
 schema\_generator: To override the logic used to generate the JSON schema, as a subclass of  
*GenerateJsonSchema* with your desired modifications  
 mode: The mode in which to generate the schema.

**Returns:**

The JSON schema for the given model class.

## model\_parametrized\_name

```

classmethod PreProcessor.model_parametrized_name(params: tuple[type[Any], ...]) →
                                                    str

```

Compute the class name for parametrizations of generic classes.

This method can be overridden to achieve a custom naming scheme for generic BaseModels.

**Args:**

**params: Tuple of types of the class. Given a generic class**

*Model* with 2 type variables and a concrete model *Model[str, int]*, the value (*str, int*)  
 would be passed to *params*.

**Returns:**

String representing the new class where *params* are passed to *cls* as type variables.

**Raises:**

*TypeError*: Raised when trying to generate concrete names for non-generic models.

### model\_post\_init

`PreProcessor.model_post_init(_BaseModel__context: Any) → None`

Override this method to perform additional initialization after `__init__` and `model_construct`. This is useful if you want to do some validation that requires the entire model to be initialized.

### model\_rebuild

**classmethod** `PreProcessor.model_rebuild(*, force: bool = False, raise_errors: bool = True, _parent_namespace_depth: int = 2, _types_namespace: dict[str, Any] | None = None) → bool | None`

Try to rebuild the pydantic-core schema for the model.

This may be necessary when one of the annotations is a `ForwardRef` which could not be resolved during the initial attempt to build the schema, and automatic rebuilding fails.

**Args:**

`force`: Whether to force the rebuilding of the model schema, defaults to *False*. `raise_errors`: Whether to raise errors, defaults to *True*. `_parent_namespace_depth`: The depth level of the parent namespace, defaults to 2. `_types_namespace`: The types namespace, defaults to *None*.

**Returns:**

Returns *None* if the schema is already “complete” and rebuilding was not required. If rebuilding `_was_ required`, returns *True* if rebuilding was successful, otherwise *False*.

### model\_validate

**classmethod** `PreProcessor.model_validate(obj: Any, *, strict: bool | None = None, from_attributes: bool | None = None, context: dict[str, Any] | None = None) → Model`

Validate a pydantic model instance.

**Args:**

`obj`: The object to validate. `strict`: Whether to raise an exception on invalid fields. `from_attributes`: Whether to extract data from object attributes. `context`: Additional context to pass to the validator.

**Raises:**

`ValidationError`: If the object could not be validated.

**Returns:**

The validated model instance.

### model\_validate\_json

**classmethod** `PreProcessor.model_validate_json(json_data: str | bytes | bytearray, *, strict: bool | None = None, context: dict[str, Any] | None = None) → Model`

Usage docs: <https://docs.pydantic.dev/2.5/concepts/json/#json-parsing>

Validate the given JSON data against the Pydantic model.

**Args:**

`json_data`: The JSON data to validate. `strict`: Whether to enforce types strictly. `context`: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**Raises:**

ValueError: If *json\_data* is not a JSON string.

**model\_validate\_strings**

**classmethod** PreProcessor.**model\_validate\_strings**(*obj*: Any, \*, *strict*: bool | None = None, *context*: dict[str, Any] | None = None) → Model

Validate the given object contains string data against the Pydantic model.

**Args:**

*obj*: The object contains string data to validate. *strict*: Whether to enforce types strictly. *context*: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**parse\_file**

**classmethod** PreProcessor.**parse\_file**(*path*: str | Path, \*, *content\_type*: str | None = None, *encoding*: str = 'utf8', *proto*: DeprecatedParseProtocol | None = None, *allow\_pickle*: bool = False) → Model

**parse\_obj**

**classmethod** PreProcessor.**parse\_obj**(*obj*: Any) → Model

**parse\_raw**

**classmethod** PreProcessor.**parse\_raw**(*b*: str | bytes, \*, *content\_type*: str | None = None, *encoding*: str = 'utf8', *proto*: DeprecatedParseProtocol | None = None, *allow\_pickle*: bool = False) → Model

**schema**

**classmethod** PreProcessor.**schema**(*by\_alias*: bool = True, *ref\_template*: str = '#/\$defs/{model}') → Dict[str, Any]

## schema\_json

```
classmethod PreProcessor.schema_json(*, by_alias: bool = True, ref_template: str =
                                     '#/$defs/{model}', **dumps_kwargs: Any) → str
```

## update\_forward\_refs

```
classmethod PreProcessor.update_forward_refs(**localns: Any) → None
```

## validate

```
classmethod PreProcessor.validate(value: Any) → Model
```

## Methods Documentation

### class Config

Bases: `object`

Config for BaseModel.

**arbitrary\_types\_allowed = True**

```
abstract apply(data: DataArray) → DataArray
```

Apply the pre-processor.

#### Parameters

**data** (`xr.DataArray`) – The data to process.

#### Returns

- `xr.DataArray`
- .. # noqa (DAR202)

```
classmethod construct(_fields_set: set[str] | None = None, **values: Any) → Model
```

```
copy(*, include: AbstractSetIntStr | MappingIntStrAny | None = None, exclude: AbstractSetIntStr
    | MappingIntStrAny | None = None, update: Dict[str, Any] | None = None, deep: bool =
    False) → Model
```

Returns a copy of the model.

#### !!! warning “Deprecated”

This method is now deprecated; use `model_copy` instead.

If you need `include` or `exclude`, use:

```
`py data = self.model_dump(include=include, exclude=exclude,
round_trip=True) data = {**data, **(update or {})} copied = self.
model_validate(data) `
```

#### Args:

##### **include: Optional set or mapping**

specifying which fields to include in the copied model.

##### **exclude: Optional set or mapping**

specifying which fields to exclude in the copied model.

##### **update: Optional dictionary of field-value pairs to override field values**

in the copied model.

**deep:** If True, the values of fields that are Pydantic models will be deep copied.

**Returns:**

A copy of the model with included, excluded and updated fields as specified.

```
dict(*, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False) → Dict[str, Any]
```

```
classmethod from_orm(obj: Any) → Model
```

```
json(*, include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, encoder: Callable[[Any], Any] | None = PydanticUndefined, models_as_dict: bool = PydanticUndefined, **dumps_kwargs: Any) → str
```

```
property model_computed_fields: Dict[str, ComputedFieldInfo]
```

Get the computed fields of this model instance.

**Returns:**

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True}
```

Configuration for the model, should be a dictionary conforming to *[ConfigDict][pydantic.config.ConfigDict]*.

```
classmethod model_construct(_fields_set: set[str] | None = None, **values: Any) → Model
```

Creates a new instance of the *Model* class with validated data.

Creates a new model setting `__dict__` and `__pydantic_fields_set__` from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if *Config.extra = 'allow'* was set since it adds all passed values

**Args:**

`_fields_set`: The set of field names accepted for the *Model* instance. `values`: Trusted or pre-validated data dictionary.

**Returns:**

A new instance of the *Model* class with validated data.

```
model_copy(*, update: dict[str, Any] | None = None, deep: bool = False) → Model
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#model\\_copy](https://docs.pydantic.dev/2.5/concepts/serialization/#model_copy)

Returns a copy of the model.

**Args:**

**update:** Values to change/add in the new model. **Note: the data is not validated** before creating the new model. You should trust this data.  
**deep:** Set to *True* to make a deep copy of the model.

**Returns:**

New model instance.

```
model_dump(*, mode: Literal['json', 'python'] | str = 'python', include: IncEx = None, exclude: IncEx = None, by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool = False, exclude_none: bool = False, round_trip: bool = False, warnings: bool = True) → dict[str, Any]
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

**Args:**

**mode:** The mode in which to *python* should run.

If mode is 'json', the dictionary will only contain JSON serializable types. If mode is 'python', the dictionary may contain any Python objects.

include: A list of fields to include in the output. exclude: A list of fields to exclude from the output. by\_alias: Whether to use the field's alias in the dictionary key if defined. exclude\_unset: Whether to exclude fields that have not been explicitly set. exclude\_defaults: Whether to exclude fields that are set to their default value from the output. exclude\_none: Whether to exclude fields that have a value of *None* from the output. round\_trip: Whether to enable serialization and deserialization round-trip support. warnings: Whether to log warnings when invalid fields are encountered.

**Returns:**

A dictionary representation of the model.

```
model_dump_json(*, indent: int | None = None, include: IncEx = None, exclude: IncEx = None,  
by_alias: bool = False, exclude_unset: bool = False, exclude_defaults: bool  
= False, exclude_none: bool = False, round_trip: bool = False, warnings:  
bool = True) → str
```

Usage docs: [https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel\\_dump\\_json](https://docs.pydantic.dev/2.5/concepts/serialization/#modelmodel_dump_json)

Generates a JSON representation of the model using Pydantic's *to\_json* method.

**Args:**

indent: Indentation to use in the JSON output. If *None* is passed, the output will be compact. include: Field(s) to include in the JSON output. Can take either a string or set of strings. exclude: Field(s) to exclude from the JSON output. Can take either a string or set of strings. by\_alias: Whether to serialize using field aliases. exclude\_unset: Whether to exclude fields that have not been explicitly set. exclude\_defaults: Whether to exclude fields that have the default value. exclude\_none: Whether to exclude fields that have a value of *None*. round\_trip: Whether to use serialization/deserialization between JSON and class instance. warnings: Whether to show any warnings that occurred during serialization.

**Returns:**

A JSON string representation of the model.

```
property model_extra: dict[str, Any] | None
```

Get extra fields set during validation.

**Returns:**

A dictionary of extra fields, or *None* if *config.extra* is not set to “allow”.

```
model_fields: ClassVar[dict[str, FieldInfo]] = {}
```

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.\_\_fields\_\_* from Pydantic V1.

```
property model_fields_set: set[str]
```

Returns the set of fields that have been explicitly set on this model instance.

**Returns:**

**A set of strings representing the fields that have been set,**  
i.e. that were not filled from defaults.

```
classmethod model_json_schema(by_alias: bool = True, ref_template: str =  
    '#/$defs/{model}', schema_generator:  
    type[~pydantic.json_schema.GenerateJsonSchema] =  
    <class 'pydantic.json_schema.GenerateJsonSchema'>,  
    mode: ~typing.Literal['validation', 'serialization'] =  
    'validation') → dict[str, Any]
```

Generates a JSON schema for a model class.

**Args:**

by\_alias: Whether to use attribute aliases or not. ref\_template: The reference template.

schema\_generator: To override the logic used to generate the JSON schema, as a subclass of

*GenerateJsonSchema* with your desired modifications

mode: The mode in which to generate the schema.

**Returns:**

The JSON schema for the given model class.

**classmethod model\_parametrized\_name**(*params: tuple[type[Any], ...]*) → *str*

Compute the class name for parametrizations of generic classes.

This method can be overridden to achieve a custom naming scheme for generic BaseModels.

**Args:**

**params: Tuple of types of the class. Given a generic class**

*Model* with 2 type variables and a concrete model *Model[str, int]*, the value (*str, int*) would be passed to *params*.

**Returns:**

String representing the new class where *params* are passed to *cls* as type variables.

**Raises:**

*TypeError*: Raised when trying to generate concrete names for non-generic models.

**model\_post\_init**(*\_BaseModel\_\_context: Any*) → *None*

Override this method to perform additional initialization after *\_\_init\_\_* and *model\_construct*.

This is useful if you want to do some validation that requires the entire model to be initialized.

**classmethod model\_rebuild**(*\*, force: bool = False, raise\_errors: bool = True, \_parent\_namespace\_depth: int = 2, \_types\_namespace: dict[str, Any] | None = None*) → *bool | None*

Try to rebuild the pydantic-core schema for the model.

This may be necessary when one of the annotations is a *ForwardRef* which could not be resolved during the initial attempt to build the schema, and automatic rebuilding fails.

**Args:**

*force*: Whether to force the rebuilding of the model schema, defaults to *False*. *raise\_errors*: Whether to raise errors, defaults to *True*. *\_parent\_namespace\_depth*: The depth level of the parent namespace, defaults to 2. *\_types\_namespace*: The types namespace, defaults to *None*.

**Returns:**

Returns *None* if the schema is already “complete” and rebuilding was not required. If rebuilding *\_was\_ required*, returns *True* if rebuilding was successful, otherwise *False*.

**classmethod model\_validate**(*obj: Any, \*, strict: bool | None = None, from\_attributes: bool | None = None, context: dict[str, Any] | None = None*) → *Model*

Validate a pydantic model instance.

**Args:**

*obj*: The object to validate. *strict*: Whether to raise an exception on invalid fields. *from\_attributes*: Whether to extract data from object attributes. *context*: Additional context to pass to the validator.

**Raises:**

*ValidationError*: If the object could not be validated.

**Returns:**

The validated model instance.

**classmethod model\_validate\_json**(*json\_data: str | bytes | bytearray, \*, strict: bool | None = None, context: dict[str, Any] | None = None*) → *Model*

Usage docs: <https://docs.pydantic.dev/2.5/concepts/json/#json-parsing>

Validate the given JSON data against the Pydantic model.

**Args:**

json\_data: The JSON data to validate. strict: Whether to enforce types strictly. context: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

**Raises:**

ValueError: If *json\_data* is not a JSON string.

```
classmethod model_validate_strings(obj: Any, *, strict: bool | None = None, context: dict[str, Any] | None = None) → Model
```

Validate the given object contains string data against the Pydantic model.

**Args:**

obj: The object contains string data to validate. strict: Whether to enforce types strictly. context: Extra variables to pass to the validator.

**Returns:**

The validated Pydantic model.

```
classmethod parse_file(path: str | Path, *, content_type: str | None = None, encoding: str = 'utf8', proto: DeprecatedParseProtocol | None = None, allow_pickle: bool = False) → Model
```

```
classmethod parse_obj(obj: Any) → Model
```

```
classmethod parse_raw(b: str | bytes, *, content_type: str | None = None, encoding: str = 'utf8', proto: DeprecatedParseProtocol | None = None, allow_pickle: bool = False) → Model
```

```
classmethod schema(by_alias: bool = True, ref_template: str = '#/$defs/{model}') → Dict[str, Any]
```

```
classmethod schema_json(*, by_alias: bool = True, ref_template: str = '#/$defs/{model}', **dump_kwargs: Any) → str
```

```
classmethod update_forward_refs(**localns: Any) → None
```

```
classmethod validate(value: Any) → Model
```

## 16.1.6 model

The glotaran model package.

### Modules

<code>glotaran.model.clp_constraint</code>	This module contains clp constraint items.
<code>glotaran.model.clp_penalties</code>	This module contains clp penalty items.
<code>glotaran.model.clp_relation</code>	This module contains clp relation items.
<code>glotaran.model.dataset_group</code>	This module contains the dataset group.
<code>glotaran.model.dataset_model</code>	This module contains the dataset model.
<code>glotaran.model.interval_item</code>	This module contains the interval item.
<code>glotaran.model.item(cls)</code>	Create an item from a class.
<code>glotaran.model.megacomplex(*[, ...])</code>	Create a megacomplex from a class.
<code>glotaran.model.model</code>	This module contains the model.
<code>glotaran.model.weight</code>	This module contains weight item.



clp\_constraint

This module contains clp constraint items.

Classes

Summary

<i>ClpConstraint</i>	Baseclass for clp constraints.
<i>OnlyConstraint</i>	Constraints the target to 0 outside the given interval.
<i>ZeroConstraint</i>	Constraints the target to 0 in the given interval.

ClpConstraint

`class glotaran.model.clp_constraint.ClpConstraint(*, interval: tuple[float, float] | list[tuple[float, float]] | None = None, type: str, target: str)`

Bases: TypedItem, [IntervalItem](#)

Baseclass for clp constraints.

There are two types: zero and equal. See the documentation of the respective classes for details.

Method generated by attrs for class ClpConstraint.

Attributes Summary

<i>target</i>
<i>type</i>
<i>interval</i>

target

ClpConstraint.target: str

## type

ClpConstraint.type: `str`

## interval

ClpConstraint.interval: `tuple[float, float] | list[tuple[float, float]] | None`

## Methods Summary

<code>applies</code>	Check if the index is in the intervals.
<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.
<code>has_interval</code>	Check if intervals are defined.

## applies

ClpConstraint.applies(index: `float | None`) → `bool`

Check if the index is in the intervals.

### Parameters

**index** (`float`) – The index.

### Return type

`bool`

## get\_item\_type

classmethod ClpConstraint.get\_item\_type() → `str`

Get the type string.

### Return type

`str`

## get\_item\_type\_class

classmethod ClpConstraint.get\_item\_type\_class(item\_type: `str`) → `Type`

Get the type for a type string.

### Parameters

**item\_type** (`str`) – The type string.

### Return type

`Type`

## get\_item\_types

**classmethod** `ClpConstraint.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

## has\_interval

`ClpConstraint.has_interval()` → `bool`

Check if intervals are defined.

**Return type**

`bool`

## Methods Documentation

**applies**(*index*: `float` | `None`) → `bool`

Check if the index is in the intervals.

**Parameters**

**index** (`float`) – The index.

**Return type**

`bool`

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**has\_interval**() → `bool`

Check if intervals are defined.

**Return type**

`bool`

**interval**: `tuple[float, float]` | `list[tuple[float, float]]` | `None`

**target**: `str`

**type**: `str`

## OnlyConstraint

```
class glotaran.model.clp_constraint.OnlyConstraint(*, interval: tuple[float, float] |  
list[tuple[float, float]] | None = None,  
target: str, type: str = 'only')
```

Bases: *ZeroConstraint*

Constraints the target to 0 outside the given interval.

Method generated by attrs for class OnlyConstraint.

### Attributes Summary

<i>type</i>
<i>target</i>
<i>interval</i>

#### type

OnlyConstraint.type: **str**

#### target

OnlyConstraint.target: **str**

#### interval

OnlyConstraint.interval: **tuple[float, float] | list[tuple[float, float]] | None**

### Methods Summary

<i>applies</i>	Check if the constraint applies on this index.
<i>get_item_type</i>	Get the type string.
<i>get_item_type_class</i>	Get the type for a type string.
<i>get_item_types</i>	Get all type strings.
<i>has_interval</i>	Check if intervals are defined.

## applies

`OnlyConstraint.applies(index: float | None) → bool`

Check if the constraint applies on this index.

**Parameters**

**index** (*float*) – The index.

**Return type**

*bool*

## get\_item\_type

`classmethod OnlyConstraint.get_item_type() → str`

Get the type string.

**Return type**

*str*

## get\_item\_type\_class

`classmethod OnlyConstraint.get_item_type_class(item_type: str) → Type`

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

## get\_item\_types

`classmethod OnlyConstraint.get_item_types() → list[str]`

Get all type strings.

**Return type**

*list[str]*

## has\_interval

`OnlyConstraint.has_interval() → bool`

Check if intervals are defined.

**Return type**

*bool*

## Methods Documentation

**applies**(*index*: *float* | *None*) → *bool*

Check if the constraint applies on this index.

**Parameters**

**index** (*float*) – The index.

**Return type**

*bool*

**classmethod** **get\_item\_type**() → *str*

Get the type string.

**Return type**

*str*

**classmethod** **get\_item\_type\_class**(*item\_type*: *str*) → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

**classmethod** **get\_item\_types**() → *list*[*str*]

Get all type strings.

**Return type**

*list*[*str*]

**has\_interval**() → *bool*

Check if intervals are defined.

**Return type**

*bool*

**interval**: *tuple*[*float*, *float*] | *list*[*tuple*[*float*, *float*]] | *None*

**target**: *str*

**type**: *str*

## ZeroConstraint

```
class glotaran.model.clp_constraint.ZeroConstraint(*, interval: tuple[float, float] |  
list[tuple[float, float]] | None = None,  
target: str, type: str = 'zero')
```

Bases: *ClpConstraint*

Constraints the target to 0 in the given interval.

Method generated by attrs for class ZeroConstraint.

## Attributes Summary

<i>type</i>
<i>target</i>
<i>interval</i>

### type

`ZeroConstraint.type: str`

### target

`ZeroConstraint.target: str`

### interval

`ZeroConstraint.interval: tuple[float, float] | list[tuple[float, float]] | None`

## Methods Summary

<i>applies</i>	Check if the index is in the intervals.
<i>get_item_type</i>	Get the type string.
<i>get_item_type_class</i>	Get the type for a type string.
<i>get_item_types</i>	Get all type strings.
<i>has_interval</i>	Check if intervals are defined.

### applies

`ZeroConstraint.applies(index: float | None) → bool`

Check if the index is in the intervals.

#### Parameters

**index** (*float*) – The index.

#### Return type

*bool*

### get\_item\_type

**classmethod** ZeroConstraint.get\_item\_type() → str

Get the type string.

**Return type**

str

### get\_item\_type\_class

**classmethod** ZeroConstraint.get\_item\_type\_class(item\_type: str) → Type

Get the type for a type string.

**Parameters**

**item\_type** (str) – The type string.

**Return type**

Type

### get\_item\_types

**classmethod** ZeroConstraint.get\_item\_types() → list[str]

Get all type strings.

**Return type**

list[str]

### has\_interval

ZeroConstraint.has\_interval() → bool

Check if intervals are defined.

**Return type**

bool

## Methods Documentation

**applies**(index: float | None) → bool

Check if the index is in the intervals.

**Parameters**

**index** (float) – The index.

**Return type**

bool

**classmethod** get\_item\_type() → str

Get the type string.

**Return type**

str

**classmethod** get\_item\_type\_class(item\_type: str) → Type

Get the type for a type string.

**Parameters**

**item\_type** (str) – The type string.



**Return type**

Type

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

**has\_interval()** → `bool`

Check if intervals are defined.

**Return type**

`bool`

**interval:** `tuple[float, float] | list[tuple[float, float]] | None`

**target:** `str`

**type:** `str`

**clp\_penalties**

This module contains clp penalty items.

**Classes**

**Summary**

<code>ClpPenalty</code>	Baseclass for clp penalties.
<code>EqualAreaPenalty</code>	Forces the area of 2 clp to be the same.

**ClpPenalty**

**class** `glotaran.model.clp_penalties.ClpPenalty(*, type: str)`

Bases: `TypedItem`

Baseclass for clp penalties.

Method generated by attrs for class `ClpPenalty`.

**Attributes Summary**

<code>type</code>
-------------------

## type

`ClpPenalty.type: str`

## Methods Summary

<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

## get\_item\_type

**classmethod** `ClpPenalty.get_item_type()` → `str`

Get the type string.

**Return type**

`str`

## get\_item\_type\_class

**classmethod** `ClpPenalty.get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

## get\_item\_types

**classmethod** `ClpPenalty.get_item_types()` → `list[str]`

Get all type strings.

**Return type**

`list[str]`

## Methods Documentation

**classmethod** `get_item_type()` → `str`

Get the type string.

**Return type**

`str`

**classmethod** `get_item_type_class(item_type: str)` → `Type`

Get the type for a type string.

**Parameters**

**item\_type** (`str`) – The type string.

**Return type**

`Type`

**classmethod** `get_item_types()` → `list[str]`

Get all type strings.

**Return type**  
`list[str]`

**type:** `str`

**EqualAreaPenalty**

**class** `glotaran.model.clp_penalties.EqualAreaPenalty`(\*`type: str = 'equal_area'`, `source: str`, `source_intervals: list[tuple[float, float]]`, `target: str`, `target_intervals: list[tuple[float, float]]`, `parameter: Parameter | str`, `weight: float`)

Bases: `ClpPenalty`

Forces the area of 2 clp to be the same.

An equal area constraint adds a the difference of the sum of a compartments in the e matrix in one or more intervals to the scaled sum of the e matrix of one or more target compartments to residual. The additional residual is scaled with the weight.

Method generated by attrs for class EqualAreaPenalty.

**Attributes Summary**

<code>type</code>
<code>source</code>
<code>source_intervals</code>
<code>target</code>
<code>target_intervals</code>
<code>parameter</code>
<code>weight</code>

**type**

`EqualAreaPenalty.type: str`

**source**

`EqualAreaPenalty.source: str`

**source\_intervals**

`EqualAreaPenalty.source_intervals: list[tuple[float, float]]`

**target**

`EqualAreaPenalty.target: str`

**target\_intervals**

`EqualAreaPenalty.target_intervals: list[tuple[float, float]]`

**parameter**

`EqualAreaPenalty.parameter: ParameterType`

**weight**

`EqualAreaPenalty.weight: float`

**Methods Summary**

<code>get_item_type</code>	Get the type string.
<code>get_item_type_class</code>	Get the type for a type string.
<code>get_item_types</code>	Get all type strings.

**get\_item\_type**

**classmethod** `EqualAreaPenalty.get_item_type() → str`

Get the type string.

**Return type**

`str`

### get\_item\_type\_class

**classmethod** EqualAreaPenalty.get\_item\_type\_class(*item\_type*: *str*) → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

### get\_item\_types

**classmethod** EqualAreaPenalty.get\_item\_types() → *list[str]*

Get all type strings.

**Return type**

*list[str]*

## Methods Documentation

**classmethod** get\_item\_type() → *str*

Get the type string.

**Return type**

*str*

**classmethod** get\_item\_type\_class(*item\_type*: *str*) → *Type*

Get the type for a type string.

**Parameters**

**item\_type** (*str*) – The type string.

**Return type**

*Type*

**classmethod** get\_item\_types() → *list[str]*

Get all type strings.

**Return type**

*list[str]*

**parameter:** *ParameterType*

**source:** *str*

**source\_intervals:** *list[tuple[float, float]]*

**target:** *str*

**target\_intervals:** *list[tuple[float, float]]*

**type:** *str*

**weight:** *float*

## clp\_relation

This module contains clp relation items.

### Classes

#### Summary

<i>ClpRelation</i>	Applies a relation between two clps.
--------------------	--------------------------------------

#### ClpRelation

```
class glotaran.model.clp_relation.ClpRelation(*, interval: tuple[float, float] |  
                                              list[tuple[float, float]] | None = None,  
                                              source: str, target: str, parameter:  
                                              Parameter | str)
```

Bases: *IntervalItem*

Applies a relation between two clps.

The relation is applied as *target* = *parameter* \* *source*.

Method generated by attrs for class ClpRelation.

#### Attributes Summary

<i>source</i>
<i>target</i>
<i>parameter</i>
<i>interval</i>

#### source

ClpRelation.**source**: **str**

**target**

ClpRelation.**target**: `str`

**parameter**

ClpRelation.**parameter**: `Parameter` | `str`

**interval**

ClpRelation.**interval**: `tuple[float, float]` | `list[tuple[float, float]]` | `None`

**Methods Summary**

<code>applies</code>	Check if the index is in the intervals.
<code>has_interval</code>	Check if intervals are defined.

**applies**

ClpRelation.**applies**(*index*: `float` | `None`) → `bool`

Check if the index is in the intervals.

**Parameters**

**index** (`float`) – The index.

**Return type**

`bool`

**has\_interval**

ClpRelation.**has\_interval**() → `bool`

Check if intervals are defined.

**Return type**

`bool`

**Methods Documentation**

**applies**(*index*: `float` | `None`) → `bool`

Check if the index is in the intervals.

**Parameters**

**index** (`float`) – The index.

**Return type**

`bool`

**has\_interval()** → bool

Check if intervals are defined.

**Return type**

bool

**interval:** tuple[float, float] | list[tuple[float, float]] | None

**parameter:** Parameter | str

**source:** str

**target:** str

## dataset\_group

This module contains the dataset group.

### Classes

#### Summary

<i>DatasetGroup</i>	A dataset group for optimization.
<i>DatasetGroupModel</i>	A group of datasets which will evaluated independently.

#### DatasetGroup

```
class glotaran.model.dataset_group.DatasetGroup(residual_function:
    Literal['variable_projection',
    'non_negative_least_squares'], link_clp:
    bool | None, model: Model, parameters:
    Parameters | None = None,
    dataset_models: dict[str, DatasetModel]
    = _Nothing.NOTHING)
```

Bases: object

A dataset group for optimization.

Method generated by attrs for class DatasetGroup.

#### Attributes Summary

<i>residual_function</i>	The residual function to use.
<i>link_clp</i>	Whether to link the clp parameter.
<i>model</i>	
<i>parameters</i>	
<i>dataset_models</i>	



### residual\_function

`DatasetGroup.residual_function: Literal['variable_projection', 'non_negative_least_squares']`

The residual function to use.

### link\_clp

`DatasetGroup.link_clp: bool | None`

Whether to link the clp parameter.

### model

`DatasetGroup.model: Model`

### parameters

`DatasetGroup.parameters: Parameters | None`

### dataset\_models

`DatasetGroup.dataset_models: dict[str, DatasetModel]`

## Methods Summary

<code>is_linkable</code>	Check if the group is linkable.
<code>set_parameters</code>	Set the group parameters.

### is\_linkable

`DatasetGroup.is_linkable(parameters: Parameters, data: Mapping[str, xr.Dataset]) → bool`

Check if the group is linkable.

#### Parameters

- **parameters** (`Parameters`) – A parameter set parameters.
- **data** (`Mapping[str, xr.Dataset]`) – A the data to link.

#### Return type

`bool`

## set\_parameters

`DatasetGroup.set_parameters(parameters: Parameters)`

Set the group parameters.

**Parameters**

**parameters** (`Parameters`) – The parameters.

## Methods Documentation

**dataset\_models:** `dict[str, DatasetModel]`

**is\_linkable**(*parameters: Parameters, data: Mapping[str, xr.Dataset]*) → `bool`

Check if the group is linkable.

**Parameters**

- **parameters** (`Parameters`) – A parameter set parameters.
- **data** (`Mapping[str, xr.Dataset]`) – A the data to link.

**Return type**

`bool`

**link\_clp:** `bool | None`

Whether to link the clp parameter.

**model:** `Model`

**parameters:** `Parameters | None`

**residual\_function:** `Literal['variable_projection', 'non_negative_least_squares']`

The residual function to use.

**set\_parameters**(*parameters: Parameters*)

Set the group parameters.

**Parameters**

**parameters** (`Parameters`) – The parameters.

## DatasetGroupModel

```
class glotaran.model.dataset_group.DatasetGroupModel(*, label: str, residual_function:
    Literal['variable_projection',
    'non_negative_least_squares'] =
    'variable_projection', link_clp:
    bool | None = None)
```

Bases: `ModelItem`

A group of datasets which will evaluated independently.

Method generated by attrs for class `DatasetGroupModel`.

## Attributes Summary

<code>residual_function</code>	The residual function to use.
<code>link_clp</code>	Whether to link the clp parameter.
<code>label</code>	

### residual\_function

`DatasetGroupModel.residual_function: Literal['variable_projection', 'non_negative_least_squares']`

The residual function to use.

### link\_clp

`DatasetGroupModel.link_clp: bool | None`

Whether to link the clp parameter.

### label

`DatasetGroupModel.label: str`

## Methods Summary

### Methods Documentation

`label: str`

`link_clp: bool | None`

Whether to link the clp parameter.

`residual_function: Literal['variable_projection', 'non_negative_least_squares']`

The residual function to use.

### dataset\_model

This module contains the dataset model.

## Functions

### Summary

<code>finalize_dataset_model</code>	Finalize a dataset by applying all megacomplex finalize methods.
<code>get_dataset_model_model_dimension</code>	Get the dataset model's model dimension.
<code>get_megacomplex_issues</code>	Get issues for megacomplexes.
<code>has_dataset_model_global_model</code>	Check if the dataset model can model the global dimension.
<code>iterate_dataset_model_global_megacomplexes</code>	Iterate the dataset model's global megacomplexes.
<code>iterate_dataset_model_megacomplexes</code>	Iterate the dataset model's megacomplexes.
<code>validate_global_megacomplexes</code>	Get issues for dataset model global megacomplexes.
<code>validate_megacomplexes</code>	Get issues for dataset model megacomplexes.

### `finalize_dataset_model`

`glotaran.model.dataset_model.finalize_dataset_model(dataset_model: DatasetModel, dataset: Dataset)`

Finalize a dataset by applying all megacomplex finalize methods.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **dataset** (`xr.Dataset`) – The dataset.

### `get_dataset_model_model_dimension`

`glotaran.model.dataset_model.get_dataset_model_model_dimension(dataset_model: DatasetModel) → str`

Get the dataset model's model dimension.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.

#### Return type

`str`

#### Raises

**ValueError** – Raised if the dataset model does not have megacomplexes or if it is not filled.

## get\_megacomplex\_issues

```
glotaran.model.dataset_model.get_megacomplex_issues(value: list[str | Megacomplex] |
None, model: Model, is_global:
bool) → list[ItemIssue]
```

Get issues for megacomplexes.

### Parameters

- **value** (`list[str | Megacomplex]` | `None`) – A list of megacomplexes.
- **model** (`Model`) – The model.
- **is\_global** (`bool`) – Whether the megacomplexes are global.

### Return type

`list[ItemIssue]`

## has\_dataset\_model\_global\_model

```
glotaran.model.dataset_model.has_dataset_model_global_model(dataset_model:
DatasetModel) → bool
```

Check if the dataset model can model the global dimension.

### Parameters

**dataset\_model** (`DatasetModel`) – The dataset model.

### Return type

`bool`

## iterate\_dataset\_model\_global\_megacomplexes

```
glotaran.model.dataset_model.iterate_dataset_model_global_megacomplexes(dataset_model:
Dataset-
Model)
→
Genera-
tor[tuple[Parameter
| str |
None,
Mega-
complex |
str],
None,
None]
```

Iterate the dataset model's global megacomplexes.

### Parameters

**dataset\_model** (`DatasetModel`) – The dataset model.

### Yields

`tuple[Parameter | str | None, Megacomplex | str]` – A scale and megacomplex.

### iterate\_dataset\_model\_megacomplexes

```
glotaran.model.dataset_model.iterate_dataset_model_megacomplexes(dataset_model:  
                                                                    DatasetModel) →  
                                                                    Genera-  
                                                                    tor[tuple[Parameter  
                                                                    | str | None,  
                                                                    Megacomplex |  
                                                                    str], None, None]
```

Iterate the dataset model's megacomplexes.

#### Parameters

**dataset\_model** ([DatasetModel](#)) – The dataset model.

#### Yields

*tuple[Parameter | str | None, Megacomplex | str]* – A scale and megacomplex.

### validate\_global\_megacomplexes

```
glotaran.model.dataset_model.validate_global_megacomplexes(value: list[str |  
                                                            Megacomplex] | None,  
                                                            dataset_model:  
                                                            DatasetModel, model:  
                                                            Model, parameters:  
                                                            Parameters | None) →  
                                                            list[ItemIssue]
```

Get issues for dataset model global megacomplexes.

#### Parameters

- **value** (*list[str | Megacomplex] | None*) – A list of megacomplexes.
- **dataset\_model** ([DatasetModel](#)) – The dataset model.
- **model** ([Model](#)) – The model.
- **parameters** ([Parameters](#) | None, ) – The parameters.

#### Return type

*list[ItemIssue]*

### validate\_megacomplexes

```
glotaran.model.dataset_model.validate_megacomplexes(value: list[str | Megacomplex],  
                                                    dataset_model: DatasetModel,  
                                                    model: Model, parameters:  
                                                    Parameters | None) →  
                                                    list[ItemIssue]
```

Get issues for dataset model megacomplexes.

#### Parameters

- **value** (*list[str | Megacomplex]*) – A list of megacomplexes.
- **dataset\_model** ([DatasetModel](#)) – The dataset model.
- **model** ([Model](#)) – The model.

- **parameters** (`Parameters` / `None`,) – The parameters.

**Return type**  
`list[ItemIssue]`

Classes

Summary

<i>DatasetModel</i>	A model for datasets.
<i>ExclusiveMegacomplexIssue</i>	Issue for exclusive megacomplexes.
<i>UniqueMegacomplexIssue</i>	Issue for unique megacomplexes.

DatasetModel

```
class glotaran.model.dataset_model.DatasetModel(*, label: str, group: str = 'default',
                                                force_index_dependent: bool = False,
                                                megacomplex: list[Megacomplex | str],
                                                megacomplex_scale: list[Parameter | str]
                                                | None = None, global_megacomplex:
                                                list[Megacomplex | str] | None = None,
                                                global_megacomplex_scale:
                                                list[Parameter | str] | None = None, scale:
                                                Parameter | str | None = None)
```

Bases: `ModelItem`  
A model for datasets.  
Method generated by attrs for class `DatasetModel`.

Attributes Summary

<i>group</i>
<i>force_index_dependent</i>
<i>megacomplex</i>
<i>megacomplex_scale</i>
<i>global_megacomplex</i>
<i>global_megacomplex_scale</i>
<i>scale</i>
<i>label</i>

**group**

`DatasetModel.group: str`

**force\_index\_dependent**

`DatasetModel.force_index_dependent: bool`

**megacomplex**

`DatasetModel.megacomplex: list[Megacomplex | str]`

**megacomplex\_scale**

`DatasetModel.megacomplex_scale: list[Parameter | str] | None`

**global\_megacomplex**

`DatasetModel.global_megacomplex: list[Megacomplex | str] | None`

**global\_megacomplex\_scale**

`DatasetModel.global_megacomplex_scale: list[Parameter | str] | None`

**scale**

`DatasetModel.scale: Parameter | str | None`

**label**

`DatasetModel.label: str`

**Methods Summary**



## Methods Documentation

`force_index_dependent: bool`  
`global_megacomplex: list[Megacomplex | str] | None`  
`global_megacomplex_scale: list[Parameter | str] | None`  
`group: str`  
`label: str`  
`megacomplex: list[Megacomplex | str]`  
`megacomplex_scale: list[Parameter | str] | None`  
`scale: Parameter | str | None`

## ExclusiveMegacomplexIssue

```
class glotaran.model.dataset_model.ExclusiveMegacomplexIssue(label: str,
                                                             megacomplex_type: str,
                                                             is_global: bool)
```

Bases: `ItemIssue`

Issue for exclusive megacomplexes.

Create an ExclusiveMegacomplexIssue.

### Parameters

- **label** (*str*) – The megacomplex label.
- **megacomplex\_type** (*str*) – The megacomplex type.
- **is\_global** (*bool*) – Whether the megacomplex is global.

## Methods Summary

<code>to_string</code>	Get the issue as string.
------------------------	--------------------------

### to\_string

ExclusiveMegacomplexIssue.**to\_string**() → *str*

Get the issue as string.

### Return type

*str*

## Methods Documentation

**to\_string()** → str

Get the issue as string.

**Return type**

str

## UniqueMegacomplexIssue

```
class glotaran.model.dataset_model.UniqueMegacomplexIssue(label: str,  
                                                           megacomplex_type: str,  
                                                           is_global: bool)
```

Bases: ItemIssue

Issue for unique megacomplexes.

Create a UniqueMegacomplexIssue.

### Parameters

- **label** (str) – The megacomplex label.
- **megacomplex\_type** (str) – The megacomplex type.
- **is\_global** (bool) – Whether the megacomplex is global.

## Methods Summary

<code>to_string</code>	Get the issue as string.
------------------------	--------------------------

## to\_string

UniqueMegacomplexIssue.**to\_string()**

Get the issue as string.

**Return type**

str

## Methods Documentation

**to\_string()**

Get the issue as string.

**Return type**

str

## interval\_item

This module contains the interval item.

### Classes

#### Summary

<i>IntervalItem</i>	An item with an interval.
---------------------	---------------------------

#### IntervalItem

```
class glotaran.model.interval_item.IntervalItem(*, interval: tuple[float, float] |
list[tuple[float, float]] | None = None)
```

Bases: `Item`

An item with an interval.

Method generated by attrs for class IntervalItem.

#### Attributes Summary

<i>interval</i>
-----------------

#### interval

IntervalItem.interval: `tuple[float, float] | list[tuple[float, float]] | None`

#### Methods Summary

<i>applies</i>	Check if the index is in the intervals.
<i>has_interval</i>	Check if intervals are defined.

#### applies

IntervalItem.applies(index: `float | None`) → `bool`

Check if the index is in the intervals.

##### Parameters

**index** (`float`) – The index.

##### Return type

`bool`

### has\_interval

IntervalItem.**has\_interval**() → bool

Check if intervals are defined.

**Return type**

bool

### Methods Documentation

**applies**(index: float | None) → bool

Check if the index is in the intervals.

**Parameters**

**index** (float) – The index.

**Return type**

bool

**has\_interval**() → bool

Check if intervals are defined.

**Return type**

bool

**interval**: tuple[float, float] | list[tuple[float, float]] | None

### item

glotaran.model.**item**(cls: type[ItemT]) → type[ItemT]

Create an item from a class.

**Parameters**

**cls** (type[ItemT]) – The class.

**Return type**

type[ItemT]

### megacomplex

glotaran.model.**megacomplex**(\*, dataset\_model\_type: type[DatasetModel] | None = None, exclusive: bool = False, unique: bool = False) → Callable

Create a megacomplex from a class.

**Parameters**

- **dataset\_model\_type** (type) – The dataset model type.
- **exclusive** (bool) – Whether the megacomplex is exclusive.
- **unique** (bool) – Whether the megacomplex is unique.

**Return type**

Callable

## model

This module contains the model.

## Classes

### Summary

<i>Model</i>	A model for global target analysis.
--------------	-------------------------------------

### Model

```
class glotaran.model.model.Model(*, clp_penalties=_Nothing.NOTHING,
                                  clp_constraints=_Nothing.NOTHING,
                                  clp_relations=_Nothing.NOTHING, dataset_groups: dict[str,
                                  DatasetGroupModel | Any] = _Nothing.NOTHING, dataset:
                                  dict[str, DatasetModel], megacomplex=_Nothing.NOTHING,
                                  weights=_Nothing.NOTHING)
```

Bases: `object`

A model for global target analysis.

Method generated by attrs for class Model.

### Attributes Summary

<i>source_path</i>
<i>clp_penalties</i>
<i>clp_constraints</i>
<i>clp_relations</i>
<i>dataset_groups</i>
<i>dataset</i>
<i>megacomplex</i>
<i>weights</i>

**source\_path**

`Model.source_path: str | None`

**clp\_penalties**

`Model.clp_penalties: list[ClpPenalty]`

**clp\_constraints**

`Model.clp_constraints: list[ClpConstraint]`

**clp\_relations**

`Model.clp_relations: list[ClpRelation]`

**dataset\_groups**

`Model.dataset_groups: dict[str, DatasetGroupModel]`

**dataset**

`Model.dataset: dict[str, DatasetModel]`

**megacomplex**

`Model.megacomplex: dict[str, Megacomplex]`

**weights**

`Model.weights: list[Weight]`

## Methods Summary

<code>as_dict</code>	Get the model as dictionary.
<code>create_class</code>	Create model class.
<code>create_class_from_megacomplexes</code>	Create model class for megacomplexes.
<code>generate_parameters</code>	Generate parameters for the model.
<code>get_dataset_groups</code>	Get the dataset groups.
<code>get_issues</code>	Get issues.
<code>get_parameter_labels</code>	Get all parameter labels.
<code>iterate_all_items</code>	Iterate the individual items.
<code>iterate_items</code>	Iterate items.
<code>loader</code>	Create a Model instance from the specs defined in a file.
<code>markdown</code>	Format the model as Markdown string.
<code>valid</code>	Check if the model is valid.
<code>validate</code>	Get a string listing all issues in the model and missing parameters if specified.

### as\_dict

`Model.as_dict()` → `dict`

Get the model as dictionary.

**Return type**

`dict`

### create\_class

**classmethod** `Model.create_class(attributes: dict[str, Attribute])` → `type`[*Model*]

Create model class.

**Parameters**

**attributes** (*dict*[*str*, *Attribute*]) – The model attributes.

**Return type**

`type`[*Model*]

### create\_class\_from\_megacomplexes

**classmethod** `Model.create_class_from_megacomplexes(megacomplexes: Iterable[type[Megacomplex]])`  
→ `type`[*Model*]

Create model class for megacomplexes.

**Parameters**

**megacomplexes** (*list*[*type*[*Megacomplex*]]) – The megacomplexes.

**Return type**

`type`[*Model*]

### generate\_parameters

`Model.generate_parameters()` → *Parameters*

Generate parameters for the model.

**Returns**

- *Parameters* – The generated parameters.
- .. # noqa (D414)

### get\_dataset\_groups

`Model.get_dataset_groups()` → `dict[str, DatasetGroup]`

Get the dataset groups.

**Return type**

`dict[str, DatasetGroup]`

**Raises**

**ModelError** – Raised if a dataset group is unknown.

### get\_issues

`Model.get_issues(*, parameters: Parameters | None = None)` → `list[ItemIssue]`

Get issues.

**Parameters**

**parameters** (*Parameters* | *None*) – The parameters.

**Return type**

`list[ItemIssue]`

### get\_parameter\_labels

`Model.get_parameter_labels()` → `set[str]`

Get all parameter labels.

**Return type**

`set[str]`

### iterate\_all\_items

`Model.iterate_all_items()` → `Generator[Item, None, None]`

Iterate the individual items.

**Yields**

*Item* – The individual item.



## iterate\_items

`Model.iterate_items()` → `Generator[tuple[str, dict[str, Item] | list[Item]], None, None]`

Iterate items.

### Yields

`tuple[str, dict[str, Item] | list[Item]]` – The name of the item and the individual items of the type.

## loader

`Model.loader(format_name: str | None = None, **kwargs: Any)` → `Model`

Create a Model instance from the specs defined in a file.

### Parameters

- **file\_name** (`StrOrPath`) – File containing the model specs.
- **format\_name** (`str`) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (`Any`) – Additional keyword arguments passes to the `load_model` implementation of the project io plugin.

### Returns

Model instance created from the file.

### Return type

`Model`

## markdown

`Model.markdown(parameters: Parameters | None = None, initial_parameters: Parameters | None = None, base_heading_level: int = 1)` → `MarkdownStr`

Format the model as Markdown string.

Parameters will be included if specified.

### Parameters

- **parameters** (`Parameters` / `None`) – Parameter to include.
- **initial\_parameters** (`Parameters` / `None`) – Initial values for the parameters.
- **base\_heading\_level** (`int`) – Base heading level of the markdown sections.

E.g.:

- If it is 1 the string will start with ‘# Model’.
- If it is 3 the string will start with ‘### Model’.

### Return type

`MarkdownStr`

**valid**

`Model.valid(parameters: Parameters | None = None) → bool`

Check if the model is valid.

**Parameters**

**parameters** (`Parameters` / `None`) – The parameters.

**Return type**

`bool`

**validate**

`Model.validate(parameters: Parameters | None = None, raise_exception: bool = False) → MarkdownStr`

Get a string listing all issues in the model and missing parameters if specified.

**Parameters**

- **parameters** (`Parameters` / `None`) – The parameters.
- **raise\_exception** (`bool`) – Whether to raise an exception on failed validation.

**Return type**

*MarkdownStr*

**Raises**

**ModelError** – Raised if validation fails and `raise_exception` is true.

**Methods Documentation**

`as_dict()` → `dict`

Get the model as dictionary.

**Return type**

`dict`

`clp_constraints: list[ClpConstraint]`

`clp_penalties: list[ClpPenalty]`

`clp_relations: list[ClpRelation]`

**classmethod** `create_class(attributes: dict[str, Attribute]) → type[Model]`

Create model class.

**Parameters**

**attributes** (`dict[str, Attribute]`) – The model attributes.

**Return type**

`type[Model]`

**classmethod** `create_class_from_megacomplexes(megacomplexes: Iterable[type[Megacomplex]]) → type[Model]`

Create model class for megacomplexes.

**Parameters**

**megacomplexes** (*list* [*type* [*Megacomplex*]]) – The megacomplexes.

**Return type**

*type* [*Model*]

**dataset:** *dict* [*str*, *DatasetModel*]

**dataset\_groups:** *dict* [*str*, *DatasetGroupModel*]

**generate\_parameters()** → *Parameters*

Generate parameters for the model.

**Returns**

- *Parameters* – The generated parameters.
- .. # noqa (D414)

**get\_dataset\_groups()** → *dict* [*str*, *DatasetGroup*]

Get the dataset groups.

**Return type**

*dict* [*str*, *DatasetGroup*]

**Raises**

**ModelError** – Raised if a dataset group is unknown.

**get\_issues**(*\**, *parameters:* *Parameters* | *None* = *None*) → *list* [*ItemIssue*]

Get issues.

**Parameters**

**parameters** (*Parameters* | *None*) – The parameters.

**Return type**

*list* [*ItemIssue*]

**get\_parameter\_labels()** → *set* [*str*]

Get all parameter labels.

**Return type**

*set* [*str*]

**iterate\_all\_items()** → *Generator* [*Item*, *None*, *None*]

Iterate the individual items.

**Yields**

*Item* – The individual item.

**iterate\_items()** → *Generator* [*tuple* [*str*, *dict* [*str*, *Item*] | *list* [*Item*]], *None*, *None*]

Iterate items.

**Yields**

*tuple* [*str*, *dict* [*str*, *Item*] | *list* [*Item*]] – The name of the item and the individual items of the type.

**loader**(*format\_name:* *str* | *None* = *None*, *\*\*kwargs:* *Any*) → *Model*

Create a Model instance from the specs defined in a file.

**Parameters**

- **file\_name** (*StrOrPath*) – File containing the model specs.

- **format\_name** (*str*) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `load_model` implementation of the project io plugin.

**Returns**

Model instance created from the file.

**Return type**

*Model*

**markdown**(*parameters*: *Parameters* | *None* = *None*, *initial\_parameters*: *Parameters* | *None* = *None*, *base\_heading\_level*: *int* = 1) → *MarkdownStr*

Format the model as Markdown string.

Parameters will be included if specified.

**Parameters**

- **parameters** (*Parameters* | *None*) – Parameter to include.
- **initial\_parameters** (*Parameters* | *None*) – Initial values for the parameters.
- **base\_heading\_level** (*int*) – Base heading level of the markdown sections.

E.g.:

- If it is 1 the string will start with ‘# Model’.
- If it is 3 the string will start with ‘### Model’.

**Return type**

*MarkdownStr*

**megacomplex**: *dict*[*str*, *Megacomplex*]

**source\_path**: *str* | *None*

**valid**(*parameters*: *Parameters* | *None* = *None*) → *bool*

Check if the model is valid.

**Parameters**

**parameters** (*Parameters* | *None*) – The parameters.

**Return type**

*bool*

**validate**(*parameters*: *Parameters* | *None* = *None*, *raise\_exception*: *bool* = *False*) → *MarkdownStr*

Get a string listing all issues in the model and missing parameters if specified.

**Parameters**

- **parameters** (*Parameters* | *None*) – The parameters.
- **raise\_exception** (*bool*) – Whether to raise an exception on failed validation.

**Return type**

*MarkdownStr*

**Raises**

**ModelError** – Raised if validation fails and `raise_exception` is true.

**weights:** `list[Weight]`

## Exceptions

### Exception Summary

<code>ModelError</code>	Raised when a model contains errors.
-------------------------	--------------------------------------

### ModelError

**exception** `glotaran.model.model.ModelError(error: str)`

Raised when a model contains errors.

Create a model error.

#### Parameters

**error** (*str*) – The error string.

## weight

This module contains weight item.

## Classes

### Summary

<i>Weight</i>	The <i>Weight</i> class describes a value by which a dataset will scaled.
---------------	---

### Weight

**class** `glotaran.model.weight.Weight(*, datasets: list[str], global_interval: tuple[float, float] | None = None, model_interval: tuple[float, float] | None = None, value: float)`

Bases: `Item`

The *Weight* class describes a value by which a dataset will scaled.

*global\_interval* and *model\_interval* are optional. The whole range of the dataset will be used if not set.

Method generated by attrs for class *Weight*.

### Attributes Summary

<i>datasets</i>
<i>global_interval</i>
<i>model_interval</i>
<i>value</i>

#### datasets

Weight.datasets: `list[str]`

#### global\_interval

Weight.global\_interval: `tuple[float, float] | None`

#### model\_interval

Weight.model\_interval: `tuple[float, float] | None`

#### value

Weight.value: `float`

### Methods Summary

#### Methods Documentation

datasets: `list[str]`

global\_interval: `tuple[float, float] | None`

model\_interval: `tuple[float, float] | None`

value: `float`

## 16.1.7 optimization

This package contains functions for optimization.

### Modules

<code>glotaran.optimization.data_provider</code>	Module containing the data provider classes.
<code>glotaran.optimization.estimation_provider</code>	Module containing the estimation provider classes.
<code>glotaran.optimization.matrix_provider</code>	Module containing the matrix provider classes.
<code>glotaran.optimization.nnls</code>	Module for residual calculation with the non-negative least-squares method.
<code>glotaran.optimization.optimization_group</code>	Module containing the optimization group class.
<code>glotaran.optimization.optimization_history</code>	Module containing the OptimizationHistory class.
<code>glotaran.optimization.optimize</code>	Module containing the optimize function.
<code>glotaran.optimization.optimizer</code>	Module containing the optimizer class.
<code>glotaran.optimization.variable_projection</code>	Module for residual calculation with the variable projection method.

### data\_provider

Module containing the data provider classes.

### Classes

#### Summary

<code>DataProvider</code>	A class to provide prepared data for optimization.
<code>DataProviderLinked</code>	A class to provide aligned data for optimization.

### DataProvider

```
class glotaran.optimization.data_provider.DataProvider(scheme: Scheme,
                                                    dataset_group: DatasetGroup)
```

Bases: `object`

A class to provide prepared data for optimization.

Initialize a data provider for a scheme and a dataset\_group.

#### Parameters

- **scheme** ([Scheme](#)) – The optimization scheme.
- **dataset\_group** ([DatasetGroup](#)) – The dataset group.

## Methods Summary

<code>add_model_weight</code>	Add model weight to data.
<code>get_axis_slice_from_interval</code>	Get a slice of indices from a min max tuple and for an axis.
<code>get_data</code>	Get data for a dataset.
<code>get_flattened_data</code>	Get flattened data for a dataset.
<code>get_flattened_weight</code>	Get flattened weight for a dataset.
<code>get_from_dataset</code>	Get a copy of data from a dataset with dimensions (model, global).
<code>get_global_axis</code>	Get the global axis for a dataset.
<code>get_global_dimension</code>	Get the global dimension for a dataset.
<code>get_model_axis</code>	Get the model axis for a dataset.
<code>get_model_dimension</code>	Get the model dimension for a dataset.
<code>get_weight</code>	Get weight for a dataset.
<code>infer_global_dimension</code>	Infer the name of the global dimension from tuple of dimensions.

### `add_model_weight`

`DataProvider.add_model_weight(model: Model, dataset_label: str, model_dimension: str, global_dimension: str)`

Add model weight to data.

#### Parameters

- **model** ([Model](#)) – The model.
- **dataset\_label** (*str*) – The label of the data.
- **model\_dimension** (*str*) – The model dimension.
- **global\_dimension** (*str*) – The global dimension.

### `get_axis_slice_from_interval`

`static DataProvider.get_axis_slice_from_interval(interval: tuple[float, float], axis: ArrayLike) → slice`

Get a slice of indices from a min max tuple and for an axis.

#### Parameters

- **interval** (*tuple*[*float*, *float*]) – The min max tuple.
- **axis** (*ArrayLike*) – The axis to slice.

#### Returns

The slice of indices.

#### Return type

*slice*



### get\_data

`DataProvider.get_data(dataset_label: str) → ArrayLike`

Get data for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The data.

**Return type**

ArrayLike

### get\_flattened\_data

`DataProvider.get_flattened_data(dataset_label: str) → ArrayLike`

Get flattened data for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The flattened data.

**Return type**

ArrayLike

### get\_flattened\_weight

`DataProvider.get_flattened_weight(dataset_label: str) → ArrayLike | None`

Get flattened weight for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The flattened weight.

**Return type**

ArrayLike | None

### get\_from\_dataset

**static** `DataProvider.get_from_dataset(dataset: xr.Dataset, name: str, model_dimension: str, global_dimension: str) → ArrayLike | None`

Get a copy of data from a dataset with dimensions (model, global).

**Parameters**

- **dataset** (*xr.Dataset*) – The dataset to retrieve from.
- **name** (*str*) – The name of the data to retrieve.
- **model\_dimension** (*str*) – The model dimension.
- **global\_dimension** (*str*) – The global dimension.

**Returns**

The copy of the data. None if name is not present in dataset.

**Return type**

ArrayLike | None

**get\_global\_axis**

`DataProvider.get_global_axis(dataset_label: str) → ArrayLike`

Get the global axis for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The global axis.

**Return type**

ArrayLike

**get\_global\_dimension**

`DataProvider.get_global_dimension(dataset_label: str) → str`

Get the global dimension for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The global dimension.

**Return type**

*str*

**get\_model\_axis**

`DataProvider.get_model_axis(dataset_label: str) → ArrayLike`

Get the model axis for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The model axis.

**Return type**

ArrayLike

### get\_model\_dimension

`DataProvider.get_model_dimension(dataset_label: str) → str`

Get the model dimension for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The model dimension.

**Return type**

*str*

### get\_weight

`DataProvider.get_weight(dataset_label: str) → ArrayLike | None`

Get weight for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The weight.

**Return type**

*ArrayLike | None*

### infer\_global\_dimension

**static** `DataProvider.infer_global_dimension(model_dimension: str, dimensions: tuple[str]) → str`

Infer the name of the global dimension from tuple of dimensions.

**Parameters**

- **model\_dimension** (*str*) – The model dimension.
- **dimensions** (*tuple[str]*) – The dimensions tuple to infer from.

**Returns**

The inferred name of the global dimension.

**Return type**

*str*

### Methods Documentation

`add_model_weight(model: Model, dataset_label: str, model_dimension: str, global_dimension: str)`

Add model weight to data.

**Parameters**

- **model** (*Model*) – The model.
- **dataset\_label** (*str*) – The label of the data.

- **model\_dimension** (*str*) – The model dimension.
- **global\_dimension** (*str*) – The global dimension.

**static** **get\_axis\_slice\_from\_interval**(*interval*: *tuple*[*float*, *float*], *axis*: *ArrayLike*) → *slice*

Get a slice of indices from a min max tuple and for an axis.

**Parameters**

- **interval** (*tuple*[*float*, *float*]) – The min max tuple.
- **axis** (*ArrayLike*) – The axis to slice.

**Returns**

The slice of indices.

**Return type**

*slice*

**get\_data**(*dataset\_label*: *str*) → *ArrayLike*

Get data for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The data.

**Return type**

*ArrayLike*

**get\_flattened\_data**(*dataset\_label*: *str*) → *ArrayLike*

Get flattened data for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The flattened data.

**Return type**

*ArrayLike*

**get\_flattened\_weight**(*dataset\_label*: *str*) → *ArrayLike* | *None*

Get flattened weight for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The flattened weight.

**Return type**

*ArrayLike* | *None*

**static** **get\_from\_dataset**(*dataset*: *xr.Dataset*, *name*: *str*, *model\_dimension*: *str*, *global\_dimension*: *str*) → *ArrayLike* | *None*

Get a copy of data from a dataset with dimensions (model, global).

**Parameters**

- **dataset** (*xr.Dataset*) – The dataset to retrieve from.

- **name** (*str*) – The name of the data to retrieve.
- **model\_dimension** (*str*) – The model dimension.
- **global\_dimension** (*str*) – The global dimension.

**Returns**

The copy of the data. None if name is not present in dataset.

**Return type**

ArrayLike | None

**get\_global\_axis**(*dataset\_label: str*) → ArrayLike

Get the global axis for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The global axis.

**Return type**

ArrayLike

**get\_global\_dimension**(*dataset\_label: str*) → *str*

Get the global dimension for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The global dimension.

**Return type**

*str*

**get\_model\_axis**(*dataset\_label: str*) → ArrayLike

Get the model axis for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The model axis.

**Return type**

ArrayLike

**get\_model\_dimension**(*dataset\_label: str*) → *str*

Get the model dimension for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The model dimension.

**Return type**

*str*

**get\_weight**(*dataset\_label: str*) → ArrayLike | None

Get weight for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The weight.

**Return type**

ArrayLike | None

**static infer\_global\_dimension**(*model\_dimension: str, dimensions: tuple[str]*) → *str*

Infer the name of the global dimension from tuple of dimensions.

**Parameters**

- **model\_dimension** (*str*) – The model dimension.
- **dimensions** (*tuple[str]*) – The dimensions tuple to infer from.

**Returns**

The inferred name of the global dimension.

**Return type**

*str*

## DataProviderLinked

```
class glotaran.optimization.data_provider.DataProviderLinked(scheme: Scheme,  
                                                           dataset_group:  
                                                           DatasetGroup)
```

Bases: *DataProvider*

A class to provide aligned data for optimization.

Initialize a linked data provider for a scheme and a dataset\_group.

**Parameters**

- **scheme** (*Scheme*) – The optimization scheme.
- **dataset\_group** (*DatasetGroup*) – The dataset group.

## Attributes Summary

<i>aligned_global_axis</i>	Get the aligned global axis for the dataset group.
<i>group_definitions</i>	Get the group definitions for the dataset group.

## aligned\_global\_axis

### DataProviderLinked.aligned\_global\_axis

Get the aligned global axis for the dataset group.

**Returns**

The aligned global axis.

**Return type**

ArrayLike

## group\_definitions

### DataProviderLinked.group\_definitions

Get the group definitions for the dataset group.

**Returns**

The group definitions.

**Return type**

dict[str, list[str]]

## Methods Summary

<code>add_model_weight</code>	Add model weight to data.
<code>align_data</code>	Align the data in a dataset group.
<code>align_dataset_indices</code>	Align the global indices in a dataset group.
<code>align_groups</code>	Align the groups in a dataset group.
<code>align_index</code>	Align an index on a target axis.
<code>align_weights</code>	Align the weights in a dataset group.
<code>create_aligned_global_axes</code>	Create aligned global axes for the dataset group.
<code>get_aligned_data</code>	Get the aligned data for an index.
<code>get_aligned_dataset_indices</code>	Get the aligned dataset indices for an index.
<code>get_aligned_group_label</code>	Get the group label for an index.
<code>get_aligned_weight</code>	Get the aligned weight for an index.
<code>get_axis_slice_from_interval</code>	Get a slice of indices from a min max tuple and for an axis.
<code>get_data</code>	Get data for a dataset.
<code>get_flattened_data</code>	Get flattened data for a dataset.
<code>get_flattened_weight</code>	Get flattened weight for a dataset.
<code>get_from_dataset</code>	Get a copy of data from a dataset with dimensions (model, global).
<code>get_global_axis</code>	Get the global axis for a dataset.
<code>get_global_dimension</code>	Get the global dimension for a dataset.
<code>get_model_axis</code>	Get the model axis for a dataset.
<code>get_model_dimension</code>	Get the model dimension for a dataset.
<code>get_weight</code>	Get weight for a dataset.
<code>infer_global_dimension</code>	Infer the name of the global dimension from tuple of dimensions.

### add\_model\_weight

`DataProviderLinked.add_model_weight(model: Model, dataset_label: str,  
model_dimension: str, global_dimension: str)`

Add model weight to data.

#### Parameters

- **model** ([Model](#)) – The model.
- **dataset\_label** (*str*) – The label of the data.
- **model\_dimension** (*str*) – The model dimension.
- **global\_dimension** (*str*) – The global dimension.

### align\_data

`DataProviderLinked.align_data(aligned_global_axes: dict[str, ArrayLike] →  
tuple[ArrayLike, list[ArrayLike]])`

Align the data in a dataset group.

#### Parameters

**aligned\_global\_axes** (*dict*[*str*, *ArrayLike*]) – The aligned global axes.

#### Returns

The aligned global axis and data.

#### Return type

*tuple*[*ArrayLike*, *list*[*ArrayLike*]]

### align\_dataset\_indices

`DataProviderLinked.align_dataset_indices(aligned_global_axes: dict[str, ArrayLike] →  
→ list[ArrayLike])`

Align the global indices in a dataset group.

#### Parameters

**aligned\_global\_axes** (*dict*[*str*, *ArrayLike*]) – The aligned global axes.

#### Returns

- *list*[*ArrayLike*]
- The aligned dataset indices.

### align\_groups

`static DataProviderLinked.align_groups(aligned_global_axes: dict[str, ArrayLike] →  
tuple[ArrayLike, dict[str, list[str]])`

Align the groups in a dataset group.

#### Parameters

**aligned\_global\_axes** (*dict*[*str*, *ArrayLike*]) – The aligned global axes.



**Returns**

The aligned grouplabels and group definitions.

**Return type**

`tuple[ArrayLike, dict[str, list[str]]]`

**align\_index**

**static** `DataProviderLinked.align_index(index: int, target_axis: ArrayLike, tolerance: float, method: Literal['nearest', 'backward', 'forward'])` → `int`

Align an index on a target axis.

**Parameters**

- **index** (*int*) – The index to align.
- **target\_axis** (*ArrayLike*) – The axis to align the index on.
- **tolerance** (*float*) – The alignment tolerance.
- **method** (*Literal*["nearest", "backward", "forward"]) – The alignment method.

**Returns**

The aligned index.

**Return type**

`int`

**align\_weights**

`DataProviderLinked.align_weights(aligned_global_axes: dict[str, ArrayLike])` → `list[ArrayLike | None]`

Align the weights in a dataset group.

**Parameters**

**aligned\_global\_axes** (*dict*[*str*, *ArrayLike*]) – The aligned global axes.

**Returns**

The aligned weights.

**Return type**

`list[ArrayLike | None]`

**create\_aligned\_global\_axes**

`DataProviderLinked.create_aligned_global_axes(scheme: Scheme)` → `dict[str, ArrayLike]`

Create aligned global axes for the dataset group.

**Parameters**

**scheme** (*Scheme*) – The optimization scheme.

**Returns**

The aligned global axes.

**Return type**`dict[str, ArrayLike]`**Raises****AlignDatasetError** – Raised when dataset alignment is ambiguous.**get\_aligned\_data**`DataProviderLinked.get_aligned_data(index: int) → ArrayLike`

Get the aligned data for an index.

**Parameters****index** (*int*) – The index on the aligned global axis.**Returns**

The aligned data.

**Return type**`ArrayLike`**get\_aligned\_dataset\_indices**`DataProviderLinked.get_aligned_dataset_indices(index: int) → ArrayLike`

Get the aligned dataset indices for an index.

**Parameters****index** (*int*) – The index on the aligned global axis.**Returns**

The aligned dataset indices.

**Return type**`ArrayLike`**get\_aligned\_group\_label**`DataProviderLinked.get_aligned_group_label(index: int) → str`

Get the group label for an index.

**Parameters****index** (*int*) – The index on the aligned global axis.**Returns**

The aligned group label.

**Return type**`str`

### get\_aligned\_weight

`DataProviderLinked.get_aligned_weight(index: int) → ArrayLike | None`

Get the aligned weight for an index.

**Parameters**

**index** (*int*) – The index on the aligned global axis.

**Returns**

The aligned weight.

**Return type**

ArrayLike | None

### get\_axis\_slice\_from\_interval

`static DataProviderLinked.get_axis_slice_from_interval(interval: tuple[float, float], axis: ArrayLike) → slice`

Get a slice of indices from a min max tuple and for an axis.

**Parameters**

- **interval** (*tuple*[float, float]) – The min max tuple.
- **axis** (*ArrayLike*) – The axis to slice.

**Returns**

The slice of indices.

**Return type**

*slice*

### get\_data

`DataProviderLinked.get_data(dataset_label: str) → ArrayLike`

Get data for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The data.

**Return type**

ArrayLike

### `get_flattened_data`

`DataProviderLinked.get_flattened_data(dataset_label: str) → ArrayLike`

Get flattened data for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The flattened data.

**Return type**

ArrayLike

### `get_flattened_weight`

`DataProviderLinked.get_flattened_weight(dataset_label: str) → ArrayLike | None`

Get flattened weight for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The flattened weight.

**Return type**

ArrayLike | None

### `get_from_dataset`

**static** `DataProviderLinked.get_from_dataset(dataset: xr.Dataset, name: str, model_dimension: str, global_dimension: str) → ArrayLike | None`

Get a copy of data from a dataset with dimensions (model, global).

**Parameters**

- **dataset** (*xr.Dataset*) – The dataset to retrieve from.
- **name** (*str*) – The name of the data to retrieve.
- **model\_dimension** (*str*) – The model dimension.
- **global\_dimension** (*str*) – The global dimension.

**Returns**

The copy of the data. None if name is not present in dataset.

**Return type**

ArrayLike | None

### get\_global\_axis

DataProviderLinked.**get\_global\_axis**(dataset\_label: *str*) → ArrayLike

Get the global axis for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The global axis.

**Return type**

ArrayLike

### get\_global\_dimension

DataProviderLinked.**get\_global\_dimension**(dataset\_label: *str*) → *str*

Get the global dimension for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The global dimension.

**Return type**

*str*

### get\_model\_axis

DataProviderLinked.**get\_model\_axis**(dataset\_label: *str*) → ArrayLike

Get the model axis for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The model axis.

**Return type**

ArrayLike

### get\_model\_dimension

DataProviderLinked.**get\_model\_dimension**(dataset\_label: *str*) → *str*

Get the model dimension for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The model dimension.

**Return type**

*str*

### get\_weight

`DataProviderLinked.get_weight(dataset_label: str) → ArrayLike | None`

Get weight for a dataset.

#### Parameters

**dataset\_label** (*str*) – The label of the data.

#### Returns

The weight.

#### Return type

ArrayLike | None

### infer\_global\_dimension

`static DataProviderLinked.infer_global_dimension(model_dimension: str, dimensions: tuple[str]) → str`

Infer the name of the global dimension from tuple of dimensions.

#### Parameters

- **model\_dimension** (*str*) – The model dimension.
- **dimensions** (*tuple[str]*) – The dimensions tuple to infer from.

#### Returns

The inferred name of the global dimension.

#### Return type

*str*

## Methods Documentation

`add_model_weight(model: Model, dataset_label: str, model_dimension: str, global_dimension: str)`

Add model weight to data.

#### Parameters

- **model** (*Model*) – The model.
- **dataset\_label** (*str*) – The label of the data.
- **model\_dimension** (*str*) – The model dimension.
- **global\_dimension** (*str*) – The global dimension.

`align_data(aligned_global_axes: dict[str, ArrayLike]) → tuple[ArrayLike, list[ArrayLike]]`

Align the data in a dataset group.

#### Parameters

**aligned\_global\_axes** (*dict[str, ArrayLike]*) – The aligned global axes.

#### Returns

The aligned global axis and data.

#### Return type

*tuple[ArrayLike, list[ArrayLike]]*

**align\_dataset\_indices**(*aligned\_global\_axes*: *dict[str, ArrayLike]*) → *list[ArrayLike]*

Align the global indices in a dataset group.

**Parameters**

**aligned\_global\_axes** (*dict[str, ArrayLike]*) – The aligned global axes.

**Returns**

- *list[ArrayLike]*
- *The aligned dataset indices.*

**static align\_groups**(*aligned\_global\_axes*: *dict[str, ArrayLike]*) → *tuple[ArrayLike, dict[str, list[str]]]*

Align the groups in a dataset group.

**Parameters**

**aligned\_global\_axes** (*dict[str, ArrayLike]*) – The aligned global axes.

**Returns**

The aligned grouplabels and group definitions.

**Return type**

*tuple[ArrayLike, dict[str, list[str]]]*

**static align\_index**(*index*: *int*, *target\_axis*: *ArrayLike*, *tolerance*: *float*, *method*: *Literal['nearest', 'backward', 'forward']*) → *int*

Align an index on a target axis.

**Parameters**

- **index** (*int*) – The index to align.
- **target\_axis** (*ArrayLike*) – The axis to align the index on.
- **tolerance** (*float*) – The alignment tolerance.
- **method** (*Literal["nearest", "backward", "forward"]*) – The alignment method.

**Returns**

The aligned index.

**Return type**

*int*

**align\_weights**(*aligned\_global\_axes*: *dict[str, ArrayLike]*) → *list[ArrayLike | None]*

Align the weights in a dataset group.

**Parameters**

**aligned\_global\_axes** (*dict[str, ArrayLike]*) – The aligned global axes.

**Returns**

The aligned weights.

**Return type**

*list[ArrayLike | None]*

**property aligned\_global\_axis**: *ArrayLike*

Get the aligned global axis for the dataset group.

**Returns**

The aligned global axis.

**Return type**

ArrayLike

**create\_aligned\_global\_axes**(*scheme*: [Scheme](#)) → [dict](#)[[str](#), ArrayLike]

Create aligned global axes for the dataset group.

**Parameters****scheme** ([Scheme](#)) – The optimization scheme.**Returns**

The aligned global axes.

**Return type**[dict](#)[[str](#), ArrayLike]**Raises****AlignDatasetError** – Raised when dataset alignment is ambiguous.**get\_aligned\_data**(*index*: [int](#)) → ArrayLike

Get the aligned data for an index.

**Parameters****index** ([int](#)) – The index on the aligned global axis.**Returns**

The aligned data.

**Return type**

ArrayLike

**get\_aligned\_dataset\_indices**(*index*: [int](#)) → ArrayLike

Get the aligned dataset indices for an index.

**Parameters****index** ([int](#)) – The index on the aligned global axis.**Returns**

The aligned dataset indices.

**Return type**

ArrayLike

**get\_aligned\_group\_label**(*index*: [int](#)) → [str](#)

Get the group label for an index.

**Parameters****index** ([int](#)) – The index on the aligned global axis.**Returns**

The aligned group label.

**Return type**[str](#)**get\_aligned\_weight**(*index*: [int](#)) → ArrayLike | [None](#)

Get the aligned weight for an index.

**Parameters****index** ([int](#)) – The index on the aligned global axis.**Returns**

The aligned weight.



**Return type**

ArrayLike | None

**static** `get_axis_slice_from_interval`(*interval*: *tuple*[*float*, *float*], *axis*: *ArrayLike*) → *slice*

Get a slice of indices from a min max tuple and for an axis.

**Parameters**

- **interval** (*tuple*[*float*, *float*]) – The min max tuple.
- **axis** (*ArrayLike*) – The axis to slice.

**Returns**

The slice of indices.

**Return type***slice*

**get\_data**(*dataset\_label*: *str*) → *ArrayLike*

Get data for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The data.

**Return type***ArrayLike*

**get\_flattened\_data**(*dataset\_label*: *str*) → *ArrayLike*

Get flattened data for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The flattened data.

**Return type***ArrayLike*

**get\_flattened\_weight**(*dataset\_label*: *str*) → *ArrayLike* | *None*

Get flattened weight for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The flattened weight.

**Return type***ArrayLike* | *None*

**static** `get_from_dataset`(*dataset*: *xr.Dataset*, *name*: *str*, *model\_dimension*: *str*, *global\_dimension*: *str*) → *ArrayLike* | *None*

Get a copy of data from a dataset with dimensions (model, global).

**Parameters**

- **dataset** (*xr.Dataset*) – The dataset to retrieve from.
- **name** (*str*) – The name of the data to retrieve.

- **model\_dimension** (*str*) – The model dimension.
- **global\_dimension** (*str*) – The global dimension.

**Returns**

The copy of the data. None if name is not present in dataset.

**Return type**

ArrayLike | None

**get\_global\_axis**(*dataset\_label: str*) → ArrayLike

Get the global axis for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The global axis.

**Return type**

ArrayLike

**get\_global\_dimension**(*dataset\_label: str*) → *str*

Get the global dimension for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The global dimension.

**Return type**

*str*

**get\_model\_axis**(*dataset\_label: str*) → ArrayLike

Get the model axis for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The model axis.

**Return type**

ArrayLike

**get\_model\_dimension**(*dataset\_label: str*) → *str*

Get the model dimension for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The model dimension.

**Return type**

*str*

**get\_weight**(*dataset\_label: str*) → ArrayLike | None

Get weight for a dataset.

**Parameters**

**dataset\_label** (*str*) – The label of the data.

**Returns**

The weight.

**Return type**

ArrayLike | None

**property group\_definitions:** `dict[str, list[str]]`

Get the group definitions for the dataset group.

**Returns**

The group definitions.

**Return type**

`dict[str, list[str]]`

**static infer\_global\_dimension**(*model\_dimension*: `str`, *dimensions*: `tuple[str]`) → `str`

Infer the name of the global dimension from tuple of dimensions.

**Parameters**

- **model\_dimension** (`str`) – The model dimension.
- **dimensions** (`tuple[str]`) – The dimensions tuple to infer from.

**Returns**

The inferred name of the global dimension.

**Return type**

`str`

## Exceptions

### Exception Summary

<code>AlignDatasetError</code>	Indicates that datasets can not be aligned.
--------------------------------	---

### `AlignDatasetError`

**exception** `glotaran.optimization.data_provider.AlignDatasetError`

Indicates that datasets can not be aligned.

Initialize a `AlignDatasetError`.

## `estimation_provider`

Module containing the estimation provider classes.

## Classes

### Summary

<i>EstimationProvider</i>	A class to provide estimation for optimization.
<i>EstimationProviderLinked</i>	A class to provide estimation for optimization of a linked dataset group.
<i>EstimationProviderUnlinked</i>	A class to provide estimation for optimization of an unlinked dataset group.

### EstimationProvider

**class** `glotaran.optimization.estimation_provider.EstimationProvider`(*dataset\_group*: `DatasetGroup`)

Bases: `object`

A class to provide estimation for optimization.

Initialize an estimation provider for a dataset group.

#### Parameters

**dataset\_group** (`DatasetGroup`) – The dataset group.

#### Raises

**UnsupportedResidualFunctionError** – Raised when residual function of the group dataset group is unsupported.

### Attributes Summary

<i>group</i>	Get the dataset group.
--------------	------------------------

### group

`EstimationProvider.group`

Get the dataset group.

#### Returns

The dataset group.

#### Return type

*DatasetGroup*

## Methods Summary

<code>calculate_clp_penalties</code>	Calculate the clp penalty.
<code>calculate_residual</code>	Calculate the clps and the residual for a matrix and data.
<code>estimate</code>	Calculate the estimation.
<code>get_additional_penalties</code>	Get the additional penalty.
<code>get_full_penalty</code>	Get the full penalty.
<code>get_result</code>	Get the results of the estimation.
<code>retrieve_clps</code>	Retrieve clp from reduced clp.

### calculate\_clp\_penalties

`EstimationProvider.calculate_clp_penalties`(*clp\_labels*: *list[list[str]]*, *clps*: *list[ndarray]*, *global\_axis*: *ndarray*) → *list[float]*

Calculate the clp penalty.

#### Parameters

- **clp\_labels** (*list[list[str]]*) – The clp labels.
- **clps** (*list[ArrayLike]*) – The clps.
- **global\_axis** (*ArrayLike*) – The global axis.

#### Returns

The clp penalty.

#### Return type

*list[float]*

### calculate\_residual

`EstimationProvider.calculate_residual`(*matrix*: *ArrayLike*, *data*: *ArrayLike*) → *tuple[ArrayLike, ArrayLike]*

Calculate the clps and the residual for a matrix and data.

#### Parameters

- **matrix** (*ArrayLike*) – The matrix.
- **data** (*ArrayLike*) – The data.

#### Returns

The estimated clp and residual.

#### Return type

*tuple[ArrayLike, ArrayLike]*

## estimate

`EstimationProvider.estimate()`

Calculate the estimation.

## get\_additional\_penalties

`EstimationProvider.get_additional_penalties() → list[float]`

Get the additional penalty.

### Returns

The additional penalty.

### Return type

`list[float]`

## get\_full\_penalty

`EstimationProvider.get_full_penalty() → ArrayLike`

Get the full penalty.

### Returns

- *ArrayLike* – The clp penalty.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

## get\_result

`EstimationProvider.get_result() → tuple[dict[str, DataArray], dict[str, DataArray]]`

Get the results of the estimation.

### Returns

- *tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]* – A tuple of the estimated clps and residuals.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

## retrieve\_clps

`EstimationProvider.retrieve_clps(clp_labels: list[str], reduced_clp_labels: list[str], reduced_clps: ArrayLike, index: int) → ArrayLike`

Retrieve clp from reduced clp.

### Parameters

- `clp_labels` (`list[str]`) – The original clp labels.
- `reduced_clp_labels` (`list[str]`) – The reduced clp labels.

- **reduced\_clps** (*ArrayLike*) – The reduced clps.
- **index** (*int*) – The index on the global axis.

**Returns**

The retrieved clps.

**Return type**

*ArrayLike*

**Methods Documentation**

**calculate\_clp\_penalties**(*clp\_labels: list[list[str]], clps: list[ndarray], global\_axis: ndarray*) → *list[float]*

Calculate the clp penalty.

**Parameters**

- **clp\_labels** (*list[list[str]]*) – The clp labels.
- **clps** (*list[ArrayLike]*) – The clps.
- **global\_axis** (*ArrayLike*) – The global axis.

**Returns**

The clp penalty.

**Return type**

*list[float]*

**calculate\_residual**(*matrix: ArrayLike, data: ArrayLike*) → *tuple[ArrayLike, ArrayLike]*

Calculate the clps and the residual for a matrix and data.

**Parameters**

- **matrix** (*ArrayLike*) – The matrix.
- **data** (*ArrayLike*) – The data.

**Returns**

The estimated clp and residual.

**Return type**

*tuple[ArrayLike, ArrayLike]*

**estimate()**

Calculate the estimation.

**get\_additional\_penalties()** → *list[float]*

Get the additional penalty.

**Returns**

The additional penalty.

**Return type**

*list[float]*

**get\_full\_penalty()** → *ArrayLike*

Get the full penalty.

**Returns**

- *ArrayLike* – The clp penalty.

- .. # noqa (DAR202)
- .. # noqa (DAR401)

**get\_result()** → tuple[dict[str, [DataArray](#)], dict[str, [DataArray](#)]]

Get the results of the estimation.

#### Returns

- tuple[dict[str, *xr.DataArray*], dict[str, *xr.DataArray*]] – A tuple of the estimated clps and residuals.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**property group:** [DatasetGroup](#)

Get the dataset group.

#### Returns

The dataset group.

#### Return type

[DatasetGroup](#)

**retrieve\_clps**(*clp\_labels*: list[str], *reduced\_clp\_labels*: list[str], *reduced\_clps*: ArrayLike, *index*: int) → ArrayLike

Retrieve clp from reduced clp.

#### Parameters

- **clp\_labels** (list[str]) – The original clp labels.
- **reduced\_clp\_labels** (list[str]) – The reduced clp labels.
- **reduced\_clps** (ArrayLike) – The reduced clps.
- **index** (int) – The index on the global axis.

#### Returns

The retrieved clps.

#### Return type

ArrayLike

### EstimationProviderLinked

```
class glotaran.optimization.estimation_provider.EstimationProviderLinked(dataset_group:  
    Dataset-  
    Group,  
    data_provider:  
    Dat-  
    aProvider-  
    Linked,  
    ma-  
    trix_provider:  
    Matrix-  
    Provider-  
    Linked)
```



Bases: *EstimationProvider*

A class to provide estimation for optimization of a linked dataset group.

Initialize an estimation provider for a linked dataset group.

#### Parameters

- **dataset\_group** (*DatasetGroup*) – The dataset group.
- **data\_provider** (*DataProviderLinked*) – The data provider.
- **matrix\_provider** (*MatrixProviderLinked*) – The matrix provider.

#### Attributes Summary

<i>group</i>	Get the dataset group.
--------------	------------------------

#### group

*EstimationProviderLinked*.**group**

Get the dataset group.

#### Returns

The dataset group.

#### Return type

*DatasetGroup*

#### Methods Summary

<i>calculate_clp_penalties</i>	Calculate the clp penalty.
<i>calculate_residual</i>	Calculate the clps and the residual for a matrix and data.
<i>estimate</i>	Calculate the estimation.
<i>get_additional_penalties</i>	Get the additional penalty.
<i>get_full_penalty</i>	Get the full penalty.
<i>get_result</i>	Get the results of the estimation.
<i>retrieve_clps</i>	Retrieve clp from reduced clp.

#### calculate\_clp\_penalties

*EstimationProviderLinked*.**calculate\_clp\_penalties**(*clp\_labels*: *list[list[str]]*, *clps*: *list[ndarray]*, *global\_axis*: *ndarray*) → *list[float]*

Calculate the clp penalty.

#### Parameters

- **clp\_labels** (*list[list[str]]*) – The clp labels.

- **clps** (*list*[*ArrayLike*]) – The clps.
- **global\_axis** (*ArrayLike*) – The global axis.

**Returns**

The clp penalty.

**Return type**

*list*[*float*]

**calculate\_residual**

`EstimationProviderLinked.calculate_residual(matrix: ArrayLike, data: ArrayLike) → tuple[ArrayLike, ArrayLike]`

Calculate the clps and the residual for a matrix and data.

**Parameters**

- **matrix** (*ArrayLike*) – The matrix.
- **data** (*ArrayLike*) – The data.

**Returns**

The estimated clp and residual.

**Return type**

*tuple*[*ArrayLike*, *ArrayLike*]

**estimate**

`EstimationProviderLinked.estimate()`

Calculate the estimation.

**get\_additional\_penalties**

`EstimationProviderLinked.get_additional_penalties() → list[float]`

Get the additional penalty.

**Returns**

The additional penalty.

**Return type**

*list*[*float*]

**get\_full\_penalty**

`EstimationProviderLinked.get_full_penalty() → ArrayLike`

Get the full penalty.

**Returns**

The clp penalty.

**Return type**

*ArrayLike*

## get\_result

`EstimationProviderLinked.get_result()` → `tuple[dict[str, DataArray], dict[str, DataArray]]`

Get the results of the estimation.

### Returns

A tuple of the estimated clps and residuals.

### Return type

`tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]`

## retrieve\_clps

`EstimationProviderLinked.retrieve_clps(clp_labels: list[str], reduced_clp_labels: list[str], reduced_clps: ArrayLike, index: int)`  
→ `ArrayLike`

Retrieve clp from reduced clp.

### Parameters

- **clp\_labels** (`list[str]`) – The original clp labels.
- **reduced\_clp\_labels** (`list[str]`) – The reduced clp labels.
- **reduced\_clps** (`ArrayLike`) – The reduced clps.
- **index** (`int`) – The index on the global axis.

### Returns

The retrieved clps.

### Return type

`ArrayLike`

## Methods Documentation

`calculate_clp_penalties(clp_labels: list[list[str]], clps: list[ndarray], global_axis: ndarray)` → `list[float]`

Calculate the clp penalty.

### Parameters

- **clp\_labels** (`list[list[str]]`) – The clp labels.
- **clps** (`list[ArrayLike]`) – The clps.
- **global\_axis** (`ArrayLike`) – The global axis.

### Returns

The clp penalty.

### Return type

`list[float]`

`calculate_residual(matrix: ArrayLike, data: ArrayLike)` → `tuple[ArrayLike, ArrayLike]`

Calculate the clps and the residual for a matrix and data.

### Parameters

- **matrix** (*ArrayLike*) – The matrix.
- **data** (*ArrayLike*) – The data.

**Returns**

The estimated clp and residual.

**Return type**

`tuple[ArrayLike, ArrayLike]`

**estimate()**

Calculate the estimation.

**get\_additional\_penalties()** → `list[float]`

Get the additional penalty.

**Returns**

The additional penalty.

**Return type**

`list[float]`

**get\_full\_penalty()** → `ArrayLike`

Get the full penalty.

**Returns**

The clp penalty.

**Return type**

`ArrayLike`

**get\_result()** → `tuple[dict[str, DataArray], dict[str, DataArray]]`

Get the results of the estimation.

**Returns**

A tuple of the estimated clps and residuals.

**Return type**

`tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]`

**property group:** *DatasetGroup*

Get the dataset group.

**Returns**

The dataset group.

**Return type**

*DatasetGroup*

**retrieve\_clps**(*clp\_labels: list[str]*, *reduced\_clp\_labels: list[str]*, *reduced\_clps: ArrayLike*, *index: int*) → `ArrayLike`

Retrieve clp from reduced clp.

**Parameters**

- **clp\_labels** (`list[str]`) – The original clp labels.
- **reduced\_clp\_labels** (`list[str]`) – The reduced clp labels.
- **reduced\_clps** (*ArrayLike*) – The reduced clps.
- **index** (`int`) – The index on the global axis.

**Returns**

The retrieved clps.

**Return type**

ArrayLike

**EstimationProviderUnlinked**

```
class glotaran.optimization.estimate_provider.EstimationProviderUnlinked(dataset_group:
                                                                    Dataset-
                                                                    Group,
                                                                    data_provider:
                                                                    Dat-
                                                                    aProvider,
                                                                    ma-
                                                                    trix_provider:
                                                                    Ma-
                                                                    trix-
                                                                    ProviderUn-
                                                                    linked)
```

Bases: *EstimationProvider*

A class to provide estimation for optimization of an unlinked dataset group.

Initialize an estimation provider for an unlinked dataset group.

**Parameters**

- **dataset\_group** (*DatasetGroup*) – The dataset group.
- **data\_provider** (*DataProvider*) – The data provider.
- **matrix\_provider** (*MatrixProviderUnlinked*) – The matrix provider.

**Attributes Summary**

<i>group</i>	Get the dataset group.
--------------	------------------------

**group**

`EstimationProviderUnlinked.group`

Get the dataset group.

**Returns**

The dataset group.

**Return type**

*DatasetGroup*

## Methods Summary

<code>calculate_clp_penalties</code>	Calculate the clp penalty.
<code>calculate_estimation</code>	Calculate the estimation for a dataset.
<code>calculate_full_model_estimation</code>	Calculate the estimation for a dataset with a full model.
<code>calculate_residual</code>	Calculate the clps and the residual for a matrix and data.
<code>estimate</code>	Calculate the estimation.
<code>get_additional_penalties</code>	Get the additional penalty.
<code>get_full_penalty</code>	Get the full penalty.
<code>get_result</code>	Get the results of the estimation.
<code>retrieve_clps</code>	Retrieve clp from reduced clp.

### `calculate_clp_penalties`

`EstimationProviderUnlinked.calculate_clp_penalties`(*clp\_labels*: *list[list[str]]*, *clps*: *list[ndarray]*, *global\_axis*: *ndarray*) → *list[float]*

Calculate the clp penalty.

#### Parameters

- **clp\_labels** (*list[list[str]]*) – The clp labels.
- **clps** (*list[ArrayLike]*) – The clps.
- **global\_axis** (*ArrayLike*) – The global axis.

#### Returns

The clp penalty.

#### Return type

*list[float]*

### `calculate_estimation`

`EstimationProviderUnlinked.calculate_estimation`(*dataset\_model*: *DatasetModel*)

Calculate the estimation for a dataset.

#### Parameters

- **dataset\_model** (*DatasetModel*) – The dataset model.

### calculate\_full\_model\_estimation

`EstimationProviderUnlinked.calculate_full_model_estimation(dataset_model: DatasetModel)`

Calculate the estimation for a dataset with a full model.

#### Parameters

**dataset\_model** (`DatasetModel`) – The dataset model.

### calculate\_residual

`EstimationProviderUnlinked.calculate_residual(matrix: ArrayLike, data: ArrayLike) → tuple[ArrayLike, ArrayLike]`

Calculate the clps and the residual for a matrix and data.

#### Parameters

- **matrix** (`ArrayLike`) – The matrix.
- **data** (`ArrayLike`) – The data.

#### Returns

The estimated clp and residual.

#### Return type

`tuple[ArrayLike, ArrayLike]`

### estimate

`EstimationProviderUnlinked.estimate()`

Calculate the estimation.

### get\_additional\_penalties

`EstimationProviderUnlinked.get_additional_penalties() → list[float]`

Get the additional penalty.

#### Returns

The additional penalty.

#### Return type

`list[float]`

### get\_full\_penalty

`EstimationProviderUnlinked.get_full_penalty() → ArrayLike`

Get the full penalty.

#### Returns

The clp penalty.

#### Return type

`ArrayLike`

### get\_result

`EstimationProviderUnlinked.get_result()` → `tuple[dict[str, list[DataArray]], dict[str, list[DataArray]]]`

Get the results of the estimation.

#### Returns

A tuple of the estimated clps and residuals.

#### Return type

`tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]`

### retrieve\_clps

`EstimationProviderUnlinked.retrieve_clps(clp_labels: list[str], reduced_clp_labels: list[str], reduced_clps: ArrayLike, index: int) → ArrayLike`

Retrieve clp from reduced clp.

#### Parameters

- **clp\_labels** (`list[str]`) – The original clp labels.
- **reduced\_clp\_labels** (`list[str]`) – The reduced clp labels.
- **reduced\_clps** (`ArrayLike`) – The reduced clps.
- **index** (`int`) – The index on the global axis.

#### Returns

The retrieved clps.

#### Return type

`ArrayLike`

## Methods Documentation

`calculate_clp_penalties(clp_labels: list[list[str]], clps: list[ndarray], global_axis: ndarray) → list[float]`

Calculate the clp penalty.

#### Parameters

- **clp\_labels** (`list[list[str]]`) – The clp labels.
- **clps** (`list[ArrayLike]`) – The clps.
- **global\_axis** (`ArrayLike`) – The global axis.

#### Returns

The clp penalty.

#### Return type

`list[float]`

`calculate_estimation(dataset_model: DatasetModel)`

Calculate the estimation for a dataset.



**Parameters****dataset\_model** ([DatasetModel](#)) – The dataset model.**calculate\_full\_model\_estimation**(*dataset\_model*: [DatasetModel](#))

Calculate the estimation for a dataset with a full model.

**Parameters****dataset\_model** ([DatasetModel](#)) – The dataset model.**calculate\_residual**(*matrix*: *ArrayLike*, *data*: *ArrayLike*) → [tuple](#)[*ArrayLike*, *ArrayLike*]

Calculate the clps and the residual for a matrix and data.

**Parameters**

- **matrix** (*ArrayLike*) – The matrix.
- **data** (*ArrayLike*) – The data.

**Returns**

The estimated clp and residual.

**Return type**[tuple](#)[*ArrayLike*, *ArrayLike*]**estimate**()

Calculate the estimation.

**get\_additional\_penalties**() → [list](#)[*float*]

Get the additional penalty.

**Returns**

The additional penalty.

**Return type**[list](#)[*float*]**get\_full\_penalty**() → *ArrayLike*

Get the full penalty.

**Returns**

The clp penalty.

**Return type***ArrayLike***get\_result**() → [tuple](#)[[dict](#)[*str*, [list](#)[*DataArray*]], [dict](#)[*str*, [list](#)[*DataArray*]]]

Get the results of the estimation.

**Returns**

A tuple of the estimated clps and residuals.

**Return type**[tuple](#)[[dict](#)[*str*, *xr.DataArray*], [dict](#)[*str*, *xr.DataArray*]]**property group**: [DatasetGroup](#)

Get the dataset group.

**Returns**

The dataset group.

**Return type**[DatasetGroup](#)

**retrieve\_clps**(*clp\_labels*: *list[str]*, *reduced\_clp\_labels*: *list[str]*, *reduced\_clps*: *ArrayLike*, *index*: *int*) → *ArrayLike*

Retrieve clp from reduced clp.

**Parameters**

- **clp\_labels** (*list[str]*) – The original clp labels.
- **reduced\_clp\_labels** (*list[str]*) – The reduced clp labels.
- **reduced\_clps** (*ArrayLike*) – The reduced clps.
- **index** (*int*) – The index on the global axis.

**Returns**

The retrieved clps.

**Return type**

*ArrayLike*

## Exceptions

### Exception Summary

<code>UnsupportedResidualFunctionError</code>	Indicates that the residual function is unsupported.
---	--

### UnsupportedResidualFunctionError

**exception** `glotaran.optimization.estation_provider.UnsupportedResidualFunctionError`(*residual\_function* *str*)

Indicates that the residual function is unsupported.

Initialize an `UnsupportedMethodError`.

**Parameters**

**residual\_function** (*str*) – The unsupported *residual\_function*.

## matrix\_provider

Module containing the matrix provider classes.

## Classes

## Summary

<i>MatrixContainer</i>	A container of matrix and the corresponding clp labels.
<i>MatrixProvider</i>	A class to provide matrix calculations for optimization.
<i>MatrixProviderLinked</i>	A class to provide matrix calculations for optimization of linked dataset groups.
<i>MatrixProviderUnlinked</i>	A class to provide matrix calculations for optimization of unlinked dataset groups.

## MatrixContainer

**class** `glotaran.optimization.matrix_provider.MatrixContainer`(*clp\_labels*: *list[str]*,  
*matrix*: *ndarray*)

Bases: `object`

A container of matrix and the corresponding clp labels.

## Attributes Summary

<i>is_index_dependent</i>	Check if the matrix is index dependent.
<i>clp_labels</i>	The clp labels.
<i>matrix</i>	The matrix.

## is\_index\_dependent

`MatrixContainer.is_index_dependent`

Check if the matrix is index dependent.

### Returns

Whether the matrix is index dependent.

### Return type

`bool`

## clp\_labels

`MatrixContainer.clp_labels`: `list[str]`

The clp labels.

## matrix

`MatrixContainer.matrix`: `ndarray`

The matrix.

## Methods Summary

<code>apply_weight</code>	Apply weight on a matrix.
<code>create_scaled_matrix</code>	Create a matrix container with a scaled matrix.
<code>create_weighted_matrix</code>	Create a matrix container with a weighted matrix.

## apply\_weight

**static** `MatrixContainer.apply_weight`(*matrix*: *ArrayLike*, *weight*: *ArrayLike*) → *ArrayLike*

Apply weight on a matrix.

### Parameters

- **matrix** (*ArrayLike*) – The matrix.
- **weight** (*ArrayLike*) – The weight.

### Returns

The weighted matrix.

### Return type

*ArrayLike*

## create\_scaled\_matrix

`MatrixContainer.create_scaled_matrix`(*scale*: *float*) → *MatrixContainer*

Create a matrix container with a scaled matrix.

### Parameters

- **scale** (*float*) – The scale.

### Returns

The scaled matrix.

### Return type

*MatrixContainer*

## create\_weighted\_matrix

`MatrixContainer.create_weighted_matrix(weight: ArrayLike) → MatrixContainer`

Create a matrix container with a weighted matrix.

### Parameters

**weight** (*ArrayLike*) – The weight.

### Returns

The weighted matrix.

### Return type

*MatrixContainer*

## Methods Documentation

`static apply_weight(matrix: ArrayLike, weight: ArrayLike) → ArrayLike`

Apply weight on a matrix.

### Parameters

- **matrix** (*ArrayLike*) – The matrix.
- **weight** (*ArrayLike*) – The weight.

### Returns

The weighted matrix.

### Return type

*ArrayLike*

`clp_labels: list[str]`

The clp labels.

`create_scaled_matrix(scale: float) → MatrixContainer`

Create a matrix container with a scaled matrix.

### Parameters

**scale** (*float*) – The scale.

### Returns

The scaled matrix.

### Return type

*MatrixContainer*

`create_weighted_matrix(weight: ArrayLike) → MatrixContainer`

Create a matrix container with a weighted matrix.

### Parameters

**weight** (*ArrayLike*) – The weight.

### Returns

The weighted matrix.

### Return type

*MatrixContainer*

**property** `is_index_dependent`: `bool`

Check if the matrix is index dependent.

**Returns**

Whether the matrix is index dependent.

**Return type**

`bool`

**matrix**: `ndarray`

The matrix.

## MatrixProvider

**class** `glotaran.optimization.matrix_provider.MatrixProvider`(*dataset\_group*:  
`DatasetGroup`)

Bases: `object`

A class to provide matrix calculations for optimization.

Initialize a matrix provider for a dataset group.

**Parameters**

**dataset\_group** (`DatasetGroup`) – The dataset group.

## Attributes Summary

<code>group</code>	Get the dataset group.
<code>number_of_clps</code>	Return number of conditionally linear parameters.

## group

`MatrixProvider.group`

Get the dataset group.

**Returns**

The dataset group.

**Return type**

`DatasetGroup`

## number\_of\_clps

`MatrixProvider.number_of_clps`

Return number of conditionally linear parameters.

**Raises**

`NotImplementedError` – This property needs to be implemented by subclasses.

**See also:**

`MatrixProviderUnlinked`, `MatrixProviderLinked`

## Methods Summary

<code>apply_constraints</code>	Apply constraints on a matrix.
<code>apply_relations</code>	Apply relations on a matrix.
<code>calculate</code>	Calculate the matrices for optimization.
<code>calculate_dataset_matrices</code>	Calculate the matrices of the datasets in the dataset group.
<code>calculate_dataset_matrix</code>	Calculate the matrix for a dataset on an index on the global axis.
<code>combine_megacomplex_matrices</code>	Calculate the matrix for a dataset on an index on the global axis.
<code>does_interval_item_apply</code>	Check if an interval item applies on an index.
<code>get_matrix_container</code>	Get the matrix container for a dataset on an index on the global axis.
<code>get_result</code>	Get the results of the matrix calculations.
<code>reduce_matrix</code>	Reduce a matrix.

### apply\_constraints

`MatrixProvider.apply_constraints(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply constraints on a matrix.

#### Parameters

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

#### Returns

The resulting matrix container.

#### Return type

`MatrixContainer`

### apply\_relations

`MatrixProvider.apply_relations(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply relations on a matrix.

#### Parameters

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

#### Returns

The resulting matrix container.

#### Return type

`MatrixContainer`

## calculate

`MatrixProvider.calculate()`

Calculate the matrices for optimization.

## calculate\_dataset\_matrices

`MatrixProvider.calculate_dataset_matrices()`

Calculate the matrices of the datasets in the dataset group.

## calculate\_dataset\_matrix

**static** `MatrixProvider.calculate_dataset_matrix(dataset_model: DatasetModel, global_axis: ArrayLike, model_axis: ArrayLike, global_matrix: bool = False) → MatrixContainer`

Calculate the matrix for a dataset on an index on the global axis.

### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **global\_matrix** (`bool`) – Calculate the global megacomplexes if `True`.

### Returns

The resulting matrix container.

### Return type

*MatrixContainer*

## combine\_megacomplex\_matrices

**static** `MatrixProvider.combine_megacomplex_matrices(matrix_left: ArrayLike, matrix_right: ArrayLike, clp_labels_left: list[str], clp_labels_right: list[str]) → tuple[list[str], ArrayLike]`

Calculate the matrix for a dataset on an index on the global axis.

### Parameters

- **matrix\_left** (`ArrayLike`) – The left matrix.
- **matrix\_right** (`ArrayLike`) – The right matrix.
- **clp\_labels\_left** (`list[str]`) – The left clp labels.
- **clp\_labels\_right** (`list[str]`) – The right clp labels.

### Returns

The combined clp labels and matrix.



**Return type**`tuple[list[str], ArrayLike]`**does\_interval\_item\_apply**

**static** `MatrixProvider.does_interval_item_apply(prop: IntervalItem, index: int | None) → bool`

Check if an interval item applies on an index.

**Parameters**

- **prop** ([IntervalItem](#)) – The interval property.
- **index** (`int` | `None`) – The index to check.

**Returns**

Whether the property applies.

**Return type**`bool`**get\_matrix\_container**

`MatrixProvider.get_matrix_container(dataset_label: str) → MatrixContainer`

Get the matrix container for a dataset on an index on the global axis.

**Parameters**

**dataset\_label** (`str`) – The label of the dataset.

**Returns**

The matrix container.

**Return type**[MatrixContainer](#)**get\_result**

`MatrixProvider.get_result() → tuple[dict[str, DataArray], dict[str, DataArray]]`

Get the results of the matrix calculations.

**Returns**

- `tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]` – A tuple of the matrices and global matrices.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

## reduce\_matrix

`MatrixProvider.reduce_matrix(matrix: MatrixContainer, global_axis: ArrayLike) → list[MatrixContainer]`

Reduce a matrix.

Applies constraints and relations.

### Parameters

- **matrix** (`MatrixContainer`) – The matrix.
- **global\_axis** (`ArrayLike`) – The global axis.

### Returns

The resulting matrix container.

### Return type

*MatrixContainer*

## Methods Documentation

`apply_constraints(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply constraints on a matrix.

### Parameters

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

### Returns

The resulting matrix container.

### Return type

*MatrixContainer*

`apply_relations(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply relations on a matrix.

### Parameters

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

### Returns

The resulting matrix container.

### Return type

*MatrixContainer*

`calculate()`

Calculate the matrices for optimization.

`calculate_dataset_matrices()`

Calculate the matrices of the datasets in the dataset group.

```
static calculate_dataset_matrix(dataset_model: DatasetModel, global_axis: ArrayLike,
                                model_axis: ArrayLike, global_matrix: bool = False)
                                → MatrixContainer
```

Calculate the matrix for a dataset on an index on the global axis.

#### Parameters

- **dataset\_model** (DatasetModel) – The dataset model.
- **global\_axis** (ArrayLike) – The global axis.
- **model\_axis** (ArrayLike) – The model axis.
- **global\_matrix** (bool) – Calculate the global megacomplexes if *True*.

#### Returns

The resulting matrix container.

#### Return type

MatrixContainer

```
static combine_megacomplex_matrices(matrix_left: ArrayLike, matrix_right: ArrayLike,
                                     clp_labels_left: list[str], clp_labels_right:
                                     list[str]) → tuple[list[str], ArrayLike]
```

Calculate the matrix for a dataset on an index on the global axis.

#### Parameters

- **matrix\_left** (ArrayLike) – The left matrix.
- **matrix\_right** (ArrayLike) – The right matrix.
- **clp\_labels\_left** (list[str]) – The left clp labels.
- **clp\_labels\_right** (list[str]) – The right clp labels.

#### Returns

The combined clp labels and matrix.

#### Return type

tuple[list[str], ArrayLike]

```
static does_interval_item_apply(prop: IntervalItem, index: int | None) → bool
```

Check if an interval item applies on an index.

#### Parameters

- **prop** (IntervalItem) – The interval property.
- **index** (int | None) – The index to check.

#### Returns

Whether the property applies.

#### Return type

bool

```
get_matrix_container(dataset_label: str) → MatrixContainer
```

Get the matrix container for a dataset on an index on the global axis.

#### Parameters

- **dataset\_label** (str) – The label of the dataset.

#### Returns

The matrix container.

**Return type***MatrixContainer***get\_result()** → tuple[dict[str, DataArray], dict[str, DataArray]]

Get the results of the matrix calculations.

**Returns**

- tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]] – A tuple of the matrices and global matrices.
- .. # noqa (DAR202)
- .. # noqa (DAR401)

**property group:** *DatasetGroup*

Get the dataset group.

**Returns**

The dataset group.

**Return type***DatasetGroup***property number\_of\_clps:** int

Return number of conditionally linear parameters.

**Raises****NotImplementedError** – This property needs to be implemented by subclasses.**See also:***MatrixProviderUnlinked, MatrixProviderLinked***reduce\_matrix**(matrix: *MatrixContainer*, global\_axis: *ArrayLike*) → list[*MatrixContainer*]

Reduce a matrix.

Applies constraints and relations.

**Parameters**

- **matrix** (*MatrixContainer*) – The matrix.
- **global\_axis** (*ArrayLike*) – The global axis.

**Returns**

The resulting matrix container.

**Return type***MatrixContainer***MatrixProviderLinked**

```
class glotaran.optimization.matrix_provider.MatrixProviderLinked(group:
                                                                    DatasetGroup,
                                                                    data_provider:
                                                                    DataProvider-
                                                                    Linked)
```

Bases: *MatrixProvider*

A class to provide matrix calculations for optimization of linked dataset groups.

Initialize a matrix provider for a linked dataset group.

#### Parameters

- **dataset\_group** ([DatasetGroup](#)) – The dataset group.
- **data\_provider** ([DataProviderLinked](#)) – The data provider.

#### Attributes Summary

<a href="#"><i>aligned_full_clp_labels</i></a>	Get the aligned full clp labels.
<a href="#"><i>group</i></a>	Get the dataset group.
<a href="#"><i>number_of_clps</i></a>	Return number of conditionally linear parameters.

#### [\*\*aligned\\_full\\_clp\\_labels\*\*](#)

`MatrixProviderLinked.aligned_full_clp_labels`

Get the aligned full clp labels.

##### Returns

The full aligned clp labels.

##### Return type

`list[list[str]]`

#### [\*\*group\*\*](#)

`MatrixProviderLinked.group`

Get the dataset group.

##### Returns

The dataset group.

##### Return type

[\*DatasetGroup\*](#)

#### [\*\*number\\_of\\_clps\*\*](#)

`MatrixProviderLinked.number_of_clps`

Return number of conditionally linear parameters.

##### Return type

`int`

## Methods Summary

<code>align_full_clp_labels</code>	Align the unreduced clp labels.
<code>align_matrices</code>	Align matrices.
<code>apply_constraints</code>	Apply constraints on a matrix.
<code>apply_relations</code>	Apply relations on a matrix.
<code>calculate</code>	Calculate the matrices for optimization.
<code>calculate_aligned_matrices</code>	Calculate the aligned matrices of the dataset group.
<code>calculate_dataset_matrices</code>	Calculate the matrices of the datasets in the dataset group.
<code>calculate_dataset_matrix</code>	Calculate the matrix for a dataset on an index on the global axis.
<code>combine_megacomplex_matrices</code>	Calculate the matrix for a dataset on an index on the global axis.
<code>does_interval_item_apply</code>	Check if an interval item applies on an index.
<code>get_aligned_matrix_container</code>	Get the aligned matrix container for an index on the aligned global axis.
<code>get_matrix_container</code>	Get the matrix container for a dataset on an index on the global axis.
<code>get_result</code>	Get the results of the matrix calculations.
<code>reduce_matrix</code>	Reduce a matrix.

### align\_full\_clp\_labels

`MatrixProviderLinked.align_full_clp_labels()` → `dict[str, list[str]]`

Align the unreduced clp labels.

**Returns**

The aligned clp for every group.

**Return type**

`dict[str, list[str]]`

### align\_matrices

**static** `MatrixProviderLinked.align_matrices(matrices: list[MatrixContainer], scales: list[float])` → `MatrixContainer`

Align matrices.

**Parameters**

- **matrices** (`list[MatrixContainer]`) – The matrices to align.
- **scales** (`list[float]`) – The scales of the matrices.

**Returns**

The aligned matrix container.

**Return type**

`MatrixContainer`

## apply\_constraints

`MatrixProviderLinked.apply_constraints(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply constraints on a matrix.

### Parameters

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

### Returns

The resulting matrix container.

### Return type

*MatrixContainer*

## apply\_relations

`MatrixProviderLinked.apply_relations(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply relations on a matrix.

### Parameters

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

### Returns

The resulting matrix container.

### Return type

*MatrixContainer*

## calculate

`MatrixProviderLinked.calculate()`

Calculate the matrices for optimization.

## calculate\_aligned\_matrices

`MatrixProviderLinked.calculate_aligned_matrices()`

Calculate the aligned matrices of the dataset group.

### calculate\_dataset\_matrices

`MatrixProviderLinked.calculate_dataset_matrices()`

Calculate the matrices of the datasets in the dataset group.

### calculate\_dataset\_matrix

`static MatrixProviderLinked.calculate_dataset_matrix(dataset_model:  
DatasetModel, global_axis:  
ArrayLike, model_axis:  
ArrayLike, global_matrix:  
bool = False) →  
MatrixContainer`

Calculate the matrix for a dataset on an index on the global axis.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **global\_matrix** (`bool`) – Calculate the global megacomplexes if `True`.

#### Returns

The resulting matrix container.

#### Return type

`MatrixContainer`

### combine\_megacomplex\_matrices

`static MatrixProviderLinked.combine_megacomplex_matrices(matrix_left: ArrayLike,  
matrix_right:  
ArrayLike,  
clp_labels_left:  
list[str],  
clp_labels_right:  
list[str]) →  
tuple[list[str],  
ArrayLike]`

Calculate the matrix for a dataset on an index on the global axis.

#### Parameters

- **matrix\_left** (`ArrayLike`) – The left matrix.
- **matrix\_right** (`ArrayLike`) – The right matrix.
- **clp\_labels\_left** (`list[str]`) – The left clp labels.
- **clp\_labels\_right** (`list[str]`) – The right clp labels.

#### Returns

The combined clp labels and matrix.



**Return type**`tuple[list[str], ArrayLike]`**does\_interval\_item\_apply**

**static** `MatrixProviderLinked.does_interval_item_apply(prop: IntervalItem, index: int | None) → bool`

Check if an interval item applies on an index.

**Parameters**

- **prop** (`IntervalItem`) – The interval property.
- **index** (`int` | `None`) – The index to check.

**Returns**

Whether the property applies.

**Return type**`bool`**get\_aligned\_matrix\_container**

`MatrixProviderLinked.get_aligned_matrix_container(global_index: int) → MatrixContainer`

Get the aligned matrix container for an index on the aligned global axis.

**Parameters**

**global\_index** (`int`) – The index on the global axis.

**Returns**

The matrix container.

**Return type***MatrixContainer***get\_matrix\_container**

`MatrixProviderLinked.get_matrix_container(dataset_label: str) → MatrixContainer`

Get the matrix container for a dataset on an index on the global axis.

**Parameters**

**dataset\_label** (`str`) – The label of the dataset.

**Returns**

The matrix container.

**Return type***MatrixContainer*

### get\_result

`MatrixProviderLinked.get_result()` → `tuple[dict[str, DataArray], dict[str, DataArray]]`

Get the results of the matrix calculations.

#### Returns

- `tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]` – A tuple of the matrices and global matrices.
- `.. # noqa (DAR202)`
- `.. # noqa (DAR401)`

### reduce\_matrix

`MatrixProviderLinked.reduce_matrix(matrix: MatrixContainer, global_axis: ArrayLike)`  
→ `list[MatrixContainer]`

Reduce a matrix.

Applies constraints and relations.

#### Parameters

- **matrix** (`MatrixContainer`) – The matrix.
- **global\_axis** (`ArrayLike`) – The global axis.

#### Returns

The resulting matrix container.

#### Return type

`MatrixContainer`

## Methods Documentation

`align_full_clp_labels()` → `dict[str, list[str]]`

Align the unreduced clp labels.

#### Returns

The aligned clp for every group.

#### Return type

`dict[str, list[str]]`

**static** `align_matrices(matrices: list[MatrixContainer], scales: list[float])` →  
`MatrixContainer`

Align matrices.

#### Parameters

- **matrices** (`list[MatrixContainer]`) – The matrices to align.
- **scales** (`list[float]`) – The scales of the matrices.

#### Returns

The aligned matrix container.

#### Return type

`MatrixContainer`

**property aligned\_full\_clp\_labels:** `list[list[str]]`

Get the aligned full clp labels.

**Returns**

The full aligned clp labels.

**Return type**

`list[list[str]]`

**apply\_constraints**(*matrices:* `list[MatrixContainer]`, *global\_axis:* `ArrayLike`) → `list[MatrixContainer]`

Apply constraints on a matrix.

**Parameters**

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

**Returns**

The resulting matrix container.

**Return type**

`MatrixContainer`

**apply\_relations**(*matrices:* `list[MatrixContainer]`, *global\_axis:* `ArrayLike`) → `list[MatrixContainer]`

Apply relations on a matrix.

**Parameters**

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

**Returns**

The resulting matrix container.

**Return type**

`MatrixContainer`

**calculate()**

Calculate the matrices for optimization.

**calculate\_aligned\_matrices()**

Calculate the aligned matrices of the dataset group.

**calculate\_dataset\_matrices()**

Calculate the matrices of the datasets in the dataset group.

**static calculate\_dataset\_matrix**(*dataset\_model:* `DatasetModel`, *global\_axis:* `ArrayLike`, *model\_axis:* `ArrayLike`, *global\_matrix:* `bool = False`) → `MatrixContainer`

Calculate the matrix for a dataset on an index on the global axis.

**Parameters**

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **global\_matrix** (`bool`) – Calculate the global megacomplexes if `True`.

**Returns**

The resulting matrix container.

**Return type**

*MatrixContainer*

```
static combine_megacomplex_matrices(matrix_left: ArrayLike, matrix_right: ArrayLike,  
                                     clp_labels_left: list[str], clp_labels_right:  
                                     list[str]) → tuple[list[str], ArrayLike]
```

Calculate the matrix for a dataset on an index on the global axis.

**Parameters**

- **matrix\_left** (*ArrayLike*) – The left matrix.
- **matrix\_right** (*ArrayLike*) – The right matrix.
- **clp\_labels\_left** (*list[str]*) – The left clp labels.
- **clp\_labels\_right** (*list[str]*) – The right clp labels.

**Returns**

The combined clp labels and matrix.

**Return type**

*tuple[list[str], ArrayLike]*

```
static does_interval_item_apply(prop: IntervalItem, index: int | None) → bool
```

Check if an interval item applies on an index.

**Parameters**

- **prop** (*IntervalItem*) – The interval property.
- **index** (*int* | *None*) – The index to check.

**Returns**

Whether the property applies.

**Return type**

*bool*

```
get_aligned_matrix_container(global_index: int) → MatrixContainer
```

Get the aligned matrix container for an index on the aligned global axis.

**Parameters**

**global\_index** (*int*) – The index on the global axis.

**Returns**

The matrix container.

**Return type**

*MatrixContainer*

```
get_matrix_container(dataset_label: str) → MatrixContainer
```

Get the matrix container for a dataset on an index on the global axis.

**Parameters**

**dataset\_label** (*str*) – The label of the dataset.

**Returns**

The matrix container.

**Return type***MatrixContainer***get\_result()** → tuple[dict[str, DataArray], dict[str, DataArray]]

Get the results of the matrix calculations.

**Returns**

- *tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]* – A tuple of the matrices and global matrices.
- *.. # noqa (DAR202)*
- *.. # noqa (DAR401)*

**property group:** *DatasetGroup*

Get the dataset group.

**Returns**

The dataset group.

**Return type***DatasetGroup***property number\_of\_clps:** *int*

Return number of conditionally linear parameters.

**Return type***int***reduce\_matrix**(*matrix: MatrixContainer, global\_axis: ArrayLike*) → list[*MatrixContainer*]

Reduce a matrix.

Applies constraints and relations.

**Parameters**

- **matrix** (*MatrixContainer*) – The matrix.
- **global\_axis** (*ArrayLike*) – The global axis.

**Returns**

The resulting matrix container.

**Return type***MatrixContainer***MatrixProviderUnlinked**

```
class glotaran.optimization.matrix_provider.MatrixProviderUnlinked(group:
                                                                    DatasetGroup,
                                                                    data_provider:
                                                                    DataProvider)
```

Bases: *MatrixProvider*

A class to provide matrix calculations for optimization of unlinked dataset groups.

Initialize a matrix provider for an unlinked dataset group.

**Parameters**

- **dataset\_group** (*DatasetGroup*) – The dataset group.

- **data\_provider** ([DataProvider](#)) – The data provider.

### Attributes Summary

<i>group</i>	Get the dataset group.
<i>number_of_clps</i>	Return number of conditionally linear parameters.

### group

`MatrixProviderUnlinked.group`

Get the dataset group.

**Returns**

The dataset group.

**Return type**

*DatasetGroup*

### number\_of\_clps

`MatrixProviderUnlinked.number_of_clps`

Return number of conditionally linear parameters.

**Return type**

`int`

## Methods Summary

<code>apply_constraints</code>	Apply constraints on a matrix.
<code>apply_relations</code>	Apply relations on a matrix.
<code>calculate</code>	Calculate the matrices for optimization.
<code>calculate_dataset_matrices</code>	Calculate the matrices of the datasets in the dataset group.
<code>calculate_dataset_matrix</code>	Calculate the matrix for a dataset on an index on the global axis.
<code>calculate_full_matrices</code>	Calculate the full matrices of the datasets in the dataset group.
<code>calculate_global_matrices</code>	Calculate the global matrices of the datasets in the dataset group.
<code>calculate_prepared_matrices</code>	Calculate the prepared matrices of the datasets in the dataset group.
<code>combine_megacomplex_matrices</code>	Calculate the matrix for a dataset on an index on the global axis.
<code>does_interval_item_apply</code>	Check if an interval item applies on an index.
<code>get_full_matrix</code>	Get the full matrix of a dataset.
<code>get_global_matrix_container</code>	Get the global matrix container for a dataset.
<code>get_matrix_container</code>	Get the matrix container for a dataset on an index on the global axis.
<code>get_prepared_matrix_container</code>	Get the prepared matrix container for a dataset on an index on the global axis.
<code>get_result</code>	Get the results of the matrix calculations.
<code>reduce_matrix</code>	Reduce a matrix.

## apply\_constraints

`MatrixProviderUnlinked.apply_constraints(matrices: list[MatrixContainer],  
global_axis: ArrayLike) →  
list[MatrixContainer]`

Apply constraints on a matrix.

### Parameters

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

### Returns

The resulting matrix container.

### Return type

`MatrixContainer`

### apply\_relations

`MatrixProviderUnlinked.apply_relations(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply relations on a matrix.

#### Parameters

- **matrices** (`list[MatrixContainer]`,) – The matrices.
- **global\_axis** (`ArrayLike`) – The global axis.

#### Returns

The resulting matrix container.

#### Return type

*MatrixContainer*

### calculate

`MatrixProviderUnlinked.calculate()`

Calculate the matrices for optimization.

### calculate\_dataset\_matrices

`MatrixProviderUnlinked.calculate_dataset_matrices()`

Calculate the matrices of the datasets in the dataset group.

### calculate\_dataset\_matrix

`static MatrixProviderUnlinked.calculate_dataset_matrix(dataset_model: DatasetModel, global_axis: ArrayLike, model_axis: ArrayLike, global_matrix: bool = False) → MatrixContainer`

Calculate the matrix for a dataset on an index on the global axis.

#### Parameters

- **dataset\_model** (`DatasetModel`) – The dataset model.
- **global\_axis** (`ArrayLike`) – The global axis.
- **model\_axis** (`ArrayLike`) – The model axis.
- **global\_matrix** (`bool`) – Calculate the global megacomplexes if `True`.

#### Returns

The resulting matrix container.

#### Return type

*MatrixContainer*



### calculate\_full\_matrices

`MatrixProviderUnlinked.calculate_full_matrices()`

Calculate the full matrices of the datasets in the dataset group.

### calculate\_global\_matrices

`MatrixProviderUnlinked.calculate_global_matrices()`

Calculate the global matrices of the datasets in the dataset group.

### calculate\_prepared\_matrices

`MatrixProviderUnlinked.calculate_prepared_matrices()`

Calculate the prepared matrices of the datasets in the dataset group.

### combine\_megacomplex\_matrices

`static MatrixProviderUnlinked.combine_megacomplex_matrices(matrix_left:  
ArrayLike,  
matrix_right:  
ArrayLike,  
clp_labels_left:  
list[str],  
clp_labels_right:  
list[str]) →  
tuple[list[str],  
ArrayLike]`

Calculate the matrix for a dataset on an index on the global axis.

#### Parameters

- **matrix\_left** (*ArrayLike*) – The left matrix.
- **matrix\_right** (*ArrayLike*) – The right matrix.
- **clp\_labels\_left** (*list[str]*) – The left clp labels.
- **clp\_labels\_right** (*list[str]*) – The right clp labels.

#### Returns

The combined clp labels and matrix.

#### Return type

*tuple[list[str], ArrayLike]*

### does\_interval\_item\_apply

**static** `MatrixProviderUnlinked.does_interval_item_apply(prop: IntervalItem, index: int | None) → bool`

Check if an interval item applies on an index.

#### Parameters

- **prop** (*IntervalItem*) – The interval property.
- **index** (*int* | *None*) – The index to check.

#### Returns

Whether the property applies.

#### Return type

*bool*

### get\_full\_matrix

`MatrixProviderUnlinked.get_full_matrix(dataset_label: str) → ArrayLike`

Get the full matrix of a dataset.

#### Parameters

**dataset\_label** (*str*) – The label of the dataset.

#### Returns

The matrix.

#### Return type

*ArrayLike*

### get\_global\_matrix\_container

`MatrixProviderUnlinked.get_global_matrix_container(dataset_label: str) → MatrixContainer`

Get the global matrix container for a dataset.

#### Parameters

**dataset\_label** (*str*) – The label of the dataset.

#### Returns

The matrix container.

#### Return type

*MatrixContainer*

### get\_matrix\_container

`MatrixProviderUnlinked.get_matrix_container(dataset_label: str) → MatrixContainer`

Get the matrix container for a dataset on an index on the global axis.

**Parameters**

**dataset\_label** (*str*) – The label of the dataset.

**Returns**

The matrix container.

**Return type**

*MatrixContainer*

### get\_prepared\_matrix\_container

`MatrixProviderUnlinked.get_prepared_matrix_container(dataset_label: str,  
global_index: int) → MatrixContainer`

Get the prepared matrix container for a dataset on an index on the global axis.

**Parameters**

- **dataset\_label** (*str*) – The label of the dataset.
- **global\_index** (*int*) – The index on the global axis.

**Returns**

The matrix container.

**Return type**

*MatrixContainer*

### get\_result

`MatrixProviderUnlinked.get_result() → tuple[dict[str, DataArray], dict[str, DataArray]]`

Get the results of the matrix calculations.

**Returns**

- *tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]* – A tuple of the matrices and global matrices.
- **.. # noqa** (*DAR202*)
- **.. # noqa** (*DAR401*)

## reduce\_matrix

`MatrixProviderUnlinked.reduce_matrix(matrix: MatrixContainer, global_axis: ArrayLike) → list[MatrixContainer]`

Reduce a matrix.

Applies constraints and relations.

### Parameters

- **matrix** ([MatrixContainer](#)) – The matrix.
- **global\_axis** ([ArrayLike](#)) – The global axis.

### Returns

The resulting matrix container.

### Return type

[MatrixContainer](#)

## Methods Documentation

`apply_constraints(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply constraints on a matrix.

### Parameters

- **matrices** (list[[MatrixContainer](#)],) – The matrices.
- **global\_axis** ([ArrayLike](#)) – The global axis.

### Returns

The resulting matrix container.

### Return type

[MatrixContainer](#)

`apply_relations(matrices: list[MatrixContainer], global_axis: ArrayLike) → list[MatrixContainer]`

Apply relations on a matrix.

### Parameters

- **matrices** (list[[MatrixContainer](#)],) – The matrices.
- **global\_axis** ([ArrayLike](#)) – The global axis.

### Returns

The resulting matrix container.

### Return type

[MatrixContainer](#)

`calculate()`

Calculate the matrices for optimization.

`calculate_dataset_matrices()`

Calculate the matrices of the datasets in the dataset group.

```
static calculate_dataset_matrix(dataset_model: DatasetModel, global_axis: ArrayLike,
                                model_axis: ArrayLike, global_matrix: bool = False)
                                → MatrixContainer
```

Calculate the matrix for a dataset on an index on the global axis.

#### Parameters

- **dataset\_model** (DatasetModel) – The dataset model.
- **global\_axis** (ArrayLike) – The global axis.
- **model\_axis** (ArrayLike) – The model axis.
- **global\_matrix** (bool) – Calculate the global megacomplexes if *True*.

#### Returns

The resulting matrix container.

#### Return type

MatrixContainer

```
calculate_full_matrices()
```

Calculate the full matrices of the datasets in the dataset group.

```
calculate_global_matrices()
```

Calculate the global matrices of the datasets in the dataset group.

```
calculate_prepared_matrices()
```

Calculate the prepared matrices of the datasets in the dataset group.

```
static combine_megacomplex_matrices(matrix_left: ArrayLike, matrix_right: ArrayLike,
                                     clp_labels_left: list[str], clp_labels_right:
                                     list[str]) → tuple[list[str], ArrayLike]
```

Calculate the matrix for a dataset on an index on the global axis.

#### Parameters

- **matrix\_left** (ArrayLike) – The left matrix.
- **matrix\_right** (ArrayLike) – The right matrix.
- **clp\_labels\_left** (list[str]) – The left clp labels.
- **clp\_labels\_right** (list[str]) – The right clp labels.

#### Returns

The combined clp labels and matrix.

#### Return type

tuple[list[str], ArrayLike]

```
static does_interval_item_apply(prop: IntervalItem, index: int | None) → bool
```

Check if an interval item applies on an index.

#### Parameters

- **prop** (IntervalItem) – The interval property.
- **index** (int | None) – The index to check.

#### Returns

Whether the property applies.

**Return type**`bool`**get\_full\_matrix**(*dataset\_label*: `str`) → `ArrayLike`

Get the full matrix of a dataset.

**Parameters****dataset\_label** (`str`) – The label of the dataset.**Returns**

The matrix.

**Return type**`ArrayLike`**get\_global\_matrix\_container**(*dataset\_label*: `str`) → `MatrixContainer`

Get the global matrix container for a dataset.

**Parameters****dataset\_label** (`str`) – The label of the dataset.**Returns**

The matrix container.

**Return type**`MatrixContainer`**get\_matrix\_container**(*dataset\_label*: `str`) → `MatrixContainer`

Get the matrix container for a dataset on an index on the global axis.

**Parameters****dataset\_label** (`str`) – The label of the dataset.**Returns**

The matrix container.

**Return type**`MatrixContainer`**get\_prepared\_matrix\_container**(*dataset\_label*: `str`, *global\_index*: `int`) → `MatrixContainer`

Get the prepared matrix container for a dataset on an index on the global axis.

**Parameters**

- **dataset\_label** (`str`) – The label of the dataset.
- **global\_index** (`int`) – The index on the global axis.

**Returns**

The matrix container.

**Return type**`MatrixContainer`**get\_result**() → `tuple[dict[str, DataArray], dict[str, DataArray]]`

Get the results of the matrix calculations.

**Returns**

- `tuple[dict[str, xr.DataArray], dict[str, xr.DataArray]]` – A tuple of the matrices and global matrices.
- `.. # noqa (DAR202)`

- .. # noqa (DAR401)

**property group:** *DatasetGroup*

Get the dataset group.

**Returns**

The dataset group.

**Return type**

*DatasetGroup*

**property number\_of\_clps:** *int*

Return number of conditionally linear parameters.

**Return type**

*int*

**reduce\_matrix**(*matrix*: *MatrixContainer*, *global\_axis*: *ArrayLike*) → *list*[*MatrixContainer*]

Reduce a matrix.

Applies constraints and relations.

**Parameters**

- **matrix** (*MatrixContainer*) – The matrix.
- **global\_axis** (*ArrayLike*) – The global axis.

**Returns**

The resulting matrix container.

**Return type**

*MatrixContainer*

## nnls

Module for residual calculation with the non-negative least-squares method.

## Functions

### Summary

<i>residual_nnls</i>	Calculate the conditionally linear parameters and residual with the NNLS method.
----------------------	--

### residual\_nnls

`glotaran.optimization.nnls.residual_nnls`(*matrix*: *ArrayLike*, *data*: *ArrayLike*) → *tuple*[*ArrayLike*, *ArrayLike*]

Calculate the conditionally linear parameters and residual with the NNLS method.

NNLS stands for ‘non-negative least-squares’.

**Parameters**

- **matrix** (*ArrayLike*) – The model matrix.

- **data** (*ArrayLike*) – The data to analyze.

**Returns**

The clps and the residual.

**Return type**

`tuple[ArrayLike, ArrayLike]`

## optimization\_group

Module containing the optimization group class.

## Classes

### Summary

<i>OptimizationGroup</i>	A class to optimize a dataset group.
--------------------------	--------------------------------------

## OptimizationGroup

```
class glotaran.optimization.optimization_group.OptimizationGroup(scheme: Scheme,  
                                                                  dataset_group:  
                                                                  DatasetGroup)
```

Bases: `object`

A class to optimize a dataset group.

Initialize an optimization group for a dataset group.

**Parameters**

- **scheme** (*Scheme*) – The optimization scheme.
- **dataset\_group** (*DatasetGroup*) – The dataset group.

### Attributes Summary

<i>number_of_clps</i>	Return number of conditionally linear parameters.
-----------------------	---

## number\_of\_clps

`OptimizationGroup.number_of_clps`

Return number of conditionally linear parameters.

**Return type**

`int`



## Methods Summary

<code>add_svd_data</code>	Add the SVD of a data matrix to a dataset.
<code>add_weight_to_result_data</code>	Add weight to result dataset if dataset is weighted.
<code>calculate</code>	Calculate the optimization group data.
<code>create_result_data</code>	Create resulting datasets.
<code>get_additional_penalties</code>	Get additional penalties.
<code>get_full_penalty</code>	Get the full penalty.

### add\_svd\_data

**static** OptimizationGroup.**add\_svd\_data**(*name*: *str*, *dataset*: Dataset, *lsv\_dim*: *str*, *rsv\_dim*: *str*)

Add the SVD of a data matrix to a dataset.

#### Parameters

- **name** (*str*) – Name of the data matrix.
- **dataset** (*xr.Dataset*) – Dataset containing the data, which will be updated with the SVD values.
- **lsv\_dim** (*str*) – The dimension name of the left singular vectors.
- **rsv\_dim** (*str*) – The dimension name of the right singular vectors.

### add\_weight\_to\_result\_data

OptimizationGroup.**add\_weight\_to\_result\_data**(*dataset\_label*: *str*, *result\_dataset*: Dataset)

Add weight to result dataset if dataset is weighted.

#### Parameters

- **dataset\_label** (*str*) – The label of the data.
- **result\_dataset** (*xr.Dataset*) – The label of the data.

### calculate

OptimizationGroup.**calculate**(*parameters*: Parameters)

Calculate the optimization group data.

#### Parameters

- **parameters** (Parameters) – The parameters.

### `create_result_data`

`OptimizationGroup.create_result_data()` → `dict[str, Dataset]`

Create resulting datasets.

#### Returns

The datasets with the results.

#### Return type

`dict[str, xr.Dataset]`

### `get_additional_penalties`

`OptimizationGroup.get_additional_penalties()` → `list[float]`

Get additional penalties.

#### Returns

The additional penalties.

#### Return type

`list[float]`

### `get_full_penalty`

`OptimizationGroup.get_full_penalty()` → `ArrayLike`

Get the full penalty.

#### Returns

The full penalty.

#### Return type

`ArrayLike`

## Methods Documentation

**static** `add_svd_data(name: str, dataset: Dataset, lsv_dim: str, rsv_dim: str)`

Add the SVD of a data matrix to a dataset.

#### Parameters

- **name** (`str`) – Name of the data matrix.
- **dataset** (`xr.Dataset`) – Dataset containing the data, which will be updated with the SVD values.
- **lsv\_dim** (`str`) – The dimension name of the left singular vectors.
- **rsv\_dim** (`str`) – The dimension name of the right singular vectors.

**add\_weight\_to\_result\_data(dataset\_label: str, result\_dataset: Dataset)**

Add weight to result dataset if dataset is weighted.

#### Parameters

- **dataset\_label** (`str`) – The label of the data.
- **result\_dataset** (`xr.Dataset`) – The label of the data.

**calculate**(*parameters*: [Parameters](#))

Calculate the optimization group data.

**Parameters**

**parameters** ([Parameters](#)) – The parameters.

**create\_result\_data**() → [dict](#)[[str](#), [Dataset](#)]

Create resulting datasets.

**Returns**

The datasets with the results.

**Return type**

[dict](#)[[str](#), [xr.Dataset](#)]

**get\_additional\_penalties**() → [list](#)[[float](#)]

Get additional penalties.

**Returns**

The additional penalties.

**Return type**

[list](#)[[float](#)]

**get\_full\_penalty**() → [ArrayLike](#)

Get the full penalty.

**Returns**

The full penalty.

**Return type**

[ArrayLike](#)

**property number\_of\_clps**: [int](#)

Return number of conditionally linear parameters.

**Return type**

[int](#)

## optimization\_history

Module containing the [OptimizationHistory](#) class.

## Classes

### Summary

[\*OptimizationHistory\*](#)

Wrapped [DataFrame](#) to hold information of the optimization and behaves like a [DataFrame](#).

## OptimizationHistory

```
class glotaran.optimization.optimization_history.OptimizationHistory(data=None,  
                                                                    source_path:  
                                                                    StrOrPath |  
                                                                    None =  
                                                                    None)
```

Bases: `object`

Wrapped DataFrame to hold information of the optimization and behaves like a DataFrame.

Ref.: <https://stackoverflow.com/a/65375904/3990615>

Ensure DataFrame has the correct columns, is numeric and has iteration as index.

### Attributes Summary

<code>data</code>	Underlying DataFrame which allows for auto-complete with static analyzers.
-------------------	--

### data

OptimizationHistory.**data**

Underlying DataFrame which allows for autocomplete with static analyzers.

#### Returns

DataFrame containing OptimizationHistory data.

#### Return type

pd.DataFrame

### Methods Summary

<code>from_csv</code>	Read OptimizationHistory from file.
<code>from_stdout_str</code>	Create OptimizationHistory instance from optimize_stdout.
<code>loader</code>	Read OptimizationHistory from file.
<code>to_csv</code>	Write a OptimizationHistory to a CSV file and set source_path.

**from\_csv**

**classmethod** OptimizationHistory.**from\_csv**(*path: StrOrPath*) → *OptimizationHistory*

Read OptimizationHistory from file.

**Parameters**

**path** (*StrOrPath*) – The path to the csv file.

**Returns**

OptimizationHistory read from file.

**Return type**

*OptimizationHistory*

**from\_stdout\_str**

**classmethod** OptimizationHistory.**from\_stdout\_str**(*optimize\_stdout: str*) → *OptimizationHistory*

Create OptimizationHistory instance from optimize\_stdout.

**Parameters**

**optimize\_stdout** (*str*) – SciPy optimization stdout string, read out via TeeContext.read().

**Returns**

OptimizationHistory instance created by parsing optimize\_stdout.

**Return type**

*OptimizationHistory*

**loader**

**classmethod** OptimizationHistory.**loader**(*path: StrOrPath*) → *OptimizationHistory*

Read OptimizationHistory from file.

**Parameters**

**path** (*StrOrPath*) – The path to the csv file.

**Returns**

OptimizationHistory read from file.

**Return type**

*OptimizationHistory*

**to\_csv**

OptimizationHistory.**to\_csv**(*path: StrOrPath, delimiter: str = ','*)

Write a OptimizationHistory to a CSV file and set source\_path.

**Parameters**

- **path** (*StrOrPath*) – The path to the CSV file.
- **delimiter** (*str*) – The delimiter of the CSV file.

## Methods Documentation

### property data: DataFrame

Underlying DataFrame which allows for autocomplete with static analyzers.

#### Returns

DataFrame containing OptimizationHistory data.

#### Return type

pd.DataFrame

### classmethod from\_csv(path: StrOrPath) → OptimizationHistory

Read OptimizationHistory from file.

#### Parameters

**path** (StrOrPath) – The path to the csv file.

#### Returns

OptimizationHistory read from file.

#### Return type

*OptimizationHistory*

### classmethod from\_stdout\_str(optimize\_stdout: str) → OptimizationHistory

Create OptimizationHistory instance from optimize\_stdout.

#### Parameters

**optimize\_stdout** (str) – SciPy optimization stdout string, read out via TeeContext.read().

#### Returns

OptimizationHistory instance created by parsing optimize\_stdout.

#### Return type

*OptimizationHistory*

### classmethod loader(path: StrOrPath) → OptimizationHistory

Read OptimizationHistory from file.

#### Parameters

**path** (StrOrPath) – The path to the csv file.

#### Returns

OptimizationHistory read from file.

#### Return type

*OptimizationHistory*

### to\_csv(path: StrOrPath, delimiter: str = ',')

Write a OptimizationHistory to a CSV file and set source\_path.

#### Parameters

- **path** (StrOrPath) – The path to the CSV file.
- **delimiter** (str) – The delimiter of the CSV file.

## optimize

Module containing the optimize function.

### Functions

#### Summary

<i>optimize</i>	Optimize a scheme.
-----------------	--------------------

#### optimize

glotaran.optimization.optimize.**optimize**(*scheme*: [Scheme](#), *verbose*: *bool* = *True*,  
*raise\_exception*: *bool* = *False*) → *Result*

Optimize a scheme.

#### Parameters

- **scheme** ([Scheme](#)) – The optimization scheme.
- **verbose** (*bool*) – Deactivate printing of logs if *False*.
- **raise\_exception** (*bool*) – Raise exceptions during optimizations instead of gracefully exiting if *True*.

#### Returns

The result of the optimization.

#### Return type

*Result*

## optimizer

Module containing the optimizer class.

### Classes

#### Summary

<i>Optimizer</i>	A class to optimize a scheme.
------------------	-------------------------------

## Optimizer

```
class glotaran.optimization.optimizer.Optimizer(scheme: Scheme, verbose: bool = True,
                                                raise_exception: bool = False)
```

Bases: `object`

A class to optimize a scheme.

Initialize an optimization group for a dataset group.

### Parameters

- **scheme** (`Scheme`) – The optimization scheme.
- **verbose** (`bool`) – Deactivate printing of logs if *False*.
- **raise\_exception** (`bool`) – Raise exceptions during optimizations instead of gracefully exiting if *True*.

### Raises

- **MissingDatasetsError** – Raised if datasets are missing.
- **ParameterNotInitializedError** – Raised if the scheme parameters are *None*.
- **UnsupportedMethodError** – Raised if the optimization method is unsupported.

## Methods Summary

<code>calculate_covariance_matrix_and_standard_errors</code>	Calculate the covariance matrix and standard errors of the optimization.
<code>calculate_penalty</code>	Calculate the penalty of the scheme.
<code>create_result</code>	Create the result of the optimization.
<code>get_current_optimization_iteration</code>	Extract current iteration from <code>optimize_stdout</code> .
<code>objective_function</code>	Calculate the objective for the optimization.
<code>optimize</code>	Perform the optimization.

## `calculate_covariance_matrix_and_standard_errors`

```
Optimizer.calculate_covariance_matrix_and_standard_errors(jacobian: ArrayLike,
                                                         root_mean_square_error: float) → ArrayLike
```

Calculate the covariance matrix and standard errors of the optimization.

### Parameters

- **jacobian** (`ArrayLike`) – The jacobian matrix.
- **root\_mean\_square\_error** (`float`) – The root mean square error.

### Returns

The covariance matrix.



**Return type**  
ArrayLike

### calculate\_penalty

Optimizer.calculate\_penalty() → ArrayLike

Calculate the penalty of the scheme.

**Returns**  
The penalty.

**Return type**  
ArrayLike

### create\_result

Optimizer.create\_result() → *Result*

Create the result of the optimization.

**Returns**  
The result of the optimization.

**Return type**  
*Result*

**Raises**  
**InitialParameterError** – Raised if the initial parameters could not be evaluated.

### get\_current\_optimization\_iteration

static Optimizer.get\_current\_optimization\_iteration(optimize\_stdout: *str*) → int

Extract current iteration from optimize\_stdout.

**Parameters**  
**optimize\_stdout** (*str*) – SciPy optimization stdout string, read out via TeeContext.read().

**Returns**  
Current iteration (0 if pattern did not match).

**Return type**  
int

### objective\_function

Optimizer.objective\_function(parameters: ArrayLike) → ArrayLike

Calculate the objective for the optimization.

**Parameters**  
**parameters** (ArrayLike) – the parameters provided by the optimizer.

**Returns**  
The objective for the optimizer.

**Return type**  
ArrayLike

## optimize

`Optimizer.optimize()`  
Perform the optimization.

**Raises**  
**Exception** – Raised if an exception occurs during optimization and `raise_exception` is `True`.

## Methods Documentation

**calculate\_covariance\_matrix\_and\_standard\_errors**(*jacobian*: ArrayLike,  
*root\_mean\_square\_error*: float) → ArrayLike

Calculate the covariance matrix and standard errors of the optimization.

### Parameters

- **jacobian** (ArrayLike) – The jacobian matrix.
- **root\_mean\_square\_error** (float) – The root mean square error.

### Returns

The covariance matrix.

**Return type**  
ArrayLike

**calculate\_penalty**() → ArrayLike  
Calculate the penalty of the scheme.

**Returns**  
The penalty.

**Return type**  
ArrayLike

**create\_result**() → *Result*  
Create the result of the optimization.

**Returns**  
The result of the optimization.

**Return type**  
*Result*

**Raises**  
**InitialParameterError** – Raised if the initial parameters could not be evaluated.

**static get\_current\_optimization\_iteration**(*optimize\_stdout*: str) → int  
Extract current iteration from `optimize_stdout`.

**Parameters**  
**optimize\_stdout** (str) – SciPy optimization stdout string, read out via `TeeContext.read()`.

**Returns**

Current iteration (0 if pattern did not match).

**Return type**

`int`

**objective\_function**(*parameters: ArrayLike*) → *ArrayLike*

Calculate the objective for the optimization.

**Parameters**

**parameters** (*ArrayLike*) – the parameters provided by the optimizer.

**Returns**

The objective for the optimizer.

**Return type**

*ArrayLike*

**optimize()**

Perform the optimization.

**Raises**

**Exception** – Raised if an exception occurs during optimization and `raise_exception` is *True*.

## Exceptions

### Exception Summary

<code>InitialParameterError</code>	Indicates that initial parameters can not be evaluated.
<code>MissingDatasetsError</code>	Indicates that datasets are missing in the scheme.
<code>ParameterNotInitializedError</code>	Indicates that scheme parameters are not initialized.
<code>UnsupportedMethodError</code>	Indicates that the optimization method is unsupported.

### InitialParameterError

**exception** `glotaran.optimization.optimizer.InitialParameterError`

Indicates that initial parameters can not be evaluated.

Initialize a `InitialParameterError`.

### MissingDatasetsError

**exception** `glotaran.optimization.optimizer.MissingDatasetsError`(*missing\_datasets*: *list[str]*)

Indicates that datasets are missing in the scheme.

Initialize a MissingDatasetsError.

**Parameters**

**missing\_datasets** (*list[str]*) – The missing datasets.

### ParameterNotInitializedError

**exception** `glotaran.optimization.optimizer.ParameterNotInitializedError`

Indicates that scheme parameters are not initialized.

Initialize a ParameterNotInitializedError.

### UnsupportedMethodError

**exception** `glotaran.optimization.optimizer.UnsupportedMethodError`(*method*: *str*)

Indicates that the optimization method is unsupported.

Initialize an UnsupportedMethodError.

**Parameters**

**method** (*str*) – The unsupported method.

## variable\_projection

Module for residual calculation with the variable projection method.

## Functions

### Summary

<i>residual_variable_projection</i>	Calculate conditionally linear parameters and residual with the variable projection method.
-------------------------------------	---

### residual\_variable\_projection

`glotaran.optimization.variable_projection.residual_variable_projection`(*matrix*: *ArrayLike*,  
*data*: *ArrayLike*)  
→ *tuple*[*ArrayLike*,  
*ArrayLike*]

Calculate conditionally linear parameters and residual with the variable projection method.

**Parameters**

- **matrix** (*ArrayLike*) – The model matrix.
- **data** (*ArrayLike*) – The data to analyze.

**Returns**

The clps and the residual.

**Return type**

`tuple[ArrayLike, ArrayLike]`

## 16.1.8 parameter

The glotaran parameter package.

**Modules**

<code>glotaran.parameter.parameter</code>	The parameter class.
<code>glotaran.parameter.parameter_history</code>	The glotaran parameter history package.
<code>glotaran.parameter.parameters</code>	The parameters class.

**parameter**

The parameter class.

**Functions****Summary**

<code>deserialize_options</code>	Replace options keys in serialized format by attribute names.
<code>serialize_options</code>	Replace options keys with serialized format by attribute names.
<code>set_transformed_expression</code>	Set the transformed expression from an expression.
<code>valid_label</code>	Check if a label is a valid label for <i>Parameter</i> .

**deserialize\_options**

`glotaran.parameter.parameter.deserialize_options(options: dict[str, Any]) → dict[str, Any]`

Replace options keys in serialized format by attribute names.

**Parameters**

**options** (`dict[str, Any]`) – The serialized options.

**Returns**

The deserialized options.

**Return type**  
`dict[str, Any]`

### **serialize\_options**

`glotaran.parameter.parameter.serialize_options(options: dict[str, Any]) → dict[str, Any]`

Replace options keys with serialized format by attribute names.

**Parameters**  
**options** (`dict[str, Any]`) – The options.

**Returns**  
The serialized options.

**Return type**  
`dict[str, Any]`

### **set\_transformed\_expression**

`glotaran.parameter.parameter.set_transformed_expression(parameter: Parameter,  
attribute: Attribute,  
expression: str | None)`

Set the transformed expression from an expression.

**Parameters**

- **parameter** (`Parameter`) – The `Parameter` instance
- **attribute** (`Attribute`) – The label field.
- **expression** (`str` | `None`) – The expression value.

### **valid\_label**

`glotaran.parameter.parameter.valid_label(parameter: Parameter, attribute: Attribute, label:  
str)`

Check if a label is a valid label for `Parameter`.

**Parameters**

- **parameter** (`Parameter`) – The `Parameter` instance
- **attribute** (`Attribute`) – The label field.
- **label** (`str`) – The label value.

**Raises**  
**ValueError** – Raise when the label is not valid.

Classes

Summary

<i>Parameter</i>	A parameter for optimization.
------------------	-------------------------------

Parameter

**class** glotaran.parameter.parameter.**Parameter**(*label*, *value*=nan, *standard\_error*: *float* = nan, *expression*: *str* | *None* = None, *maximum*: *float* = inf, *minimum*: *float* = -inf, *non\_negative*: *bool* = False, *vary*: *bool* = True)

Bases: `_SupportsArray`  
A parameter for optimization.  
Method generated by attrs for class Parameter.

Attributes Summary

<i>label</i>	
<i>value</i>	
<i>standard_error</i>	
<i>expression</i>	
<i>maximum</i>	
<i>minimum</i>	
<i>non_negative</i>	
<i>vary</i>	
<i>transformed_expression</i>	
<i>label_short</i>	Get short label.

### **label**

`Parameter.label: str`

### **value**

`Parameter.value: float`

### **standard\_error**

`Parameter.standard_error: float`

### **expression**

`Parameter.expression: str | None`

### **maximum**

`Parameter.maximum: float`

### **minimum**

`Parameter.minimum: float`

### **non\_negative**

`Parameter.non_negative: bool`

### **vary**

`Parameter.vary: bool`

### **transformed\_expression**

`Parameter.transformed_expression: str | None`



## label\_short

`Parameter.label_short`

Get short label.

**Returns**

The short label.

**Return type**

str

## Methods Summary

<code>as_dict</code>	Get the parameter as a dictionary.
<code>as_list</code>	Get the parameter as a dictionary.
<code>copy</code>	Create a copy of the <i>Parameter</i> .
<code>from_list</code>	Create a parameter from a list.
<code>get_value_and_bounds_for_optimization</code>	Get the parameter value and bounds with expression and non-negative constraints applied.
<code>markdown</code>	Get a markdown representation of the parameter.
<code>set_value_from_optimization</code>	Set the value from an optimization result and reverses non-negative transformation.

## as\_dict

`Parameter.as_dict()` → dict[str, Any]

Get the parameter as a dictionary.

**Returns**

The parameter as dictionary.

**Return type**

dict[str, Any]

## as\_list

`Parameter.as_list(label_short: bool = False)` → list[str | float | dict[str, Any]]

Get the parameter as a dictionary.

**Parameters**

**label\_short** (bool) – If true, the label will be replaced by the shortened label.

**Returns**

The parameter as dictionary.

**Return type**

dict[str, Any]

## copy

`Parameter.copy()` → *Parameter*

Create a copy of the *Parameter*.

### Returns

A copy of the *Parameter*.

### Return type

*Parameter*

## from\_list

**classmethod** `Parameter.from_list(values: list[Any], *, default_options: dict[str, Any] | None = None)` → *Parameter*

Create a parameter from a list.

### Parameters

- **values** (*list*[Any]) – The list of parameter definitions.
- **default\_options** (*dict*[str, Any] | None) – A dictionary of default options.

### Returns

The created *Parameter*.

### Return type

*Parameter*

## get\_value\_and\_bounds\_for\_optimization

`Parameter.get_value_and_bounds_for_optimization()` → *tuple*[float, float, float]

Get the parameter value and bounds with expression and non-negative constraints applied.

### Returns

A tuple containing the value, the lower and the upper bound.

### Return type

*tuple*[float, float, float]

## markdown

`Parameter.markdown(all_parameters: Parameters | None = None, initial_parameters: Parameters | None = None)` → *MarkdownStr*

Get a markdown representation of the parameter.

### Parameters

- **all\_parameters** (*Parameters* | None) – A parameter group containing the whole parameter set (used for expression lookup).
- **initial\_parameters** (*Parameters* | None) – The initial parameter.

### Returns

The parameter as markdown string.

**Return type***MarkdownStr***set\_value\_from\_optimization**`Parameter.set_value_from_optimization(value: float)`

Set the value from an optimization result and reverses non-negative transformation.

**Parameters**

**value** (*float*) – Value from optimization.

**Methods Documentation**`as_dict() → dict[str, Any]`

Get the parameter as a dictionary.

**Returns**

The parameter as dictionary.

**Return type***dict[str, Any]*`as_list(label_short: bool = False) → list[str | float | dict[str, Any]]`

Get the parameter as a dictionary.

**Parameters**

**label\_short** (*bool*) – If true, the label will be replaced by the shortened label.

**Returns**

The parameter as dictionary.

**Return type***dict[str, Any]*`copy() → Parameter`

Create a copy of the *Parameter*.

**Returns**

A copy of the *Parameter*.

**Return type***Parameter***expression:** *str* | *None*`classmethod from_list(values: list[Any], *, default_options: dict[str, Any] | None = None) → Parameter`

Create a parameter from a list.

**Parameters**

- **values** (*list[Any]*) – The list of parameter definitions.
- **default\_options** (*dict[str, Any] | None*) – A dictionary of default options.

**Returns**

The created *Parameter*.

**Return type***Parameter***get\_value\_and\_bounds\_for\_optimization()** → `tuple[float, float, float]`

Get the parameter value and bounds with expression and non-negative constraints applied.

**Returns**

A tuple containing the value, the lower and the upper bound.

**Return type**`tuple[float, float, float]`**label:** `str`**property label\_short:** `str`

Get short label.

**Returns**

The short label.

**Return type**`str`**markdown**(*all\_parameters: Parameters | None = None, initial\_parameters: Parameters | None = None*) → *MarkdownStr*

Get a markdown representation of the parameter.

**Parameters**

- **all\_parameters** (*Parameters* / *None*) – A parameter group containing the whole parameter set (used for expression lookup).
- **initial\_parameters** (*Parameters* / *None*) – The initial parameter.

**Returns**

The parameter as markdown string.

**Return type***MarkdownStr***maximum:** `float`**minimum:** `float`**non\_negative:** `bool`**set\_value\_from\_optimization**(*value: float*)

Set the value from an optimization result and reverses non-negative transformation.

**Parameters**

**value** (*float*) – Value from optimization.

**standard\_error:** `float`**transformed\_expression:** `str` | `None`**value:** `float`**vary:** `bool`

## parameter\_history

The glotaran parameter history package.

### Classes

#### Summary

<i>ParameterHistory</i>	A class representing a history of parameters.
-------------------------	---

#### ParameterHistory

**class** glotaran.parameter.parameter\_history.**ParameterHistory**

Bases: `object`

A class representing a history of parameters.

#### Attributes Summary

<i>number_of_records</i>	Return the number of records in the history.
<i>parameter_labels</i>	Return the labels of the parameters in the history.
<i>parameters</i>	Return the parameters in the history.

#### number\_of\_records

**ParameterHistory.number\_of\_records**

Return the number of records in the history.

##### Returns

The number of records.

##### Return type

`int`

#### parameter\_labels

**ParameterHistory.parameter\_labels**

Return the labels of the parameters in the history.

##### Returns

A list of parameter labels.

##### Return type

`list[str]`

## parameters

### ParameterHistory.parameters

Return the parameters in the history.

#### Returns

A list of parameters in the history.

#### Return type

`list[np.ndarray]`

## Methods Summary

<code>append</code>	Append Parameters to the history.
<code>from_csv</code>	Create a history from a csv file.
<code>from_dataframe</code>	Create a history from a pandas data frame.
<code>get_parameters</code>	Get parameters for a history index.
<code>loader</code>	Create a history from a csv file.
<code>to_csv</code>	Write a <i>ParameterHistory</i> to a CSV file.
<code>to_dataframe</code>	Create a data frame from the history.

## append

ParameterHistory.append(parameters: Parameters, current\_iteration: int = 0)

Append Parameters to the history.

#### Parameters

- **parameters** (Parameters) – The group to append.
- **current\_iteration** (int) – Current iteration of the optimizer.

#### Raises

**ValueError** – Raised if the parameter labels differs from previous.

## from\_csv

classmethod ParameterHistory.from\_csv(path: str) → ParameterHistory

Create a history from a csv file.

#### Parameters

**path** (str) – The path to the csv file.

#### Returns

The created history.

#### Return type

*ParameterHistory*

### from\_dataframe

**classmethod** `ParameterHistory.from_dataframe(history_df: DataFrame) → ParameterHistory`

Create a history from a pandas data frame.

**Parameters**

**history\_df** (*pd.DataFrame*) – The source data frame.

**Returns**

The created history.

**Return type**

*ParameterHistory*

### get\_parameters

`ParameterHistory.get_parameters(index: int) → ndarray`

Get parameters for a history index.

**Parameters**

**index** (*int*) – The history index.

**Returns**

The parameter values at the history index as array.

**Return type**

`np.ndarray`

### loader

**classmethod** `ParameterHistory.loader(path: str) → ParameterHistory`

Create a history from a csv file.

**Parameters**

**path** (*str*) – The path to the csv file.

**Returns**

The created history.

**Return type**

*ParameterHistory*

### to\_csv

`ParameterHistory.to_csv(file_name: str | PathLike[str], delimiter: str = ',')`

Write a *ParameterHistory* to a CSV file.

**Parameters**

- **file\_name** (*str*) – The path to the CSV file.
- **delimiter** (*str*) – The delimiter of the CSV file.

### `to_dataframe`

`ParameterHistory.to_dataframe()` → `DataFrame`

Create a data frame from the history.

#### Returns

The created data frame.

#### Return type

`pd.DataFrame`

## Methods Documentation

`append(parameters: Parameters, current_iteration: int = 0)`

Append `Parameters` to the history.

#### Parameters

- **parameters** (`Parameters`) – The group to append.
- **current\_iteration** (`int`) – Current iteration of the optimizer.

#### Raises

**ValueError** – Raised if the parameter labels differs from previous.

`classmethod from_csv(path: str) → ParameterHistory`

Create a history from a csv file.

#### Parameters

**path** (`str`) – The path to the csv file.

#### Returns

The created history.

#### Return type

`ParameterHistory`

`classmethod from_dataframe(history_df: DataFrame) → ParameterHistory`

Create a history from a pandas data frame.

#### Parameters

**history\_df** (`pd.DataFrame`) – The source data frame.

#### Returns

The created history.

#### Return type

`ParameterHistory`

`get_parameters(index: int) → ndarray`

Get parameters for a history index.

#### Parameters

**index** (`int`) – The history index.

#### Returns

The parameter values at the history index as array.

#### Return type

`np.ndarray`



**classmethod loader**(*path: str*) → *ParameterHistory*

Create a history from a csv file.

**Parameters**

**path** (*str*) – The path to the csv file.

**Returns**

The created history.

**Return type**

*ParameterHistory*

**property number\_of\_records: int**

Return the number of records in the history.

**Returns**

The number of records.

**Return type**

int

**property parameter\_labels: list[str]**

Return the labels of the parameters in the history.

**Returns**

A list of parameter labels.

**Return type**

list[str]

**property parameters: list[ndarray]**

Return the parameters in the history.

**Returns**

A list of parameters in the history.

**Return type**

list[np.ndarray]

**to\_csv**(*file\_name: str | PathLike[str], delimiter: str = ','*)

Write a *ParameterHistory* to a CSV file.

**Parameters**

- **file\_name** (*str*) – The path to the CSV file.
- **delimiter** (*str*) – The delimiter of the CSV file.

**to\_dataframe**() → DataFrame

Create a data frame from the history.

**Returns**

The created data frame.

**Return type**

pd.DataFrame

## parameters

The parameters class.

## Functions

### Summary

<code>flatten_parameter_dict</code>	Flatten a parameter dictionary.
<code>param_dict_to_markdown</code>	Format the <i>Parameters</i> as markdown string.

### flatten\_parameter\_dict

`glotaran.parameter.parameters.flatten_parameter_dict(parameter_dict: dict)` →  
Generator[tuple[str, list[Any],  
dict[str, Any] | None], None, None]

Flatten a parameter dictionary.

#### Parameters

**parameter\_dict** (*dict*) – The parameter dictionary.

#### Yields

*tuple[str, list[Any], dict | None* – The concatenated keys, the parameter definition and default options.

### param\_dict\_to\_markdown

`glotaran.parameter.parameters.param_dict_to_markdown(parameters: dict | list,  
float_format: str = '.3e', depth: int  
= 0, label: str | None = None)` →  
*MarkdownStr*

Format the *Parameters* as markdown string.

This is done by recursing the nested *Parameters* tree.

#### Parameters

- **parameters** (*dict | list*) – The parameter dict or list.
- **float\_format** (*str*) – Format string for floating point numbers, by default “.3e”
- **depth** (*int*) – The depth of the parameter dict.
- **label** (*str | None*) – The label of the parameter dict.

#### Returns

The markdown representation as string.

#### Return type

*MarkdownStr*

## Classes

### Summary

<i>Parameters</i>	A container for <code>Parameter</code> .
-------------------	--

### Parameters

**class** `glotaran.parameter.parameters.Parameters`(*parameters*: `dict[str, Parameter]`)

Bases: `object`

A container for `Parameter`.

Create *Parameters*.

#### **Parameters**

**parameters** (`dict[str, Parameter]`) – A parameter list containing parameters

#### **Returns**

The created *Parameters*.

#### **Return type**

‘Parameters’

### Attributes Summary

<i>labels</i>	List of all labels.
---------------	---------------------

### labels

`Parameters.labels`

List of all labels.

#### **Return type**

`list[str]`

## Methods Summary

<code>all</code>	Iterate over all parameters.
<code>copy</code>	Create a copy of the <i>Parameters</i> .
<code>from_dataframe</code>	Create a <i>Parameters</i> from a pandas. DataFrame.
<code>from_dict</code>	Create a <i>Parameters</i> from a dictionary.
<code>from_list</code>	Create <i>Parameters</i> from a list.
<code>from_parameter_dict_list</code>	Create <i>Parameters</i> from a list of parameter dictionaries.
<code>get</code>	Get a Parameter by its label.
<code>get_label_value_and_bounds_arrays</code>	Return a arrays of all parameter labels, values and bounds.
<code>has</code>	Check if a parameter with the given label is in the group or in a subgroup.
<code>loader</code>	Create a <i>Parameters</i> instance from the specs defined in a file.
<code>markdown</code>	Format the ParameterGroup as markdown string.
<code>set_from_history</code>	Update the <i>Parameters</i> with values from a parameter history.
<code>set_from_label_and_value_arrays</code>	Update the parameter values from a list of labels and values.
<code>to_dataframe</code>	Create a pandas data frame from the group.
<code>to_parameter_dict_list</code>	Create list of parameter dictionaries from the group.
<code>to_parameter_dict_or_list</code>	Convert to a dict or list of parameter definitions.
<code>update_parameter_expression</code>	Update all parameters which have an expression.

### all

`Parameters.all()` → `Generator[Parameter, None, None]`

Iterate over all parameters.

#### Yields

*Parameter* – A parameter in the parameters.

### copy

`Parameters.copy()` → *Parameters*

Create a copy of the *Parameters*.

#### Returns

- *Parameters* – A copy of the *Parameters*.
- .. # noqa (D414)

## from\_dataframe

**classmethod** `Parameters.from_dataframe(df: DataFrame, source: str = 'DataFrame') → Parameters`

Create a *Parameters* from a `pandas.DataFrame`.

### Parameters

- **df** (`pd.DataFrame`) – The source data frame.
- **source** (`str`) – Optional name of the source file, used for error messages.

### Raises

**ValueError** – Raised if the columns ‘label’ or ‘value’ doesn’t exist. Also raised if the columns ‘minimum’, ‘maximum’ or ‘values’ contain non numeric values or if the columns ‘non-negative’ or ‘vary’ are no boolean.

### Returns

- *Parameters* – The created parameter group.
- .. # noqa (D414)

## from\_dict

**classmethod** `Parameters.from_dict(parameter_dict: dict[str, dict[str, Any] | list[float | list[Any]]]) → Parameters`

Create a *Parameters* from a dictionary.

### Parameters

**parameter\_dict** (`dict[str, dict[str, Any] | list[float | list[Any]]]`) – A parameter dictionary containing parameters.

### Returns

- *Parameters* – The created *Parameters*
- .. # noqa (D414)

## from\_list

**classmethod** `Parameters.from_list(parameter_list: list[float | int | str | list[Any] | dict[str, Any]]) → Parameters`

Create *Parameters* from a list.

### Parameters

**parameter\_list** (`list[float | list[Any]]`) – A parameter list containing parameters

### Returns

- *Parameters* – The created *Parameters*.
- .. # noqa (D414)

### from\_parameter\_dict\_list

**classmethod** `Parameters.from_parameter_dict_list`(*parameter\_dict\_list*: *list[dict[str, Any]]*) → *Parameters*

Create *Parameters* from a list of parameter dictionaries.

**Parameters**

**parameter\_dict\_list** (*list[dict[str, Any]]*) – A list of parameter dictionaries.

**Returns**

- *Parameters* – The created *Parameters*.
- .. # noqa (D414)

### get

`Parameters.get`(*label*: *str*) → *Parameter*

Get a *Parameter* by its label.

**Parameters**

**label** (*str*) – The label of the parameter, with its path in a *ParameterGroup* prepended.

**Returns**

The parameter.

**Return type**

*Parameter*

**Raises**

**ParameterNotFoundException** – Raised if no parameter with the given label exists.

### get\_label\_value\_and\_bounds\_arrays

`Parameters.get_label_value_and_bounds_arrays`(*exclude\_non\_vary*: *bool = False*) → *tuple[list[str], ndarray, ndarray, ndarray]*

Return a arrays of all parameter labels, values and bounds.

**Parameters**

**exclude\_non\_vary** (*bool*) – If true, parameters with *vary=False* are excluded.

**Returns**

A tuple containing a list of parameter labels and an array of the values, lower and upper bounds.

**Return type**

*tuple[list[str], np.ndarray, np.ndarray, np.ndarray]*

## has

`Parameters.has(label: str) → bool`

Check if a parameter with the given label is in the group or in a subgroup.

### Parameters

**label** (*str*) – The label of the parameter, with its path in a `ParameterGroup` prepended.

### Returns

Whether a parameter with the given label exists in the group.

### Return type

`bool`

## loader

`Parameters.loader(format_name: str | None = None, **kwargs) → Parameters`

Create a `Parameters` instance from the specs defined in a file.

### Parameters

- **file\_name** (*StrOrPath*) – File containing the parameter specs.
- **format\_name** (*str* | *None*) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `load_parameters` implementation of the project io plugin.

### Returns

- *Parameters* – `Parameters` instance created from the file.
- `.. # noqa (D414)`

## markdown

`Parameters.markdown(float_format: str = '.3e') → MarkdownStr`

Format the `ParameterGroup` as markdown string.

This is done by recursing the nested `ParameterGroup` tree.

### Parameters

**float\_format** (*str*) – Format string for floating point numbers, by default “.3e”

### Returns

The markdown representation as string.

### Return type

`MarkdownStr`

### set\_from\_history

`Parameters.set_from_history(history: ParameterHistory, index: int)`

Update the *Parameters* with values from a parameter history.

#### Parameters

- **history** (*ParameterHistory*) – The parameter history.
- **index** (*int*) – The history index.

### set\_from\_label\_and\_value\_arrays

`Parameters.set_from_label_and_value_arrays(labels: list[str], values: ndarray)`

Update the parameter values from a list of labels and values.

#### Parameters

- **labels** (*list[str]*) – A list of parameter labels.
- **values** (*np.ndarray*) – An array of parameter values.

#### Raises

**ValueError** – Raised if the size of the labels does not match the size of values.

### to\_dataframe

`Parameters.to_dataframe() → DataFrame`

Create a pandas data frame from the group.

#### Returns

The created data frame.

#### Return type

`pd.DataFrame`

### to\_parameter\_dict\_list

`Parameters.to_parameter_dict_list() → list[dict[str, Any]]`

Create list of parameter dictionaries from the group.

#### Returns

A list of parameter dictionaries.

#### Return type

`list[dict[str, Any]]`



### to\_parameter\_dict\_or\_list

`Parameters.to_parameter_dict_or_list(serialize_parameters: bool = False) → dict | list`

Convert to a dict or list of parameter definitions.

**Parameters**

**serialize\_parameters** (*bool*) – If true, the parameters will be serialized into list representation.

**Returns**

A dict or list of parameter definitions.

**Return type**

dict | list

### update\_parameter\_expression

`Parameters.update_parameter_expression()`

Update all parameters which have an expression.

**Raises**

**ValueError** – Raised if an expression evaluates to a non-numeric value.

## Methods Documentation

`all() → Generator[Parameter, None, None]`

Iterate over all parameters.

**Yields**

*Parameter* – A parameter in the parameters.

`copy() → Parameters`

Create a copy of the *Parameters*.

**Returns**

- *Parameters* – A copy of the *Parameters*.
- .. # noqa (D414)

`classmethod from_dataframe(df: DataFrame, source: str = 'DataFrame') → Parameters`

Create a *Parameters* from a `pandas.DataFrame`.

**Parameters**

- **df** (*pd.DataFrame*) – The source data frame.
- **source** (*str*) – Optional name of the source file, used for error messages.

**Raises**

**ValueError** – Raised if the columns ‘label’ or ‘value’ doesn’t exist. Also raised if the columns ‘minimum’, ‘maximum’ or ‘values’ contain non numeric values or if the columns ‘non-negative’ or ‘vary’ are no boolean.

**Returns**

- *Parameters* – The created parameter group.
- .. # noqa (D414)

**classmethod** `from_dict`(*parameter\_dict*: `dict[str, dict[str, Any] | list[float] | list[Any]]`) → *Parameters*

Create a *Parameters* from a dictionary.

**Parameters**

**parameter\_dict** (`dict[str, dict[str, Any] | list[float] | list[Any]]`) – A parameter dictionary containing parameters.

**Returns**

- *Parameters* – The created *Parameters*
- .. # noqa (D414)

**classmethod** `from_list`(*parameter\_list*: `list[float | int | str | list[Any] | dict[str, Any]]`) → *Parameters*

Create *Parameters* from a list.

**Parameters**

**parameter\_list** (`list[float | list[Any]]`) – A parameter list containing parameters

**Returns**

- *Parameters* – The created *Parameters*.
- .. # noqa (D414)

**classmethod** `from_parameter_dict_list`(*parameter\_dict\_list*: `list[dict[str, Any]]`) → *Parameters*

Create *Parameters* from a list of parameter dictionaries.

**Parameters**

**parameter\_dict\_list** (`list[dict[str, Any]]`) – A list of parameter dictionaries.

**Returns**

- *Parameters* – The created *Parameters*.
- .. # noqa (D414)

**get**(*label*: *str*) → *Parameter*

Get a *Parameter* by its label.

**Parameters**

**label** (*str*) – The label of the parameter, with its path in a *ParameterGroup* prepended.

**Returns**

The parameter.

**Return type**

*Parameter*

**Raises**

**ParameterNotFoundException** – Raised if no parameter with the given label exists.

**get\_label\_value\_and\_bounds\_arrays**(*exclude\_non\_vary*: *bool* = *False*) → `tuple[list[str], ndarray, ndarray, ndarray]`

Return a arrays of all parameter labels, values and bounds.

**Parameters**

**exclude\_non\_vary** (*bool*) – If true, parameters with *vary=False* are excluded.

**Returns**

A tuple containing a list of parameter labels and an array of the values, lower and upper bounds.

**Return type**

*tuple*[*list*[*str*], *np.ndarray*, *np.ndarray*, *np.ndarray*]

**has**(*label: str*) → *bool*

Check if a parameter with the given label is in the group or in a subgroup.

**Parameters**

**label** (*str*) – The label of the parameter, with its path in a *ParameterGroup* prepended.

**Returns**

Whether a parameter with the given label exists in the group.

**Return type**

*bool*

**property labels:** *list*[*str*]

List of all labels.

**Return type**

*list*[*str*]

**loader**(*format\_name: str | None = None, \*\*kwargs*) → *Parameters*

Create a *Parameters* instance from the specs defined in a file.

**Parameters**

- **file\_name** (*StrOrPath*) – File containing the parameter specs.
- **format\_name** (*str | None*) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the *load\_parameters* implementation of the project io plugin.

**Returns**

- *Parameters* – *Parameters* instance created from the file.
- .. # noqa (D414)

**markdown**(*float\_format: str = '.3e'*) → *MarkdownStr*

Format the *ParameterGroup* as markdown string.

This is done by recursing the nested *ParameterGroup* tree.

**Parameters**

**float\_format** (*str*) – Format string for floating point numbers, by default “.3e”

**Returns**

The markdown representation as string.

**Return type**

*MarkdownStr*

**set\_from\_history**(*history*: [ParameterHistory](#), *index*: *int*)

Update the [Parameters](#) with values from a parameter history.

**Parameters**

- **history** ([ParameterHistory](#)) – The parameter history.
- **index** (*int*) – The history index.

**set\_from\_label\_and\_value\_arrays**(*labels*: *list[str]*, *values*: *ndarray*)

Update the parameter values from a list of labels and values.

**Parameters**

- **labels** (*list[str]*) – A list of parameter labels.
- **values** (*np.ndarray*) – An array of parameter values.

**Raises**

[ValueError](#) – Raised if the size of the labels does not match the size of values.

**to\_dataframe**() → [DataFrame](#)

Create a pandas data frame from the group.

**Returns**

The created data frame.

**Return type**

[pd.DataFrame](#)

**to\_parameter\_dict\_list**() → *list[dict[str, Any]]*

Create list of parameter dictionaries from the group.

**Returns**

A list of parameter dictionaries.

**Return type**

*list[dict[str, Any]]*

**to\_parameter\_dict\_or\_list**(*serialize\_parameters*: *bool = False*) → *dict | list*

Convert to a dict or list of parameter definitions.

**Parameters**

**serialize\_parameters** (*bool*) – If true, the parameters will be serialized into list representation.

**Returns**

A dict or list of parameter definitions.

**Return type**

*dict | list*

**update\_parameter\_expression**()

Update all parameters which have an expression.

**Raises**

[ValueError](#) – Raised if an expression evaluates to a non-numeric value.

## Exceptions

### Exception Summary

<code>ParameterNotFoundException</code>	Raised when a Parameter is not found.
---	---------------------------------------

### ParameterNotFoundException

**exception** `glotaran.parameter.parameters.ParameterNotFoundException(label: str)`

Raised when a Parameter is not found.

## 16.1.9 plugin\_system

Plugin system package containing all plugin related implementations.

### Modules

<code>glotaran.plugin_system.base_registry</code>	Functionality to register, initialize and retrieve glotaran plugins.
<code>glotaran.plugin_system.data_io_registration</code>	Data Io registration convenience functions.
<code>glotaran.plugin_system.io_plugin_utils</code>	Utility functions for io plugin.
<code>glotaran.plugin_system.megacomplex_registration</code>	Megacomplex registration convenience functions.
<code>glotaran.plugin_system.project_io_registration</code>	Project Io registration convenience functions.

### base\_registry

Functionality to register, initialize and retrieve glotaran plugins.

Since this module is imported at the root `__init__.py` file all other glotaran imports should be used for typechecking only in the ‘if TYPE\_CHECKING’ block. This is to prevent issues with circular imports.

### Functions

## Summary

<code>add_instantiated_plugin_to_registry</code>	Add instances of <code>plugin_class</code> to the given registry.
<code>add_plugin_to_registry</code>	Add a plugin with name <code>plugin_register_key</code> to the given registry.
<code>full_plugin_name</code>	Full name of a plugin instance/class similar to the <code>repr</code> .
<code>get_method_from_plugin</code>	Retrieve a method callable from an class or instance <code>plugin</code> .
<code>get_plugin_from_registry</code>	Retrieve a plugin with name <code>plugin_register_key</code> is registered in a given registry.
<code>is_registered_plugin</code>	Check if a plugin with name <code>plugin_register_key</code> is registered in the given registry.
<code>load_plugins</code>	Initialize plugins registered under the entrypoint 'glotaran.plugins'.
<code>methods_differ_from_baseclass</code>	Check if a plugins methods implementation differ from its baseclass.
<code>methods_differ_from_baseclass_table</code>	Create table of which plugins methods differ from their baseclass.
<code>registered_plugins</code>	Names of the plugins in the given registry.
<code>set_plugin</code>	Set a plugins short name to a specific plugin referred by its full name.
<code>show_method_help</code>	Show help on a method as if it was called directly on it.
<code>supported_file_extensions</code>	Get file extensions for plugins that support all methods in <code>method_names</code> .

## `add_instantiated_plugin_to_registry`

```
glotaran.plugin_system.base_registry.add_instantiated_plugin_to_registry(plugin_register_keys:  
    str |  
    list[str],  
    plugin_class:  
    type[_PluginInstantiableType],  
    plugin_registry:  
    MutableMapping[str,  
        _PluginInstantiableType],  
    plugin_set_func_name:  
    str) →  
    None
```

Add instances of `plugin_class` to the given registry.

**Parameters**

- **plugin\_register\_keys** (*str* | *list[str]*) – Name/-s of the plugin under which it is registered.
- **plugin\_class** (*type[\_PluginInstantiableType]*) – Pluginclass which should be instantiated with `plugin_register_keys` and added to the registry.
- **plugin\_registry** (*MutableMapping[str, \_PluginInstantiableType]*) – Registry the plugin should be added to.
- **plugin\_set\_func\_name** (*str*) – Name of the function used to pin a plugin.

See also:

`add_plugin_to_register`

**add\_plugin\_to\_registry**

```
glotaran.plugin_system.base_registry.add_plugin_to_registry(plugin_register_key: str,
                                                            plugin: _PluginType,
                                                            plugin_registry:
                                                                MutableMapping[str,
                                                                _PluginType],
                                                            plugin_set_func_name:
                                                                str, instance_identifier:
                                                                str = "") → None
```

Add a plugin with name `plugin_register_key` to the given registry.

In addition it also adds the plugin with it full import path name as key, which allows for a better reproducibility in case there are conflicting plugins.

**Parameters**

- **plugin\_register\_key** (*str*) – Name of the plugin under which it is registered.
- **plugin** (*\_PluginType*) – Plugin to be added to the registry.
- **plugin\_registry** (*MutableMapping[str, \_PluginType]*) – Registry the plugin should be added to.
- **plugin\_set\_func\_name** (*str*) – Name of the function used to pin a plugin.
- **instance\_identifier** (*str*) – Used to differentiate between plugin instances (e.g. different format for IO plugins)

**Raises**

**ValueError** – If `plugin_register_key` has the character ‘.’ in it.

See also:

`add_instantiated_plugin_to_register`, *full\_plugin\_name*

## full\_plugin\_name

glotaran.plugin\_system.base\_registry.**full\_plugin\_name**(*plugin: object | type[object]*) → *str*

Full name of a plugin instance/class similar to the repr.

### Parameters

**plugin** (*object* | *type[object]*) – plugin instance/class

### Examples

```
>>> from glotaran.builtin.io.sdt.sdt_file_reader import SdtDataIo
>>> full_plugin_name(SdtDataIo)
"glotaran.builtin.io.sdt.sdt_file_reader.SdtDataIo"
>>> full_plugin_name(SdtDataIo("sdt"))
"glotaran.builtin.io.sdt.sdt_file_reader.SdtDataIo"
```

### Returns

Full name of the plugin.

### Return type

*str*

## get\_method\_from\_plugin

glotaran.plugin\_system.base\_registry.**get\_method\_from\_plugin**(*plugin: object | type[object]*, *method\_name: str*) → *Callable[..., Any]*

Retrieve a method callable from an class or instance plugin.

### Parameters

- **plugin** (*object* | *type[object]*,) – Plugin instance or class.
- **method\_name** (*str*) – Method name, e.g. load\_megacomplex.

### Returns

Method callable.

### Return type

*Callable[..., Any]*

### Raises

- **ValueError** – If plugin has an attribute with that name but it isn't callable.
- **ValueError** – If plugin misses the attribute.



## get\_plugin\_from\_registry

```
glotaran.plugin_system.base_registry.get_plugin_from_registry(plugin_register_key:
    str, plugin_registry:
    MutableMapping[str,
    _PluginType],
    not_found_error_message:
    str) → _PluginType
```

Retrieve a plugin with name `plugin_register_key` is registered in a given registry.

### Parameters

- **plugin\_register\_key** (*str*) – Name of the plugin under which it is registered.
- **plugin\_registry** (*MutableMapping[str, \_PluginType]*) – Registry to search in.
- **not\_found\_error\_message** (*str*) – Error message to be shown if the plugin wasn't found.

### Returns

Plugin from the plugin Registry.

### Return type

*\_PluginType*

### Raises

**ValueError** – If there was no plugin registered under the name `plugin_register_key`.

## is\_registered\_plugin

```
glotaran.plugin_system.base_registry.is_registered_plugin(plugin_register_key: str,
    plugin_registry:
    MutableMapping[str,
    _PluginType]) → bool
```

Check if a plugin with name `plugin_register_key` is registered in the given registry.

### Parameters

- **plugin\_register\_key** (*str*) – Name of the plugin under which it is registered.
- **plugin\_registry** (*MutableMapping[str, \_PluginType]*) – Registry to search in.

### Returns

Whether or not a plugin is in the registry.

### Return type

*bool*

## load\_plugins

`glotaran.plugin_system.base_registry.load_plugins()`

Initialize plugins registered under the entrypoint 'glotaran.plugins'.

For an entry\_point to be considered a glotaran plugin it just needs to start with 'glotaran.plugins', which allows for an easy extendability.

Currently used builtin entrypoints are:

- `glotaran.plugins.data_io`
- `glotaran.plugins.megacomplex`
- `glotaran.plugins.project_io`

## methods\_differ\_from\_baseclass

```
glotaran.plugin_system.base_registry.methods_differ_from_baseclass(method_names:  
                                                                    str |  
                                                                    Sequence[str],  
                                                                    plugin: Generic-  
                                                                    PluginInstance |  
                                                                    type[GenericPluginInstance],  
                                                                    base_class:  
                                                                    type[GenericPluginInstance])  
                                                                    →  
                                                                    Generator[bool,  
                                                                    None, None]
```

Check if a plugins methods implementation differ from its baseclass.

Based on the assumption that `base_class` didn't implement the methods (e.g. `DataIoInterface` or `ProjectIoInterface`), this can be used to to create a 'supported methods' list.

### Parameters

- **method\_names** (*str* | *list*[*str*]) – Name|s of the method|s
- **plugin** (*GenericPluginInstance* | *type*[*GenericPluginInstance*])  
– Plugin class or instance.
- **base\_class** (*type*[*GenericPluginInstance*]) – Base class the plugin in-  
herited from.

### Yields

*bool* – Whether or not a plugins method differs from the implementation in `base_class`.

## methods\_differ\_from\_baseclass\_table

```

glotaran.plugin_system.base_registry.methods_differ_from_baseclass_table(method_names:
    str | Sequence[str],
    plugin_registry_keys:
    str | Sequence[str],
    get_plugin_function:
    Callable[[str],
    GenericPluginInstance],
    base_class:
    type[GenericPluginInstance],
    plugin_names:
    bool =
    False)
    →
    Generator[list[str
    | bool],
    None,
    None]

```

Create table of which plugins methods differ from their baseclass.

This uses the assumption that all plugins have the same `base_class`.

The main purpose of this function is to show the user which plugin implements which methods differently than its baseclass.

Based on the assumption that `base_class` didn't implement the methods (e.g. `DataIoInterface` or `ProjectIoInterface`), this can be used to to create a 'supported methods' table.

### Parameters

- **method\_names** (*str* | *list*[*str*]) – Name|s of the method|s.
- **plugin\_registry\_keys** (*str* | *list*[*str*]) – Keys the plugins are registered under (e.g. return value of the implementation of `func:registered_plugins`)
- **get\_plugin\_function** (*Callable*[[*str*], *GenericPluginInstance* | *type*[*GenericPluginInstance*]]) – Function to get plugin from plugin registry.
- **base\_class** (*type*[*GenericPluginInstance*]) – Base class the plugin inherited from.
- **plugin\_names** (*bool*) – Whether or not to add the names of the plugins to the lists.

### Yields

*list*[*str* | *bool*] – Row with the first value being the `plugin_registry_key` and the others whether or not a plugins method differs from `base_class`.

See also:

[`methods\_differ\_from\_baseclass`](#)

## registered\_plugins

```
glotaran.plugin_system.base_registry.registered_plugins(plugin_registry:
    MutableMapping[str,
    _PluginType], full_names:
    bool = False) → list[str]
```

Names of the plugins in the given registry.

### Parameters

- **plugin\_registry** (*MutableMapping*[*str*, *\_PluginType*]) – Registry to search in.
- **full\_names** (*bool*) – Whether to display the full names the plugins are registered under as well.

### Returns

List of plugin names in plugin\_registry.

### Return type

*list*[*str*]

## set\_plugin

```
glotaran.plugin_system.base_registry.set_plugin(plugin_register_key: str,
    full_plugin_name: str, plugin_registry:
    MutableMapping[str, _PluginType],
    plugin_register_key_name: str =
    'format_name') → None
```

Set a plugins short name to a specific plugin referred by its full name.

This can be used to ensure that a specific plugin is used in case there are conflicting plugins installed.

### Parameters

- **plugin\_register\_key** (*str*) – Name of the plugin under which it is registered.
- **full\_plugin\_name** (*str*) – Full name (import path) of the registered plugin.
- **plugin\_registry** (*MutableMapping*[*str*, *\_PluginType*]) – Registry the plugin should be set in to.
- **plugin\_register\_key\_name** (*str*) – Name of the arg passed `plugin_register_key` in the function that implements `set_plugin`.

### Raises

- **ValueError** – If `plugin_register_key` has the character ‘.’ in it.
- **ValueError** – If there isn’t a registered plugin with the key `full_plugin_name`.

See also:

[`add\_plugin\_to\_registry`](#), [`full\_plugin\_name`](#)

## show\_method\_help

glotaran.plugin\_system.base\_registry.**show\_method\_help**(*plugin*: *object* | *type[object]*,  
*method\_name*: *str*) → *None*

Show help on a method as if it was called directly on it.

### Parameters

- **plugin** (*object* | *type[object]*,) – Plugin instance or class.
- **method\_name** (*str*) – Method name, e.g. load\_megacomplex.

## supported\_file\_extensions

glotaran.plugin\_system.base\_registry.**supported\_file\_extensions**(*method\_names*: *str* |  
*Sequence[str]*,  
*plugin\_registry\_keys*:  
*str* | *Sequence[str]*,  
*get\_plugin\_function*:  
*Callable[[str],*  
*GenericPluginIn-*  
*stance* |  
*type[GenericPluginInstance]]*,  
*base\_class*:  
*type[GenericPluginInstance]*)  
→ *Generator[str,*  
*None, None]*

Get file extensions for plugins that support all methods in *method\_names*.

### Parameters

- **method\_names** (*str* | *list[str]*) – Name|s of the method|s.
- **plugin\_registry\_keys** (*str* | *list[str]*) – Keys the plugins are registered under (e.g. return value of the implementation of *func:registered\_plugins*)
- **get\_plugin\_function** (*Callable[[str], GenericPluginInstance* | *type[GenericPluginInstance]]*) – Function to get plugin from plugin registry.
- **base\_class** (*type[GenericPluginInstance]*) – Base class the plugin inherited from.

### Yields

*Generator[str, None, None]* – File extension supported by all methods in *method\_names*.

### See also:

[\*methods\\_differ\\_from\\_baseclass\*](#), [\*methods\\_differ\\_from\\_baseclass\\_table\*](#)

## Exceptions

### Exception Summary

PluginOverwriteWarning	Warning used if a plugin tries to overwrite and existing plugin.
------------------------	--

### PluginOverwriteWarning

```
exception glotaran.plugin_system.base_registry.PluginOverwriteWarning(*args: Any,
                                                                       old_key: str,
                                                                       old_plugin:
                                                                       object |
                                                                       type[object],
                                                                       new_plugin:
                                                                       object |
                                                                       type[object],
                                                                       plu-
                                                                       gin_set_func_name:
                                                                       str)
```

Warning used if a plugin tries to overwrite and existing plugin.

Use old and new plugin and keys to give verbose warning message.

#### Parameters

- **old\_key** (*str*) – Old registry key.
- **old\_plugin** (*object* | *type[object]*) – Old plugin ('registry[old\_key]').
- **new\_plugin** (*object* | *type[object]*) – New Plugin ('registry[new\_key]').
- **plugin\_set\_func\_name** (*str*) – Name of the function used to pin a plugin.
- **\*args** (*Any*) – Additional args passed to the super constructor.

## data\_io\_registration

Data Io registration convenience functions.

---

**Note:** The [call-arg] type error would be raised since the base methods doesn't have a **\*\*kwargs** argument, but we rather ignore this error here, than adding **\*\*kwargs** to the base method and causing an [override] type error in the plugins implementation.

---

## Functions

### Summary

<code>data_io_plugin_table</code>	Return registered data io plugins and which functions they support as markdown table.
<code>get_data_io</code>	Retrieve a data io plugin from the data_io registry.
<code>get_data_loader</code>	Retrieve implementation of the <code>read_dataset</code> functionality for the format 'format_name'.
<code>get_data_saver</code>	Retrieve implementation of the <code>save_dataset</code> functionality for the format 'format_name'.
<code>is_known_data_format</code>	Check if a data format is in the data_io registry.
<code>known_data_formats</code>	Names of the registered data io plugins.
<code>load_dataset</code>	Read data from a file to <code>xarray.Dataset</code> or <code>xarray.DataArray</code> .
<code>register_data_io</code>	Register data io plugins to one or more formats.
<code>save_dataset</code>	Save data from <code>xarray.Dataset</code> or <code>xarray.DataArray</code> to a file.
<code>set_data_plugin</code>	Set the plugin used for a specific data format.
<code>show_data_io_method_help</code>	Show help for the implementation of data io plugin methods.
<code>supported_file_extensions_data_io</code>	Get data io formats that support all methods in <code>method_names</code> .

### data\_io\_plugin\_table

```
glotaran.plugin_system.data_io_registration.data_io_plugin_table(*, plugin_names:
                                                                    bool = False,
                                                                    full_names: bool =
                                                                    False) →
                                                                    MarkdownStr
```

Return registered data io plugins and which functions they support as markdown table.

This is especially useful when you work with new plugins.

#### Parameters

- **plugin\_names** (*bool*) – Whether or not to add the names of the plugins to the table.
- **full\_names** (*bool*) – Whether to display the full names the plugins are registered under as well.

#### Returns

Markdown table of data io plugins.

#### Return type

*MarkdownStr*

## get\_data\_io

glotaran.plugin\_system.data\_io\_registration.**get\_data\_io**(format\_name: *str*) → *DataIoInterface*

Retrieve a data io plugin from the data\_io registry.

### Parameters

**format\_name** (*str*) – Name of the data io plugin under which it is registered.

### Returns

Data io plugin instance.

### Return type

*DataIoInterface*

## get\_dataloader

glotaran.plugin\_system.data\_io\_registration.**get\_dataloader**(format\_name: *str*) → *DataLoader*

Retrieve implementation of the read\_dataset functionality for the format 'format\_name'.

This allows to get the proper help and autocomplete for the function, which is especially valuable if the function provides additional options.

### Parameters

**format\_name** (*str*) – Format the dataloader should be able to read.

### Returns

Function to load data of format format\_name as *xarray.Dataset* or *xarray.DataArray*.

### Return type

*DataLoader*

## get\_datasaver

glotaran.plugin\_system.data\_io\_registration.**get\_datasaver**(format\_name: *str*) → *DataSaver*

Retrieve implementation of the save\_dataset functionality for the format 'format\_name'.

This allows to get the proper help and autocomplete for the function, which is especially valuable if the function provides additional options.

### Parameters

**format\_name** (*str*) – Format the datawriter should be able to write.

### Returns

Function to write *xarray.Dataset* to the format format\_name .

### Return type

*DataSaver*



### is\_known\_data\_format

glotaran.plugin\_system.data\_io\_registration.is\_known\_data\_format(*format\_name: str*)  
→ bool

Check if a data format is in the data\_io registry.

#### Parameters

**format\_name** (*str*) – Name of the data io plugin under which it is registered.

#### Returns

Whether or not the data format is a registered data io plugins.

#### Return type

bool

### known\_data\_formats

glotaran.plugin\_system.data\_io\_registration.known\_data\_formats(*full\_names: bool = False*) → list[str]

Names of the registered data io plugins.

#### Parameters

**full\_names** (*bool*) – Whether to display the full names the plugins are registered under as well.

#### Returns

List of registered data io plugins.

#### Return type

list[str]

### load\_dataset

glotaran.plugin\_system.data\_io\_registration.load\_dataset(*file\_name: StrOrPath, format\_name: str | None = None, \*\*kwargs: Any*) → xr.Dataset

Read data from a file to `xarray.Dataset` or `xarray.DataArray`.

#### Parameters

- **file\_name** (*StrOrPath*) – File containing the data.
- **format\_name** (*str*) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `read_dataset` implementation of the data io plugin. If you aren't sure about those use `get_data_loader` to get the implementation with the proper help and autocomplete.

#### Returns

Data loaded from the file.

#### Return type

xr.Dataset

## register\_data\_io

```
glotaran.plugin_system.data_io_registration.register_data_io(format_names: str |  
list[str]) →  
Callable[[type[DataIoInterface]],  
type[DataIoInterface]]
```

Register data io plugins to one or more formats.

Decorate a data io plugin class with `@register_data_io(format_name | [*format_names])` to add it to the registry.

### Parameters

**format\_names** (*str* | *list[str]*) – Name of the data io plugin under which it is registered.

### Returns

Inner decorator function.

### Return type

Callable[[type[DataIoInterface]], type[DataIoInterface]]

## Examples

```
>>> @register_data_io("my_format_1")  
... class MyDataIo1(DataIoInterface):  
...     pass
```

```
>>> @register_data_io(["my_format_1", "my_format_1_alias"])  
... class MyDataIo2(DataIoInterface):  
...     pass
```

## save\_dataset

```
glotaran.plugin_system.data_io_registration.save_dataset(dataset: xr.Dataset |  
xr.DataArray, file_name:  
StrOrPath, format_name: str  
| None = None, *,  
data_filters: list[str] | None =  
None, allow_overwrite: bool  
= False, update_source_path:  
bool = True, **kwargs: Any)  
→ None
```

Save data from `xarray.Dataset` or `xarray.DataArray` to a file.

### Parameters

- **dataset** (*xr.Dataset* | *xr.DataArray*) – Data to be written to file.
- **file\_name** (*StrOrPath*) – File to write the data to.
- **format\_name** (*str*) – Format the file should be in, if not provided it will be inferred from the file extension.
- **data\_filters** (*list[str]* | *None*) – Optional list of items in the dataset to be saved.

- **allow\_overwrite** (*bool*) – Whether or not to allow overwriting existing files, by default False
- **update\_source\_path** (*bool*) – Whether or not to update the `source_path` attribute to `file_name` when saving. by default True
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `write_dataset` implementation of the data io plugin. If you aren't sure about those use `get_datasaver` to get the implementation with the proper help and autocomplete.

## set\_data\_plugin

```
glotaran.plugin_system.data_io_registration.set_data_plugin(format_name: str,
                                                           full_plugin_name: str)
                                                           → None
```

Set the plugin used for a specific data format.

This function is useful when you want to resolve conflicts of installed plugins or overwrite the plugin used for a specific format.

Effectuated functions:

- `load_dataset()`
- `save_dataset()`

### Parameters

- **format\_name** (*str*) – Format name used to refer to the plugin when used for save and load functions.
- **full\_plugin\_name** (*str*) – Full name (import path) of the registered plugin.

## show\_data\_io\_method\_help

```
glotaran.plugin_system.data_io_registration.show_data_io_method_help(format_name:
                                                                       str,
                                                                       method_name:
                                                                       Lit-
                                                                       eral['load_dataset',
                                                                       'save_dataset'])
                                                                       → None
```

Show help for the implementation of data io plugin methods.

### Parameters

- **format\_name** (*str*) – Format the method should support.
- **method\_name** (`{'load_dataset', 'save_dataset'}`) – Method name

## supported\_file\_extensions\_data\_io

```
glotaran.plugin_system.data_io_registration.supported_file_extensions_data_io(method_names:  
                                     str  
                                     |  
                                     Sequence[str])  
                                     →  
                                     Generator[str,  
                                     None,  
                                     None]
```

Get data io formats that support all methods in `method_names`.

### Parameters

**method\_names** (*str* / *Sequence[str]*) – Names of Methods that need to support the file extension.

### Yields

*Generator[str, None, None]* – File extension supported by all methods in `method_names`.

### See also:

`supported_file_extensions`, `DATA_IO_METHODS`

## io\_plugin\_utils

Utility functions for io plugin.

## Functions

### Summary

<i>bool_str_repr</i>	Replace boolean value with string repr.
<i>bool_table_repr</i>	Replace boolean value with string repr for all table values.
<i>infer_file_format</i>	Infer format of a file if it exists.
<i>not_implemented_to_value_error</i>	Decorate a function to raise <code>ValueError</code> instead of <code>NotImplementedError</code> .
<i>protect_from_overwrite</i>	Raise <code>FileExistsError</code> if files already exists and <code>allow_overwrite</code> isn't <code>True</code> .

## bool\_str\_repr

glotaran.plugin\_system.io\_plugin\_utils.**bool\_str\_repr**(value: *Any*, true\_repr: *str* = '\*', false\_repr: *str* = '/') → *Any*

Replace boolean value with string repr.

This function is a helper for table representation (e.g. with tabulate) of boolean values.

### Parameters

- **value** (*Any*) – Arbitrary value
- **true\_repr** (*str*) – Desired repr for True, by default “\*”
- **false\_repr** (*str*) – Desired repr for False, by default “/”

### Returns

Original value or desired repr for bool

### Return type

*Any*

## Examples

```
>>> table_data = [{"foo", True, False}, {"bar", False, True}]
>>> print(tabulate(map(lambda x: map(bool_table_repr, x), table_data)))
--- - -
foo  *  /
bar  /  *
--- - -
```

## bool\_table\_repr

glotaran.plugin\_system.io\_plugin\_utils.**bool\_table\_repr**(table\_data: *Iterable[Iterable[Any]]*, true\_repr: *str* = '\*', false\_repr: *str* = '/') → *Iterator[Iterator[Any]]*

Replace boolean value with string repr for all table values.

This function is an implementation of [bool\\_str\\_repr\(\)](#) for a 2D table, for easy usage with tabulate.

### Parameters

- **table\_data** (*Iterable[Iterable[Any]]*) – Data of the table e.g. a list of lists.
- **true\_repr** (*str*) – Desired repr for True, by default “\*”
- **false\_repr** (*str*) – Desired repr for False, by default “/”

### Returns

table\_data with original values or desired repr for bool

### Return type

*Iterator[Iterator[Any]]*

See also:

[`bool\_str\_repr`](#)

## Examples

```
>>> table_data = [{"foo", True, False}, {"bar", False, True}]
>>> print(tabulate(bool_table_repr(table_data)))
--- - -
foo  *  /
bar  /  *
--- - -
```

## `infer_file_format`

`glotaran.plugin_system.io_plugin_utils.infer_file_format`(*file\_path*: *StrOrPath*, \*,  
*needs\_to\_exist*: *bool* = *True*,  
*allow\_folder*=*False*) → *str*

Infer format of a file if it exists.

### Parameters

- **file\_path** (*StrOrPath*) – Path/str to the file.
- **needs\_to\_exist** (*bool*) – Whether or not a file need to exists for an successful format inferring. While write functions don't need the file to exists, load functions do.
- **allow\_folder** (*bool*) – Whether or not to allow the format to be folder. This is only used in `save_result`.

### Returns

File extension without the leading dot.

### Return type

*str*

### Raises

- **ValueError** – If file doesn't exists.
- **ValueError** – If file has no extension.

## `not_implemented_to_value_error`

`glotaran.plugin_system.io_plugin_utils.not_implemented_to_value_error`(*func*: *DecoratedFunc*)  
→  
*DecoratedFunc*

Decorate a function to raise `ValueError` instead of `NotImplementedError`.

This decorator is supposed to be used on functions which call functions that might raise a `NotImplementedError`, but raise `ValueError` instead with the same error text.

### Parameters

**func** (*DecoratedFunc*) – Function to be decorated.

**Returns**

Wrapped function.

**Return type**

DecoratedFunc

**protect\_from\_overwrite**

```
glotaran.plugin_system.io_plugin_utils.protect_from_overwrite(path: str |
    PathLike[str], *,
    allow_overwrite: bool
    = False) → None
```

Raise FileExistsError if files already exists and allow\_overwrite isn't True.

As a side effect this also creates the parent directory of a file if it does not exist.

**Parameters**

- **path** (*str*) – Path to a file or folder.
- **allow\_overwrite** (*bool*) – Whether or not to allow overwriting existing files, by default False

**Raises**

- **FileExistsError** – If path points to an existing file.
- **FileExistsError** – If path points to an existing folder which is not empty.

**megacomplex\_registration**

Megacomplex registration convenience functions.

**Functions****Summary**

<i>get_megacomplex</i>	Retrieve a megacomplex from the megacomplex registry.
<i>is_known_megacomplex</i>	Check if a megacomplex is in the megacomplex registry.
<i>known_megacomplex_names</i>	Names of the registered megacomplexes.
<i>megacomplex_plugin_table</i>	Return registered megacomplex plugins as mark-down table.
<i>register_megacomplex</i>	Add a megacomplex to the megacomplex registry.
<i>set_megacomplex_plugin</i>	Set the plugin used for a specific megacomplex name.

### get\_megacomplex

```
glotaran.plugin_system.megacomplex_registration.get_megacomplex(megacomplex_type:  
                                                                str) →  
                                                                type[Megacomplex]
```

Retrieve a megacomplex from the megacomplex registry.

**Parameters**

**megacomplex\_type** (*str*) – Name of the megacomplex under which it is registered.

**Returns**

Megacomplex class

**Return type**

type[Megacomplex]

### is\_known\_megacomplex

```
glotaran.plugin_system.megacomplex_registration.is_known_megacomplex(megacomplex_type:  
                                                                    str) → bool
```

Check if a megacomplex is in the megacomplex registry.

**Parameters**

**megacomplex\_type** (*str*) – Name of the megacomplex under which it is registered.

**Returns**

Whether or not the megacomplex is registered.

**Return type**

bool

### known\_megacomplex\_names

```
glotaran.plugin_system.megacomplex_registration.known_megacomplex_names(full_names:  
                                                                    bool =  
                                                                    False) →  
                                                                    list[str]
```

Names of the registered megacomplexes.

**Parameters**

**full\_names** (*bool*) – Whether to display the full names the plugins are registered under as well.

**Returns**

List of registered megacomplexes.

**Return type**

list[str]



## megacomplex\_plugin\_table

```
glotaran.plugin_system.megacomplex_registration.megacomplex_plugin_table(*, plugin_names:
bool =
False,
full_names:
bool =
False)
→
MarkdownStr
```

Return registered megacomplex plugins as markdown table.

This is especially useful when you work with new plugins.

### Parameters

- **plugin\_names** (*bool*) – Whether or not to add the names of the plugins to the table.
- **full\_names** (*bool*) – Whether to display the full names the plugins are registered under as well.

### Returns

Markdown table of megacomplexnames.

### Return type

*MarkdownStr*

## register\_megacomplex

```
glotaran.plugin_system.megacomplex_registration.register_megacomplex(megacomplex_type:
str, megacomplex:
type[Megacomplex])
→ None
```

Add a megacomplex to the megacomplex registry.

### Parameters

- **megacomplex\_type** (*str*) – Name of the megacomplex under which it is registered.
- **megacomplex** (*type[Megacomplex]*) – megacomplex class to be registered.

## set\_megacomplex\_plugin

```
glotaran.plugin_system.megacomplex_registration.set_megacomplex_plugin(megacomplex_name:  
                                str,  
                                full_plugin_name:  
                                str) →  
                                None
```

Set the plugin used for a specific megacomplex name.

This function is useful when you want to resolve conflicts of installed plugins or overwrite the plugin used for a specific megacomplex name.

Effected functions:

- optimize()

### Parameters

- **megacomplex\_name** (*str*) – Name of the megacomplex to use the plugin for.
- **full\_plugin\_name** (*str*) – Full name (import path) of the registered plugin.

## project\_io\_registration

Project Io registration convenience functions.

---

**Note:** The [call-arg] type error would be raised since the base methods doesn't have a **\*\*kwargs** argument, but we rather ignore this error here, than adding **\*\*kwargs** to the base method and causing an [override] type error in the plugins implementation.

---

## Functions

## Summary

<code>get_project_io</code>	Retrieve a data io plugin from the project_io registry.
<code>get_project_io_method</code>	Retrieve implementation of project io functionality for the format 'format_name'.
<code>is_known_project_format</code>	Check if a data format is in the project_io registry.
<code>known_project_formats</code>	Names of the registered project io plugins.
<code>load_model</code>	Create a <code>Model</code> instance from the specs defined in a file.
<code>load_parameters</code>	Create a <code>Parameters</code> instance from the specs defined in a file.
<code>load_result</code>	Create a <code>Result</code> instance from the specs defined in a file.
<code>load_scheme</code>	Create a <code>Scheme</code> instance from the specs defined in a file.
<code>project_io_plugin_table</code>	Return registered project io plugins and which functions they support as markdown table.
<code>register_project_io</code>	Register project io plugins to one or more formats.
<code>save_model</code>	Save a <code>Model</code> instance to a spec file.
<code>save_parameters</code>	Save a <code>Parameters</code> instance to a spec file.
<code>save_result</code>	Write a <code>Result</code> instance to a spec file.
<code>save_scheme</code>	Save a <code>Scheme</code> instance to a spec file.
<code>set_project_plugin</code>	Set the plugin used for a specific project format.
<code>show_project_io_method_help</code>	Show help for the implementation of project io plugin methods.
<code>supported_file_extensions_project_io</code>	Get project io formats that support all methods in <code>method_names</code> .

## get\_project\_io

`glotaran.plugin_system.project_io_registration.get_project_io(format_name: str) → ProjectIoInterface`

Retrieve a data io plugin from the project\_io registry.

### Parameters

**format\_name** (*str*) – Name of the data io plugin under which it is registered.

### Returns

Project io plugin instance.

### Return type

*ProjectIoInterface*

## get\_project\_io\_method

```
glotaran.plugin_system.project_io_registration.get_project_io_method(format_name:
                                                                    str,
                                                                    method_name:
                                                                    ProjectIoMethods)
→ Callable[..., Any]
```

Retrieve implementation of project io functionality for the format ‘format\_name’.

This allows to get the proper help and autocomplete for the function, which is especially valuable if the function provides additional options.

### Parameters

- **format\_name** (*str*) – Format the dataloader should be able to read.
- **method\_name** (*{'load\_model', 'write\_model', 'load\_parameters', 'write\_parameters', 'load\_scheme', 'write\_scheme', 'load\_result', 'write\_result'}*) – Method name, e.g. load\_model.

### Returns

The function which is called in the background by the convenience functions.

### Return type

Callable[..., Any]

## is\_known\_project\_format

```
glotaran.plugin_system.project_io_registration.is_known_project_format(format_name:
                                                                    str) →
                                                                    bool
```

Check if a data format is in the project\_io registry.

### Parameters

**format\_name** (*str*) – Name of the project io plugin under which it is registered.

### Returns

Whether or not the data format is a registered project io plugin.

### Return type

*bool*

## known\_project\_formats

```
glotaran.plugin_system.project_io_registration.known_project_formats(full_names:
                                                                    bool = False)
→ list[str]
```

Names of the registered project io plugins.

### Parameters

**full\_names** (*bool*) – Whether to display the full names the plugins are registered under as well.

**Returns**

List of registered project io plugins.

**Return type**

`list[str]`

**load\_model**

```
glotaran.plugin_system.project_io_registration.load_model(file_name: StrOrPath,
                                                         format_name: str | None =
                                                         None, **kwargs: Any) →
                                                         Model
```

Create a Model instance from the specs defined in a file.

**Parameters**

- **file\_name** (*StrOrPath*) – File containing the model specs.
- **format\_name** (*str*) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `load_model` implementation of the project io plugin.

**Returns**

Model instance created from the file.

**Return type**

*Model*

**load\_parameters**

```
glotaran.plugin_system.project_io_registration.load_parameters(file_name: StrOrPath,
                                                             format_name: str |
                                                             None = None,
                                                             **kwargs) →
                                                             Parameters
```

Create a Parameters instance from the specs defined in a file.

**Parameters**

- **file\_name** (*StrOrPath*) – File containing the parameter specs.
- **format\_name** (*str* | *None*) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `load_parameters` implementation of the project io plugin.

**Returns**

- *Parameters* – Parameters instance created from the file.
- .. # noqa (D414)

## load\_result

```
glotaran.plugin_system.project_io_registration.load_result(result_path: StrOrPath,  
                                                          format_name: str | None =  
                                                          None, **kwargs: Any) →  
                                                          Result
```

Create a `Result` instance from the specs defined in a file.

### Parameters

- **result\_path** (*StrOrPath*) – Path containing the result data.
- **format\_name** (*str* | *None*) – Format the result is in, if not provided and it is a file it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `load_result` implementation of the project io plugin.

### Returns

`Result` instance created from the saved format.

### Return type

*Result*

## load\_scheme

```
glotaran.plugin_system.project_io_registration.load_scheme(file_name: StrOrPath,  
                                                          format_name: str | None =  
                                                          None, **kwargs: Any) →  
                                                          Scheme
```

Create a `Scheme` instance from the specs defined in a file.

### Parameters

- **file\_name** (*StrOrPath*) – File containing the parameter specs.
- **format\_name** (*str* | *None*) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `load_scheme` implementation of the project io plugin.

### Returns

`Scheme` instance created from the file.

### Return type

*Scheme*

## project\_io\_plugin\_table

```
glotaran.plugin_system.project_io_registration.project_io_plugin_table(*, plu-
    gin_names:
        bool =
        False,
    full_names:
        bool =
        False) →
    Markdown-
    Str
```

Return registered project io plugins and which functions they support as markdown table.

This is especially useful when you work with new plugins.

### Parameters

- **plugin\_names** (*bool*) – Whether or not to add the names of the plugins to the table.
- **full\_names** (*bool*) – Whether to display the full names the plugins are registered under as well.

### Returns

Markdown table of project io plugins.

### Return type

*MarkdownStr*

## register\_project\_io

```
glotaran.plugin_system.project_io_registration.register_project_io(format_names:
    str | list[str]) →
    Callable[[type[ProjectIoInterface]],
    type[ProjectIoInterface]]
```

Register project io plugins to one or more formats.

Decorate a project io plugin class with `@register_project_io(format_name | [*format_names])` to add it to the registry.

### Parameters

**format\_names** (*str* | *list[str]*) – Name of the project io plugin under which it is registered.

### Returns

Inner decorator function.

### Return type

`Callable[[type[ProjectIoInterface]], type[ProjectIoInterface]]`

## Examples

```
>>> @register_project_io("my_format_1")
... class MyProjectIo1(ProjectIoInterface):
...     pass
```

```
>>> @register_project_io(["my_format_1", "my_format_1_alias"])
... class MyProjectIo2(ProjectIoInterface):
...     pass
```

## save\_model

glotaran.plugin\_system.project\_io\_registration.**save\_model**(*model*: [Model](#), *file\_name*: *StrOrPath*, *format\_name*: *str* | *None* = *None*, \*, *allow\_overwrite*: *bool* = *False*, *update\_source\_path*: *bool* = *True*, *\*\*kwargs*: *Any*) → *None*

Save a `Model` instance to a spec file.

### Parameters

- **model** ([Model](#)) – `Model` instance to save to specs file.
- **file\_name** (*StrOrPath*) – File to write the model specs to.
- **format\_name** (*str* | *None*) – Format the file should be in, if not provided it will be inferred from the file extension.
- **allow\_overwrite** (*bool*) – Whether or not to allow overwriting existing files, by default `False`
- **update\_source\_path** (*bool*) – Whether or not to update the `source_path` attribute to `file_name` when saving. by default `True`
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `save_model` implementation of the project io plugin.

## save\_parameters

glotaran.plugin\_system.project\_io\_registration.**save\_parameters**(*parameters*: [Parameters](#), *file\_name*: *StrOrPath*, *format\_name*: *str* | *None* = *None*, \*, *allow\_overwrite*: *bool* = *False*, *update\_source\_path*: *bool* = *True*, *\*\*kwargs*: *Any*) → *None*

Save a `Parameters` instance to a spec file.



### Parameters

- **parameters** ([Parameters](#)) – Parameters instance to save to specs file.
- **file\_name** (*StrOrPath*) – File to write the parameter specs to.
- **format\_name** (*str* / *None*) – Format the file should be in, if not provided it will be inferred from the file extension.
- **allow\_overwrite** (*bool*) – Whether or not to allow overwriting existing files, by default False
- **update\_source\_path** (*bool*) – Whether or not to update the `source_path` attribute to `file_name` when saving. by default True
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `save_parameters` implementation of the project io plugin.

### save\_result

```
glotaran.plugin_system.project_io_registration.save_result(result: Result, result_path:
    StrOrPath, format_name:
    str | None = None, *,
    allow_overwrite: bool =
    False,
    update_source_path: bool
    = True, saving_options:
    SavingOptions =
    SavingOp-
    tions(data_filter=None,
    data_format='nc',
    parameter_format='csv',
    report=True), **kwargs:
    Any) → list[str]
```

Write a `Result` instance to a spec file.

### Parameters

- **result** ([Result](#)) – `Result` instance to write.
- **result\_path** (*StrOrPath*) – Path to write the result data to.
- **format\_name** (*str* / *None*) – Format the result should be saved in, if not provided and it is a file it will be inferred from the file extension.
- **allow\_overwrite** (*bool*) – Whether or not to allow overwriting existing files, by default False
- **update\_source\_path** (*bool*) – Whether or not to update the `source_path` attribute to `result_path` when saving. by default True
- **saving\_options** ([SavingOptions](#)) – Options for the saved result.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `save_result` implementation of the project io plugin.

### Returns

List of file paths which were saved.

### Return type

*list*[*str*] | *None*

## save\_scheme

```
glotaran.plugin_system.project_io_registration.save_scheme(scheme: Scheme,  
                                                         file_name: StrOrPath,  
                                                         format_name: str | None =  
                                                         None, *, allow_overwrite:  
                                                         bool = False,  
                                                         update_source_path: bool  
                                                         = True, **kwargs: Any)  
                                                         → None
```

Save a Scheme instance to a spec file.

### Parameters

- **scheme** ([Scheme](#)) – Scheme instance to save to specs file.
- **file\_name** ([StrOrPath](#)) – File to write the scheme specs to.
- **format\_name** ([str](#) | [None](#)) – Format the file should be in, if not provided it will be inferred from the file extension.
- **allow\_overwrite** ([bool](#)) – Whether or not to allow overwriting existing files, by default False
- **update\_source\_path** ([bool](#)) – Whether or not to update the `source_path` attribute to `file_name` when saving. by default True
- **\*\*kwargs** ([Any](#)) – Additional keyword arguments passes to the `save_scheme` implementation of the project io plugin.

## set\_project\_plugin

```
glotaran.plugin_system.project_io_registration.set_project_plugin(format_name: str,  
                                                                  full_plugin_name:  
                                                                  str) → None
```

Set the plugin used for a specific project format.

This function is useful when you want to resolve conflicts of installed plugins or overwrite the plugin used for a specific format.

Effected functions:

- [load\\_model\(\)](#)
- [save\\_model\(\)](#)
- [load\\_parameters\(\)](#)
- [save\\_parameters\(\)](#)
- [load\\_scheme\(\)](#)
- [save\\_scheme\(\)](#)
- [load\\_result\(\)](#)
- [save\\_result\(\)](#)

### Parameters

- **format\_name** (*str*) – Format name used to refer to the plugin when used for save and load functions.
- **full\_plugin\_name** (*str*) – Full name (import path) of the registered plugin.

### show\_project\_io\_method\_help

```
glotaran.plugin_system.project_io_registration.show_project_io_method_help(format_name:
                                                                            str,
                                                                            method_name:
                                                                            Pro-
                                                                            jec-
                                                                            tlIoMeth-
                                                                            ods)
                                                                            →
                                                                            None
```

Show help for the implementation of project io plugin methods.

#### Parameters

- **format\_name** (*str*) – Format the method should support.
- **method\_name** (*{'load\_model', 'write\_model', 'load\_parameters', 'write\_parameters', 'load\_scheme', 'write\_scheme', 'load\_result', 'write\_result'}*) – Method name.

### supported\_file\_extensions\_project\_io

```
glotaran.plugin_system.project_io_registration.supported_file_extensions_project_io(method_names:
                                                                                    str
                                                                                    |
                                                                                    Se-
                                                                                    quence[str])
                                                                                    →
                                                                                    Gen-
                                                                                    er-
                                                                                    a-
                                                                                    tor[str,
                                                                                    None,
                                                                                    None]
```

Get project io formats that support all methods in *method\_names*.

#### Parameters

**method\_names** (*str* | *Sequence[str]*) – Names of Methods that need to support the file extension.

#### Yields

*Generator[str, None, None]* – File extension supported by all methods in *method\_names*.

See also:

supported\_file\_extensions, PROJECT\_IO\_METHODS

### 16.1.10 project

The glotaran project package.

#### Modules

<code>glotaran.project.dataclass_helpers</code>	Contains helper methods for dataclasses.
<code>glotaran.project.generators</code>	The glotaran generator package.
<code>glotaran.project.project</code>	The glotaran project module.
<code>glotaran.project.project_data_registry</code>	The glotaran data registry module.
<code>glotaran.project.project_model_registry</code>	The glotaran model registry module.
<code>glotaran.project.project_parameter_registry</code>	The glotaran parameter registry module.
<code>glotaran.project.project_registry</code>	The glotaran registry module.
<code>glotaran.project.project_result_registry</code>	The glotaran result registry module.
<code>glotaran.project.result</code>	The result class for global analysis.
<code>glotaran.project.scheme</code>	The module for :class:Scheme.

#### dataclass\_helpers

Contains helper methods for dataclasses.

#### Functions

##### Summary

<code>asdict</code>	Create a dictionary containing all fields of the dataclass.
<code>exclude_from_dict_field</code>	Create a dataclass field with which will be excluded from asdict.
<code>file_loadable_field</code>	Create a dataclass field which can be an object of type <code>targetClass</code> or file path.
<code>file_loader_factory</code>	Create <code>file_loader</code> functions to load <code>targetClass</code> from file.
<code>fromdict</code>	Create a dataclass instance from a dict and loads all file represented fields.
<code>init_file_loadable_fields</code>	Load objects into class when dataclass is initialized with paths.

#### asdict

`glotaran.project.dataclass_helpers.asdict(dataclass: DataclassInstance, folder: Path | None = None) → dict[str, Any]`

Create a dictionary containing all fields of the dataclass.

##### Parameters

- **dataclass** (`DataclassInstance`) – A dataclass instance.

- **folder** (*Path* / *None*) – Parent folder of `FileLoadable` fields. by default `None`

**Returns**

The dataclass represented as a dictionary.

**Return type**

`dict[str, Any]`

**exclude\_from\_dict\_field**

```
glotaran.project.dataclass_helpers.exclude_from_dict_field(default: DefaultType =
                                                         <data-
                                                         classes._MISSING_TYPE
                                                         object>) → DefaultType
```

Create a dataclass field with which will be excluded from `asdict`.

**Parameters**

**default** (*DefaultType*) – The default value of the field.

**Returns**

The created field.

**Return type**

`DefaultType`

**file\_loadable\_field**

```
glotaran.project.dataclass_helpers.file_loadable_field(targetClass:
                                                         type[FileLoadable], *,
                                                         is_wrapper_class=False) →
                                                         FileLoadable
```

Create a dataclass field which can be and object of type `targetClass` or file path.

**Parameters**

- **targetClass** (*type[FileLoadable]*) – Class the resulting value should be an instance of.
- **is\_wrapper\_class** (*bool*) – Whether or not `targetClass` is a wrapper class, so the `isinstance` check will be ignored and instead the responsibility for supported types lies at the implementation of the loader.

**Notes**

This also requires to add `init_file_loadable_fields` in the `__post_init__` method.

**Returns**

Instance of `targetClass`.

**Return type**

`FileLoadable`

**See also:**

`init_file_loadable_fields`

## file\_loader\_factory

```
glotaran.project.dataclass_helpers.file_loader_factory(targetClass:
                                                         type[FileLoadable], *,
                                                         is_wrapper_class: bool =
                                                         False) →
                                                         Callable[[FileLoadable | str |
                                                         Path], FileLoadable]
```

Create `file_loader` functions to load `targetClass` from file.

### Parameters

- **targetClass** (`type[FileLoadable]`) – Class the loader function should return an instance of.
- **is\_wrapper\_class** (`bool`) – Whether or not `targetClass` is a wrapper class, so the `isinstance` check will be ignored and instead the responsibility for supported types lies at the implementation of the loader.

### Returns

**file\_loader** – Function to load `FileLoadable` from source file or return instance if already loaded.

### Return type

`Callable[[FileLoadable | str | Path], FileLoadable]`

## fromdict

```
glotaran.project.dataclass_helpers.fromdict(dataclass_type: type[DataclassInstanceType],
                                             dataclass_dict: dict[str, Any], folder: Path |
                                             None = None) → DataclassInstanceType
```

Create a dataclass instance from a dict and loads all file represented fields.

### Parameters

- **dataclass\_type** (`type[DataclassInstanceType]`) – A dataclass type.
- **dataclass\_dict** (`dict[str, Any]`) – A dict for instancing the the dataclass.
- **folder** (`Path`) – The root folder for file paths. If `None` file paths are consider absolute.

### Returns

Created instance of `dataclass_type`.

### Return type

`DataclassInstanceType`

## init\_file\_loadable\_fields

`glotaran.project.dataclass_helpers.init_file_loadable_fields(dataclass_instance:  
DataclassInstance)`

Load objects into class when dataclass is initialized with paths.

If the class has `file_loadable` fields, this needs be called in the `__post_init__` method of that class.

### Parameters

**`dataclass_instance`** (*DataclassInstance*) – Instance of the dataclass being initialized. When used inside of `__post_init__` for the class itself use `self`.

**See also:**

[\*file\\_loadable\\_field\*](#)

## generators

The glotaran generator package.

## Modules

<a href="#"><i>glotaran.project.generators.generator</i></a>	The glotaran generator module.
--	--------------------------------

## generator

The glotaran generator module.

## Functions

### Summary

<a href="#"><i>generate_model</i></a>	Generate a model.
<a href="#"><i>generate_model_yaml</i></a>	Generate a model as yaml string.
<a href="#"><i>generate_parallel_decay_model</i></a>	Generate a parallel decay model dictionary.
<a href="#"><i>generate_parallel_spectral_decay_model</i></a>	Generate a parallel spectral decay model dictionary.
<a href="#"><i>generate_sequential_decay_model</i></a>	Generate a sequential decay model dictionary.
<a href="#"><i>generate_sequential_spectral_decay_model</i></a>	Generate a sequential spectral decay model dictionary.

## generate\_model

```
glotaran.project.generators.generator.generate_model(*, generator_name: str,  
                                                    generator_arguments:  
                                                    GeneratorArguments) → Model
```

Generate a model.

### Parameters

- **generator\_name** (*str*) – The generator to use.
- **generator\_arguments** (*GeneratorArguments*) – Arguments for the generator.

### Returns

The generated model

### Return type

*Model*

### See also:

*generate\_parallel\_decay\_model*, *generate\_parallel\_spectral\_decay\_model*,  
*generate\_sequential\_decay\_model*, *generate\_sequential\_spectral\_decay\_model*

### Raises

**ValueError** – Raised when an unknown generator is specified.

## generate\_model\_yaml

```
glotaran.project.generators.generator.generate_model_yaml(*, generator_name: str,  
                                                         generator_arguments:  
                                                         GeneratorArguments) → str
```

Generate a model as yaml string.

### Parameters

- **generator\_name** (*str*) – The generator to use.
- **generator\_arguments** (*GeneratorArguments*) – Arguments for the generator.

### Returns

The generated model yaml string.

### Return type

*str*

### See also:

*generate\_parallel\_decay\_model*, *generate\_parallel\_spectral\_decay\_model*,  
*generate\_sequential\_decay\_model*, *generate\_sequential\_spectral\_decay\_model*

### Raises

**ValueError** – Raised when an unknown generator is specified.



## generate\_parallel\_decay\_model

```
glotaran.project.generators.generator.generate_parallel_decay_model(*,
                                                                    nr_compartments:
                                                                    int = 1, irf:
                                                                    bool = False)
                                                                    → dict[str,
                                                                    Any]
```

Generate a parallel decay model dictionary.

### Parameters

- **nr\_compartments** (*int*) – The number of compartments.
- **irf** (*bool*) – Whether to add a gaussian irf.

### Returns

The generated model dictionary.

### Return type

`dict[str, Any]`

## generate\_parallel\_spectral\_decay\_model

```
glotaran.project.generators.generator.generate_parallel_spectral_decay_model(*,
                                                                              nr_compartments:
                                                                              int
                                                                              =
                                                                              1,
                                                                              irf:
                                                                              bool
                                                                              =
                                                                              False)
                                                                              →
                                                                              dict[str,
                                                                              Any]
```

Generate a parallel spectral decay model dictionary.

### Parameters

- **nr\_compartments** (*int*) – The number of compartments.
- **irf** (*bool*) – Whether to add a gaussian irf.

### Returns

The generated model dictionary.

### Return type

`dict[str, Any]`

### generate\_sequential\_decay\_model

```
glotaran.project.generators.generator.generate_sequential_decay_model(nr_compartments:  
    int = 1, irf:  
    bool =  
    False) →  
    dict[str,  
    Any]
```

Generate a sequential decay model dictionary.

#### Parameters

- **nr\_compartments** (*int*) – The number of compartments.
- **irf** (*bool*) – Whether to add a gaussian irf.

#### Returns

The generated model dictionary.

#### Return type

`dict[str, Any]`

### generate\_sequential\_spectral\_decay\_model

```
glotaran.project.generators.generator.generate_sequential_spectral_decay_model(*,  
    nr_compartments:  
    int  
    =  
    1,  
    irf:  
    bool  
    =  
    False)  
    →  
    dict[str,  
    Any]
```

Generate a sequential spectral decay model dictionary.

#### Parameters

- **nr\_compartments** (*int*) – The number of compartments.
- **irf** (*bool*) – Whether to add a gaussian irf.

#### Returns

The generated model dictionary.

#### Return type

`dict[str, Any]`

## Classes

### Summary

<i>GeneratorArguments</i>	Arguments used by <code>generate_model</code> and <code>generate_model</code> .
---------------------------	---

### GeneratorArguments

**class** `glotaran.project.generators.generator.GeneratorArguments`

Bases: `TypedDict`

Arguments used by `generate_model` and `generate_model`.

#### Parameters

- **`nr_compartments`** (*int*) – The number of compartments.
- **`irf`** (*bool*) – Whether to add a gaussian irf.

See also:

*`generate_model`*, *`generate_model.yml`*

### Attributes Summary

*`nr_compartments`*

*`irf`*

### `nr_compartments`

`GeneratorArguments.nr_compartments`: *int*

### `irf`

`GeneratorArguments.irf`: *bool*

## Methods Summary

<i>clear</i>	
<i>copy</i>	
<i>fromkeys</i>	Create a new dictionary with keys from iterable and values set to value.
<i>get</i>	Return the value for key if key is in the dictionary, else default.
<i>items</i>	
<i>keys</i>	
<i>pop</i>	If the key is not found, return the default if given; otherwise, raise a KeyError.
<i>popitem</i>	Remove and return a (key, value) pair as a 2-tuple.
<i>setdefault</i>	Insert key with a value of default if key is not in the dictionary.
<i>update</i>	If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
<i>values</i>	

### clear

`GeneratorArguments.clear()` → None. Remove all items from D.

### copy

`GeneratorArguments.copy()` → a shallow copy of D

### fromkeys

`GeneratorArguments.fromkeys(iterable, value=None, /)`

Create a new dictionary with keys from iterable and values set to value.

### get

`GeneratorArguments.get(key, default=None, /)`

Return the value for key if key is in the dictionary, else default.

### items

`GeneratorArguments.items()` → a set-like object providing a view on D's items

### keys

`GeneratorArguments.keys()` → a set-like object providing a view on D's keys

### pop

`GeneratorArguments.pop(key, default=<unrepresentable>, /)`

If the key is not found, return the default if given; otherwise, raise a `KeyError`.

### popitem

`GeneratorArguments.popitem(/)`

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises `KeyError` if the dict is empty.

### setdefault

`GeneratorArguments.setdefault(key, default=None, /)`

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

### update

`GeneratorArguments.update([E, ]**F)` → None. Update D from dict/iterable E and F.

If E is present and has a `.keys()` method, then does: for k in E: D[k] = E[k] If E is present and lacks a `.keys()` method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

## values

`GeneratorArguments.values()` → an object providing a view on D's values

## Methods Documentation

**clear()** → None. Remove all items from D.

**copy()** → a shallow copy of D

**fromkeys**(*iterable*, *value=None*, /)

Create a new dictionary with keys from iterable and values set to value.

**get**(*key*, *default=None*, /)

Return the value for key if key is in the dictionary, else default.

**irf**: **bool**

**items()** → a set-like object providing a view on D's items

**keys()** → a set-like object providing a view on D's keys

**nr\_compartments**: **int**

**pop**(*key*, *default=<unrepresentable>*, /)

If the key is not found, return the default if given; otherwise, raise a `KeyError`.

**popitem**(/)

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises `KeyError` if the dict is empty.

**setdefault**(*key*, *default=None*, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

**update**([*E*, ]\*\**F*) → None. Update D from dict/iterable E and F.

If E is present and has a `.keys()` method, then does: for k in E: D[k] = E[k] If E is present and lacks a `.keys()` method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

**values()** → an object providing a view on D's values

## project

The glotaran project module.

## Classes

### Summary

<i>Project</i>	A project represents a projectfolder on disk which contains a project file.
----------------	---

### Project

**class** `glotaran.project.project.Project`(*file*: *Path*, *folder*: *Path* | *None* = *None*)

Bases: `object`

A project represents a projectfolder on disk which contains a project file.

A project file is a file in *yml* format with name *project.gta*

### Attributes Summary

<i>data</i>	Get all project datasets.
<i>folder</i>	
<i>has_data</i>	Check if the project has datasets.
<i>has_models</i>	Check if the project has models.
<i>has_parameters</i>	Check if the project has parameters.
<i>has_results</i>	Check if the project has results.
<i>models</i>	Get all project models.
<i>parameters</i>	Get all project parameters.
<i>results</i>	Get all project results.
<i>version</i>	
<i>file</i>	

### data

**Project.data**

Get all project datasets.

#### Returns

The models of the datasets.

#### Return type

Mapping[*str*, *Path*]

## **folder**

`Project.folder: Path = None`

## **has\_data**

`Project.has_data`

Check if the project has datasets.

### **Returns**

Whether the project has datasets.

### **Return type**

`bool`

## **has\_models**

`Project.has_models`

Check if the project has models.

### **Returns**

Whether the project has models.

### **Return type**

`bool`

## **has\_parameters**

`Project.has_parameters`

Check if the project has parameters.

### **Returns**

Whether the project has parameters.

### **Return type**

`bool`

## **has\_results**

`Project.has_results`

Check if the project has results.

### **Returns**

Whether the project has results.

### **Return type**

`bool`



## models

### Project.models

Get all project models.

#### Returns

The models of the project.

#### Return type

Mapping[str, Path]

## parameters

### Project.parameters

Get all project parameters.

#### Returns

The parameters of the project.

#### Return type

Mapping[str, Path]

## results

### Project.results

Get all project results.

#### Returns

The results of the project.

#### Return type

Mapping[str, Path]

## version

Project.version: str

## file

Project.file: Path

## Methods Summary

<code>create</code>	Create a new project folder and file.
<code>create_scheme</code>	Create a scheme for optimization.
<code>generate_model</code>	Generate a model.
<code>generate_parameters</code>	Generate parameters for a model.
<code>get_latest_result_path</code>	Get the path to a result with name <code>name</code> .
<code>get_models_directory</code>	Get the path to the model directory of the project.
<code>get_parameters_directory</code>	Get the path to the parameter directory of the project.
<code>get_result_path</code>	Get the path to a result with name <code>name</code> .
<code>import_data</code>	Import a dataset by saving it as an <code>.nc</code> file in the project's data folder.
<code>load_data</code>	Load a dataset, with SVD data if <code>add_svd</code> is <code>True</code> .
<code>load_latest_result</code>	Load a result.
<code>load_model</code>	Load a model.
<code>load_parameters</code>	Load parameters.
<code>load_result</code>	Load a result.
<code>markdown</code>	Format the project as a markdown text.
<code>open</code>	Open a new project.
<code>optimize</code>	Optimize a model.
<code>show_model_definition</code>	Show model definition file content with syntax highlighting.
<code>show_parameters_definition</code>	Show parameters definition file content with syntax highlighting.
<code>validate</code>	Check that the model is valid, list all issues in the model if there are any.

### create

**static** `Project.create(folder: str | Path, allow_overwrite: bool = False) → Project`

Create a new project folder and file.

#### Parameters

- **folder** (`str` | `Path` | `None`) – The folder where the project will be created. If `None`, the current work directory will be used.
- **allow\_overwrite** (`bool`) – Whether to overwrite an existing project file.

#### Returns

The created project.

#### Return type

`Project`

#### Raises

**FileExistsError** – Raised if the project file already exists and `allow_overwrite=False`.

## create\_scheme

`Project.create_scheme(model_name: str, parameters_name: str, maximum_number_function_evaluations: int | None = None, clp_link_tolerance: float = 0.0, data_lookup_override: Mapping[str, LoadableDataset] | None = None) → Scheme`

Create a scheme for optimization.

### Parameters

- **model\_name** (*str*) – The model to optimize.
- **parameters\_name** (*str*) – The initial parameters.
- **maximum\_number\_function\_evaluations** (*int* | *None*) – The maximum number of function evaluations.
- **clp\_link\_tolerance** (*float*) – The CLP link tolerance.
- **data\_lookup\_override** (*Mapping[str, LoadableDataset]* | *None*) – Allows to bypass the default dataset lookup in the project data folder and use a different dataset for the optimization without changing the model. This is especially useful when working with preprocessed data. Defaults to *None*.

### Returns

The created scheme.

### Return type

*Scheme*

## generate\_model

`Project.generate_model(model_name: str, generator_name: str, generator_arguments: dict[str, Any], *, allow_overwrite: bool = False, ignore_existing: bool = False)`

Generate a model.

### Parameters

- **model\_name** (*str*) – The name of the model.
- **generator\_name** (*str*) – The generator for the model.
- **generator\_arguments** (*dict[str, Any]*) – Arguments for the generator.
- **allow\_overwrite** (*bool*) – Whether to overwrite an existing model.
- **ignore\_existing** (*bool*) – Whether to ignore generation of a model file if it already exists.

### generate\_parameters

```
Project.generate_parameters(model_name: str, parameters_name: str | None = None, *,
                           format_name: Literal['yaml', 'yaml', 'csv'] = 'csv',
                           allow_overwrite: bool = False, ignore_existing: bool =
                           False)
```

Generate parameters for a model.

#### Parameters

- **model\_name** (*str*) – The model.
- **parameters\_name** (*str* | *None*) – The name of the parameters. If *None* it will be <model\_name>\_parameters.
- **format\_name** (*Literal*["yaml", "yaml", "csv"]) – The parameter format.
- **allow\_overwrite** (*bool*) – Whether to overwrite existing parameters.
- **ignore\_existing** (*bool*) – Whether to ignore generation of a parameter file if it already exists.

### get\_latest\_result\_path

```
Project.get_latest_result_path(result_name: str) → Path
```

Get the path to a result with name *name*.

#### Parameters

**result\_name** (*str*) – The name of the result.

#### Returns

The path to the result.

#### Return type

Path

#### Raises

**ValueError** – Raised if result does not exist.

### get\_models\_directory

```
Project.get_models_directory() → Path
```

Get the path to the model directory of the project.

#### Returns

The path to the project's model directory.

#### Return type

Path

### get\_parameters\_directory

Project.get\_parameters\_directory() → Path

Get the path to the parameter directory of the project.

#### Returns

The path to the project's parameter directory.

#### Return type

Path

### get\_result\_path

Project.get\_result\_path(result\_name: str, \*, latest: bool = False) → Path

Get the path to a result with name name.

#### Parameters

- **result\_name** (str) – The name of the result.
- **latest** (bool) – Flag to deactivate warning about using latest result. Defaults to False

#### Returns

The path to the result.

#### Return type

Path

#### Raises

**ValueError** – Raised if result does not exist.

### import\_data

Project.import\_data(dataset: LoadableDataset | Mapping[str, LoadableDataset],  
dataset\_name: str | None = None, allow\_overwrite: bool = False,  
ignore\_existing: bool = True)

Import a dataset by saving it as an .nc file in the project's data folder.

#### Parameters

- **dataset** (LoadableDataset) – Dataset instance or path to a dataset.
- **dataset\_name** (str | None) – The name of the dataset (needs to be provided when dataset is an xarray instance). Defaults to None.
- **allow\_overwrite** (bool) – Whether to overwrite an existing dataset.
- **ignore\_existing** (bool) – Whether to skip import if the dataset already exists and allow\_overwrite is False. Defaults to True.

## load\_data

`Project.load_data(dataset_name: str, *, add_svd: bool = False, lsv_dim: Hashable = 'time',  
rsv_dim: Hashable = 'spectral') → xr.Dataset`

Load a dataset, with SVD data if `add_svd` is `True`.

### Parameters

- **dataset\_name** (*str*) – The name of the dataset.
- **add\_svd** (*bool*) – Whether or not to calculate and add SVD data. Defaults to `False`.
- **lsv\_dim** (*Hashable*) – Dimension of the left singular vectors. Defaults to “time”.
- **rsv\_dim** (*Hashable*) – Dimension of the right singular vectors. Defaults to “spectral”,

### Returns

The loaded dataset, with SVD data if `add_svd` is `True`.

### Return type

`xr.Dataset`

### Raises

**ValueError** – Raised if the dataset does not exist.

## load\_latest\_result

`Project.load_latest_result(result_name: str) → Result`

Load a result.

### Parameters

**result\_name** (*str*) – The name of the result.

### Returns

The loaded result.

### Return type

*Result*

### Raises

**ValueError** – Raised if result does not exist.

## load\_model

`Project.load_model(model_name: str) → Model`

Load a model.

### Parameters

**model\_name** (*str*) – The name of the model.

### Returns

The loaded model.

### Return type

*Model*

**Raises**

**ValueError** – Raised if the model does not exist.

**load\_parameters**

`Project.load_parameters(parameters_name: str) → Parameters`

Load parameters.

**Parameters**

**parameters\_name** (*str*) – The name of the parameters.

**Raises**

**ValueError** – Raised if parameters do not exist.

**Returns**

The loaded parameters.

**Return type**

*Parameters*

**load\_result**

`Project.load_result(result_name: str, *, latest: bool = False) → Result`

Load a result.

**Parameters**

- **result\_name** (*str*) – The name of the result.
- **latest** (*bool*) – Flag to deactivate warning about using latest result. Defaults to False

**Returns**

The loaded result.

**Return type**

*Result*

**Raises**

**ValueError** – Raised if result does not exist.

**markdown**

`Project.markdown() → MarkdownStr`

Format the project as a markdown text.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

*str*

## open

**classmethod** `Project.open(project_folder_or_file: str | Path, create_if_not_exist: bool = True) → Project`

Open a new project.

### Parameters

- **project\_folder\_or\_file** (*str* | *Path*) – The path to a project folder or file.
- **create\_if\_not\_exist** (*bool*) – Create the project if not existent.

### Returns

The project instance.

### Return type

*Project*

### Raises

**FileNotFoundError** – Raised when the project file does not exist and *create\_if\_not\_exist* is *False*.

## optimize

`Project.optimize(model_name: str, parameters_name: str, result_name: str | None = None, maximum_number_function_evaluations: int | None = None, clp_link_tolerance: float = 0.0, data_lookup_override: Mapping[str, LoadableDataset] | None = None) → Result`

Optimize a model.

### Parameters

- **model\_name** (*str*) – The model to optimize.
- **parameters\_name** (*str*) – The initial parameters.
- **result\_name** (*str* | *None*) – The name of the result.
- **maximum\_number\_function\_evaluations** (*int* | *None*) – The maximum number of function evaluations.
- **clp\_link\_tolerance** (*float*) – The CLP link tolerance.
- **data\_lookup\_override** (*Mapping*[*str*, *LoadableDataset*] | *None*) – Allows to bypass the default dataset lookup in the project data folder and use a different dataset for the optimization without changing the model. This is especially useful when working with preprocessed data. Defaults to *None*.

### Returns

Result of the optimization.

### Return type

*Result*



## show\_model\_definition

`Project.show_model_definition(model_name: str, syntax: str | None = None) → MarkdownStr`

Show model definition file content with syntax highlighting.

### Parameters

- **model\_name** (*str*) – The name of the model.
- **syntax** (*str* | *None*) – Syntax used for syntax highlighting. Defaults to *None* which means that the syntax is inferred based on the file extension. Pass the value *""* to deactivate syntax highlighting.

### Returns

Model definition file content with syntax highlighting to render in ipython.

### Return type

*MarkdownStr*

## show\_parameters\_definition

`Project.show_parameters_definition(parameters_name: str, syntax: str | None = None, *, as_dataframe: bool | None = None) → MarkdownStr | DataFrame`

Show parameters definition file content with syntax highlighting.

### Parameters

- **parameters\_name** (*str*) – The name of the parameters.
- **syntax** (*str* | *None*) – Syntax used for syntax highlighting. Defaults to *None* which means that the syntax is inferred based on the file extension. Pass the value *""* to deactivate syntax highlighting.
- **as\_dataframe** (*bool* | *None*) – Whether or not to show the Parameters definition as *pandas.DataFrame* (mostly useful for non string formats). Defaults to *None* which means that it will be inferred to *True* for known non string formats like *xlsx*.

### Returns

Parameters definition file content with syntax highlighting to render in ipython.

### Return type

*MarkdownStr* | *pd.DataFrame*

## validate

`Project.validate(model_name: str, parameters_name: str | None = None) → MarkdownStr`

Check that the model is valid, list all issues in the model if there are any.

If *parameters\_name* also consider the Parameters when validating.

### Parameters

- **model\_name** (*str*) – The name of the model to validate.

- **parameters\_name** (*str* | *None*) – The name of the parameters to use when validating. Defaults to *None* which means that parameters are not considered when validating the model.

**Returns**

Text indicating if the model is valid or not.

**Return type**

*MarkdownStr*

## Methods Documentation

**static create**(*folder*: *str* | *Path*, *allow\_overwrite*: *bool* = *False*) → *Project*

Create a new project folder and file.

**Parameters**

- **folder** (*str* | *Path* | *None*) – The folder where the project will be created. If *None*, the current work directory will be used.
- **allow\_overwrite** (*bool*) – Whether to overwrite an existing project file.

**Returns**

The created project.

**Return type**

*Project*

**Raises**

**FileExistsError** – Raised if the project file already exists and *allow\_overwrite=False*.

**create\_scheme**(*model\_name*: *str*, *parameters\_name*: *str*,  
                  *maximum\_number\_function\_evaluations*: *int* | *None* = *None*,  
                  *clp\_link\_tolerance*: *float* = 0.0, *data\_lookup\_override*: *Mapping*[*str*,  
  *LoadableDataset*] | *None* = *None*) → *Scheme*

Create a scheme for optimization.

**Parameters**

- **model\_name** (*str*) – The model to optimize.
- **parameters\_name** (*str*) – The initial parameters.
- **maximum\_number\_function\_evaluations** (*int* | *None*) – The maximum number of function evaluations.
- **clp\_link\_tolerance** (*float*) – The CLP link tolerance.
- **data\_lookup\_override** (*Mapping*[*str*, *LoadableDataset*] | *None*) – Allows to bypass the default dataset lookup in the project data folder and use a different dataset for the optimization without changing the model. This is especially useful when working with preprocessed data. Defaults to *None*.

**Returns**

The created scheme.

**Return type**

*Scheme*

**property data:** `Mapping[str, Path]`

Get all project datasets.

**Returns**

The models of the datasets.

**Return type**

`Mapping[str, Path]`

**file:** `Path`

**folder:** `Path = None`

**generate\_model**(*model\_name: str, generator\_name: str, generator\_arguments: dict[str, Any], \*, allow\_overwrite: bool = False, ignore\_existing: bool = False*)

Generate a model.

**Parameters**

- **model\_name** (*str*) – The name of the model.
- **generator\_name** (*str*) – The generator for the model.
- **generator\_arguments** (*dict[str, Any]*) – Arguments for the generator.
- **allow\_overwrite** (*bool*) – Whether to overwrite an existing model.
- **ignore\_existing** (*bool*) – Whether to ignore generation of a model file if it already exists.

**generate\_parameters**(*model\_name: str, parameters\_name: str | None = None, \*, format\_name: Literal['yaml', 'yml', 'csv'] = 'csv', allow\_overwrite: bool = False, ignore\_existing: bool = False*)

Generate parameters for a model.

**Parameters**

- **model\_name** (*str*) – The model.
- **parameters\_name** (*str | None*) – The name of the parameters. If `None` it will be `<model_name>_parameters`.
- **format\_name** (*Literal["yaml", "yml", "csv"]*) – The parameter format.
- **allow\_overwrite** (*bool*) – Whether to overwrite existing parameters.
- **ignore\_existing** (*bool*) – Whether to ignore generation of a parameter file if it already exists.

**get\_latest\_result\_path**(*result\_name: str*) → `Path`

Get the path to a result with name `name`.

**Parameters**

**result\_name** (*str*) – The name of the result.

**Returns**

The path to the result.

**Return type**

`Path`

**Raises**

**ValueError** – Raised if result does not exist.

**get\_models\_directory()** → [Path](#)

Get the path to the model directory of the project.

**Returns**

The path to the project's model directory.

**Return type**

[Path](#)

**get\_parameters\_directory()** → [Path](#)

Get the path to the parameter directory of the project.

**Returns**

The path to the project's parameter directory.

**Return type**

[Path](#)

**get\_result\_path(result\_name: [str](#), \*, latest: [bool](#) = False)** → [Path](#)

Get the path to a result with name name.

**Parameters**

- **result\_name** ([str](#)) – The name of the result.
- **latest** ([bool](#)) – Flag to deactivate warning about using latest result. Defaults to False

**Returns**

The path to the result.

**Return type**

[Path](#)

**Raises**

[ValueError](#) – Raised if result does not exist.

**property has\_data:** [bool](#)

Check if the project has datasets.

**Returns**

Whether the project has datasets.

**Return type**

[bool](#)

**property has\_models:** [bool](#)

Check if the project has models.

**Returns**

Whether the project has models.

**Return type**

[bool](#)

**property has\_parameters:** [bool](#)

Check if the project has parameters.

**Returns**

Whether the project has parameters.

**Return type**

[bool](#)

**property has\_results:** `bool`

Check if the project has results.

**Returns**

Whether the project has results.

**Return type**

`bool`

**import\_data**(*dataset: LoadableDataset | Mapping[str, LoadableDataset], dataset\_name: str | None = None, allow\_overwrite: bool = False, ignore\_existing: bool = True*)

Import a dataset by saving it as an .nc file in the project's data folder.

**Parameters**

- **dataset** (*LoadableDataset*) – Dataset instance or path to a dataset.
- **dataset\_name** (*str | None*) – The name of the dataset (needs to be provided when dataset is an xarray instance). Defaults to None.
- **allow\_overwrite** (*bool*) – Whether to overwrite an existing dataset.
- **ignore\_existing** (*bool*) – Whether to skip import if the dataset already exists and allow\_overwrite is False. Defaults to True.

**load\_data**(*dataset\_name: str, \*, add\_svd: bool = False, lsv\_dim: Hashable = 'time', rsv\_dim: Hashable = 'spectral'*) → *xr.Dataset*

Load a dataset, with SVD data if add\_svd is True.

**Parameters**

- **dataset\_name** (*str*) – The name of the dataset.
- **add\_svd** (*bool*) – Whether or not to calculate and add SVD data. Defaults to False.
- **lsv\_dim** (*Hashable*) – Dimension of the left singular vectors. Defaults to “time”.
- **rsv\_dim** (*Hashable*) – Dimension of the right singular vectors. Defaults to “spectral”,

**Returns**

The loaded dataset, with SVD data if add\_svd is True.

**Return type**

*xr.Dataset*

**Raises**

**ValueError** – Raised if the dataset does not exist.

**load\_latest\_result**(*result\_name: str*) → *Result*

Load a result.

**Parameters**

**result\_name** (*str*) – The name of the result.

**Returns**

The loaded result.

**Return type**

*Result*

**Raises**

**ValueError** – Raised if result does not exist.

**load\_model**(*model\_name: str*) → *Model*

Load a model.

**Parameters**

**model\_name** (*str*) – The name of the model.

**Returns**

The loaded model.

**Return type**

*Model*

**Raises**

**ValueError** – Raised if the model does not exist.

**load\_parameters**(*parameters\_name: str*) → *Parameters*

Load parameters.

**Parameters**

**parameters\_name** (*str*) – The name of the parameters.

**Raises**

**ValueError** – Raised if parameters do not exist.

**Returns**

The loaded parameters.

**Return type**

*Parameters*

**load\_result**(*result\_name: str, \*, latest: bool = False*) → *Result*

Load a result.

**Parameters**

- **result\_name** (*str*) – The name of the result.
- **latest** (*bool*) – Flag to deactivate warning about using latest result. Defaults to False

**Returns**

The loaded result.

**Return type**

*Result*

**Raises**

**ValueError** – Raised if result does not exist.

**markdown**() → *MarkdownStr*

Format the project as a markdown text.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

*str*

**property models:** `Mapping[str, Path]`

Get all project models.

**Returns**

The models of the project.

**Return type**

`Mapping[str, Path]`

**classmethod** `open(project_folder_or_file: str | Path, create_if_not_exist: bool = True) → Project`

Open a new project.

**Parameters**

- **project\_folder\_or\_file** (`str | Path`) – The path to a project folder or file.
- **create\_if\_not\_exist** (`bool`) – Create the project if not existent.

**Returns**

The project instance.

**Return type**

`Project`

**Raises**

**FileNotFoundError** – Raised when the project file does not exist and `create_if_not_exist` is `False`.

**optimize**(`model_name: str, parameters_name: str, result_name: str | None = None, maximum_number_function_evaluations: int | None = None, clp_link_tolerance: float = 0.0, data_lookup_override: Mapping[str, LoadableDataset] | None = None`) → `Result`

Optimize a model.

**Parameters**

- **model\_name** (`str`) – The model to optimize.
- **parameters\_name** (`str`) – The initial parameters.
- **result\_name** (`str | None`) – The name of the result.
- **maximum\_number\_function\_evaluations** (`int | None`) – The maximum number of function evaluations.
- **clp\_link\_tolerance** (`float`) – The CLP link tolerance.
- **data\_lookup\_override** (`Mapping[str, LoadableDataset] | None`) – Allows to bypass the default dataset lookup in the project data folder and use a different dataset for the optimization without changing the model. This is especially useful when working with preprocessed data. Defaults to `None`.

**Returns**

Result of the optimization.

**Return type**

`Result`

**property parameters:** `Mapping[str, Path]`

Get all project parameters.

**Returns**

The parameters of the project.

**Return type**

Mapping[str, Path]

**property results:** Mapping[str, Path]

Get all project results.

**Returns**

The results of the project.

**Return type**

Mapping[str, Path]

**show\_model\_definition**(model\_name: str, syntax: str | None = None) → MarkdownStr

Show model definition file content with syntax highlighting.

**Parameters**

- **model\_name** (str) – The name of the model.
- **syntax** (str | None) – Syntax used for syntax highlighting. Defaults to None which means that the syntax is inferred based on the file extension. Pass the value "" to deactivate syntax highlighting.

**Returns**

Model definition file content with syntax highlighting to render in ipython.

**Return type**

MarkdownStr

**show\_parameters\_definition**(parameters\_name: str, syntax: str | None = None, \*, as\_dataframe: bool | None = None) → MarkdownStr | DataFrame

Show parameters definition file content with syntax highlighting.

**Parameters**

- **parameters\_name** (str) – The name of the parameters.
- **syntax** (str | None) – Syntax used for syntax highlighting. Defaults to None which means that the syntax is inferred based on the file extension. Pass the value "" to deactivate syntax highlighting.
- **as\_dataframe** (bool | None) – Whether or not to show the Parameters definition as pandas.DataFrame (mostly useful for non string formats). Defaults to None which means that it will be inferred to True for known non string formats like `xlsx`.

**Returns**

Parameters definition file content with syntax highlighting to render in ipython.

**Return type**

MarkdownStr | pd.DataFrame

**validate**(model\_name: str, parameters\_name: str | None = None) → MarkdownStr

Check that the model is valid, list all issues in the model if there are any.

If parameters\_name also consider the Parameters when validating.

**Parameters**



- **model\_name** (*str*) – The name of the model to validate.
- **parameters\_name** (*str* / *None*) – The name of the parameters to use when validating. Defaults to *None* which means that parameters are not considered when validating the model.

**Returns**

Text indicating if the model is valid or not.

**Return type**

*MarkdownStr*

**version:** *str*

**project\_data\_registry**

The glotaran data registry module.

**Classes****Summary**

<i>ProjectDataRegistry</i>	A registry for data.
----------------------------	----------------------

**ProjectDataRegistry**

**class** glotaran.project.project\_data\_registry.**ProjectDataRegistry**(*directory*: *Path*)

Bases: *ProjectRegistry*

A registry for data.

Initialize a data registry.

**Parameters**

**directory** (*Path*) – The registry directory.

**Attributes Summary**

<i>directory</i>	Get the registry directory.
<i>empty</i>	Whether the registry is empty.
<i>items</i>	Get the items of the registry.

## directory

ProjectDataRegistry.**directory**

Get the registry directory.

**Returns**

The registry directory.

**Return type**

Path

## empty

ProjectDataRegistry.**empty**

Whether the registry is empty.

**Returns**

Whether the registry is empty.

**Return type**

bool

## items

ProjectDataRegistry.**items**

Get the items of the registry.

**Returns**

The items of the registry.

**Return type**

*ItemMapping*

## Methods Summary

<i>import_data</i>	Import a dataset.
<i>is_item</i>	Check if the path contains an registry item.
<i>load_item</i>	Load an registry item by it's name.
<i>markdown</i>	Format the registry items as a markdown text.

## import\_data

ProjectDataRegistry.**import\_data**(*dataset: str | Path | Dataset | DataArray, dataset\_name: str | None = None, allow\_overwrite: bool = False, ignore\_existing: bool = False*)

Import a dataset.

**Parameters**

- **dataset** (*str | Path | xr.Dataset | xr.DataArray*) – Dataset instance or path to a dataset.

- **dataset\_name** (*str* / *None*) – The name of the dataset (needs to be provided when dataset is an xarray instance). Defaults to *None*.
- **allow\_overwrite** (*bool*) – Whether to overwrite an existing dataset.
- **ignore\_existing** (*bool*) – Whether to ignore import if the dataset already exists.

**Raises**

**ValueError** – When importing from xarray object and not providing a name.

**is\_item**

`ProjectDataRegistry.is_item(path: Path) → bool`

Check if the path contains an registry item.

**Parameters**

**path** (*Path*) – The path to check.

**Returns**

Whether the path contains an item.

**Return type**

*bool*

**load\_item**

`ProjectDataRegistry.load_item(name: str) → Any`

Load an registry item by it's name.

**Parameters**

**name** (*str*) – The item name.

**Returns**

The loaded item.

**Return type**

*Any*

**Raises**

**ValueError** – Raise if the item does not exist.

**markdown**

`ProjectDataRegistry.markdown(join_indentation: int = 0) → MarkdownStr`

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with `dedent` when used in an indented f-string. Defaults to 0.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

*str*

## Methods Documentation

**property directory:** `Path`

Get the registry directory.

**Returns**

The registry directory.

**Return type**

`Path`

**property empty:** `bool`

Whether the registry is empty.

**Returns**

Whether the registry is empty.

**Return type**

`bool`

**import\_data**(*dataset*: `str` | `Path` | `Dataset` | `DataArray`, *dataset\_name*: `str` | `None` = `None`,  
*allow\_overwrite*: `bool` = `False`, *ignore\_existing*: `bool` = `False`)

Import a dataset.

**Parameters**

- **dataset** (`str` | `Path` | `xr.Dataset` | `xr.DataArray`) – Dataset instance or path to a dataset.
- **dataset\_name** (`str` | `None`) – The name of the dataset (needs to be provided when dataset is an xarray instance). Defaults to `None`.
- **allow\_overwrite** (`bool`) – Whether to overwrite an existing dataset.
- **ignore\_existing** (`bool`) – Whether to ignore import if the dataset already exists.

**Raises**

**ValueError** – When importing from xarray object and not providing a name.

**is\_item**(*path*: `Path`) → `bool`

Check if the path contains an registry item.

**Parameters**

**path** (`Path`) – The path to check.

**Returns**

Whether the path contains an item.

**Return type**

`bool`

**property items:** `ItemMapping`

Get the items of the registry.

**Returns**

The items of the registry.

**Return type**

`ItemMapping`

**load\_item**(*name: str*) → *Any*

Load an registry item by it's name.

**Parameters**

**name** (*str*) – The item name.

**Returns**

The loaded item.

**Return type**

*Any*

**Raises**

**ValueError** – Raise if the item does not exist.

**markdown**(*join\_indentation: int = 0*) → *MarkdownStr*

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with dedent when used in an indented f-string. Defaults to 0.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

*str*

## project\_model\_registry

The glotaran model registry module.

## Classes

### Summary

*ProjectModelRegistry*

A registry for models.

## ProjectModelRegistry

**class** glotaran.project.project\_model\_registry.**ProjectModelRegistry**(*directory: Path*)

Bases: *ProjectRegistry*

A registry for models.

Initialize a model registry.

**Parameters**

**directory** (*Path*) – The registry directory.

## Attributes Summary

<i>directory</i>	Get the registry directory.
<i>empty</i>	Whether the registry is empty.
<i>items</i>	Get the items of the registry.

## directory

`ProjectModelRegistry.directory`

Get the registry directory.

**Returns**

The registry directory.

**Return type**

Path

## empty

`ProjectModelRegistry.empty`

Whether the registry is empty.

**Returns**

Whether the registry is empty.

**Return type**

bool

## items

`ProjectModelRegistry.items`

Get the items of the registry.

**Returns**

The items of the registry.

**Return type**

*ItemMapping*

## Methods Summary

<i>generate_model</i>	Generate a model.
<i>is_item</i>	Check if the path contains an registry item.
<i>load_item</i>	Load an registry item by it's name.
<i>markdown</i>	Format the registry items as a markdown text.

## generate\_model

ProjectModelRegistry.**generate\_model**(*name*: *str*, *generator\_name*: *str*,  
*generator\_arguments*: GeneratorArguments, \*,  
*allow\_overwrite*: *bool* = False, *ignore\_existing*:  
*bool* = False)

Generate a model.

### Parameters

- **name** (*str*) – The name of the model.
- **generator\_name** (*str*) – The generator for the model.
- **generator\_arguments** (GeneratorArguments) – Arguments for the generator.
- **allow\_overwrite** (*bool*) – Whether to overwrite an existing model.
- **ignore\_existing** (*bool*) – Whether to ignore generation of a model file if it already exists.

### Raises

**FileExistsError** – Raised if model is already existing and *allow\_overwrite=False*.

## is\_item

ProjectModelRegistry.**is\_item**(*path*: Path) → bool

Check if the path contains an registry item.

### Parameters

**path** (Path) – The path to check.

### Returns

Whether the path contains an item.

### Return type

bool

## load\_item

ProjectModelRegistry.**load\_item**(*name*: *str*) → Any

Load an registry item by it's name.

### Parameters

**name** (*str*) – The item name.

### Returns

The loaded item.

### Return type

Any

### Raises

**ValueError** – Raise if the item does not exist.

## markdown

ProjectModelRegistry.**markdown**(join\_indentation: *int* = 0) → *MarkdownStr*

Format the registry items as a markdown text.

### Parameters

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with dedent when used in an indented f-string. Defaults to 0.

### Returns

**MarkdownStr** – The markdown string.

### Return type

*str*

## Methods Documentation

**property directory:** *Path*

Get the registry directory.

### Returns

The registry directory.

### Return type

*Path*

**property empty:** *bool*

Whether the registry is empty.

### Returns

Whether the registry is empty.

### Return type

*bool*

**generate\_model**(name: *str*, generator\_name: *str*, generator\_arguments: *GeneratorArguments*, \*, allow\_overwrite: *bool* = False, ignore\_existing: *bool* = False)

Generate a model.

### Parameters

- **name** (*str*) – The name of the model.
- **generator\_name** (*str*) – The generator for the model.
- **generator\_arguments** (*GeneratorArguments*) – Arguments for the generator.
- **allow\_overwrite** (*bool*) – Whether to overwrite an existing model.
- **ignore\_existing** (*bool*) – Whether to ignore generation of a model file if it already exists.

### Raises

**FileExistsError** – Raised if model is already existing and *allow\_overwrite=False*.



**is\_item**(*path*: *Path*) → bool

Check if the path contains an registry item.

**Parameters**

**path** (*Path*) – The path to check.

**Returns**

Whether the path contains an item.

**Return type**

bool

**property items**: *ItemMapping*

Get the items of the registry.

**Returns**

The items of the registry.

**Return type**

*ItemMapping*

**load\_item**(*name*: *str*) → *Any*

Load an registry item by it's name.

**Parameters**

**name** (*str*) – The item name.

**Returns**

The loaded item.

**Return type**

Any

**Raises**

**ValueError** – Raise if the item does not exist.

**markdown**(*join\_indentation*: *int* = 0) → *MarkdownStr*

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with dedent when used in an indented f-string. Defaults to 0.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

str

## project\_parameter\_registry

The glotaran parameter registry module.

## Classes

### Summary

<i>ProjectParameterRegistry</i>	A registry for parameters.
---------------------------------	----------------------------

### ProjectParameterRegistry

**class** glotaran.project.project\_parameter\_registry.**ProjectParameterRegistry**(*directory*:  
*Path*)

Bases: *ProjectRegistry*

A registry for parameters.

Initialize a parameter registry.

#### Parameters

**directory** (*Path*) – The registry directory.

### Attributes Summary

<i>directory</i>	Get the registry directory.
<i>empty</i>	Whether the registry is empty.
<i>items</i>	Get the items of the registry.

### directory

**ProjectParameterRegistry.directory**

Get the registry directory.

#### Returns

The registry directory.

#### Return type

Path

### empty

**ProjectParameterRegistry.empty**

Whether the registry is empty.

#### Returns

Whether the registry is empty.

#### Return type

bool

## items

`ProjectParameterRegistry.items`

Get the items of the registry.

### Returns

The items of the registry.

### Return type

*ItemMapping*

## Methods Summary

<code>generate_parameters</code>	Generate parameters for a model.
<code>is_item</code>	Check if the path contains an registry item.
<code>load_item</code>	Load an registry item by it's name.
<code>markdown</code>	Format the registry items as a markdown text.

## generate\_parameters

`ProjectParameterRegistry.generate_parameters`(*model*: `Model`, *name*: `str` | `None`, \*,  
*format\_name*: `Literal`['yaml', 'yaml', 'csv'],  
*'csv']* = 'csv', *allow\_overwrite*: `bool` =  
`False`, *ignore\_existing*: `bool` = `False`)

Generate parameters for a model.

### Parameters

- **model** (`Model`) – The model.
- **name** (`str` | `None`) – The name of the parameters.
- **format\_name** (`Literal`["yaml", "yaml", "csv"]) – The parameter format.
- **allow\_overwrite** (`bool`) – Whether to overwrite existing parameters.
- **ignore\_existing** (`bool`) – Whether to ignore generation of a parameter file if it already exists.

### Raises

**FileExistsError** – Raised if parameters is already existing and *allow\_overwrite=False*.

## is\_item

`ProjectParameterRegistry.is_item`(*path*: `Path`) → `bool`

Check if the path contains an registry item.

### Parameters

**path** (`Path`) – The path to check.

### Returns

Whether the path contains an item.

**Return type**`bool`**load\_item**

`ProjectParameterRegistry.load_item(name: str) → Any`

Load an registry item by it's name.

**Parameters**

**name** (*str*) – The item name.

**Returns**

The loaded item.

**Return type**`Any`**Raises**

**ValueError** – Raise if the item does not exist.

**markdown**

`ProjectParameterRegistry.markdown(join_indentation: int = 0) → MarkdownStr`

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with `dedent` when used in an indented f-string. Defaults to 0.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**`str`**Methods Documentation**

**property directory:** `Path`

Get the registry directory.

**Returns**

The registry directory.

**Return type**`Path`

**property empty:** `bool`

Whether the registry is empty.

**Returns**

Whether the registry is empty.

**Return type**`bool`

**generate\_parameters**(*model*: [Model](#), *name*: *str* | *None*, \*, *format\_name*: *Literal*['yaml', 'yaml', 'csv'] = 'csv', *allow\_overwrite*: *bool* = *False*, *ignore\_existing*: *bool* = *False*)

Generate parameters for a model.

#### Parameters

- **model** ([Model](#)) – The model.
- **name** (*str* | *None*) – The name of the parameters.
- **format\_name** (*Literal*["yaml", "yaml", "csv"]) – The parameter format.
- **allow\_overwrite** (*bool*) – Whether to overwrite existing parameters.
- **ignore\_existing** (*bool*) – Whether to ignore generation of a parameter file if it already exists.

#### Raises

**FileExistsError** – Raised if parameters is already existing and *allow\_overwrite=False*.

**is\_item**(*path*: [Path](#)) → *bool*

Check if the path contains an registry item.

#### Parameters

**path** ([Path](#)) – The path to check.

#### Returns

Whether the path contains an item.

#### Return type

*bool*

**property items**: [ItemMapping](#)

Get the items of the registry.

#### Returns

The items of the registry.

#### Return type

[ItemMapping](#)

**load\_item**(*name*: *str*) → *Any*

Load an registry item by it's name.

#### Parameters

**name** (*str*) – The item name.

#### Returns

The loaded item.

#### Return type

*Any*

#### Raises

**ValueError** – Raise if the item does not exist.

**markdown**(*join\_indentation*: *int* = 0) → [MarkdownStr](#)

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with `dedent` when used in an indented f-string. Defaults to 0.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

*str*

## project\_registry

The glotaran registry module.

## Classes

### Summary

<i>ItemMapping</i>	Container class for <code>ProjectRegistry</code> items.
<i>ProjectRegistry</i>	A registry base class.

## ItemMapping

```
class glotaran.project.project_registry.ItemMapping(data: Mapping[str, Path],  
                                                    item_name: str)
```

Bases: `Mapping`

Container class for `ProjectRegistry` items.

The main purpose of this class is to show a user friendly error when accessing none existing items.

Initialize class instance as wrapper around data.

**Parameters**

- **data** (*Mapping*[*str*, *Path*]) – Underlying data that are used for mapping.
- **item\_name** (*str*) – Name of items in the registry used to format warning and exception (e.g. 'Parameters').

### Methods Summary

<i>get</i>
<i>items</i>
<i>keys</i>
<i>values</i>

**get**

`ItemMapping.get(k[, d])` → `D[k]` if `k` in `D`, else `d`. `d` defaults to `None`.

**items**

`ItemMapping.items()` → a set-like object providing a view on `D`'s items

**keys**

`ItemMapping.keys()` → a set-like object providing a view on `D`'s keys

**values**

`ItemMapping.values()` → an object providing a view on `D`'s values

**Methods Documentation**

`get(k[, d])` → `D[k]` if `k` in `D`, else `d`. `d` defaults to `None`.

`items()` → a set-like object providing a view on `D`'s items

`keys()` → a set-like object providing a view on `D`'s keys

`values()` → an object providing a view on `D`'s values

**ProjectRegistry**

```
class glotaran.project.project_registry.ProjectRegistry(directory: Path, file_suffix: str
| Iterable[str], loader:
Callable, item_name: str)
```

Bases: `object`

A registry base class.

Initialize a registry.

**Parameters**

- **directory** (*Path*) – The registry directory.
- **file\_suffix** (*str* | *Iterable[str]*) – The suffixes of item files.
- **loader** (*Callable*) – A loader for the registry items.
- **item\_name** (*str*) – Name of items in the registry used to format warning and exception (e.g. 'Parameters').

## Attributes Summary

<i>directory</i>	Get the registry directory.
<i>empty</i>	Whether the registry is empty.
<i>items</i>	Get the items of the registry.

## directory

`ProjectRegistry.directory`

Get the registry directory.

**Returns**

The registry directory.

**Return type**

Path

## empty

`ProjectRegistry.empty`

Whether the registry is empty.

**Returns**

Whether the registry is empty.

**Return type**

bool

## items

`ProjectRegistry.items`

Get the items of the registry.

**Returns**

The items of the registry.

**Return type**

*ItemMapping*

## Methods Summary

<i>is_item</i>	Check if the path contains an registry item.
<i>load_item</i>	Load an registry item by it's name.
<i>markdown</i>	Format the registry items as a markdown text.



### is\_item

ProjectRegistry.**is\_item**(path: *Path*) → bool

Check if the path contains an registry item.

**Parameters**

**path** (*Path*) – The path to check.

**Returns**

Whether the path contains an item.

**Return type**

bool

### load\_item

ProjectRegistry.**load\_item**(name: *str*) → Any

Load an registry item by it's name.

**Parameters**

**name** (*str*) – The item name.

**Returns**

The loaded item.

**Return type**

Any

**Raises**

**ValueError** – Raise if the item does not exist.

### markdown

ProjectRegistry.**markdown**(join\_indentation: *int* = 0) → *MarkdownStr*

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with dedent when used in an indented f-string. Defaults to 0.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

str

## Methods Documentation

**property directory:** *Path*

Get the registry directory.

**Returns**

The registry directory.

**Return type**

*Path*

**property empty:** *bool*

Whether the registry is empty.

**Returns**

Whether the registry is empty.

**Return type**

*bool*

**is\_item**(*path: Path*) → *bool*

Check if the path contains an registry item.

**Parameters**

**path** (*Path*) – The path to check.

**Returns**

Whether the path contains an item.

**Return type**

*bool*

**property items:** *ItemMapping*

Get the items of the registry.

**Returns**

The items of the registry.

**Return type**

*ItemMapping*

**load\_item**(*name: str*) → *Any*

Load an registry item by it's name.

**Parameters**

**name** (*str*) – The item name.

**Returns**

The loaded item.

**Return type**

*Any*

**Raises**

**ValueError** – Raise if the item does not exist.

**markdown**(*join\_indentation: int = 0*) → *MarkdownStr*

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the

parts. This is intended to be used with `dedent` when used in an indented f-string.  
Defaults to 0.

#### Returns

**MarkdownStr** – The markdown string.

#### Return type

`str`

## Exceptions

### Exception Summary

<b>AmbiguousNameWarning</b>	Warning thrown when an item with the same name already exists.
-----------------------------	--

### AmbiguousNameWarning

**exception** `glotaran.project.project_registry.AmbiguousNameWarning`(\*, *item\_name*: `str`, *items*: `dict[str, Path]`, *item\_key*: `str`, *unique\_item\_key*: `str`, *project\_root\_path*: `Path`)

Warning thrown when an item with the same name already exists.

This is the case if two files with the same name but different extensions exist next to each other.

Initialize `AmbiguousNameWarning` with a formatted message.

#### Parameters

- **item\_name** (`str`) – Name of items in the registry (e.g. ‘Parameters’).
- **items** (`dict[str, Path]`) – Known items at this iteration point.
- **item\_key** (`str`) – Key that would have been used if the file names weren’t ambiguous.
- **unique\_item\_key** (`str`) – Unique key for the item with ambiguous file name.
- **project\_root\_path** (`Path`) – Root path of the project.

## project\_result\_registry

The glotaran result registry module.

### Classes

#### Summary

<i>ProjectResultRegistry</i>	A registry for results.
------------------------------	-------------------------

#### ProjectResultRegistry

**class** glotaran.project.project\_result\_registry.**ProjectResultRegistry**(*directory*:  
*Path*)

Bases: *ProjectRegistry*

A registry for results.

Initialize a result registry.

##### Parameters

**directory** (*Path*) – The registry directory.

#### Attributes Summary

<i>directory</i>	Get the registry directory.
<i>empty</i>	Whether the registry is empty.
<i>items</i>	Get the items of the registry.
<i>result_pattern</i>	

#### directory

**ProjectResultRegistry.directory**

Get the registry directory.

##### Returns

The registry directory.

##### Return type

Path

## empty

ProjectResultRegistry.**empty**

Whether the registry is empty.

### Returns

Whether the registry is empty.

### Return type

`bool`

## items

ProjectResultRegistry.**items**

Get the items of the registry.

### Returns

The items of the registry.

### Return type

*ItemMapping*

## result\_pattern

ProjectResultRegistry.**result\_pattern** = `re.compile('._run_\\d{4}$')`

## Methods Summary

<code>create_result_run_name</code>	Create a result name for a model.
<code>is_item</code>	Check if the path contains an registry item.
<code>load_item</code>	Load an registry item by it's name.
<code>markdown</code>	Format the registry items as a markdown text.
<code>previous_result_paths</code>	List previous result paths with base_name.
<code>save</code>	Save a result.

## create\_result\_run\_name

ProjectResultRegistry.**create\_result\_run\_name**(*base\_name: str*) → *str*

Create a result name for a model.

### Parameters

**base\_name** (*str*) – The base name for the result provided by user or derived from model name.

### Returns

Folder name for the new result to be saved in.

### Return type

`str`

### is\_item

ProjectResultRegistry.**is\_item**(path: *Path*) → bool

Check if the path contains an registry item.

**Parameters**

**path** (*Path*) – The path to check.

**Returns**

Whether the path contains an item.

**Return type**

bool

### load\_item

ProjectResultRegistry.**load\_item**(name: *str*) → Any

Load an registry item by it's name.

**Parameters**

**name** (*str*) – The item name.

**Returns**

The loaded item.

**Return type**

Any

**Raises**

**ValueError** – Raise if the item does not exist.

### markdown

ProjectResultRegistry.**markdown**(join\_indentation: *int* = 0) → *MarkdownStr*

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with dedent when used in an indented f-string. Defaults to 0.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

str

## previous\_result\_paths

`ProjectResultRegistry.previous_result_paths(base_name: str) → list[Path]`

List previous result paths with base\_name.

### Parameters

**base\_name** (*str*) – The base name for the result provided by user or derived from model name.

### Returns

Paths to previous results with name base\_name.

### Return type

`list[Path]`

## save

`ProjectResultRegistry.save(name: str, result: Result)`

Save a result.

### Parameters

- **name** (*str*) – The name of the result.
- **result** (*Result*) – The result to save.

## Methods Documentation

`create_result_run_name(base_name: str) → str`

Create a result name for a model.

### Parameters

**base\_name** (*str*) – The base name for the result provided by user or derived from model name.

### Returns

Folder name for the new result to be saved in.

### Return type

`str`

**property directory:** `Path`

Get the registry directory.

### Returns

The registry directory.

### Return type

`Path`

**property empty:** `bool`

Whether the registry is empty.

### Returns

Whether the registry is empty.

### Return type

`bool`

**is\_item**(*path*: *Path*) → bool

Check if the path contains an registry item.

**Parameters**

**path** (*Path*) – The path to check.

**Returns**

Whether the path contains an item.

**Return type**

bool

**property items**: *ItemMapping*

Get the items of the registry.

**Returns**

The items of the registry.

**Return type**

*ItemMapping*

**load\_item**(*name*: *str*) → *Any*

Load an registry item by it's name.

**Parameters**

**name** (*str*) – The item name.

**Returns**

The loaded item.

**Return type**

Any

**Raises**

**ValueError** – Raise if the item does not exist.

**markdown**(*join\_indentation*: *int* = 0) → *MarkdownStr*

Format the registry items as a markdown text.

**Parameters**

**join\_indentation** (*int*) – Number of whitespaces to indent when joining the parts. This is intended to be used with dedent when used in an indented f-string. Defaults to 0.

**Returns**

**MarkdownStr** – The markdown string.

**Return type**

str

**previous\_result\_paths**(*base\_name*: *str*) → list[*Path*]

List previous result paths with base\_name.

**Parameters**

**base\_name** (*str*) – The base name for the result provided by user or derived from model name.

**Returns**

Paths to previous results with name base\_name.

**Return type**

list[*Path*]



```
result_pattern = re.compile('._run_\\d{4}$')
```

```
save(name: str, result: Result)
```

Save a result.

#### Parameters

- **name** (*str*) – The name of the result.
- **result** (*Result*) – The result to save.

## result

The result class for global analysis.

## Classes

### Summary

<i>Result</i>	The result of a global analysis.
---------------	----------------------------------

### Result

```
class glotaran.project.result.Result(number_of_function_evaluations: int, success: bool,
    termination_reason: str, glotaran_version: str,
    free_parameter_labels: list[str], scheme: Scheme,
    initial_parameters: Parameters, optimized_parameters:
    Parameters, parameter_history: ParameterHistory,
    optimization_history: OptimizationHistory, data:
    Mapping[str, xr.Dataset], additional_penalty:
    list[np.ndarray] | None = None, cost: ArrayLike | None
    = None, chi_square: float | None = None,
    covariance_matrix: ArrayLike | None = None,
    degrees_of_freedom: int | None = None,
    number_of_clps: int | None = None, jacobian:
    ArrayLike | list | None = None, number_of_residuals:
    int | None = None, number_of_jacobian_evaluations:
    int | None = None, number_of_free_parameters: int |
    None = None, optimality: float | None = None,
    reduced_chi_square: float | None = None,
    root_mean_square_error: float | None = None)
```

Bases: *object*

The result of a global analysis.

## Attributes Summary

<i>additional_penalty</i>	A vector with the value for each additional penalty, or None
<i>chi_square</i>	The chi-square of the optimization.
<i>cost</i>	The final cost.
<i>covariance_matrix</i>	Covariance matrix.
<i>degrees_of_freedom</i>	Degrees of freedom in optimization $N - N_{vars} - N_{clps}$ .
<i>jacobian</i>	Modified Jacobian matrix at the solution
<i>model</i>	Return the model used to fit result.
<i>number_of_clps</i>	Number of conditionally linear parameters $N_{clps}$ .
<i>number_of_data_points</i>	Return the number of values in the residual vector $N$ .
<i>number_of_free_parameters</i>	Number of free parameters in optimization $N_{vars}$
<i>number_of_jacobian_evaluations</i>	The number of jacobian evaluations.
<i>number_of_parameters</i>	Return the number of free parameters in optimization $N_{vars}$ .
<i>number_of_residuals</i>	Number of values in the residual vector $N$ .
<i>optimality</i>	
<i>reduced_chi_square</i>	The reduced chi-square of the optimization.
<i>root_mean_square_error</i>	The root mean square error the optimization.
<i>source_path</i>	
<i>number_of_function_evaluations</i>	The number of function evaluations.
<i>success</i>	Indicates if the optimization was successful.
<i>termination_reason</i>	The reason (message when) the optimizer terminated
<i>glotaran_version</i>	The glotaran version used to create the result.
<i>free_parameter_labels</i>	List of labels of the free parameters used in optimization.
<i>scheme</i>	
<i>initial_parameters</i>	
<i>optimized_parameters</i>	
<i>parameter_history</i>	The parameter history.
<i>optimization_history</i>	The optimization history.
<i>data</i>	The resulting data as a dictionary of <code>xarray.Dataset</code> .

**additional\_penalty**

`Result.additional_penalty: list[np.ndarray] | None = None`

A vector with the value for each additional penalty, or None

**chi\_square**

`Result.chi_square: float | None = None`

The chi-square of the optimization.

$$\chi^2 = \sum_i^N [\text{Residual}_i]^2.$$

**cost**

`Result.cost: ArrayLike | None = None`

The final cost.

**covariance\_matrix**

`Result.covariance_matrix: ArrayLike | None = None`

Covariance matrix.

The rows and columns are corresponding to *free\_parameter\_labels*.

**degrees\_of\_freedom**

`Result.degrees_of_freedom: int | None = None`

Degrees of freedom in optimization  $N - N_{vars} - N_{clps}$ .

**jacobian**

`Result.jacobian: ArrayLike | list | None = None`

Modified Jacobian matrix at the solution

See also: `scipy.optimize.least_squares()`

**model**

`Result.model`

Return the model used to fit result.

**Returns**

The model instance.

**Return type**

*Model*

**number\_of\_clps**

`Result.number_of_clps: int | None = None`

Number of conditionally linear parameters  $N_{clps}$ .

**number\_of\_data\_points**

`Result.number_of_data_points`

Return the number of values in the residual vector  $N$ .

Deprecated since it returned the wrong value.

**Returns**

Number of values in the residual vector  $N$ .

**Return type**

`int | None`

**number\_of\_free\_parameters**

`Result.number_of_free_parameters: int | None = None`

Number of free parameters in optimization  $N_{vars}$

**number\_of\_jacobian\_evaluations**

`Result.number_of_jacobian_evaluations: int | None = None`

The number of jacobian evaluations.

**number\_of\_parameters**

`Result.number_of_parameters`

Return the number of free parameters in optimization  $N_{vars}$ .

**Returns**

Number of free parameters in optimization  $N_{vars}$ .

**Return type**

`int | None`

**number\_of\_residuals**

`Result.number_of_residuals: int | None = None`

Number of values in the residual vector  $N$ .

### optimality

`Result.optimality: float | None = None`

### reduced\_chi\_square

`Result.reduced_chi_square: float | None = None`

The reduced chi-square of the optimization.

$$\chi_{red}^2 = \chi^2 / (N - N_{vars} - N_{clps}).$$

### root\_mean\_square\_error

`Result.root_mean_square_error: float | None = None`

The root mean square error the optimization.

$$rms = \sqrt{\chi_{red}^2}$$

### source\_path

`Result.source_path: StrOrPath = 'result.yml'`

### number\_of\_function\_evaluations

`Result.number_of_function_evaluations: int`

The number of function evaluations.

### success

`Result.success: bool`

Indicates if the optimization was successful.

### termination\_reason

`Result.termination_reason: str`

The reason (message when) the optimizer terminated

### glotaran\_version

`Result.glotaran_version: str`

The glotaran version used to create the result.

**free\_parameter\_labels**

`Result.free_parameter_labels: list[str]`

List of labels of the free parameters used in optimization.

**scheme**

`Result.scheme: Scheme`

**initial\_parameters**

`Result.initial_parameters: Parameters`

**optimized\_parameters**

`Result.optimized_parameters: Parameters`

**parameter\_history**

`Result.parameter_history: ParameterHistory`

The parameter history.

**optimization\_history**

`Result.optimization_history: OptimizationHistory`

The optimization history.

**data**

`Result.data: Mapping[str, xr.Dataset]`

The resulting data as a dictionary of `xarray.Dataset`.

**Notes**

The actual content of the data depends on the actual model and can be found in the documentation for the model.

## Methods Summary

<code>create_clp_guide_dataset</code>	Create dataset for clp guidance.
<code>get_scheme</code>	Return a new scheme from the Result object with optimized parameters.
<code>loader</code>	Create a <a href="#">Result</a> instance from the specs defined in a file.
<code>markdown</code>	Format the model as a markdown text.
<code>recreate</code>	Recreate a result from the initial parameters.
<code>save</code>	Save the result to given folder.
<code>verify</code>	Verify a result.

### create\_clp\_guide\_dataset

`Result.create_clp_guide_dataset(clp_label: str, dataset_name: str) → Dataset`

Create dataset for clp guidance.

#### Parameters

- **clp\_label** (*str*) – Label of the clp to guide.
- **dataset\_name** (*str*) – Name of dataset to extract the guide from.

#### Returns

DataArray containing the clp guide, with `clp_label` dimension replaced by the model dimensions first value.

#### Return type

`xr.Dataset`

#### Raises

- **ValueError** – If `dataset_name` is not in result.
- **ValueError** – If `clp_labels` is not in result.

### Examples

Extracting the clp guide from an optimization result object.

```
from glotaran.io import save_dataset

clp_guide = result.create_clp_guide_dataset("species_1", "dataset_1")
save_dataset(clp_guide, "clp_guide__result_dataset_1__species_1.nc")
```

## get\_scheme

`Result.get_scheme()` → *Scheme*

Return a new scheme from the `Result` object with optimized parameters.

### Returns

A new scheme with the parameters set to the optimized values. For the dataset weights the (precomputed) weights from the original scheme are used.

### Return type

*Scheme*

## loader

`Result.loader(format_name: str | None = None, **kwargs: Any)` → *Result*

Create a *Result* instance from the specs defined in a file.

### Parameters

- **result\_path** (*StrOrPath*) – Path containing the result data.
- **format\_name** (*str* | *None*) – Format the result is in, if not provided and it is a file it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `load_result` implementation of the project io plugin.

### Returns

*Result* instance created from the saved format.

### Return type

*Result*

## markdown

`Result.markdown(with_model: bool = True, *, base_heading_level: int = 1, wrap_model_in_details: bool = False)` → *MarkdownStr*

Format the model as a markdown text.

### Parameters

- **with\_model** (*bool*) – If *True*, the model will be printed with initial and optimized parameters filled in.
- **base\_heading\_level** (*int*) – The level of the base heading.
- **wrap\_model\_in\_details** (*bool*) – Wraps model into details tag. Defaults to *False*

### Returns

**MarkdownStr** – The scheme as markdown string.

### Return type

*str*



**recreate**`Result.recreate() → Result`

Recreate a result from the initial parameters.

**Returns**

The recreated result.

**Return type**

*Result*

**save**`Result.save(path: StrOrPath, saving_options: SavingOptions = SavingOptions(data_filter=None, data_format='nc', parameter_format='csv', report=True)) → list[str]`

Save the result to given folder.

**Parameters**

- **path** (*StrOrPath*) – The path to the folder in which to save the result.
- **saving\_options** (*SavingOptions*) – Options for the saved result.

**Returns**

Paths to all the saved files.

**Return type**

`list[str]`

**verify**`Result.verify() → bool`

Verify a result.

**Returns**

Whether the recreated result is equal to this result.

**Return type**

`bool`

**Methods Documentation****additional\_penalty:** `list[np.ndarray] | None = None`

A vector with the value for each additional penalty, or None

**chi\_square:** `float | None = None`

The chi-square of the optimization.

$$\chi^2 = \sum_i^N [Residual_i]^2.$$

**cost:** `ArrayLike | None = None`

The final cost.

**covariance\_matrix:** `ArrayLike | None = None`

Covariance matrix.

The rows and columns are corresponding to *free\_parameter\_labels*.

**create\_clp\_guide\_dataset**(*clp\_label: str, dataset\_name: str*) → `Dataset`

Create dataset for clp guidance.

#### Parameters

- **clp\_label** (*str*) – Label of the clp to guide.
- **dataset\_name** (*str*) – Name of dataset to extract the guide from.

#### Returns

`dataArray` containing the clp guide, with `clp_label` dimension replaced by the model dimensions first value.

#### Return type

`xr.Dataset`

#### Raises

- **ValueError** – If `dataset_name` is not in result.
- **ValueError** – If `clp_labels` is not in result.

### Examples

Extracting the clp guide from an optimization result object.

```
from glotaran.io import save_dataset

clp_guide = result.create_clp_guide_dataset("species_1", "dataset_1")
save_dataset(clp_guide, "clp_guide__result_dataset_1__species_1.nc")
```

**data:** `Mapping[str, xr.Dataset]`

The resulting data as a dictionary of `xarray.Dataset`.

### Notes

The actual content of the data depends on the actual model and can be found in the documentation for the model.

**degrees\_of\_freedom:** `int | None = None`

Degrees of freedom in optimization  $N - N_{vars} - N_{clps}$ .

**free\_parameter\_labels:** `list[str]`

List of labels of the free parameters used in optimization.

**get\_scheme()** → *Scheme*

Return a new scheme from the Result object with optimized parameters.

#### Returns

A new scheme with the parameters set to the optimized values. For the dataset weights the (precomputed) weights from the original scheme are used.

#### Return type

*Scheme*

**glotaran\_version:** `str`

The glotaran version used to create the result.

**initial\_parameters:** `Parameters`

**jacobian:** `ArrayLike | list | None = None`

Modified Jacobian matrix at the solution

See also: `scipy.optimize.least_squares()`

**loader**(*format\_name: str | None = None, \*\*kwargs: Any*) → `Result`

Create a `Result` instance from the specs defined in a file.

#### Parameters

- **result\_path** (`StrOrPath`) – Path containing the result data.
- **format\_name** (`str | None`) – Format the result is in, if not provided and it is a file it will be inferred from the file extension.
- **\*\*kwargs** (`Any`) – Additional keyword arguments passes to the `load_result` implementation of the project io plugin.

#### Returns

`Result` instance created from the saved format.

#### Return type

`Result`

**markdown**(*with\_model: bool = True, \*, base\_heading\_level: int = 1, wrap\_model\_in\_details: bool = False*) → `MarkdownStr`

Format the model as a markdown text.

#### Parameters

- **with\_model** (`bool`) – If `True`, the model will be printed with initial and optimized parameters filled in.
- **base\_heading\_level** (`int`) – The level of the base heading.
- **wrap\_model\_in\_details** (`bool`) – Wraps model into details tag. Defaults to `False`

#### Returns

`MarkdownStr` – The scheme as markdown string.

#### Return type

`str`

**property model:** `Model`

Return the model used to fit result.

#### Returns

The model instance.

#### Return type

`Model`

**number\_of\_clps:** `int | None = None`

Number of conditionally linear parameters  $N_{clps}$ .

**property number\_of\_data\_points:** `int` | `None`

Return the number of values in the residual vector  $N$ .

Deprecated since it returned the wrong value.

**Returns**

Number of values in the residual vector  $N$ .

**Return type**

`int` | `None`

**number\_of\_free\_parameters:** `int` | `None` = `None`

Number of free parameters in optimization  $N_{vars}$

**number\_of\_function\_evaluations:** `int`

The number of function evaluations.

**number\_of\_jacobian\_evaluations:** `int` | `None` = `None`

The number of jacobian evaluations.

**property number\_of\_parameters:** `int` | `None`

Return the number of free parameters in optimization  $N_{vars}$ .

**Returns**

Number of free parameters in optimization  $N_{vars}$ .

**Return type**

`int` | `None`

**number\_of\_residuals:** `int` | `None` = `None`

Number of values in the residual vector  $N$ .

**optimality:** `float` | `None` = `None`

**optimization\_history:** *OptimizationHistory*

The optimization history.

**optimized\_parameters:** *Parameters*

**parameter\_history:** *ParameterHistory*

The parameter history.

**recreate()**  $\rightarrow$  *Result*

Recreate a result from the initial parameters.

**Returns**

The recreated result.

**Return type**

*Result*

**reduced\_chi\_square:** `float` | `None` = `None`

The reduced chi-square of the optimization.

$$\chi_{red}^2 = \chi^2 / (N - N_{vars} - N_{clps}).$$

**root\_mean\_square\_error:** `float` | `None` = `None`

The root mean square error the optimization.

$$rms = \sqrt{\chi_{red}^2}$$

**save**(*path*: *StrOrPath*, *saving\_options*: *SavingOptions* = *SavingOptions*(*data\_filter*=None, *data\_format*='nc', *parameter\_format*='csv', *report*=True)) → *list*[*str*]

Save the result to given folder.

#### Parameters

- **path** (*StrOrPath*) – The path to the folder in which to save the result.
- **saving\_options** (*SavingOptions*) – Options for the saved result.

#### Returns

Paths to all the saved files.

#### Return type

*list*[*str*]

**scheme**: *Scheme*

**source\_path**: *StrOrPath* = 'result.yml'

**success**: *bool*

Indicates if the optimization was successful.

**termination\_reason**: *str*

The reason (message when) the optimizer terminated

**verify**() → *bool*

Verify a result.

#### Returns

Whether the recreated result is equal to this result.

#### Return type

*bool*

## scheme

The module for :class:Scheme.

## Classes

### Summary

*Scheme*

A scheme is a collection of a model, parameters and a dataset.

## Scheme

```
class glotaran.project.scheme.Scheme(model: Model, parameters: Parameters, data:
    Mapping[str, xr.Dataset], clp_link_tolerance: float =
    0.0, clp_link_method: Literal['nearest', 'backward',
    'forward'] = 'nearest',
    maximum_number_function_evaluations: int | None =
    None, add_svd: bool = True, ftol: float = 1e-08, gtol:
    float = 1e-08, xtol: float = 1e-08, optimization_method:
    Literal['TrustRegionReflection', 'Dogbox',
    'Levenberg-Marquardt'] = 'TrustRegionReflection',
    result_path: str | None = None)
```

Bases: `object`

A scheme is a collection of a model, parameters and a dataset.

A scheme also holds options for optimization.

## Attributes Summary

<code>add_svd</code>
<code>clp_link_method</code>
<code>clp_link_tolerance</code>
<code>ftol</code>
<code>gtol</code>
<code>maximum_number_function_evaluations</code>
<code>optimization_method</code>
<code>result_path</code>
<code>source_path</code>
<code>xtol</code>
<code>model</code>
<code>parameters</code>
<code>data</code>

### `add_svd`

`Scheme.add_svd: bool = True`

### `clp_link_method`

`Scheme.clp_link_method: Literal['nearest', 'backward', 'forward'] = 'nearest'`

### `clp_link_tolerance`

`Scheme.clp_link_tolerance: float = 0.0`

### `ftol`

`Scheme.ftol: float = 1e-08`

### `gtol`

`Scheme.gtol: float = 1e-08`

### `maximum_number_function_evaluations`

`Scheme.maximum_number_function_evaluations: int | None = None`

### `optimization_method`

`Scheme.optimization_method: Literal['TrustRegionReflection', 'Dogbox', 'Levenberg-Marquardt'] = 'TrustRegionReflection'`

### `result_path`

`Scheme.result_path: str | None = None`

### `source_path`

`Scheme.source_path: StrOrPath = 'scheme.yml'`

**xtol**

Scheme.xtol: float = 1e-08

**model**

Scheme.model: Model

**parameters**

Scheme.parameters: Parameters

**data**

Scheme.data: Mapping[str, xr.Dataset]

**Methods Summary**

<i>loader</i>	Create a <i>Scheme</i> instance from the specs defined in a file.
<i>markdown</i>	Format the <i>Scheme</i> as markdown string.
<i>valid</i>	Check if there are no problems with the model or the parameters.
<i>validate</i>	Return a string listing all problems in the model and missing parameters.

**loader**

Scheme.loader(format\_name: str | None = None, \*\*kwargs: Any) → Scheme

Create a *Scheme* instance from the specs defined in a file.

**Parameters**

- **file\_name** (StrOrPath) – File containing the parameter specs.
- **format\_name** (str | None) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (Any) – Additional keyword arguments passes to the load\_scheme implementation of the project io plugin.

**Returns**

*Scheme* instance created from the file.

**Return type**

*Scheme*



## markdown

`Scheme.markdown()`

Format the *Scheme* as markdown string.

**Returns**

The scheme as markdown string.

**Return type**

*MarkdownStr*

## valid

`Scheme.valid() → bool`

Check if there are no problems with the model or the parameters.

**Returns**

Whether the scheme is valid.

**Return type**

`bool`

## validate

`Scheme.validate() → MarkdownStr`

Return a string listing all problems in the model and missing parameters.

**Returns**

A user-friendly string containing all the problems of a model if any. Defaults to 'Your model is valid.' if no problems are found.

**Return type**

*MarkdownStr*

## Methods Documentation

`add_svd: bool = True`

`clp_link_method: Literal['nearest', 'backward', 'forward'] = 'nearest'`

`clp_link_tolerance: float = 0.0`

`data: Mapping[str, xr.Dataset]`

`ftol: float = 1e-08`

`gtol: float = 1e-08`

`loader(format_name: str | None = None, **kwargs: Any) → Scheme`

Create a *Scheme* instance from the specs defined in a file.

**Parameters**

- **file\_name** (*StrOrPath*) – File containing the parameter specs.

- **format\_name** (*str* / *None*) – Format the file is in, if not provided it will be inferred from the file extension.
- **\*\*kwargs** (*Any*) – Additional keyword arguments passes to the `load_scheme` implementation of the project io plugin.

**Returns**

*Scheme* instance created from the file.

**Return type**

*Scheme*

**markdown()**

Format the *Scheme* as markdown string.

**Returns**

The scheme as markdown string.

**Return type**

*MarkdownStr*

**maximum\_number\_function\_evaluations:** *int* | *None* = *None*

**model:** *Model*

**optimization\_method:** *Literal*['TrustRegionReflection', 'Dogbox', 'Levenberg-Marquardt'] = 'TrustRegionReflection'

**parameters:** *Parameters*

**result\_path:** *str* | *None* = *None*

**source\_path:** *StrOrPath* = 'scheme.yml'

**valid()** → *bool*

Check if there are no problems with the model or the parameters.

**Returns**

Whether the scheme is valid.

**Return type**

*bool*

**validate()** → *MarkdownStr*

Return a string listing all problems in the model and missing parameters.

**Returns**

A user-friendly string containing all the problems of a model if any. Defaults to 'Your model is valid.' if no problems are found.

**Return type**

*MarkdownStr*

**xtol:** *float* = 1e-08

### 16.1.11 simulation

Package containing code for simulation of dataset models.

#### Modules

<code>glotaran.simulation.simulation</code>	Functions for simulating a dataset using a global optimization model.
---	---

#### simulation

Functions for simulating a dataset using a global optimization model.

#### Functions

##### Summary

<code>simulate</code>	Simulate a dataset using a model.
<code>simulate_from_clp</code>	Simulate a dataset model from pre-defined conditionally linear parameters.
<code>simulate_full_model</code>	Simulate a dataset model with global megacomplexes.

#### simulate

`glotaran.simulation.simulation.simulate(model: Model, dataset: str, parameters: Parameters, coordinates: dict[str, ArrayLike], clp: xr.DataArray | None = None, noise: bool = False, noise_std_dev: float = 1.0, noise_seed: int | None = None) → xr.Dataset`

Simulate a dataset using a model.

##### Parameters

- **model** ([Model](#)) – The model containing the dataset model.
- **dataset** (*str*) – Label of the dataset to simulate
- **parameters** ([Parameters](#)) – The parameters for the simulation.
- **coordinates** (*dict*[*str*, *ArrayLike*]) – A dictionary with the coordinates used for simulation (e.g. time, wavelengths, ...).
- **clp** (*xr.DataArray* | *None*) – A matrix with conditionally linear parameters (e.g. spectra, pixel intensity, ...). Will be used instead of the dataset's global megacomplexes if not *None*.
- **noise** (*bool*) – Add noise to the simulation.
- **noise\_std\_dev** (*float*) – The standard deviation for noise simulation.
- **noise\_seed** (*int* | *None*) – The seed for the noise simulation.

**Returns**

The simulated dataset.

**Return type**

xr.Dataset

**Raises**

**ValueError** – Raised if dataset model has no global megacomplex and no clp are provided.

### `simulate_from_clp`

```
glotaran.simulation.simulation.simulate_from_clp(dataset_model: DatasetModel,  
                                                  global_dimension: str, global_axis:  
                                                  ArrayLike, model_dimension: str,  
                                                  model_axis: ArrayLike, clp:  
                                                  xr.DataArray) → xr.Dataset
```

Simulate a dataset model from pre-defined conditionally linear parameters.

**Parameters**

- **dataset\_model** ([DatasetModel](#)) – The dataset model to simulate.
- **global\_dimension** (*str*) – The global dimension of the dataset.
- **global\_axis** (*ArrayLike*) – The global axis of the dataset.
- **model\_dimension** (*str*) – The model dimension of the dataset.
- **model\_axis** (*ArrayLike*) – The model axis of the dataset.
- **clp** (*xr.DataArray*) – A matrix with conditionally linear parameters.

**Returns**

The simulated dataset.

**Return type**

xr.Dataset

**Raises**

**ValueError** – Raised if the clp are missing the dimension ‘clp\_label’.

### `simulate_full_model`

```
glotaran.simulation.simulation.simulate_full_model(dataset_model: DatasetModel,  
                                                  global_dimension: str, global_axis:  
                                                  ArrayLike, model_dimension: str,  
                                                  model_axis: ArrayLike) →  
                                                  xr.Dataset
```

Simulate a dataset model with global megacomplexes.

**Parameters**

- **dataset\_model** ([DatasetModel](#)) – The dataset model to simulate.
- **global\_dimension** (*str*) – The global dimension of the dataset.
- **global\_axis** (*ArrayLike*) – The global axis of the dataset.
- **model\_dimension** (*str*) – The model dimension of the dataset.

- **model\_axis** (*ArrayLike*) – The model axis of the dataset.

**Returns**

The simulated dataset.

**Return type**

xr.Dataset

**Raises**

**ValueError** – Raised if at least one of the dataset model’s global megacomplexes is index dependent.

16.1.12 testing

Testing framework package for glotaran itself and plugins.

Modules

<i>glotaran.testing.plugin_system</i>	Mock functionality for the plugin system.
<i>glotaran.testing.simulated_data</i>	Package containing simulated data for testing and quick demos.

plugin\_system

Mock functionality for the plugin system.

Functions

Summary

<i>monkeypatch_plugin_registry</i>	Contextmanager to monkeypatch multiple plugin registries at once.
<i>monkeypatch_plugin_registry_data_io</i>	Monkeypatch the DataIoInterface registry.
<i>monkeypatch_plugin_registry_megacomplex</i>	Monkeypatch the Megacomplex registry.
<i>monkeypatch_plugin_registry_project_io</i>	Monkeypatch the ProjectIoInterface registry.

monkeypatch\_plugin\_registry

```
glotaran.testing.plugin_system.monkeypatch_plugin_registry(*, test_megacomplex:
    MutableMapping[str,
    type[Megacomplex]] |
    None = None,
    test_data_io:
    MutableMapping[str,
    DataIoInterface] | None =
    None, test_project_io:
    MutableMapping[str,
    ProjectIoInterface] | None
    = None,
    create_new_registry: bool
    = False) →
    Generator[None, None,
    None]
```

Contextmanager to monkeypatch multiple plugin registries at once.

#### Parameters

- **test\_megacomplex** (*MutableMapping[str, type[Megacomplex]]*, *optional*) – Registry to to update or replace the Megacomplex registry with. , by default None
- **test\_data\_io** (*MutableMapping[str, DataIoInterface]*, *optional*) – Registry to to update or replace the DataIoInterface registry with. , by default None
- **test\_project\_io** (*MutableMapping[str, ProjectIoInterface]*, *optional*) – Registry to to update or replace the ProjectIoInterface registry with. , by default None
- **create\_new\_registry** (*bool*) – Whether to update the actual registry or create a new one from the arguments. , by default False

#### Yields

*Generator[None, None, None]* – Just keeps all context manager alive

#### See also:

[\*monkeypatch\\_plugin\\_registry\\_megacomplex\*](#), [\*monkeypatch\\_plugin\\_registry\\_data\\_io\*](#),  
[\*monkeypatch\\_plugin\\_registry\\_project\\_io\*](#)

### **monkeypatch\_plugin\_registry\_data\_io**

```
glotaran.testing.plugin_system.monkeypatch_plugin_registry_data_io(test_data_io:
    MutableMap-
    ping[str,
    DataIoInterface]
    | None = None,
    cre-
    ate_new_registry:
    bool = False) →
    Generator[None,
    None, None]
```

Monkeypatch the DataIoInterface registry.

#### Parameters

- **test\_data\_io** (*MutableMapping*[*str*, *DataIoInterface*], *optional*)  
– Registry to to update or replace the *DataIoInterface* registry with. , by default *None*
- **create\_new\_registry** (*bool*) – Whether to update the actual registry or create a new one from *test\_data\_io* , by default *False*

**Yields**

*Generator*[*None*, *None*, *None*] – Just to keep the context alive.

**monkeypatch\_plugin\_registry\_megacomplex**

```
glotaran.testing.plugin_system.monkeypatch_plugin_registry_megacomplex(test_megacomplex:
    Muta-
    bleMap-
    ping[str,
    type[Megacomplex]]
    | None =
    None, create_new_registry:
    bool =
    False) →
    Genera-
    tor[None,
    None,
    None]
```

Monkeypatch the Megacomplex registry.

**Parameters**

- **test\_megacomplex** (*MutableMapping*[*str*, *type*[*Megacomplex*]], *optional*) – Registry to to update or replace the Megacomplex registry with. , by default *None*
- **create\_new\_registry** (*bool*) – Whether to update the actual registry or create a new one from *test\_megacomplex* , by default *False*

**Yields**

*Generator*[*None*, *None*, *None*] – Just to keep the context alive.

## monkeypatch\_plugin\_registry\_project\_io

```
glotaran.testing.plugin_system.monkeypatch_plugin_registry_project_io(test_project_io:  
    MutableMapping[str, ProjectIoInterface] |  
    None = None, create_new_registry:  
    bool = False) →  
    Generator[None, None, None]
```

Monkeypatch the ProjectIoInterface registry.

### Parameters

- **test\_project\_io** (*MutableMapping*[*str*, *ProjectIoInterface*], *optional*) – Registry to to update or replace the ProjectIoInterface registry with. , by default *None*
- **create\_new\_registry** (*bool*) – Whether to update the actual registry or create a new one from *test\_data\_io* , by default *False*

### Yields

*Generator*[*None*, *None*, *None*] – Just to keep the context alive.

## simulated\_data

Package containing simulated data for testing and quick demos.

### Modules

<code>glotaran.testing.simulated_data.parallel_spectral_decay</code>	A simple parallel decay for testing purposes.
<code>glotaran.testing.simulated_data.sequential_spectral_decay</code>	A simple sequential decay for testing purposes.
<code>glotaran.testing.simulated_data.shared_decay</code>	Shared variables for simulated decays.



### parallel\_spectral\_decay

A simple parallel decay for testing purposes.

### sequential\_spectral\_decay

A simple sequential decay for testing purposes.

### shared\_decay

Shared variables for simulated decays.

## 16.1.13 typing

Glottaran specific typing module.

### Modules

<code>glottaran.typing.protocols</code>	Protocol like type definitions.
<code>glottaran.typing.types</code>	Glottaran types module containing commonly used types.

### protocols

Protocol like type definitions.

### Classes

#### Summary

<code>FileLoadableProtocol</code>	Protocol class that a file loadable class need adherer to.
-----------------------------------	--

### FileLoadableProtocol

```
class glottaran.typing.protocols.FileLoadableProtocol(*args, **kwargs)
```

Bases: `Protocol`

Protocol class that a file loadable class need adherer to.

## Attributes Summary

<i>loader</i>
<i>source_path</i>

### loader

`FileLoadableProtocol.loader`: `Callable[[StrOrPath | Sequence[StrOrPath] | Mapping[str, StrOrPath]], FileLoadableProtocol]`

### source\_path

`FileLoadableProtocol.source_path`: `StrOrPath | Sequence[StrOrPath] | Mapping[str, StrOrPath]`

## Methods Summary

### Methods Documentation

`loader`: `Callable[[StrOrPath | Sequence[StrOrPath] | Mapping[str, StrOrPath]], FileLoadableProtocol]`

`source_path`: `StrOrPath | Sequence[StrOrPath] | Mapping[str, StrOrPath]`

## types

Glotaran types module containing commonly used types.

### 16.1.14 utils

Glotaran utility function/class package.

## Modules

<code>glotaran.utils.attrs_helper</code>	Helper functions for attrs.
<code>glotaran.utils.helpers</code>	Module containing general helper functions.
<code>glotaran.utils.io</code>	Glotalan IO utility module.
<code>glotaran.utils.ipython</code>	Glotalan module with utilities for <code>ipython</code> integration (e.g. <code>notebooks</code> ).
<code>glotaran.utils.regex</code>	Glotalan module with regular expression patterns and functions.
<code>glotaran.utils.sanitize</code>	Glotalan module with utilities for sanitation of parsed content.
<code>glotaran.utils.tee</code>	Module containing context manager to capture stdout output and still forward it to stdout.

## attrs\_helper

Helper functions for attrs.

## Functions

### Summary

<code>no_default_vals_in_repr</code>	Class decorator to omits attributes from repr that have their default value.
--------------------------------------	--

### no\_default\_vals\_in\_repr

`glotaran.utils.attrs_helper.no_default_vals_in_repr(cls)`

Class decorator to omits attributes from repr that have their default value.

Needs to be on top of the `attr.define` decorator. Based on: <https://stackoverflow.com/a/47663099/3990615>

#### Parameters

**cls** – Class decorated with `attr.define`.

#### Return type

`type[cls]`

## helpers

Module containing general helper functions.

## Functions

### Summary

<code>nan_or_equal</code>	Compare values which can be nan for equality.
---------------------------	---

### `nan_or_equal`

`glotaran.utils.helpers.nan_or_equal(lhs: Any, rhs: Any) → bool`

Compare values which can be nan for equality.

This helper function is needed because `np.nan == np.nan` returns `False`.

#### Parameters

- **lhs** (*Any*) – Left hand side value.
- **rhs** (*Any*) – Right hand side value.

#### Returns

Whether or not values are equal.

#### Return type

`bool`

## io

Glotaran IO utility module.

## Functions

### Summary

<code>chdir_context</code>	Context manager to change directory to <code>folder_path</code> .
<code>create_clp_guide_dataset</code>	Create dataset for clp guidance.
<code>get_script_dir</code>	Get the parent folder a script is executed in.
<code>load_datasets</code>	Load multiple datasets into a mapping (convenience function).
<code>make_path_absolute_if_relative</code>	Get a path as absolute if relative.
<code>relative_posix_path</code>	Ensure that <code>source_path</code> is a posix path, relative to <code>base_path</code> if defined.
<code>safe_dataframe_fillna</code>	Fill NaN values with <code>fill_value</code> if the column exists or do nothing.
<code>safe_dataframe_replace</code>	Replace column values with <code>replace_value</code> if the column exists or do nothing.

## chdir\_context

`glotaran.utils.io.chdir_context(folder_path: StrOrPath) → Generator[Path, None, None]`

Context manager to change directory to `folder_path`.

### Parameters

**folder\_path** (*StrOrPath*) – Path to change to.

### Yields

*Generator[Path, None, None]* – Resolved path of `folder_path`.

### Raises

**ValueError** – If `folder_path` is an existing file.

## create\_clp\_guide\_dataset

`glotaran.utils.io.create_clp_guide_dataset(result: Result | xr.Dataset, clp_label: str, dataset_name: str | None = None) → xr.Dataset`

Create dataset for clp guidance.

### Parameters

- **result** (*Result* / *xr.Dataset*) – Optimization result object or dataset, created with `pyglotaran` ≥ 0.6.0.
- **clp\_label** (*str*) – Label of the clp to guide.
- **dataset\_name** (*str* / *None*) – Name of dataset to extract the guide from. Defaults to `None`.

### Returns

`DataArray` containing the clp guide, with `clp_label` dimension replaced by the model dimensions first value.

### Return type

`xr.Dataset`

### Raises

- **ValueError** – If `result` is an instance of `Result` and `dataset_name` is `None` or not in `result`.
- **ValueError** – If `clp_labels` is not in `result`.
- **ValueError** – The result dataset was created with `pyglotaran` < 0.6.0.

## Examples

Extracting the clp guide from an optimization result object.

```
from glotaran.io import save_dataset
from glotaran.utils.io import create_clp_guide_dataset

clp_guide = create_clp_guide_dataset(result, "species_1", "dataset_1")
save_dataset(clp_guide, "clp_guide__result_dataset_1__species_1.nc")
```

Extracting the clp guide from a result dataset loaded from file.

```
from glotaran.io import load_dataset
from glotaran.io import save_dataset
from glotaran.utils.io import create_clp_guide_dataset

result_dataset = load_dataset("result_dataset_1.nc")
clp_guide = create_clp_guide_dataset(result_dataset, "species_1")
save_dataset(clp_guide, "clp_guide__result_dataset_1__species_1.nc")
```

## get\_script\_dir

glotaran.utils.io.get\_script\_dir(\*, nesting: *int* = 0) → *Path*

Get the parent folder a script is executed in.

This is a helper function for cross compatibility with jupyter notebooks. In notebooks the global `__file__` variable isn't set, thus we need different means to get the folder a script is defined in, which doesn't change with the current working director the `python` interpreter was called from.  
:param nesting: Number to go up in the call stack to get to the initially calling function.

This is only needed for library code and not for user code. , by default 0 (direct call)

### Returns

Path to the folder the script was resides in.

### Return type

*Path*

## load\_datasets

glotaran.utils.io.load\_datasets(*dataset\_mappable*: *str* | *Path* | *Dataset* | *DataArray* |  
*Sequence*[*str* | *Path* | *Dataset* | *DataArray*] | *Mapping*[*str*, *str* |  
*Path* | *Dataset* | *DataArray*]) → *DatasetMapping*

Load multiple datasets into a mapping (convenience function).

This is used for `file_loadable_field` of a dataset mapping e.g. in `Scheme`

### Parameters

**dataset\_mappable** (*DatasetMappable*) – Single dataset/file path to a dataset or sequence or mapping of it.

### Returns

Mapping of dataset with string keys, where datasets hare ensured to have the `source_path` attr.

### Return type

*DatasetMapping*

### make\_path\_absolute\_if\_relative

`glotaran.utils.io.make_path_absolute_if_relative(path: Path) → Path`

Get a path as absolute if relative.

#### Parameters

**path** (*Path*) – The path to make absolute.

#### Returns

Either the original path or the path as absolute relative to the script directory.

#### Return type

*Path*

### relative\_posix\_path

`glotaran.utils.io.relative_posix_path(source_path: StrOrPath, base_path: StrOrPath | None = None) → str`

Ensure that `source_path` is a posix path, relative to `base_path` if defined.

For `source_path` to be converted to a relative path it either needs to be an absolute path or `base_path` needs to be a parent directory of `source_path`. On Windows if `source_path` and `base_path` are on different drives, it will return the absolute posix path to the file.

#### Parameters

- **source\_path** (*StrOrPath*) – Path which should be converted to a relative posix path.
- **base\_path** (*StrOrPath*, *optional*) – Base path the resulting path string should be relative to. Defaults to `None`.

#### Returns

`source_path` as posix path relative to `base_path` if defined.

#### Return type

*str*

### safe\_dataframe\_fillna

`glotaran.utils.io.safe_dataframe_fillna(df: pd.DataFrame, column_name: str, fill_value: Any) → None`

Fill NaN values with `fill_value` if the column exists or do nothing.

#### Parameters

- **df** (*pd.DataFrame*) – DataFrame from which specific column values will be replaced
- **column\_name** (*str*) – Name of column of `df` to fill NaNs
- **fill\_value** (*Any*) – Value to fill NaNs with

### safe\_dataframe\_replace

```
glotaran.utils.io.safe_dataframe_replace(df: pd.DataFrame, column_name: str,  
                                         to_be_replaced_values: Any, replace_value: Any)  
                                         → None
```

Replace column values with `replace_value` if the column exists or do nothing.

If `to_be_replaced_values` is not list or tuple format, convert into list with same `to_be_replaced_values` as element.

#### Parameters

- **df** (*pd.DataFrame*) – DataFrame from which specific column values will be replaced
- **column\_name** (*str*) – Name of column of df to replace values for
- **to\_be\_replaced\_values** (*Any*) – Values to be replaced
- **replace\_value** (*Any*) – Value to replace `to_be_replaced_values` with

## Classes

### Summary

<i>DatasetMapping</i>	Wrapper class for a mapping of datasets which can be used for a <code>file_loadable_field</code> .
-----------------------	--

### DatasetMapping

```
class glotaran.utils.io.DatasetMapping(init_map: Mapping[str, Dataset] | None = None)
```

Bases: `MutableMapping`

Wrapper class for a mapping of datasets which can be used for a `file_loadable_field`.

Initialize an instance of *DatasetMapping*.

#### Parameters

**init\_dict** (*dict[str, xr.Dataset] | None*) – Mapping to initially populate the instance. Defaults to `None`.

### Attributes Summary

<i>source_path</i>	Map the <code>source_path</code> attribute of each dataset to a standalone mapping.
--------------------	---



source\_path

DatasetMapping.source\_path

Map the source\_path attribute of each dataset to a standalone mapping.

**Note:** When the source\_path attribute of the dataset gets updated (e.g. by calling save\_dataset with the default update\_source\_path=True) this value will be updated as well.

Returns

Mapping of the dataset source paths.

Return type

Mapping[str, str]

Methods Summary

<i>clear</i>	
<i>get</i>	
<i>items</i>	
<i>keys</i>	
<i>loader</i>	Loader function utilized by file_loadable_field.
<i>pop</i>	If key is not found, d is returned if given, otherwise KeyError is raised.
<i>popitem</i>	as a 2-tuple; but raise KeyError if D is empty.
<i>setdefault</i>	
<i>update</i>	If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v
<i>values</i>	

**clear**

`DatasetMapping.clear()` → None. Remove all items from D.

**get**

`DatasetMapping.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

**items**

`DatasetMapping.items()` → a set-like object providing a view on D's items

**keys**

`DatasetMapping.keys()` → a set-like object providing a view on D's keys

**loader**

**classmethod** `DatasetMapping.loader(dataset_mappable: str | Path | Dataset | DataArray | Sequence[str | Path | Dataset | DataArray] | Mapping[str, str | Path | Dataset | DataArray])` → *DatasetMapping*

Loader function utilized by `file_loadable_field`.

**Parameters**

**dataset\_mappable** (*DatasetMappable*) – Mapping of datasets to initialize *DatasetMapping*.

**Returns**

Populated instance of *DatasetMapping*.

**Return type**

*DatasetMapping*

**pop**

`DatasetMapping.pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised.

## popitem

`DatasetMapping.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

## setdefault

`DatasetMapping.setdefault(k[, d])` → `D.get(k,d)`, also set `D[k]=d` if k not in D

## update

`DatasetMapping.update([E, ]**F)` → None. Update D from mapping/iterable E and F.  
 If E present and has a `.keys()` method, does: for k in E: `D[k] = E[k]` If E present and lacks `.keys()` method, does: for (k, v) in E: `D[k] = v` In either case, this is followed by: for k, v in `F.items()`: `D[k] = v`

## values

`DatasetMapping.values()` → an object providing a view on D's values

## Methods Documentation

`clear()` → None. Remove all items from D.

`get(k[, d])` → `D[k]` if k in D, else d. d defaults to None.

`items()` → a set-like object providing a view on D's items

`keys()` → a set-like object providing a view on D's keys

**classmethod loader**(*dataset\_mappable*: *str* | *Path* | *Dataset* | *dataArray* | *Sequence*[*str* | *Path* | *Dataset* | *dataArray*] | *Mapping*[*str*, *str* | *Path* | *Dataset* | *dataArray*]) → *DatasetMapping*

Loader function utilized by `file_loadable_field`.

### Parameters

**dataset\_mappable** (*DatasetMappable*) – Mapping of datasets to initialize *DatasetMapping*.

### Returns

Populated instance of *DatasetMapping*.

### Return type

*DatasetMapping*

`pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised.

`popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

**setdefault**(*k*[, *d*]) → *D*.get(*k*,*d*), also set *D*[*k*]=*d* if *k* not in *D*

**property source\_path**

Map the `source_path` attribute of each dataset to a standalone mapping.

---

**Note:** When the `source_path` attribute of the dataset gets updated (e.g. by calling `save_dataset` with the default `update_source_path=True`) this value will be updated as well.

---

**Returns**

Mapping of the dataset source paths.

**Return type**

Mapping[*str*, *str*]

**update**([*E*, ]\*\**F*) → None. Update *D* from mapping/iterable *E* and *F*.

If *E* present and has a `.keys()` method, does: for *k* in *E*: *D*[*k*] = *E*[*k*] If *E* present and lacks `.keys()` method, does: for (*k*, *v*) in *E*: *D*[*k*] = *v* In either case, this is followed by: for *k*, *v* in *F*.items(): *D*[*k*] = *v*

**values**() → an object providing a view on *D*'s values

## ipython

Glotaran module with utilities for `ipython` integration (e.g. notebooks).

## Functions

### Summary

<code>display_file</code>	Display a file with syntax highlighting <code>syntax</code> .
---------------------------	---

### display\_file

`glotaran.utils.ipython.display_file(path: str | Path, *, syntax: str | None = None)` → *MarkdownStr*

Display a file with syntax highlighting `syntax`.

**Parameters**

- **path** (*StrOrPath*) – Paths to the file
- **syntax** (*str* | *None*) – Syntax used for syntax highlighting. Defaults to *None* which means that the syntax is inferred based on the file extension. Pass the value `""` to deactivate syntax highlighting.

**Returns**

File content with syntax highlighting to render in `ipython`.

**Return type**

*MarkdownStr*

Classes

Summary

<i>MarkdownStr</i>	String wrapper class for rich display integration of markdown in ipython.
--------------------	---

MarkdownStr

**class** glotaran.utils.ipython.**MarkdownStr**(*wrapped\_str*: *str*, \*, *syntax*: *str* | *None* = *None*)  
Bases: *UserString*

String wrapper class for rich display integration of markdown in ipython.  
Initialize string class that is automatically displayed as markdown by ipython.

Parameters

- **wrapped\_str** (*str*) – String to be wrapped.
- **syntax** (*str*) – Syntax highlighting which should be applied, by default None

**Note:** Possible syntax highlighting values can e.g. be found here: <https://support.codebasehq.com/articles/tips-tricks/syntax-highlighting-in-markdown>

Methods Summary

<i>capitalize</i>	
<i>casefold</i>	
<i>center</i>	
<i>count</i>	
<i>encode</i>	
<i>endswith</i>	
<i>expandtabs</i>	
<i>find</i>	
<i>format</i>	
<i>format_map</i>	
<i>index</i>	Raises ValueError if the value is not present.

continues on next page

Table 1 – continued from previous page

<i>isalnum</i>	
<i>isalpha</i>	
<i>isascii</i>	
<i>isdecimal</i>	
<i>isdigit</i>	
<i>isidentifier</i>	
<i>islower</i>	
<i>isnumeric</i>	
<i>isprintable</i>	
<i>isspace</i>	
<i>istitle</i>	
<i>isupper</i>	
<i>join</i>	
<i>ljust</i>	
<i>lower</i>	
<i>lstrip</i>	
<i>maketrans</i>	Return a translation table usable for <code>str.translate()</code> .
<i>partition</i>	
<i>removeprefix</i>	
<i>removesuffix</i>	
<i>replace</i>	
<i>rfind</i>	
<i>rindex</i>	
<i>rjust</i>	
<i>rpartition</i>	
<i>rsplit</i>	

continues on next page

Table 1 – continued from previous page

<i>rstrip</i>
<i>split</i>
<i>splitlines</i>
<i>startswith</i>
<i>strip</i>
<i>swapcase</i>
<i>title</i>
<i>translate</i>
<i>upper</i>
<i>zfill</i>

**capitalize**

MarkdownStr.**capitalize**()

**casefold**

MarkdownStr.**casefold**()

**center**

MarkdownStr.**center**(width, \*args)

**count**

MarkdownStr.**count**(value) → integer -- return number of occurrences of value

**encode**

MarkdownStr.**encode**(encoding='utf-8', errors='strict')

**endswith**

MarkdownStr.**endswith**(*suffix*, *start*=0, *end*=9223372036854775807)

**expandtabs**

MarkdownStr.**expandtabs**(*tabsize*=8)

**find**

MarkdownStr.**find**(*sub*, *start*=0, *end*=9223372036854775807)

**format**

MarkdownStr.**format**(\**args*, \*\**kws*)

**format\_map**

MarkdownStr.**format\_map**(*mapping*)

**index**

MarkdownStr.**index**(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

**isalnum**

MarkdownStr.**isalnum**()

**isalpha**

MarkdownStr.**isalpha**()

**isascii**

MarkdownStr.**isascii**()



**isdecimal**

MarkdownStr.**isdecimal**()

**isdigit**

MarkdownStr.**isdigit**()

**isidentifier**

MarkdownStr.**isidentifier**()

**islower**

MarkdownStr.**islower**()

**isnumeric**

MarkdownStr.**isnumeric**()

**isprintable**

MarkdownStr.**isprintable**()

**isspace**

MarkdownStr.**isspace**()

**istitle**

MarkdownStr.**istitle**()

**isupper**

MarkdownStr.**isupper**()

**join**

MarkdownStr.**join**(*seq*)

**ljust**

MarkdownStr.**ljust**(*width*, *\*args*)

**lower**

MarkdownStr.**lower**()

**lstrip**

MarkdownStr.**lstrip**(*chars=None*)

**maketrans**

MarkdownStr.**maketrans**(*x*, *y=<unrepresentable>*, *z=<unrepresentable>*, /)

Return a translation table usable for str.translate().

If there is only one argument, it must be a dictionary mapping Unicode ordinals (integers) or characters to Unicode ordinals, strings or None. Character keys will be then converted to ordinals. If there are two arguments, they must be strings of equal length, and in the resulting dictionary, each character in x will be mapped to the character at the same position in y. If there is a third argument, it must be a string, whose characters will be mapped to None in the result.

**partition**

MarkdownStr.**partition**(*sep*)

**removeprefix**

MarkdownStr.**removeprefix**(*prefix*, /)

**removesuffix**

MarkdownStr.**removesuffix**(*suffix*, /)

**replace**

MarkdownStr.**replace**(*old*, *new*, *maxsplit=-1*)

**rfind**

MarkdownStr.**rfind**(*sub*, *start=0*, *end=9223372036854775807*)

**rindex**

MarkdownStr.**rindex**(*sub*, *start=0*, *end=9223372036854775807*)

**rjust**

MarkdownStr.**rjust**(*width*, *\*args*)

**rpartition**

MarkdownStr.**rpartition**(*sep*)

**rsplit**

MarkdownStr.**rsplit**(*sep=None*, *maxsplit=-1*)

**rstrip**

MarkdownStr.**rstrip**(*chars=None*)

**split**

MarkdownStr.**split**(*sep=None*, *maxsplit=-1*)

**splitlines**

MarkdownStr.**splitlines**(*keepends=False*)

**startswith**

MarkdownStr.**startswith**(*prefix*, *start*=0, *end*=9223372036854775807)

**strip**

MarkdownStr.**strip**(*chars*=None)

**swapcase**

MarkdownStr.**swapcase**()

**title**

MarkdownStr.**title**()

**translate**

MarkdownStr.**translate**(\**args*)

**upper**

MarkdownStr.**upper**()

**zfill**

MarkdownStr.**zfill**(*width*)

**Methods Documentation**

**capitalize**()

**casefold**()

**center**(*width*, \**args*)

**count**(*value*) → integer -- return number of occurrences of value

**encode**(*encoding*='utf-8', *errors*='strict')

**endswith**(*suffix*, *start*=0, *end*=9223372036854775807)

**expandtabs**(*tabsize*=8)

**find**(*sub*, *start*=0, *end*=9223372036854775807)

**format**(\*args, \*\*kwargs)

**format\_map**(mapping)

**index**(value[, start[, stop]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

**isalnum**()

**isalpha**()

**isascii**()

**isdecimal**()

**isdigit**()

**isidentifier**()

**islower**()

**isnumeric**()

**isprintable**()

**isspace**()

**istitle**()

**isupper**()

**join**(seq)

**ljust**(width, \*args)

**lower**()

**lstrip**(chars=None)

**maketrans**(x, y=<unrepresentable>, z=<unrepresentable>, /)

Return a translation table usable for str.translate().

If there is only one argument, it must be a dictionary mapping Unicode ordinals (integers) or characters to Unicode ordinals, strings or None. Character keys will be then converted to ordinals. If there are two arguments, they must be strings of equal length, and in the resulting dictionary, each character in x will be mapped to the character at the same position in y. If there is a third argument, it must be a string, whose characters will be mapped to None in the result.

**partition**(sep)

**removeprefix**(prefix, /)

**removesuffix**(suffix, /)

**replace**(old, new, maxsplit=-1)

**rfind**(sub, start=0, end=9223372036854775807)

```
rindex(sub, start=0, end=9223372036854775807)
rjust(width, *args)
rpartition(sep)
rsplit(sep=None, maxsplit=-1)
rstrip(chars=None)
split(sep=None, maxsplit=-1)
splitlines(keepends=False)
startswith(prefix, start=0, end=9223372036854775807)
strip(chars=None)
swapcase()
title()
translate(*args)
upper()
zfill(width)
```

## regex

Glotaran module with regular expression patterns and functions.

## Classes

### Summary

<i>RegexPattern</i>	An 'Enum' of (compiled) regular expression patterns (rp).
---------------------	---

### RegexPattern

```
class glotaran.utils.regex.RegexPattern
```

Bases: `object`

An 'Enum' of (compiled) regular expression patterns (rp).

## Attributes Summary

<code>elements_in_string_of_list</code>
<code>group</code>
<code>list_with_tuples</code>
<code>number</code>
<code>number_scientific</code>
<code>optimization_stdout</code>
<code>tuple_number</code>
<code>tuple_word</code>
<code>word</code>

### elements\_in\_string\_of\_list

RegexPattern.elements\_in\_string\_of\_list: **Pattern** =  
`re.compile('(\\(.(+?\\)|[-+\\.\\d]+)')`

### group

RegexPattern.group: **Pattern** = `re.compile('(\\(.(+?\\))')`

### list\_with\_tuples

RegexPattern.list\_with\_tuples: **Pattern** = `re.compile('(\\[[.+\\(.(+\\)\\.+\\]])')`

### number

RegexPattern.number: **Pattern** = `re.compile('[\\d.+-]+')`

**number\_scientific**

```
RegexPattern.number_scientific: Pattern =
re.compile('[+-]?[0-9]*\\.?[0-9]+([eE][+-]?[0-9]+)')
```

## optimization\_stdout

```
RegexPattern.optimization_stdout: Pattern =
re.compile('^\s+(?P<iteration>\d+)\s+(?P<nfev>\d+)\s+(?P<cost>\d\.\.\d+e[+-]\d+)\s+(\s+(?P<cost_reduction>\d\.\.\d+e[+-]\d+)\s+(?P<step_norm>\d\.\.\d+e[+-]\d+)\s+)\s+(?P<optimality>\d\.\.\d+e[+-]\d+),
re.MULTILINE)
```

**tuple\_number**

```
RegexPattern.tuple_number: Pattern =
re.compile('(\\([\\s\\d.+]+?[\\s\\d.+]*?\\))')
```

**tuple\_word**

```
RegexPattern.tuple_word: Pattern =
re.compile('(\\([\\.\\s\\w\\d]+?[,.\\s\\w\\d]*?\\))')
```

**word**

```
RegexPattern.word: Pattern = re.compile('[\w]+')
```

## Methods Summary

## Methods Documentation

```
elements_in_string_of_list: Pattern = re.compile('(\\"(.+?"\\")|[-+\\.\\d]+)')
```

```
group: Pattern = re.compile('(\\"(.+?\\"))')
```

```
list_with_tuples: Pattern = re.compile('(\\[.+\\(\\.+\\)\\.+\\])')
```

```
number: Pattern = re.compile('[\\d.+\\-]+')
```

```
number_scientific: Pattern =  
re.compile('[-+]?[0-9]*\\.?[0-9]+([eE] [-+]?[0-9]+)')
```



```

optimization_stdout: Pattern =
re.compile('^\\s+(?P<iteration>\\d+)\\s+(?P<nfev>\\d+)\\s+(?P<cost>\\d\\.\\.\\d+e[+-]\\d+)(\\s+(?P<cost_reduction>\\d\\.\\.\\d+e[+-]\\d+)\\s+(?P<step_norm>\\d\\.\\.\\d+e[+-]\\d+)|\\s+)\\s+(?P<optimality>\\d\\.\\.\\d+e[+-]\\d+),
re.MULTILINE)

tuple_number: Pattern = re.compile('\\(\\([\\s\\d.+-]?[,\\s\\d.+-]?*\\)\\)')
tuple_word: Pattern = re.compile('\\(\\([\\s\\w\\d]+?[,\\.\\s\\w\\d]*?\\)\\)')
word: Pattern = re.compile('[\\w]+')

```

## sanitize

Glotaran module with utilities for sanitation of parsed content.

## Functions

### Summary

<i>convert_scientific_to_float</i>	Convert value to float if it matches scientific notation string.
<i>list_string_to_tuple</i>	Convert a list of strings (representing tuples) to a list of tuples.
<i>pretty_format_numerical</i>	Format value with with at most <code>decimal_places</code> decimal places.
<i>sanitize_dict_keys</i>	Sanitize the stringified tuple dict keys in a yaml parsed dict.
<i>sanitize_dict_values</i>	Sanitizes a dict with broken tuples inside modifying it in-place.
<i>sanitize_list_with_broken_tuples</i>	Sanitize a list with 'broken' tuples.
<i>sanitize_parameter_list</i>	Replace in a list strings matching scientific notation with floats.
<i>sanitize_yaml</i>	Sanitize a yaml-returned dict for key or (list) values containing tuples.
<i>sanity_scientific_notation_conversion_string_to_tuple</i>	Convert scientific notation string values to floats. Convert a string to a tuple if it matches a tuple pattern.

### convert\_scientific\_to\_float

`glotaran.utils.sanitize.convert_scientific_to_float(value: str) → float | str`

Convert value to float if it matches scientific notation string.

#### Parameters

**value** (*str*) – value to convert from string to float if it matches scientific notation

#### Returns

return float if value was scientific notation string, else turn original value

#### Return type

float | string

### list\_string\_to\_tuple

```
glotaran.utils.sanitize.list_string_to_tuple(a_list: list[str]) → list[tuple[float, ...] |  
tuple[str, ...] | float | str]
```

Convert a list of strings (representing tuples) to a list of tuples.

#### Parameters

**a\_list** (*List*[*str*]) – A list of strings, some of them representing (numbered) tuples

#### Returns

A list of the (numbered) tuples represted by the incoming a\_list

#### Return type

List[Union[float, str]]

### pretty\_format\_numerical

```
glotaran.utils.sanitize.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.

#### 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

### sanitize\_dict\_keys

```
glotaran.utils.sanitize.sanitize_dict_keys(d: dict) → dict
```

Sanitize the stringified tuple dict keys in a yaml parsed dict.

**Keys representing a tuple, e.g. '(s1, s2)' are converted to a tuple of strings**  
e.g. ('s1', 's2')

#### Parameters

**d** (*dict*) – A dict containing tuple-like string keys

#### Returns

A dict with tuple-like string keys converted to tuple keys

#### Return type

dict

## sanitize\_dict\_values

glotaran.utils.sanitize.**sanitize\_dict\_values**(*d*: *dict*[*str*, *Any*] | *list*[*Any*])

Sanitizes a dict with broken tuples inside modifying it in-place.

Broken tuples are tuples that are turned into strings by the yaml parser. This functions calls *sanitize\_list\_with\_broken\_tuples* to glue the broken strings together and then calls *list\_to\_tuple* to turn the list with tuple strings back to number tuples.

### Parameters

**d** (*dict*) – A (complex) dict containing (possibly nested) values of broken tuple strings.

## sanitize\_list\_with\_broken\_tuples

glotaran.utils.sanitize.**sanitize\_list\_with\_broken\_tuples**(*mangled\_list*: *list*[*str* | *float*])  
→ *list*[*str*]

Sanitize a list with ‘broken’ tuples.

A list of broken tuples as returned by yaml when parsing tuples. e.g parsing the list of tuples [(3,100), (4,200)] results in a list of str [‘(3’, ‘100)’, ‘(4’, ‘200)’] which can be restored to a list with the tuples restored as strings [‘(3, 100)’, ‘(4, 200)’]

### Parameters

**mangled\_list** (*List*[*Union*[*str*, *float*]]) – A list with strings representing tuples broken up by round brackets.

### Returns

A list containing the restores tuples (in string form) which can be converted back to numbered tuples using *list\_string\_to\_tuple*

### Return type

*List*[*str*]

## sanitize\_parameter\_list

glotaran.utils.sanitize.**sanitize\_parameter\_list**(*parameter\_list*: *list*[*str* | *float*]) → *list*[*str* | *float*]

Replace in a list strings matching scientific notation with floats.

### Parameters

**parameter\_list** (*list*) – A list of parameters where some elements may be strings like 1E7

### Returns

A list where strings matching a scientific number have been converted to float

### Return type

*list*

## sanitize\_yaml

glotaran.utils.sanitize.**sanitize\_yaml**(*d*: *dict*, *do\_keys*: *bool* = *True*, *do\_values*: *bool* = *False*) → *dict*

Sanitize a yaml-returned dict for key or (list) values containing tuples.

### Parameters

- **d** (*dict*) – a dict resulting from parsing a pyglotaran model spec yaml file
- **do\_keys** (*bool*) – toggle sanitization of dict keys, by default True
- **do\_values** (*bool*) – toggle sanitization of dict values, by default False

### Returns

a sanitized dict with (broken) string tuples restored as proper tuples

### Return type

*dict*

## sanity\_scientific\_notation\_conversion

glotaran.utils.sanitize.**sanity\_scientific\_notation\_conversion**(*d*: *dict*[*str*, *Any*] | *list*[*Any*])

Convert scientific notation string values to floats.

### Parameters

**d** (*dict*[*str*, *Any*] | *list*[*Any*]) – Iterable which should be checked for scientific notation values.

## string\_to\_tuple

glotaran.utils.sanitize.**string\_to\_tuple**(*tuple\_str*: *str*, *from\_list*=*False*) → *tuple*[*float*, ...] | *tuple*[*str*, ...] | *float* | *str*

Convert a string to a tuple if it matches a tuple pattern.

### Parameters

- **tuple\_str** (*str*) – A string representing some tuple to convert the numbers inside the string tuple are mapped to float
- **from\_list** (*bool*, *optional*) – only if true will a single number string be converted to float, otherwise returned as-is since it may represent a label, by default False

### Returns

Returns the tuple intended by the string

### Return type

*tuple*[*float*], *tuple*[*str*], *float*, *str*

## tee

Module containing context manager to capture stdout output and still forward it to stdout.

## Classes

### Summary

<i>TeeContext</i>	Context manager that allows to work with string written to stdout.
-------------------	--

### TeeContext

**class** `glotaran.utils.tee.TeeContext`

Bases: `object`

Context manager that allows to work with string written to stdout.

This context manager behaves similar to the `tee` shell command. <https://linuxize.com/post/linux-tee-command>

Create new `StringIO` buffer and save reference to original `sys.stdout`.

### Methods Summary

<i>flush</i>	Flush values in the buffer.
<i>read</i>	Return values written to buffer.
<i>write</i>	Write to buffer and original <code>sys.stdout</code> .

### flush

`TeeContext.flush()` → `None`

Flush values in the buffer.

### read

`TeeContext.read()` → `str`

Return values written to buffer.

#### Returns

Text written to buffer.

#### Return type

`str`

**write**

`TeeContext.write(data: str) → None`

Write to buffer and original `sys.stdout`.

**Parameters**

**data** (*str*) – String to write to stdout and buffer.

**Methods Documentation**

**flush()** → *None*

Flush values in the buffer.

**read()** → *str*

Return values written to buffer.

**Returns**

Text written to buffer.

**Return type**

*str*

**write(data: str) → None**

Write to buffer and original `sys.stdout`.

**Parameters**

**data** (*str*) – String to write to stdout and buffer.

## PLUGIN DEVELOPMENT

If you don't find the plugin that fits your needs you can always write your own. This sections will explain you how and what you need to know.

In time we will also provide you with a [cookiecutter](#) template, to kickstart your new plugin for publishing as a package on PyPi.

The following section was generated from docs/source/notebooks/plugin\_system/plugin\_howto\_write\_a\_io\_plugin.ipynb .....

### 17.1 How to Write your own io plugin

There are all kinds of different data formats, so it is quite likely that your experimental setup uses a format which isn't yet supported by a `glotaran` plugin and want to write your own `DataIo` plugin to support this format.

Since `json` is very common format (admittedly not for data, but in general) and python has builtin support for it we will use it as an example.

First let's have a look which `DataIo` plugins are already installed and which functions they support.

```
[1]: from glotaran.io import data_io_plugin_table
```

```
[2]: data_io_plugin_table()
```

```
[2]:
```

Format name	load_dataset	save_dataset
ascii	*	*
nc	*	*
sdt	*	/

Looks like there isn't a `json` plugin installed yet, but maybe someone else did already write one, so have a look at the `3rd party plugins` list in the user documentation <[https://pyglotaran.readthedocs.io/en/latest/user\\_documentation/using\\_plugins.html](https://pyglotaran.readthedocs.io/en/latest/user_documentation/using_plugins.html)>`\_\_` before you start writing your own plugin.

For the sake of the example, we will write our `json` plugin even if there already exists one by the time you read this.

First you need to import all needed libraries and functions.

- `from __future__ import annotations`: needed to write python 3.10 typing syntax (`|`), even with a lower python version
- `json,xarray`: Needed for reading and writing itself
- `DataIoInterface`: needed to subclass from, this way you get the proper type and especially signature checking
- `register_data_io`: registers the `DataIo` plugin under the given `format_names`

```
[3]: from __future__ import annotations

import json

import xarray as xr

from glotaran.io.interface import DataIoInterface
from glotaran.plugin_system.data_io_registration import register_data_io
```

DataIoInterface has two methods we could implement `load_dataset` and `save_dataset`, which are used by the identically named functions in `glotaran.io`.

We will just implement both for our example to be complete. the quickest way to get started is to just copy over the code from DataIoInterface which already has the right signatures and some boilerplate docstrings, for the method arguments.

If the default arguments aren't enough for your plugin and you need your methods to have additional option, you can just add those. Note the `*` between `file_name` and `my_extra_option`, this tell python that `my_extra_option` is an `keyword only argument` and ``mypy` <https://github.com/python/mypy>` won't raise an `[override]` type error for changing the signature of the method. To help others who might use your plugin and your future self, it is good practice to documents what each parameter does in the methods docstring, which will be accessed by the help function.

Finally add the `@register_data_io` with the `format_name`'s you want to register the plugin to, in our case `json` and `my_json`.

Pro tip: You don't need to implement the whole functionality inside of the method itself,

```
[4]: @register_data_io(["json", "my_json"])
class JsonDataIo(DataIoInterface):
    """My new shiny glotaran plugin for json data io"""

    def load_dataset(
        self, file_name: str, *, my_extra_option: str = None
    ) -> xr.Dataset | xr.DataArray:
        """Read json data to xarray.Dataset

        Parameters
        -----
        file_name : str
            File containing the data.
        my_extra_option: str
            This argument is only for demonstration
        """
        if my_extra_option is not None:
            print(f"Using my extra option loading json: {my_extra_option}")

        with open(file_name) as json_file:
            data_dict = json.load(json_file)
        return xr.Dataset.from_dict(data_dict)

    def save_dataset(
        self, dataset: xr.Dataset | xr.DataArray, file_name: str, *, my_extra_option=None
    ):
        """Write xarray.Dataset to a json file
```

(continues on next page)



(continued from previous page)

```

Parameters
-----
dataset : xr.Dataset
    Dataset to be saved to file.
file_name : str
    File to write the result data to.
my_extra_option: str
    This argument is only for demonstration
"""
if my_extra_option is not None:
    print(f"Using my extra option for writing json: {my_extra_option}")

data_dict = dataset.to_dict()
with open(file_name, "w") as json_file:
    json.dump(data_dict, json_file)

```

Let's verify that our new plugin was registered successfully under the `format_names` `json` and `my_json`.

```
[5]: data_io_plugin_table()
```

```
[5]:
```

Format name	load_dataset	save_dataset
ascii	*	*
json	*	*
my_json	*	*
nc	*	*
sdt	*	/

Now let's use the example data from the quickstart to test the reading and writing capabilities of our plugin.

```
[6]: from glotaran.io import load_dataset
      from glotaran.io import save_dataset
      from glotaran.testing.simulated_data.sequential_spectral_decay import DATASET as dataset
```

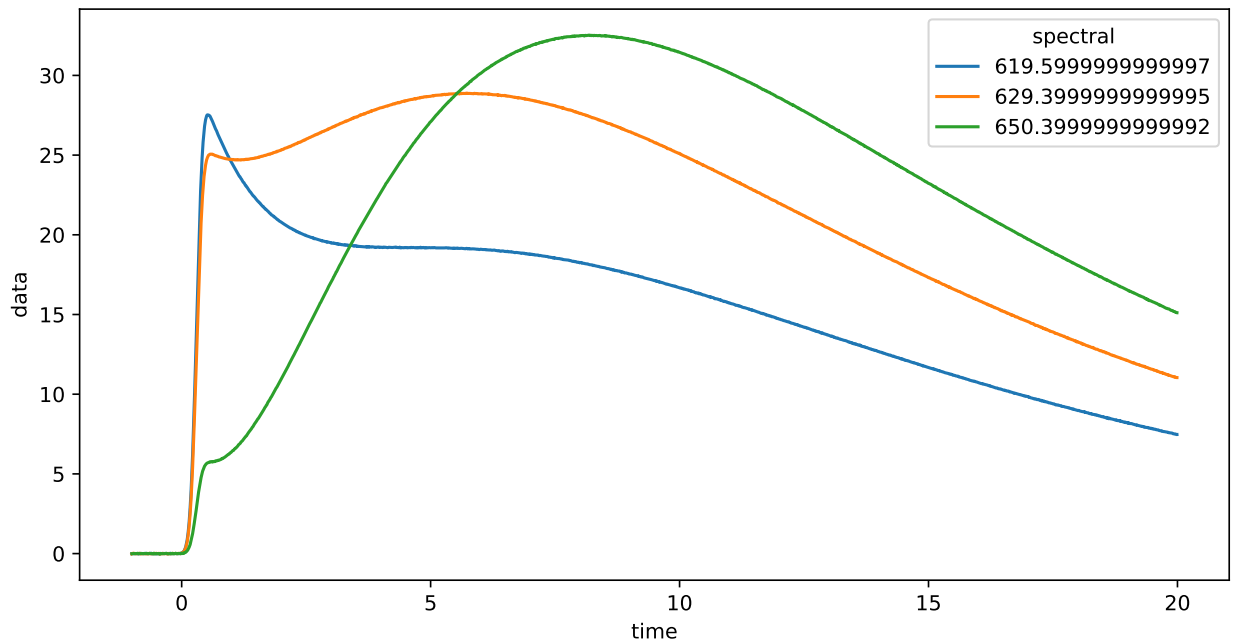
```
[7]: dataset
```

```
[7]: <xarray.Dataset>
Dimensions:  (time: 2100, spectral: 72)
Coordinates:
  * time      (time) float64 -1.0 -0.99 -0.98 -0.97 ... 19.96 19.97 19.98 19.99
  * spectral  (spectral) float64 600.0 601.4 602.8 604.2 ... 696.6 698.0 699.4
Data variables:
  data        (time, spectral) float64 0.00423 -0.01779 ... 2.551 2.323
Attributes:
  source_path:  dataset_1.nc
```

To get a feeling for our data, let's plot some traces.

```
[8]: plot_data = dataset.data.sel(spectral=[620, 630, 650], method="nearest")
      plot_data.plot.line(x="time", aspect=2, size=5)
```

```
[8]: [<matplotlib.lines.Line2D at 0x7fe4d894f5e0>,
      <matplotlib.lines.Line2D at 0x7fe4d894f610>,
      <matplotlib.lines.Line2D at 0x7fe4d894f700>]
```



Since we want to see a difference of our saved and loaded data, we divide the amplitudes by 2 for no reason.

```
[9]: dataset["data"] = dataset.data / 2
```

Now that we changed the data, let's write them to a file.

But in which order were the arguments again? And are there any additional option?

Good thing we documented our new plugin, so we can just lookup the help.

```
[10]: from glotaran.io import show_data_io_method_help

show_data_io_method_help("json", "save_dataset")

Help on method save_dataset in module __main__:

save_dataset(dataset: 'xr.Dataset | xr.DataArray', file_name: 'str', *, my_extra_
↪option=None) method of __main__.JsonDataIo instance
    Write xarray.Dataset to a json file

Parameters
-----
dataset : xr.Dataset
    Dataset to be saved to file.
file_name : str
    File to write the result data to.
my_extra_option: str
    This argument is only for demonstration
```

Note that the **function** `save_dataset` has additional arguments:

- `format_name`: overwrites the inferred plugin selection
- `allow_overwrite`: Allows to overwrite existing files (**USE WITH CAUTION!!!**)

```
[11]: help(save_dataset)
```

```
Help on function save_dataset in module glotaran.plugin_system.data_io_registration:

save_dataset(dataset: 'xr.Dataset | xr.DataArray', file_name: 'StrOrPath', format_name:
↳ 'str | None' = None, *, data_filters: 'list[str] | None' = None, allow_overwrite: 'bool
↳ ' = False, update_source_path: 'bool' = True, **kwargs: 'Any') -> 'None'
    Save data from :xarraydoc:`Dataset` or :xarraydoc:`DataArray` to a file.

Parameters
-----
dataset : xr.Dataset | xr.DataArray
    Data to be written to file.
file_name : StrOrPath
    File to write the data to.
format_name : str
    Format the file should be in, if not provided it will be inferred from the file_
↳ extension.
data_filters : list[str] | None
    Optional list of items in the dataset to be saved.
allow_overwrite : bool
    Whether or not to allow overwriting existing files, by default False
update_source_path: bool
    Whether or not to update the ``source_path`` attribute to ``file_name`` when_
↳ saving.
    by default True
**kwargs : Any
    Additional keyword arguments passes to the ``write_dataset`` implementation
    of the data io plugin. If you aren't sure about those use ``get_datasaver``
    to get the implementation with the proper help and autocomplete.
```

Since this is just an example and we don't overwrite important data we will use `allow_overwrite=True`. Also it makes writing this documentation easier, not having to manually delete the test file each time you run the cell.

```
[12]: save_dataset(
    dataset, "half_intensity.json", allow_overwrite=True, my_extra_option="just as an_
↳ example"
)
```

Using my extra option for writing json: just as an example

Now let's test our data loading functionality.

```
[13]: reloaded_data = load_dataset("half_intensity.json", my_extra_option="just as an example")
reloaded_data
```

Using my extra option loading json: just as an example

```
[13]: <xarray.Dataset>
Dimensions:  (time: 2100, spectral: 72)
```

(continues on next page)

(continued from previous page)

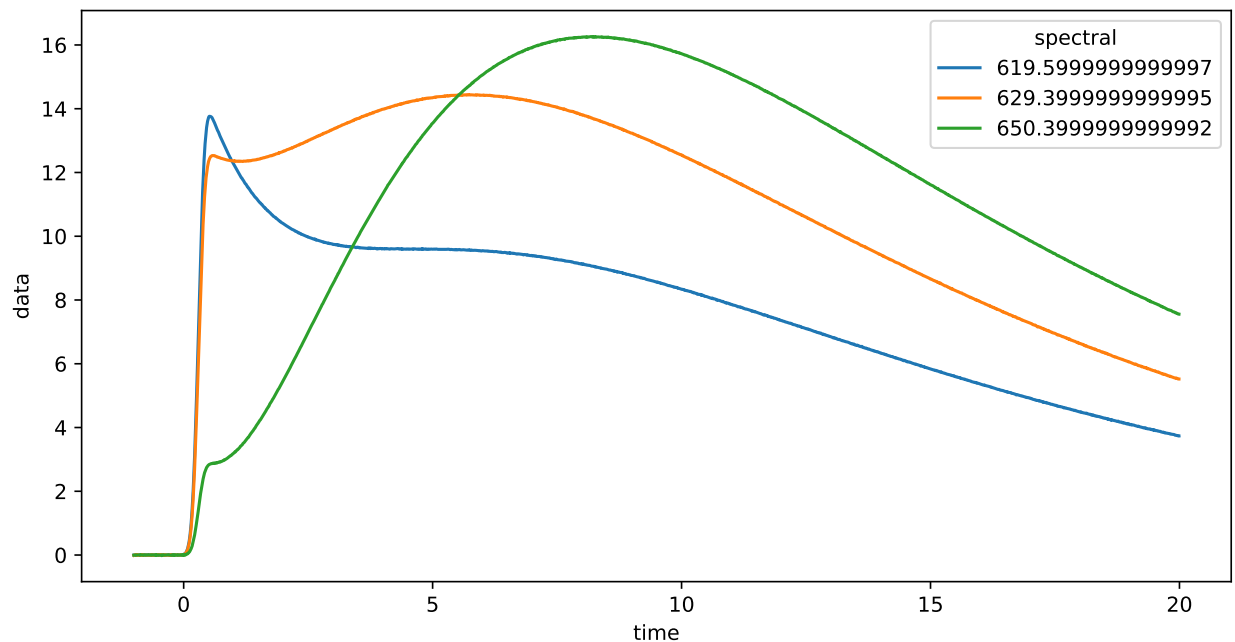
```

Coordinates:
  * time      (time) float64 -1.0 -0.99 -0.98 -0.97 ... 19.96 19.97 19.98 19.99
  * spectral  (spectral) float64 600.0 601.4 602.8 604.2 ... 696.6 698.0 699.4
Data variables:
  data        (time, spectral) float64 0.002115 -0.008893 ... 1.275 1.161
Attributes:
  source_path: half_intensity.json
  loader:      <function load_dataset at 0x7fe4eb2d5630>

```

```
[14]: reloaded_plot_data = reloaded_data.data.sel(spectral=[620, 630, 650], method="nearest")
reloaded_plot_data.plot.line(x="time", aspect=2, size=5)
```

```
[14]: [<matplotlib.lines.Line2D at 0x7fe4d83d1450>,
<matplotlib.lines.Line2D at 0x7fe4d83d1510>,
<matplotlib.lines.Line2D at 0x7fe4d83d1600>]
```



Since this looks like the above plot, but with half the amplitudes, so writing and reading our data worked as we hoped it would.

Writing a ProjectIo plugin words analogous:

	DataIo plugin	ProjectIo plugin
Register function	<code>glotaran.plugin_system.data_io_registration.register_data_io</code>	<code>glotaran.plugin_system.project_io_registration.register_project_io</code>
Base-class	<code>glotaran.io.interface.DataIoInterface</code>	<code>glotaran.io.interface.DataIoInterface</code>
Possible methods	<code>load_dataset</code> , <code>save_dataset</code>	<code>load_model</code> , <code>save_model</code> , <code>load_parameters</code> , <code>save_parameters</code> , <code>load_scheme</code> , <code>save_scheme</code> , <code>load_result</code> , <code>save_result</code>

Of course you don't have to implement all methods (sometimes that doesn't even make sense), but only the ones you need.

Last but not least:

Chances are that if you need a plugin someone else does too, so it would be awesome if you would publish it open source, so the wheel isn't reinvented over and over again.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





## BIBLIOGRAPHY

- [1] [https://glotaran.github.io/legacy/file\\_formats](https://glotaran.github.io/legacy/file_formats)
- [1] [https://glotaran.github.io/legacy/file\\_formats](https://glotaran.github.io/legacy/file_formats)



## PYTHON MODULE INDEX

### g

- glotaran, 61
- glotaran.analysis, 62
- glotaran.builtin, 62
- glotaran.builtin.io, 62
- glotaran.builtin.io.ascii, 62
- glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file, 63
- glotaran.builtin.io.folder, 73
- glotaran.builtin.io.folder.folder\_plugin, 73
- glotaran.builtin.io.netCDF, 83
- glotaran.builtin.io.netCDF.netCDF, 83
- glotaran.builtin.io.pandas, 85
- glotaran.builtin.io.pandas.csv, 85
- glotaran.builtin.io.pandas.tsv, 89
- glotaran.builtin.io.pandas.xlsx, 94
- glotaran.builtin.io.sdt, 98
- glotaran.builtin.io.sdt.sdt\_file\_reader, 98
- glotaran.builtin.io.yml, 100
- glotaran.builtin.io.yml.utils, 100
- glotaran.builtin.io.yml.yml, 101
- glotaran.builtin.megacomplexes, 106
- glotaran.builtin.megacomplexes.baseline, 106
- glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex, 106
- glotaran.builtin.megacomplexes.clp\_guide, 110
- glotaran.builtin.megacomplexes.clp\_guide.clp\_guide\_megacomplex, 110
- glotaran.builtin.megacomplexes.coherent\_artifact, 115
- glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact\_megacomplex, 115
- glotaran.builtin.megacomplexes.damped\_oscillation, 120
- glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation\_megacomplex, 120
- glotaran.builtin.megacomplexes.decay, 129
- glotaran.builtin.megacomplexes.decay.decay\_matrix\_gaussian\_fit, 129
- glotaran.builtin.megacomplexes.decay.decay\_megacomplex, 131
- glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex, 140
- glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex, 150
- glotaran.builtin.megacomplexes.decay.initial\_concentration, 155
- glotaran.builtin.megacomplexes.decay.irf, 158
- glotaran.builtin.megacomplexes.decay.k\_matrix, 180
- glotaran.builtin.megacomplexes.decay.util, 186
- glotaran.builtin.megacomplexes.spectral, 190
- glotaran.builtin.megacomplexes.spectral.shape, 190
- glotaran.builtin.megacomplexes.spectral.spectral\_megacomplex, 205
- glotaran.cli, 214
- glotaran.cli.commands, 214
- glotaran.cli.commands.explore, 214
- glotaran.cli.commands.export, 215
- glotaran.cli.commands.optimize, 215
- glotaran.cli.commands.pluginlist, 215
- glotaran.cli.commands.print, 215
- glotaran.cli.commands.util, 216
- glotaran.cli.commands.validate, 222
- glotaran.deprecation, 222
- glotaran.deprecation.deprecation\_utils, 223
- glotaran.deprecation.modules, 233
- glotaran.deprecation.modules.builtin\_io\_yaml, 233
- glotaran.deprecation.modules.examples, 234
- glotaran.deprecation.modules.examples.sequential, 234
- glotaran.io, 234
- glotaran.io.interface, 235
- glotaran.io.prepare\_dataset, 243
- glotaran.io.preprocessor, 243
- glotaran.io.preprocessor.pipeline, 243
- glotaran.io.preprocessor.preprocessor, 258
- glotaran.model, 300
- glotaran.model.clp\_constraint, 301
- glotaran.model.clp\_penalties, 309
- glotaran.model.clp\_relation, 314

- glotaran.model.dataset\_group, 316
- glotaran.model.dataset\_model, 319
- glotaran.model.interval\_item, 327
- glotaran.model.model, 329
- glotaran.model.weight, 337
- glotaran.optimization, 339
- glotaran.optimization.data\_provider, 339
- glotaran.optimization.estimate\_provider, 359
- glotaran.optimization.matrix\_provider, 374
- glotaran.optimization.nnls, 403
- glotaran.optimization.optimization\_group, 404
- glotaran.optimization.optimization\_history, 407
- glotaran.optimization.optimize, 411
- glotaran.optimization.optimizer, 411
- glotaran.optimization.variable\_projection, 416
- glotaran.parameter, 417
- glotaran.parameter.parameter, 417
- glotaran.parameter.parameter\_history, 425
- glotaran.parameter.parameters, 430
- glotaran.plugin\_system, 441
- glotaran.plugin\_system.base\_registry, 441
- glotaran.plugin\_system.data\_io\_registration, 450
- glotaran.plugin\_system.io\_plugin\_utils, 456
- glotaran.plugin\_system.megacomplex\_registration, 459
- glotaran.plugin\_system.project\_io\_registration, 462
- glotaran.project, 472
- glotaran.project.dataclass\_helpers, 472
- glotaran.project.generators, 475
- glotaran.project.generators.generator, 475
- glotaran.project.project, 482
- glotaran.project.project\_data\_registry, 501
- glotaran.project.project\_model\_registry, 505
- glotaran.project.project\_parameter\_registry, 509
- glotaran.project.project\_registry, 514
- glotaran.project.project\_result\_registry, 520
- glotaran.project.result, 525
- glotaran.project.scheme, 537
- glotaran.simulation, 543
- glotaran.simulation.simulation, 543
- glotaran.testing, 545
- glotaran.testing.plugin\_system, 545
- glotaran.testing.simulated\_data, 548
- glotaran.testing.simulated\_data.parallel\_spectral\_decay, 549
- glotaran.testing.simulated\_data.sequential\_spectral\_decay, 549
- glotaran.testing.simulated\_data.shared\_decay, 549
- glotaran.typing, 549
- glotaran.typing.protocols, 549
- glotaran.typing.types, 550
- glotaran.utils, 550
- glotaran.utils.attrs\_helper, 551
- glotaran.utils.helpers, 551
- glotaran.utils.io, 552
- glotaran.utils.ipython, 560
- glotaran.utils.regex, 570
- glotaran.utils.sanitize, 573
- glotaran.utils.tee, 577

## Symbols

--data  
     glotaran-optimize command line option, 49  
 --dataformat  
     glotaran-optimize command line option, 49  
 --model\_file  
     glotaran-optimize command line option, 50  
     glotaran-print command line option, 50  
     glotaran-validate command line option, 51  
 --nfev  
     glotaran-optimize command line option, 49  
 --nnls  
     glotaran-optimize command line option, 50  
 --out  
     glotaran-optimize command line option, 49  
 --outformat  
     glotaran-optimize command line option, 49  
 --parameters\_file  
     glotaran-optimize command line option, 50  
     glotaran-print command line option, 50  
     glotaran-validate command line option, 51  
 --version  
     glotaran command line option, 49  
 --yes  
     glotaran-optimize command line option, 50  
 -d  
     glotaran-optimize command line option, 49  
 -dfmt  
     glotaran-optimize command line option, 49  
 -m  
     glotaran-optimize command line option, 50  
     glotaran-print command line option, 50  
     glotaran-validate command line option, 51  
 -n  
     glotaran-optimize command line option, 49  
 -o  
     glotaran-optimize command line option, 49  
 -ofmt  
     glotaran-optimize command line option, 49  
 -p  
     glotaran-optimize command line option, 50  
     glotaran-print command line option, 50

glotaran-validate command line option, 51  
 -y  
     glotaran-optimize command line option, 50

## A

a\_matrix() (*glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix* method), 185  
 a\_matrix\_as\_markdown()  
     (*glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix* method), 185  
 a\_matrix\_general() (*glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix* method), 185  
 a\_matrix\_sequential()  
     (*glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix* method), 185  
 action(*glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage* attribute), 268  
 action(*glotaran.io.preprocessor.preprocessor.CorrectBaselineValue* attribute), 282  
 actions(*glotaran.io.preprocessor.pipeline.PreProcessingPipeline* attribute), 253  
 add\_data\_row() (*glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file.T* method), 70  
 add\_data\_row() (*glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file.W* method), 73  
 add\_instantiated\_plugin\_to\_registry() (in module *glotaran.plugin\_system.base\_registry*), 442  
 add\_model\_weight() (*glotaran.optimization.data\_provider.DataProvider* method), 343  
 add\_model\_weight() (*glotaran.optimization.data\_provider.DataProvider* method), 354  
 add\_plugin\_to\_registry() (in module *glotaran.plugin\_system.base\_registry*), 443  
 add\_svd(*glotaran.project.scheme.Scheme* attribute), 541  
 add\_svd\_data() (*glotaran.optimization.optimization\_group.OptimizationGroup* static method), 406  
 add\_svd\_to\_dataset() (in module *glotaran.io.prepare\_dataset*), 242  
 add\_weight\_to\_result\_data()  
     (*glotaran.optimization.optimization\_group.OptimizationGroup* method), 406  
 additional\_penalty(*glotaran.project.result.Result* at-

tribute), 533

`align_data()` (glotaran.optimization.data\_provider.DataProviderLinked method), 354

`align_dataset_indices()` (glotaran.optimization.data\_provider.DataProviderLinked method), 354

`align_full_clp_labels()` (glotaran.optimization.matrix\_provider.MatrixProviderLinked method), 390

`align_groups()` (glotaran.optimization.data\_provider.DataProviderLinked static method), 355

`align_index()` (glotaran.optimization.data\_provider.DataProviderLinked static method), 355

`align_matrices()` (glotaran.optimization.matrix\_provider.MatrixProviderLinked static method), 390

`align_weights()` (glotaran.optimization.data\_provider.DataProviderLinked method), 355

`aligned_full_clp_labels` (glotaran.optimization.matrix\_provider.MatrixProviderLinked property), 390

`aligned_global_axis` (glotaran.optimization.data\_provider.DataProviderLinked property), 355

`all()` (glotaran.parameter.parameters.Parameters method), 437

`amplitude` (glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian attribute), 195

`amplitude` (glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeS2y2Gaussian attribute), 202

`append()` (glotaran.parameter.parameter\_history.ParameterHistory method), 428

`applies()` (glotaran.model.clp\_constraint.ClpConstraint method), 303

`applies()` (glotaran.model.clp\_constraint.OnlyConstraint method), 306

`applies()` (glotaran.model.clp\_constraint.ZeroConstraint method), 308

`applies()` (glotaran.model.clp\_relation.ClpRelation method), 315

`applies()` (glotaran.model.interval\_item.IntervalItem method), 328

`apply()` (glotaran.io.preprocessor.pipeline.PreProcessingPipeline method), 253

`apply()` (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage method), 268

`apply()` (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue method), 282

`apply()` (glotaran.io.preprocessor.preprocessor.PreProcessor.Config method), 296

`apply_constraints()` (glotaran.optimization.matrix\_provider.MatrixProviderLinked method), 382

`apply_constraints()` (glotaran.optimization.matrix\_provider.MatrixProviderLinked method), 391

`apply_constraints()` (glotaran.optimization.matrix\_provider.MatrixProviderUnlinked method), 400

`applied_relations()` (glotaran.optimization.matrix\_provider.MatrixProviderLinked method), 382

`apply_relations()` (glotaran.optimization.matrix\_provider.MatrixProviderLinked method), 391

`apply_relations()` (glotaran.optimization.matrix\_provider.MatrixProviderUnlinked method), 400

`apply_weight()` (glotaran.optimization.matrix\_provider.MatrixContainer method), 377

`arbitrary_types_allowed` (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage attribute), 268

`arbitrary_types_allowed` (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue attribute), 282

`arbitrary_types_allowed` (glotaran.io.preprocessor.preprocessor.PreProcessor.Config attribute), 296

`as_dict()` (glotaran.cli.commands.util.ValOrRangeOrList attribute), 221

`as_dict()` (glotaran.model.model.Model method), 334

`as_dict()` (glotaran.parameter.parameter.Parameter method), 437

`as_list()` (glotaran.parameter.parameter.Parameter method), 437

`AsciiDataIo` (class in glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file), 63

`asdict()` (in module glotaran.project.dataclass\_helpers), 472

## B

`backsweep` (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 163

`backsweep` (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 167

`backsweep` (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian attribute), 173

`backsweep` (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute), 179

`backsweep_period` (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 163

`backsweep_period` (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 167

`backsweep_period` (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian attribute), 173

`backsweep_period` (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute), 179

`BaselineMegacomplex` (class in glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex), 406

`bool_str_repr()` (in module `glotaran.plugin_system.io_plugin_utils`), 457  
`bool_table_repr()` (in module `glotaran.plugin_system.io_plugin_utils`), 457  
**C**  
`calculate()` (`glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian` method), 163  
`calculate()` (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian` method), 167  
`calculate()` (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralZero` method), 173  
`calculate()` (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralShapeGaussian` method), 179  
`calculate()` (`glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian` method), 195  
`calculate()` (`glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeZero` method), 198  
`calculate()` (`glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeZero` method), 202  
`calculate()` (`glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeZero` method), 205  
`calculate()` (`glotaran.optimization.matrix_provider.MatrixProviderLinked` method), 382  
`calculate()` (`glotaran.optimization.matrix_provider.MatrixProviderUnlinked` method), 391  
`calculate()` (`glotaran.optimization.matrix_provider.MatrixProviderUnlinked` method), 400  
`calculate()` (`glotaran.optimization.optimization_group.OptimizationGroup` method), 406  
`calculate_aligned_matrices()` (`glotaran.optimization.matrix_provider.MatrixProviderLinked` method), 391  
`calculate_clp_penalties()` (`glotaran.optimization.termination_provider.TerminationProviderLinked` method), 363  
`calculate_clp_penalties()` (`glotaran.optimization.termination_provider.TerminationProviderUnlinked` method), 367  
`calculate_clp_penalties()` (`glotaran.optimization.termination_provider.TerminationProviderUnlinked` method), 372  
`calculate_covariance_matrix_and_standard_errors()` (`glotaran.optimization.optimizer.Optimizer` method), 414  
`calculate_damped_oscillation_matrix_gaussian_irf()` (in module `glotaran.builtin.megacomplexes.damped_oscillation_matrix_gaussian_irf`), 121  
`calculate_damped_oscillation_matrix_gaussian_irf_on_index()` (in module `glotaran.builtin.megacomplexes.damped_oscillation_matrix_gaussian_irf`), 121  
`calculate_damped_oscillation_matrix_no_irf()` (in module `glotaran.builtin.megacomplexes.damped_oscillation_matrix_no_irf`), 122  
`calculate_dataset_matrices()` (`glotaran.optimization.matrix_provider.MatrixProvider` method), 382  
`calculate_dataset_matrices()` (`glotaran.optimization.matrix_provider.MatrixProviderLinked` method), 391  
`calculate_dataset_matrices()` (`glotaran.optimization.matrix_provider.MatrixProviderUnlinked` method), 400  
`calculate_dataset_matrix()` (`glotaran.optimization.matrix_provider.MatrixProvider` static method), 382  
`calculate_dataset_matrix()` (`glotaran.optimization.matrix_provider.MatrixProviderLinked` static method), 391  
`calculate_dataset_matrix()` (`glotaran.optimization.matrix_provider.MatrixProviderUnlinked` static method), 400  
`calculate_decay_matrix_gaussian_irf()` (in module `glotaran.builtin.megacomplexes.decay.decay_matrix_gaussian_irf`), 139  
`calculate_decay_matrix_gaussian_irf_on_index()` (in module `glotaran.builtin.megacomplexes.decay.decay_matrix_gaussian_irf`), 130  
`calculate_decay_matrix_no_irf()` (in module `glotaran.builtin.megacomplexes.decay.util`), 137  
`calculate_dispersion()` (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian` method), 173  
`calculate_dispersion()` (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian` method), 179  
`calculate_estimation()` (`glotaran.optimization.termination_provider.TerminationProviderUnlinked` method), 372  
`calculate_full_matrices()` (`glotaran.optimization.matrix_provider.MatrixProviderUnlinked` method), 401  
`calculate_full_model_estimation()` (`glotaran.optimization.termination_provider.TerminationProviderUnlinked` method), 373  
`calculate_gamma()` (in module `glotaran.builtin.megacomplexes.decay.k_matrix`), 181  
`calculate_global_matrices()` (`glotaran.optimization.matrix_provider.MatrixProviderUnlinked` method), 401  
`calculate_index_matrix()` (`glotaran.builtin.megacomplexes.baseline.baseline_matrix` method), 199  
`calculate_matrix()` (`glotaran.builtin.megacomplexes.clp_guide.clp_guide` method), 114



[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.coherent\_artifact.megacomplex.CoherentArtifactMegacomplex method), 119  
[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.damped\_oscillation.megacomplex.DampedOscillationMegacomplex method), 127  
[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.decay.decay\_parallel.megacomplex.DecayParallelMegacomplex method), 139  
[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.decay.decay\_sequential.megacomplex.DecaySequentialMegacomplex method), 148  
[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.decay.decay\_parallel.megacomplex.DecayParallelMegacomplex method), 148  
[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 154  
[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 167  
[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.decay.irf.IrfSpectral attribute), 173  
[calculate\\_matrix\(\)](#) (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute), 179  
[calculate\\_matrix\(\)](#) (in module glotaran.builtin.megacomplexes.decay.util), 187  
[calculate\\_penalty\(\)](#) (glotaran.optimization.optimizer.Optimizer method), 414  
[calculate\\_prepared\\_matrices\(\)](#) (glotaran.optimization.matrix\_provider.MatrixProviderUnlinked method), 401  
[calculate\\_residual\(\)](#) (glotaran.optimization.estimation\_provider.EstimationProviderUnlinked method), 363  
[calculate\\_residual\(\)](#) (glotaran.optimization.estimation\_provider.EstimationProviderUnlinked method), 367  
[calculate\\_residual\(\)](#) (glotaran.optimization.estimation\_provider.EstimationProviderUnlinked method), 373  
[capitalize\(\)](#) (glotaran.utils.ipython.MarkdownStr method), 568  
[casefold\(\)](#) (glotaran.utils.ipython.MarkdownStr method), 568  
[center](#) (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 163  
[center](#) (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 167  
[center](#) (glotaran.builtin.megacomplexes.decay.irf.IrfSpectral attribute), 173  
[center](#) (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute), 179  
[center\(\)](#) (glotaran.utils.ipython.MarkdownStr method), 568  
[center\\_dispersion\\_coefficients](#) (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian attribute), 173  
[center\\_dispersion\\_coefficients](#) (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute), 179  
[chdir\\_context\(\)](#) (in module glotaran.utils.io), 553  
[check\\_overdue\(\)](#) (in module glotaran.deprecation.deprecation\_utils), 223  
[check\\_qualtrics\\_run\\_tests\(\)](#) (glotaran.deprecation.deprecation\_utils), 223  
[chi\\_square](#) (glotaran.project.result.Result attribute), 533  
[clear\(\)](#) (glotaran.project.generators.generator.GeneratorArguments attribute), 482  
[clear\(\)](#) (glotaran.utils.io.DatasetMapping method), 559  
[clp\\_constraint](#) (glotaran.model.model.Model attribute), 334  
[ClpConstraint](#) (class in glotaran.model.clp\_constraint), 301  
[ClpGuideMegacomplex](#) (class in glotaran.builtin.megacomplexes.clp\_guide.clp\_guide\_megacomplex), 111  
[ClpPenaltyLink](#) (class in glotaran.model.clp\_penalties), 309  
[ClpRelation](#) (class in glotaran.model.clp\_relation), 314  
[CoherentArtifactMegacomplex](#) (class in glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact\_megacomplex), 115  
[collect\\_megacomplexes\(\)](#) (in module glotaran.builtin.megacomplexes.decay.util), 188  
[combine\(\)](#) (glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix method), 185  
[combine\\_megacomplex\\_matrices\(\)](#) (glotaran.optimization.matrix\_provider.MatrixProvider static method), 383  
[combine\\_megacomplex\\_matrices\(\)](#) (glotaran.optimization.matrix\_provider.MatrixProviderLinked static method), 392  
[combine\\_megacomplex\\_matrices\(\)](#) (glotaran.optimization.matrix\_provider.MatrixProviderUnlinked static method), 401  
[compartments](#) (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex attribute), 148  
[compartments](#) (glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex attribute), 154  
[compartments](#) (glotaran.builtin.megacomplexes.decay.initial\_concentration\_megacomplex attribute), 157  
[compartments\(\)](#) (glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact\_megacomplex method), 119  
[construct\(\)](#) (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 253



`construct()` (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` class method), 268  
`construct()` (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` class method), 282  
`construct()` (`glotaran.io.preprocessor.preprocessor.PreProcessor` class method), 296  
`convert()` (`glotaran.cli.commands.util.ValOrRangeOrList` method), 221  
`convert_scientific_to_float()` (`glotaran.utils.sanitize` method), 573  
`copy()` (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` method), 253  
`copy()` (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` method), 268  
`copy()` (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` method), 282  
`copy()` (`glotaran.io.preprocessor.preprocessor.PreProcessor` method), 296  
`copy()` (`glotaran.parameter.parameter.Parameter` method), 423  
`copy()` (`glotaran.parameter.parameters.Parameters` method), 437  
`copy()` (`glotaran.project.generators.generator.GeneratorArguments` method), 482  
`correct_baseline_average()` (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` method), 254  
`correct_baseline_value()` (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` method), 254  
`CorrectBaselineAverage` (class in `glotaran.io.preprocessor.preprocessor`), 258  
`CorrectBaselineAverage.Config` (class in `glotaran.io.preprocessor.preprocessor`), 268  
`CorrectBaselineValue` (class in `glotaran.io.preprocessor.preprocessor`), 273  
`CorrectBaselineValue.Config` (class in `glotaran.io.preprocessor.preprocessor`), 282  
`cost` (`glotaran.project.result.Result` attribute), 533  
`count()` (`glotaran.utils.ipython.MarkdownStr` method), 568  
`covariance_matrix` (`glotaran.project.result.Result` attribute), 533  
`create()` (`glotaran.project.project.Project` static method), 494  
`create_aligned_global_axes()` (`glotaran.optimization.data_provider.DataProviderLinked` method), 356  
`create_class()` (`glotaran.model.model.Model` class method), 334  
`create_class_from_megacomplexes()` (`glotaran.model.model.Model` class method), 334  
`create_clp_guide_dataset()` (`glotaran.project.result.Result` method), 534  
`create_clp_guide_dataset()` (in module `glotaran.utils.io`), 553  
`create_result()` (`glotaran.optimization.optimizer.Optimizer` method), 414  
`create_result_data()` (`glotaran.optimization.optimization_group.OptimizationGroup` method), 407  
`create_result_run_name()` (`glotaran.project.project_result_registry.ProjectResultRegistry` method), 523  
`create_scaled_matrix()` (`glotaran.optimization.matrix_provider.MatrixContainer` method), 377  
`create_value_scheme()` (`glotaran.project.project.Project` method), 494  
`create_weighted_matrix()` (`glotaran.optimization.matrix_provider.MatrixContainer` method), 377  
`CsvProjectIo` (class in `glotaran.builtin.io.pandas.csv`), 85

## D

`DampedOscillationMegacomplex` (class in `glotaran.builtin.megacomplexes.damped_oscillation.damped_oscillation`), 124  
`data` (`glotaran.optimization.optimization_history.OptimizationHistory` property), 410  
`data` (`glotaran.project.project.Project` property), 494  
`data` (`glotaran.project.result.Result` attribute), 534  
`data` (`glotaran.project.scheme.Scheme` attribute), 541  
`data_filter` (`glotaran.io.interface.SavingOptions` attribute), 242  
`data_format` (`glotaran.io.interface.SavingOptions` attribute), 242  
`data_io_plugin_table()` (in module `glotaran.plugin_system.data_io_registration`), 451  
`DataFileType` (class in `glotaran.builtin.io.ascii.wavelength_time_explicit_file`), 65  
`DataIoInterface` (class in `glotaran.io.interface`), 235  
`DataProvider` (class in `glotaran.optimization.data_provider`), 339  
`DataProviderLinked` (class in `glotaran.optimization.data_provider`), 346  
`dataset` (`glotaran.model.model.Model` attribute), 335  
`dataset()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile` method), 67  
`dataset()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.TimeExplicitFile` method), 70  
`dataset()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.WavelengthTimeExplicitFile` method), 73

dataset\_groups (*glotaran.model.model.Model* attribute), 335  
 dataset\_models (*glotaran.model.dataset\_group.DatasetGroup* attribute), 318  
 DatasetGroup (class in *glotaran.model.dataset\_group*), 316  
 DatasetGroupModel (class in *glotaran.model.dataset\_group*), 318  
 DatasetMapping (class in *glotaran.utils.io*), 556  
 DatasetModel (class in *glotaran.model.dataset\_model*), 323  
 datasets (*glotaran.model.weight.Weight* attribute), 338  
 decay\_matrix\_implementation\_index\_dependent() (in module *glotaran.builtin.megacomplexes.decay.util*), 188  
 decay\_matrix\_implementation\_index\_independent() (in module *glotaran.builtin.megacomplexes.decay.util*), 188  
 DecayDatasetModel (class in *glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex*), 132  
 DecayDatasetModel (class in *glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex*), 141  
 DecayMegacomplex (class in *glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex*), 136  
 DecayParallelMegacomplex (class in *glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex*), 145  
 DecaySequentialMegacomplex (class in *glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex*), 150  
 degrees\_of\_freedom (*glotaran.project.result.Result* attribute), 534  
 deprecate() (in module *glotaran.deprecation.deprecation\_utils*), 224  
 deprecate\_dict\_entry() (in module *glotaran.deprecation.deprecation\_utils*), 225  
 deprecate\_module\_attribute() (in module *glotaran.deprecation.deprecation\_utils*), 227  
 deprecate\_submodule() (in module *glotaran.deprecation.deprecation\_utils*), 228  
 deserialize\_options() (in module *glotaran.parameter.parameter*), 417  
 dict() (*glotaran.io.preprocessor.pipeline.PreProcessingPipeline* method), 254  
 dict() (*glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage* method), 269  
 dict() (*glotaran.io.preprocessor.preprocessor.CorrectBaselineValue* method), 283  
 dict() (*glotaran.io.preprocessor.preprocessor.PreProcessor* method), 297  
 dimension (*glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex* attribute), 109  
 dimension (*glotaran.builtin.megacomplexes.clp\_guide.clp\_guide\_megacomplex* attribute), 114  
 dimension (*glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact\_megacomplex* attribute), 119  
 dimension (*glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation\_megacomplex* attribute), 127  
 dimension (*glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex* attribute), 149  
 dimension (*glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex* attribute), 154  
 dimension (*glotaran.builtin.megacomplexes.spectral.spectral\_megacomplex* attribute), 213  
 directory (*glotaran.project.project\_data\_registry.ProjectDataRegistry* property), 504  
 directory (*glotaran.project.project\_model\_registry.ProjectModelRegistry* property), 508  
 directory (*glotaran.project.project\_parameter\_registry.ProjectParameterRegistry* property), 512  
 directory (*glotaran.project.project\_registry.ProjectRegistry* property), 518  
 directory (*glotaran.project.project\_result\_registry.ProjectResultRegistry* property), 523  
 dispersion\_center (*glotaran.builtin.megacomplexes.decay.irf.IrfSpectrum* attribute), 173  
 dispersion\_center (*glotaran.builtin.megacomplexes.decay.irf.IrfSpectrum* attribute), 179  
 display\_file() (in module *glotaran.utils.ipython*), 560  
 does\_interval\_item\_apply() (*glotaran.optimization.matrix\_provider.MatrixProvider* static method), 383  
 does\_interval\_item\_apply() (*glotaran.optimization.matrix\_provider.MatrixProviderLinked* static method), 392  
 does\_interval\_item\_apply() (*glotaran.optimization.matrix\_provider.MatrixProviderUnlinked* static method), 401

## E

eigen() (*glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix* method), 185  
 elements\_in\_string\_of\_list (*glotaran.utils.regex.RegexPattern* attribute), 572  
 empty (*glotaran.project.project\_data\_registry.ProjectDataRegistry* property), 504  
 empty (*glotaran.project.project\_model\_registry.ProjectModelRegistry* property), 508

[empty \(glotaran.project.project\\_parameter\\_registry.ProjectParameterRegistry property\), 512](#)  
[empty \(glotaran.project.project\\_registry.ProjectRegistry property\), 518](#)  
[empty \(glotaran.project.project\\_result\\_registry.ProjectResultRegistry property\), 523](#)  
[empty\(\) \(glotaran.builtin.megacomplexes.decay.k\\_matrix.KMatrix class method\), 185](#)  
[encode\(\) \(glotaran.utils.ipython.MarkdownStr method\), 568](#)  
[endswith\(\) \(glotaran.utils.ipython.MarkdownStr method\), 568](#)  
[envvar\\_list\\_splitter \(glotaran.cli.commands.util.ValOrRangeOrList attribute\), 221](#)  
[EqualAreaPenalty \(class in glotaran.model.clp\\_penalties\), 311](#)  
[estimate\(\) \(glotaran.optimization.estation\\_provider.EstimationProvider method\), 363](#)  
[estimate\(\) \(glotaran.optimization.estation\\_provider.EstimationProviderLinked method\), 368](#)  
[estimate\(\) \(glotaran.optimization.estation\\_provider.EstimationProviderUnlinked method\), 373](#)  
[EstimationProvider \(class in glotaran.optimization.estation\\_provider\), 360](#)  
[EstimationProviderLinked \(class in glotaran.optimization.estation\\_provider\), 364](#)  
[EstimationProviderUnlinked \(class in glotaran.optimization.estation\\_provider\), 369](#)  
[ExcelProjectIo \(class in glotaran.builtin.io.pandas.xlsx\), 94](#)  
[exclude \(glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage attribute\), 269](#)  
[exclude\\_from\\_dict\\_field\(\) \(in module glotaran.project.dataclass\\_helpers\), 473](#)  
[exclude\\_from\\_normalize \(glotaran.builtin.megacomplexes.decay.initial\\_concentration.InitialConcentration attribute\), 157](#)  
[ExclusiveMegacomplexIssue \(class in glotaran.model.dataset\\_model\), 325](#)  
[expandtabs\(\) \(glotaran.utils.ipython.MarkdownStr method\), 568](#)  
[ExplicitFile \(class in glotaran.builtin.io.ascii.wavelength\\_time\\_explicit\\_file\), 65](#)  
[export\(\) \(in module glotaran.cli.commands.explore\), 214](#)  
[expression \(glotaran.parameter.parameter.Parameter attribute\), 423](#)  
[fail\(\) \(glotaran.cli.commands.util.ValOrRangeOrList method\), 221](#)  
[file \(glotaran.project.project.Project attribute\), 495](#)  
[file\\_loadable\\_field\(\) \(in module glotaran.project.dataclass\\_helpers\), 473](#)  
[file\\_loader\\_factory\(\) \(in module glotaran.project.dataclass\\_helpers\), 474](#)  
[FileLoadableProtocol \(class in glotaran.typing.protocols\), 549](#)  
[finalize\\_data\(\) \(glotaran.builtin.megacomplexes.baseline.baseline\\_megacomplex method\), 109](#)  
[finalize\\_data\(\) \(glotaran.builtin.megacomplexes.clp\\_guide.clp\\_guide\\_method\), 114](#)  
[finalize\\_data\(\) \(glotaran.builtin.megacomplexes.coherent\\_artifact.coherent\\_artifact method\), 119](#)  
[finalize\\_data\(\) \(glotaran.builtin.megacomplexes.damped\\_oscillation.damped\\_oscillation method\), 127](#)  
[finalize\\_data\(\) \(glotaran.builtin.megacomplexes.decay.decay\\_megacomplex method\), 139](#)  
[finalize\\_data\(\) \(glotaran.builtin.megacomplexes.decay.decay\\_parallel\\_megacomplex method\), 149](#)  
[finalize\\_data\(\) \(glotaran.builtin.megacomplexes.decay.decay\\_sequential\\_megacomplex method\), 154](#)  
[finalize\\_data\(\) \(glotaran.builtin.megacomplexes.spectral.spectral\\_megacomplex method\), 213](#)  
[finalize\\_data\(\) \(in module glotaran.builtin.megacomplexes.decay.util\), 189](#)  
[finalize\\_dataset\\_model\(\) \(in module glotaran.model.dataset\\_model\), 320](#)  
[find\(\) \(glotaran.utils.ipython.MarkdownStr method\), 568](#)  
[flatten\\_parameter\\_dict\(\) \(in module glotaran.parameter.parameters\), 430](#)  
[flush\(\) \(glotaran.utils.tee.TeeContext method\), 578](#)  
[folder \(glotaran.project.project.Project attribute\), 495](#)  
[FolderProjectIo \(class in glotaran.builtin.io.folder.folder\\_plugin\), 74](#)  
[force\\_index\\_dependent \(glotaran.builtin.megacomplexes.decay.decay\\_megacomplex.DecayMegacomplex attribute\), 135](#)  
[force\\_index\\_dependent \(glotaran.builtin.megacomplexes.decay.decay\\_parallel\\_megacomplex.DecayParallelMegacomplex attribute\), 144](#)  
[force\\_index\\_dependent \(glotaran.builtin.megacomplexes.spectral.spectral\\_megacomplex.SpectralMegacomplex attribute\), 209](#)  
[force\\_index\\_dependent \(glotaran.model.dataset\\_model.DatasetModel attribute\), 325](#)  
[format\(\) \(glotaran.utils.ipython.MarkdownStr method\), 568](#)  
[format\\_map\(\) \(glotaran.utils.ipython.MarkdownStr method\), 568](#)

method), 569

free\_parameter\_labels (glotaran.project.result.Result attribute), 534

frequencies (glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation\_megacomplexes module), 127

from\_csv() (glotaran.optimization.optimization\_history.OptimizationHistory class method), 410

from\_csv() (glotaran.parameter.parameter\_history.ParameterHistory class method), 428

from\_dataframe() (glotaran.parameter.parameter\_history.ParameterHistory class method), 428

from\_dataframe() (glotaran.parameter.parameters.Parameters class method), 437

from\_dict() (glotaran.parameter.parameters.Parameters class method), 437

from\_list() (glotaran.parameter.parameter.Parameter class method), 423

from\_list() (glotaran.parameter.parameters.Parameters class method), 438

from\_orm() (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 254

from\_orm() (glotaran.io.preprocessor.preprocessor.CorrectedLineAggregation class method), 269

from\_orm() (glotaran.io.preprocessor.preprocessor.CorrectedJitter class method), 283

from\_orm() (glotaran.io.preprocessor.preprocessor.PreProcessor class method), 297

from\_parameter\_dict\_list() (glotaran.parameter.parameters.Parameters class method), 438

from\_stdout\_str() (glotaran.optimization.optimization\_history.OptimizationHistory class method), 410

fromdict() (in module glotaran.project.dataclass\_helpers), 474

fromkeys() (glotaran.project.generators.generator.GeneratorArguments class method), 482

ftol (glotaran.project.scheme.Scheme attribute), 541

full() (glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix class method), 185

full\_plugin\_name() (in module glotaran.plugin\_system.base\_registry), 444

**G**

generate\_model() (glotaran.project.project.Project class method), 495

generate\_model() (glotaran.project.project\_model\_registry.ProjectModelRegistry class method), 508

generate\_model() (in module glotaran.project.generators.generator), 476

generate\_model\_yaml() (in module glotaran.project.generators.generator), 476

generate\_parallel\_decay\_model() (in module glotaran.project.generators.generator), 477

generate\_parallel\_spectral\_decay\_model() (in module glotaran.project.generators.generator), 477

generate\_parameters() (glotaran.project.project.Project class method), 495

generate\_parameters() (glotaran.project.project\_parameter\_registry.ProjectParameterRegistry class method), 512

generate\_sequential\_decay\_model() (in module glotaran.project.generators.generator), 478

generate\_sequential\_spectral\_decay\_model() (in module glotaran.project.generators.generator), 478

GeneratorArguments (class in glotaran.project.generators.generator), 479

get() (glotaran.parameter.parameters.Parameters class method), 438

get() (glotaran.project.generators.generator.GeneratorArguments class method), 482

get() (glotaran.project.project\_registry.ItemMapping class method), 515

get() (glotaran.utils.io.DatasetMapping class method), 559

get\_a\_matrix() (glotaran.builtin.megacomplexes.decay.decay\_megacomplexes class method), 139

get\_a\_matrix() (glotaran.builtin.megacomplexes.decay.decay\_parallel\_matrix class method), 149

get\_a\_matrix() (glotaran.builtin.megacomplexes.decay.decay\_sequential\_matrix class method), 154

get\_additional\_penalties() (glotaran.optimization.optimization\_history.OptimizationHistory class method), 363

get\_additional\_penalties() (glotaran.optimization.termination\_provider.TerminationProvider class method), 368

get\_additional\_penalties() (glotaran.optimization.termination\_provider.TerminationProvider class method), 373

get\_additional\_penalties() (glotaran.optimization.termination\_provider.TerminationProvider class method), 407

get\_aligned\_data() (glotaran.optimization.data\_provider.DataProvider class method), 356

get\_aligned\_dataset\_indices() (glotaran.optimization.data\_provider.DataProviderLinked class method), 356

get\_aligned\_group\_label() (glotaran.optimization.data\_provider.DataProviderLinked class method), 356

get\_aligned\_matrix\_container() (glotaran.optimization.matrix\_provider.MatrixProviderLinked class method), 392



`get_aligned_weight()`  
    (`glotaran.optimization.data_provider.DataProviderLinked`  
        method), 356  
`get_axis_slice_from_interval()`  
    (`glotaran.optimization.data_provider.DataProvider`  
        static method), 344  
`get_axis_slice_from_interval()`  
    (`glotaran.optimization.data_provider.DataProviderLinked`  
        static method), 357  
`get_compartments()` (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex.DecayParallelMegaComplex`  
    method), 139  
`get_compartments()` (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex.DecayParallelMegaComplex`  
    method), 149  
`get_compartments()` (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex.DecayParallelMegaComplex`  
    method), 154  
`get_current_optimization_iteration()`  
    (`glotaran.optimization.optimizer.Optimizer`  
        static method), 414  
`get_data()` (`glotaran.optimization.data_provider.DataProvider`  
    method), 344  
`get_data()` (`glotaran.optimization.data_provider.DataProvider`  
    method), 357  
`get_data_file_format()` (in module  
    `glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`)  
    63  
`get_data_io()` (in module  
    `glotaran.plugin_system.data_io_registration`), 452  
`get_data_row()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
    method), 67  
`get_data_row()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
    method), 70  
`get_data_row()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
    method), 73  
`get_dataloader()` (in module  
    `glotaran.plugin_system.data_io_registration`), 452  
`get_datasaver()` (in module  
    `glotaran.plugin_system.data_io_registration`), 452  
`get_dataset_groups()` (`glotaran.model.model.Model`  
    method), 335  
`get_dataset_model_dimension()` (in module  
    `glotaran.model.dataset_model`), 320  
`get_dataset_model_type()`  
    (`glotaran.builtin.megacomplexes.baseline.baseline_broadening.MagneticBroadening`  
        class method), 109  
`get_dataset_model_type()`  
    (`glotaran.builtin.megacomplexes.clp_guide.clp_guide_megacluster.ClpGuideMegaCluster`  
        class method), 114  
`get_dataset_model_type()`  
    (`glotaran.builtin.megacomplexes.coherent_artifact_complex.CoherentArtifactMagneticComplex`  
        class method), 119  
`get_dataset_model_type()`  
    (`glotaran.builtin.megacomplexes.damped_oscillation.damped_oscillation_megacluster.DampedOscillationMegaCluster`  
        class method), 127  
`get_dataset_model_type()`  
    (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex.DecayParallelMegaComplex`  
        class method), 139  
`get_dataset_model_type()`  
    (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex.DecayParallelMegaComplex`  
        class method), 149  
`get_dataset_model_type()`  
    (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex.DecayParallelMegaComplex`  
        class method), 154  
`get_dataset_model_type()`  
    (`glotaran.builtin.megacomplexes.spectral.spectral_megacluster.SpectralMegaCluster`  
        class method), 163  
`get_explicit_axis()`  
    (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
        method), 67  
`get_explicit_axis()`  
    (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.TimeExplicitFile`  
        method), 70  
`get_explicit_axis()`  
    (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.WavelengthTimeExplicitFile`  
        method), 73  
`get_flattened_data()`  
    (`glotaran.optimization.data_provider.DataProvider`  
        method), 344  
`get_flattened_data()`  
    (`glotaran.optimization.data_provider.DataProviderLinked`  
        method), 357  
`get_flattened_weight()`  
    (`glotaran.optimization.data_provider.DataProvider`  
        method), 344  
`get_full_penalty()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
    method), 67  
`get_full_penalty()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
    method), 70  
`get_full_penalty()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
    method), 73  
`get_full_penalty()` (`glotaran.optimization.data_provider.DataProvider`  
    static method), 344  
`get_full_penalty()` (`glotaran.optimization.data_provider.DataProvider`  
    static method), 357  
`get_full_penalty()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
    method), 402  
`get_full_penalty()` (`glotaran.optimization.estimation_provider.EstimationProvider`  
    method), 368  
`get_full_penalty()` (`glotaran.optimization.estimation_provider.EstimationProvider`  
    method), 373  
`get_full_penalty()` (`glotaran.optimization.group.OptimizationGroup`  
    method), 373

method), 407

get\_global\_axis() (glotaran.optimization.data\_provider.DataProvider method), 345

get\_global\_axis() (glotaran.optimization.data\_provider.DataProvider method), 358

get\_global\_dimension() (glotaran.optimization.data\_provider.DataProvider method), 345

get\_global\_dimension() (glotaran.optimization.data\_provider.DataProvider method), 358

get\_global\_matrix\_container() (glotaran.optimization.matrix\_provider.MatrixProvider method), 402

get\_initial\_concentration() (glotaran.builtin.megacomplexes.decay.decay\_megacomplex method), 140

get\_initial\_concentration() (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex method), 149

get\_initial\_concentration() (glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex method), 154

get\_interval\_number() (in module glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file), 63

get\_irf\_parameter() (glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact method), 119

get\_issues() (glotaran.model.model.Model method), 335

get\_item\_type() (glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex class method), 110

get\_item\_type() (glotaran.builtin.megacomplexes.clp\_guide.clp\_guide class method), 114

get\_item\_type() (glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact class method), 119

get\_item\_type() (glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation class method), 127

get\_item\_type() (glotaran.builtin.megacomplexes.decay.decay\_megacomplex class method), 140

get\_item\_type() (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex class method), 149

get\_item\_type() (glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex class method), 154

get\_item\_type() (glotaran.builtin.megacomplexes.decay.irf.Irf class method), 159

get\_item\_type() (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian class method), 163

get\_item\_type() (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian class method), 167

get\_item\_type() (glotaran.builtin.megacomplexes.decay.irf.IrfSpectral class method), 173

get\_item\_type() (glotaran.builtin.megacomplexes.decay.irf.IrfSpectral class method), 173

get\_item\_type() (glotaran.builtin.megacomplexes.spectral.shape.Spectral class method), 179

get\_item\_type() (glotaran.builtin.megacomplexes.spectral.shape.Spectral class method), 192

get\_item\_type() (glotaran.builtin.megacomplexes.spectral.shape.Spectral class method), 196

get\_item\_type() (glotaran.builtin.megacomplexes.spectral.shape.Spectral class method), 198

get\_item\_type() (glotaran.builtin.megacomplexes.spectral.shape.Spectral class method), 202

get\_item\_type() (glotaran.builtin.megacomplexes.spectral.shape.Spectral class method), 205

get\_item\_type() (glotaran.builtin.megacomplexes.spectral.spectral\_megacomplex class method), 213

get\_item\_type() (glotaran.model.clp\_constraint.ClpConstraint class method), 303

get\_item\_type() (glotaran.model.clp\_constraint.OnlyConstraint class method), 306

get\_item\_type() (glotaran.model.clp\_constraint.ZeroConstraint class method), 308

get\_item\_type() (glotaran.model.clp\_penalties.ClpPenalty class method), 310

get\_item\_type() (glotaran.model.clp\_penalties.MegadropPenalty class method), 313

get\_item\_type\_class() (glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex class method), 110

get\_item\_type\_class() (glotaran.builtin.megacomplexes.clp\_guide.clp\_guide class method), 114

get\_item\_type\_class() (glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact class method), 119

get\_item\_type\_class() (glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation class method), 127

get\_item\_type\_class() (glotaran.builtin.megacomplexes.decay.decay\_megacomplex class method), 140

get\_item\_type\_class() (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex class method), 149

get\_item\_type\_class() (glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex class method), 154

get\_item\_type\_class() (glotaran.builtin.megacomplexes.decay.irf.Irf class method), 159

get\_item\_type\_class() (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian class method), 163

get\_item\_type\_class() (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian class method), 167

get\_item\_type\_class() (glotaran.builtin.megacomplexes.decay.irf.IrfSpectral class method), 173

get\_item\_type\_class() (glotaran.builtin.megacomplexes.decay.irf.IrfSpectral class method), 173

`(glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian)`  
`class method`), 173  
`get_item_type_class()`  
`(glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian)`  
`class method`), 180  
`get_item_type_class()`  
`(glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian)`  
`class method`), 192  
`get_item_type_class()`  
`(glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian)`  
`class method`), 196  
`get_item_type_class()`  
`(glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian)`  
`class method`), 198  
`get_item_type_class()`  
`(glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian)`  
`class method`), 203  
`get_item_type_class()`  
`(glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian)`  
`class method`), 205  
`get_item_type_class()`  
`(glotaran.builtin.megacomplexes.spectral.spectral_megacomplex.SpectralMegaComplex)`  
`class method`), 213  
`get_item_type_class()`  
`(glotaran.model.clp_constraint.ClpConstraint)`  
`class method`), 303  
`get_item_type_class()`  
`(glotaran.model.clp_constraint.OnlyConstraint)`  
`class method`), 306  
`get_item_type_class()`  
`(glotaran.model.clp_constraint.ZeroConstraint)`  
`class method`), 308  
`get_item_type_class()`  
`(glotaran.model.clp_penalties.ClpPenalty)`  
`class method`), 310  
`get_item_type_class()`  
`(glotaran.model.clp_penalties.EqualAreaPenalty)`  
`class method`), 313  
`get_item_types()` (`glotaran.builtin.megacomplexes.baseline.baseline_megacomplex.BaselineMegaComplex`  
`class method`), 110  
`get_item_types()` (`glotaran.builtin.megacomplexes.clp_guide.clp_guide_megacomplex.ClpGuideMegaComplex`),  
`class method`), 114  
`get_item_types()` (`glotaran.builtin.megacomplexes.coherent_artifact_megacomplex.CoherentArtifactMegaComplex`),  
`class method`), 120  
`get_item_types()` (`glotaran.builtin.megacomplexes.damped_oscillation_megacomplex.DampedOscillationMegaComplex`),  
`class method`), 127  
`get_item_types()` (`glotaran.builtin.megacomplexes.decay.decay_megacomplex.DecayMegaComplex`),  
`class method`), 140  
`get_item_types()` (`glotaran.builtin.megacomplexes.decay.parallel_megacomplex.DecayParallelMegaComplex`),  
`class method`), 149  
`get_item_types()` (`glotaran.builtin.megacomplexes.decay.sequential_megacomplex.DecaySequentialMegaComplex`),  
`class method`), 155  
`get_item_types()` (`glotaran.builtin.megacomplexes.irf.Irf` `glotaran.plugin_system.megacomplex_registration`),  
`class method`), 159

`get_megacomplex_issues()` (in module `glotaran.model.dataset_model`), 321  
`get_metavar()` (`glotaran.cli.commands.util.ValOrRangeOrList` method), 221  
`get_method_from_plugin()` (in module `glotaran.plugin_system.base_registry`), 444  
`get_missing_message()` (`glotaran.cli.commands.util.ValOrRangeOrList` method), 221  
`get_model_axis()` (`glotaran.optimization.data_provider.DataProviderLinked` method), 345  
`get_model_axis()` (`glotaran.optimization.data_provider.DataProviderUnlinked` method), 358  
`get_model_dimension()` (`glotaran.optimization.data_provider.DataProviderLinked` method), 345  
`get_model_dimension()` (`glotaran.optimization.data_provider.DataProviderUnlinked` method), 358  
`get_models_directory()` (`glotaran.project.project.Project` method), 495  
`get_observations()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile` method), 67  
`get_observations()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile` method), 70  
`get_observations()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.WavelengthExplicitFile` method), 73  
`get_parameter_labels()` (`glotaran.model.model.Model` method), 335  
`get_parameters()` (`glotaran.parameter.parameter_history.ParameterHistory` method), 428  
`get_parameters_directory()` (`glotaran.project.project.Project` method), 496  
`get_plugin_from_registry()` (in module `glotaran.plugin_system.base_registry`), 445  
`get_prepared_matrix_container()` (`glotaran.optimization.matrix_provider.MatrixProviderLinked` method), 402  
`get_project_io()` (in module `glotaran.plugin_system.project_io_registration`), 463  
`get_project_io_method()` (in module `glotaran.plugin_system.project_io_registration`), 464  
`get_result()` (`glotaran.optimization.estation_provider.EstimationProviderLinked` method), 364  
`get_result()` (`glotaran.optimization.estation_provider.EstimationProviderUnlinked` method), 368  
`get_result()` (`glotaran.optimization.estation_provider.EstimationProviderUnlinked` method), 373  
`get_result()` (`glotaran.optimization.matrix_provider.MatrixProviderLinked` method), 384  
`get_result()` (`glotaran.optimization.matrix_provider.MatrixProviderUnlinked` method), 393  
`get_result()` (`glotaran.optimization.matrix_provider.MatrixProviderUnlinked` method), 402  
`get_result_path()` (`glotaran.project.project.Project` method), 496  
`get_scheme()` (`glotaran.project.result.Result` method), 534  
`get_script_dir()` (in module `glotaran.utils.io`), 554  
`get_secondary_axis()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile` method), 67  
`get_secondary_axis()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.TimeExplicitFile` method), 70  
`get_secondary_axis()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.WavelengthExplicitFile` method), 73  
`get_value_and_bounds_for_optimization()` (`glotaran.parameter.parameter.Parameter` method), 424  
`get_weight()` (`glotaran.optimization.data_provider.DataProviderLinked` method), 456  
`get_weight()` (`glotaran.optimization.data_provider.DataProviderUnlinked` method), 458  
`global_interval` (`glotaran.model.weight.Weight` attribute), 228  
`global_megacomplex` (`glotaran.builtin.megacomplexes.decay.decay_megacomplex` attribute), 135  
`global_megacomplex` (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex` attribute), 144  
`global_megacomplex` (`glotaran.builtin.megacomplexes.spectral.spectral_megacomplex` attribute), 209  
`global_megacomplex` (`glotaran.model.dataset_model.DatasetModel` attribute), 325  
`global_megacomplex_scale` (`glotaran.builtin.megacomplexes.decay.decay_megacomplex` attribute), 135  
`global_parallel_megacomplex_scale` (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex` attribute), 144  
`global_spectral_megacomplex_scale` (`glotaran.builtin.megacomplexes.spectral.spectral_megacomplex` attribute), 209  
`global_megacomplex_scale` (`glotaran.model.dataset_model.DatasetModel` attribute), 325  
`glotaran` module, 62  
`glotaran` command line option, 40  
`glotaran.analysis` module, 62  
`glotaran.builtin`



module, 62  
 glotaran.builtin.io  
   module, 62  
 glotaran.builtin.io.ascii  
   module, 62  
 glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file  
   module, 63  
 glotaran.builtin.io.folder  
   module, 73  
 glotaran.builtin.io.folder.folder\_plugin  
   module, 73  
 glotaran.builtin.io.netCDF  
   module, 83  
 glotaran.builtin.io.netCDF.netCDF  
   module, 83  
 glotaran.builtin.io.pandas  
   module, 85  
 glotaran.builtin.io.pandas.csv  
   module, 85  
 glotaran.builtin.io.pandas.tsv  
   module, 89  
 glotaran.builtin.io.pandas.xlsx  
   module, 94  
 glotaran.builtin.io.sdt  
   module, 98  
 glotaran.builtin.io.sdt.sdt\_file\_reader  
   module, 98  
 glotaran.builtin.io.yml  
   module, 100  
 glotaran.builtin.io.yml.utils  
   module, 100  
 glotaran.builtin.io.yml.yml  
   module, 101  
 glotaran.builtin.megacomplexes  
   module, 106  
 glotaran.builtin.megacomplexes.baseline  
   module, 106  
 glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex  
   module, 106  
 glotaran.builtin.megacomplexes.clp\_guide  
   module, 110  
 glotaran.builtin.megacomplexes.clp\_guide.clp\_guide\_megacomplex  
   module, 110  
 glotaran.builtin.megacomplexes.coherent\_artifact  
   module, 115  
 glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact\_megacomplex  
   module, 115  
 glotaran.builtin.megacomplexes.damped\_oscillation  
   module, 120  
 glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation\_megacomplex  
   module, 120  
 glotaran.builtin.megacomplexes.decay  
   module, 129  
 glotaran.builtin.megacomplexes.decay.decay\_matrix  
   module, 129  
 glotaran.builtin.megacomplexes.decay.decay\_megacomplex  
   module, 131  
 glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex  
   module, 140  
 glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex  
   module, 150  
 glotaran.builtin.megacomplexes.decay.initial\_concentration  
   module, 155  
 glotaran.builtin.megacomplexes.decay.irf  
   module, 158  
 glotaran.builtin.megacomplexes.decay.k\_matrix  
   module, 180  
 glotaran.builtin.megacomplexes.decay.util  
   module, 186  
 glotaran.builtin.megacomplexes.spectral  
   module, 190  
 glotaran.builtin.megacomplexes.spectral.shape  
   module, 190  
 glotaran.builtin.megacomplexes.spectral.spectral\_megacomplex  
   module, 205  
 glotaran.cli  
   module, 214  
 glotaran.cli.commands  
   module, 214  
 glotaran.cli.commands.explore  
   module, 214  
 glotaran.cli.commands.export  
   module, 215  
 glotaran.cli.commands.optimize  
   module, 215  
 glotaran.cli.commands.pluginlist  
   module, 215  
 glotaran.cli.commands.print  
   module, 215  
 glotaran.cli.commands.util  
   module, 216  
 glotaran.cli.commands.validate  
   module, 222  
 glotaran.deprecation  
   module, 222  
 glotaran.deprecation.deprecation\_utils  
   module, 223  
 glotaran.deprecation.modules  
   module, 233  
 glotaran.deprecation.modules.examples  
   module, 234  
 glotaran.deprecation.modules.megacomplexes  
   module, 234  
 glotaran.deprecation.modules.megacomplexes.sequential  
   module, 234  
 glotaran.io  
   module, 234  
 glotaran.io.gaussian\_interface  
   module, 234

- module, 235
- glotaran.io.prepare\_dataset
  - module, 242
- glotaran.io.preprocessor
  - module, 243
- glotaran.io.preprocessor.pipeline
  - module, 243
- glotaran.io.preprocessor.preprocessor
  - module, 258
- glotaran.model
  - module, 300
- glotaran.model.clp\_constraint
  - module, 301
- glotaran.model.clp\_penalties
  - module, 309
- glotaran.model.clp\_relation
  - module, 314
- glotaran.model.dataset\_group
  - module, 316
- glotaran.model.dataset\_model
  - module, 319
- glotaran.model.interval\_item
  - module, 327
- glotaran.model.model
  - module, 329
- glotaran.model.weight
  - module, 337
- glotaran.optimization
  - module, 339
- glotaran.optimization.data\_provider
  - module, 339
- glotaran.optimization.estimation\_provider
  - module, 359
- glotaran.optimization.matrix\_provider
  - module, 374
- glotaran.optimization.nnls
  - module, 403
- glotaran.optimization.optimization\_group
  - module, 404
- glotaran.optimization.optimization\_history
  - module, 407
- glotaran.optimization.optimize
  - module, 411
- glotaran.optimization.optimizer
  - module, 411
- glotaran.optimization.variable\_projection
  - module, 416
- glotaran.parameter
  - module, 417
- glotaran.parameter.parameter
  - module, 417
- glotaran.parameter.parameter\_history
  - module, 425
- glotaran.parameter.parameters
  - module, 430
- glotaran.plugin\_system
  - module, 441
- glotaran.plugin\_system.base\_registry
  - module, 441
- glotaran.plugin\_system.data\_io\_registration
  - module, 450
- glotaran.plugin\_system.io\_plugin\_utils
  - module, 456
- glotaran.plugin\_system.megacomplex\_registration
  - module, 459
- glotaran.plugin\_system.project\_io\_registration
  - module, 462
- glotaran.project
  - module, 472
- glotaran.project.dataclass\_helpers
  - module, 472
- glotaran.project.generators
  - module, 475
- glotaran.project.generators.generator
  - module, 475
- glotaran.project.project
  - module, 482
- glotaran.project.project\_data\_registry
  - module, 501
- glotaran.project.project\_model\_registry
  - module, 505
- glotaran.project.project\_parameter\_registry
  - module, 509
- glotaran.project.project\_registry
  - module, 514
- glotaran.project.project\_result\_registry
  - module, 520
- glotaran.project.result
  - module, 525
- glotaran.project.scheme
  - module, 537
- glotaran.simulation
  - module, 543
- glotaran.simulation.simulation
  - module, 543
- glotaran.testing
  - module, 545
- glotaran.testing.plugin\_system
  - module, 545
- glotaran.testing.simulated\_data
  - module, 548
- glotaran.testing.simulated\_data.parallel\_spectral\_decay
  - module, 549
- glotaran.testing.simulated\_data.sequential\_spectral\_decay
  - module, 549
- glotaran.testing.simulated\_data.shared\_decay
  - module, 549
- glotaran.typing

module, 549  
 glotaran.typing.protocols  
   module, 549  
 glotaran.typing.types  
   module, 550  
 glotaran.utils  
   module, 550  
 glotaran.utils.attrs\_helper  
   module, 551  
 glotaran.utils.helpers  
   module, 551  
 glotaran.utils.io  
   module, 552  
 glotaran.utils.ipython  
   module, 560  
 glotaran.utils.regex  
   module, 570  
 glotaran.utils.sanitize  
   module, 573  
 glotaran.utils.tee  
   module, 577  
 glotaran\_version (glotaran.project.result.Result attribute), 534  
 glotaran\_version() (in module  
   glotaran.deprecation.deprecation\_utils),  
   229  
 glotaran-optimize command line option  
   --data, 49  
   --dataformat, 49  
   --model\_file, 50  
   --nfev, 49  
   --nnls, 50  
   --out, 49  
   --outformat, 49  
   --parameters\_file, 50  
   --yes, 50  
   -d, 49  
   -dfmt, 49  
   -m, 50  
   -n, 49  
   -o, 49  
   -ofmt, 49  
   -p, 50  
   -y, 50  
   SCHEME\_FILE, 50  
 glotaran-print command line option  
   --model\_file, 50  
   --parameters\_file, 50  
   -m, 50  
   -p, 50  
   SCHEME\_FILE, 50  
 glotaran-validate command line option  
   --model\_file, 51  
   --parameters\_file, 51

-m, 51  
 -p, 51  
 SCHEME\_FILE, 51  
 group (glotaran.builtin.megacomplexes.decay.decay\_megacomplex.DecayL  
   attribute), 135  
 group (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomple  
   attribute), 144  
 group (glotaran.builtin.megacomplexes.spectral.spectral\_megacomplex.Spe  
   attribute), 209  
 group (glotaran.model.dataset\_model.DatasetModel attribute), 325  
 group (glotaran.optimization.estimate\_provider.EstimationProvider  
   property), 364  
 group (glotaran.optimization.estimate\_provider.EstimationProviderLinke  
   property), 368  
 group (glotaran.optimization.estimate\_provider.EstimationProviderUnlin  
   property), 373  
 group (glotaran.optimization.matrix\_provider.MatrixProvider  
   property), 384  
 group (glotaran.optimization.matrix\_provider.MatrixProviderLinked  
   property), 393  
 group (glotaran.optimization.matrix\_provider.MatrixProviderUnlinked  
   property), 403  
 group (glotaran.utils.regex.RegexPattern attribute), 572  
 group\_definitions (glotaran.optimization.data\_provider.DataProviderL  
   property), 359  
 gtol (glotaran.project.scheme.Scheme attribute), 541

## H

has() (glotaran.parameter.parameters.Parameters  
   method), 439  
 has\_data (glotaran.project.project.Project property),  
   496  
 has\_dataset\_model\_global\_model() (in module  
   glotaran.model.dataset\_model), 321  
 has\_interval() (glotaran.model.clp\_constraint.ClpConstraint  
   method), 303  
 has\_interval() (glotaran.model.clp\_constraint.OnlyConstraint  
   method), 306  
 has\_interval() (glotaran.model.clp\_constraint.ZeroConstraint  
   method), 309  
 has\_interval() (glotaran.model.clp\_relation.ClpRelation  
   method), 315  
 has\_interval() (glotaran.model.interval\_item.IntervalItem  
   method), 328  
 has\_models (glotaran.project.project.Project property),  
   496  
 has\_parameters (glotaran.project.project.Project prop-  
   erty), 496  
 has\_results (glotaran.project.project.Project prop-  
   erty), 496

## I

import\_data() (glotaran.project.project.Project

method), 497

`import_data()` (glotaran.project.project\_data\_registry.ProjectDataRegistry method), 504

`index()` (glotaran.utils.ipython.MarkdownStr method), 569

`index_dependent()` (in module `glotaran.builtin.megacomplexes.decay.util`), 189

`infer_file_format()` (in module `glotaran.plugin_system.io_plugin_utils`), 458

`infer_global_dimension()` (glotaran.optimization.data\_provider.DataProvider static method), 346

`infer_global_dimension()` (glotaran.optimization.data\_provider.DataProvider static method), 359

`init_file_loadable_fields()` (in module `glotaran.project.dataclass_helpers`), 475

`initial_concentration` (glotaran.builtin.megacomplexes.decay.decay\_megacomplex attribute), 135

`initial_parameters` (glotaran.project.result.Result attribute), 535

`InitialConcentration` (class in `glotaran.builtin.megacomplexes.decay.initial_concentration`), 155

`interval` (glotaran.model.clp\_constraint.ClpConstraint attribute), 303

`interval` (glotaran.model.clp\_constraint.OnlyConstraint attribute), 306

`interval` (glotaran.model.clp\_constraint.ZeroConstraint attribute), 309

`interval` (glotaran.model.clp\_relation.ClpRelation attribute), 316

`interval` (glotaran.model.interval\_item.IntervalItem attribute), 328

`IntervalItem` (class in `glotaran.model.interval_item`), 327

`involved_compartments()` (glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix method), 185

`Irf` (class in `glotaran.builtin.megacomplexes.decay.irf`), 158

`irf` (glotaran.builtin.megacomplexes.decay.decay\_megacomplex.DecayModel attribute), 135

`irf` (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex.DecayModel attribute), 144

`irf` (glotaran.project.generators.generator.GeneratorArguments attribute), 482

`IrfGaussian` (class in `glotaran.builtin.megacomplexes.decay.irf`), 160

`IrfMultiGaussian` (class in `glotaran.builtin.megacomplexes.decay.irf`), 168

`glotaran.builtin.megacomplexes.decay.irf`, 168

`IrfSpectralGaussian` (class in `glotaran.builtin.megacomplexes.decay.irf`), 174

`IrfSpectralMultiGaussian` (class in `glotaran.builtin.megacomplexes.decay.irf`), 174

`is_composite` (glotaran.cli.commands.util.ValOrRangeOrList attribute), 221

`is_index_dependent` (glotaran.optimization.matrix\_provider.MatrixContainer property), 377

`is_index_dependent()` (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian method), 163

`is_index_dependent()` (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian method), 168

`is_index_dependent()` (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian method), 168

`is_index_dependent()` (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian method), 180

`is_item()` (glotaran.project.project\_data\_registry.ProjectDataRegistry method), 504

`is_item()` (glotaran.project.project\_model\_registry.ProjectModelRegistry method), 508

`is_item()` (glotaran.project.project\_parameter\_registry.ProjectParameterRegistry method), 513

`is_item()` (glotaran.project.project\_registry.ProjectRegistry method), 518

`is_item()` (glotaran.project.project\_result\_registry.ProjectResultRegistry method), 523

`is_known_data_format()` (in module `glotaran.plugin_system.data_io_registration`), 453

`is_known_megacomplex()` (in module `glotaran.plugin_system.megacomplex_registration`), 460

`is_known_project_format()` (in module `glotaran.plugin_system.project_io_registration`), 464

`is_linkable()` (glotaran.model.dataset\_group.DatasetGroup method), 185

`is_registered_plugin()` (in module `glotaran.plugin_system.plugin_registry`), 445

`is_sequential()` (glotaran.builtin.megacomplexes.decay.k\_matrix.KMatrix method), 185

`isalnum()` (glotaran.utils.ipython.MarkdownStr method), 569

`isalpha()` (glotaran.utils.ipython.MarkdownStr method), 569

`isascii()` (glotaran.utils.ipython.MarkdownStr method), 569

- method), 569
- isdecimal() (glotaran.utils.ipython.MarkdownStr method), 569
- isdigit() (glotaran.utils.ipython.MarkdownStr method), 569
- isidentifier() (glotaran.utils.ipython.MarkdownStr method), 569
- islower() (glotaran.utils.ipython.MarkdownStr method), 569
- isnumeric() (glotaran.utils.ipython.MarkdownStr method), 569
- isprintable() (glotaran.utils.ipython.MarkdownStr method), 569
- isspace() (glotaran.utils.ipython.MarkdownStr method), 569
- istitle() (glotaran.utils.ipython.MarkdownStr method), 569
- isupper() (glotaran.utils.ipython.MarkdownStr method), 569
- item() (in module glotaran.model), 328
- ItemMapping (class in glotaran.project.project\_registry), 514
- items (glotaran.project.project\_data\_registry.ProjectDataRegistry property), 504
- items (glotaran.project.project\_model\_registry.ProjectModelRegistry property), 509
- items (glotaran.project.project\_parameter\_registry.ProjectParameterRegistry property), 513
- items (glotaran.project.project\_registry.ProjectRegistry property), 518
- items (glotaran.project.project\_result\_registry.ProjectResultRegistry property), 524
- items() (glotaran.project.generators.generator.GeneratorArguments method), 482
- items() (glotaran.project.project\_registry.ItemMapping method), 515
- items() (glotaran.utils.io.DatasetMapping method), 559
- iterate\_all\_items() (glotaran.model.model.Model method), 335
- iterate\_dataset\_model\_global\_megacomplexes() (in module glotaran.model.dataset\_model), 321
- iterate\_dataset\_model\_megacomplexes() (in module glotaran.model.dataset\_model), 322
- iterate\_items() (glotaran.model.model.Model method), 335
- ## J
- jacobian (glotaran.project.result.Result attribute), 535
- join() (glotaran.utils.ipython.MarkdownStr method), 569
- json() (glotaran.io.preprocessor.pipeline.PreProcessingPipeline method), 254
- json() (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue method), 269
- json() (glotaran.io.preprocessor.preprocessor.PreProcessor method), 297
- ## K
- k\_matrix (glotaran.builtin.megacomplexes.decay.decay\_megacomplex.DecayM attribute), 140
- keys() (glotaran.project.generators.generator.GeneratorArguments method), 482
- keys() (glotaran.project.project\_registry.ItemMapping method), 515
- keys() (glotaran.utils.io.DatasetMapping method), 559
- KMatrix (class in glotaran.builtin.megacomplexes.decay.k\_matrix), 181
- known\_data\_formats() (in module glotaran.plugin\_system.data\_io\_registration), 453
- known\_megacomplex\_names() (in module glotaran.plugin\_system.megacomplex\_registration), 460
- known\_project\_formats() (in module glotaran.plugin\_system.project\_io\_registration), 464
- ## L
- label (glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex.Ba attribute), 110
- label (glotaran.builtin.megacomplexes.clp\_guide.clp\_guide\_megacomplex attribute), 114
- label (glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact attribute), 120
- label (glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation attribute), 128
- label (glotaran.builtin.megacomplexes.decay.decay\_megacomplex.DecayL attribute), 136
- label (glotaran.builtin.megacomplexes.decay.decay\_megacomplex.DecayM attribute), 140
- label (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex attribute), 144
- label (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex attribute), 149
- label (glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex attribute), 155
- label (glotaran.builtin.megacomplexes.decay.initial\_concentration.InitialC attribute), 157
- label (glotaran.builtin.megacomplexes.decay.irf.Irf attribute), 159
- label (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 163
- label (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 168
- label (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian attribute), 173



[label \(glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute\), 180](#)  
[label \(glotaran.builtin.megacomplexes.decay.k\\_matrix.KMatrix attribute\), 186](#)  
[label \(glotaran.builtin.megacomplexes.spectral.shape.SpectralShape attribute\), 192](#)  
[label \(glotaran.builtin.megacomplexes.spectral.shape.SpectralShape attribute\), 196](#)  
[label \(glotaran.builtin.megacomplexes.spectral.shape.SpectralShape attribute\), 198](#)  
[label \(glotaran.builtin.megacomplexes.spectral.shape.SpectralShape attribute\), 203](#)  
[label \(glotaran.builtin.megacomplexes.spectral.shape.SpectralShape attribute\), 205](#)  
[label \(glotaran.builtin.megacomplexes.spectral.spectral\\_megacomplex attribute\), 209](#)  
[label \(glotaran.builtin.megacomplexes.spectral.spectral\\_megacomplex attribute\), 213](#)  
[label \(glotaran.model.dataset\\_group.DatasetGroupModel attribute\), 319](#)  
[label \(glotaran.model.dataset\\_model.DatasetModel attribute\), 325](#)  
[label \(glotaran.parameter.parameter.Parameter attribute\), 424](#)  
[label\\_short \(glotaran.parameter.parameter.Parameter property\), 424](#)  
[labels \(glotaran.builtin.megacomplexes.damped\\_oscillation.damped\\_oscillation\\_megacomplex.DampedOscillationMegacomplex attribute\), 128](#)  
[labels \(glotaran.parameter.parameters.Parameters property\), 439](#)  
[LegacyProjectIo \(class in glotaran.builtin.io.folder.folder\\_plugin.FolderProjectIo\), 78](#)  
[link\\_clp \(glotaran.model.dataset\\_group.DatasetGroup attribute\), 318](#)  
[link\\_clp \(glotaran.model.dataset\\_group.DatasetGroupModel attribute\), 319](#)  
[list\\_string\\_to\\_tuple\(\) \(in module glotaran.utils.sanitize\), 574](#)  
[list\\_with\\_tuples \(glotaran.utils.regex.RegexPattern attribute\), 572](#)  
[ljust\(\) \(glotaran.utils.ipython.MarkdownStr method\), 569](#)  
[load\\_data\(\) \(glotaran.project.project.Project method\), 497](#)  
[load\\_dataset\(\) \(glotaran.builtin.io.ascii.wavelength\\_time\\_explicit\\_parameters.ExplicitParameters method\), 64](#)  
[load\\_dataset\(\) \(glotaran.builtin.io.netCDF.netCDF.NetCDFParameters method\), 84](#)  
[load\\_dataset\(\) \(glotaran.builtin.io.sdt.sdt\\_file\\_reader.SdtFileReader method\), 99](#)  
[load\\_dataset\(\) \(glotaran.io.interface.DataIoInterface method\), 236](#)  
[load\\_dataset\(\) \(in module glotaran.plugin\\_system.data\\_io\\_registration\),](#)  
[load\\_dataset\\_file\(\) \(in module glotaran.cli.commands.util\), 216](#)  
[load\\_datasets\(\) \(in module glotaran.utils.io\), 554](#)  
[load\\_dict\(\) \(in module glotaran.builtin.io.yml.utils\), 101](#)  
[load\\_interface\(\) \(glotaran.project.project\\_data\\_registry.ProjectDataRegistry method\), 504](#)  
[load\\_interface\(\) \(glotaran.project.project\\_model\\_registry.ProjectModelRegistry method\), 509](#)  
[load\\_interface\(\) \(glotaran.project.project\\_parameter\\_registry.ProjectParameterRegistry method\), 513](#)  
[load\\_interface\(\) \(glotaran.project.project\\_registry.ProjectRegistry method\), 518](#)  
[load\\_interface\(\) \(glotaran.project.project\\_result\\_registry.ProjectResultRegistry method\), 524](#)  
[load\\_model\(\) \(glotaran.builtin.io.folder.folder\\_plugin.FolderProjectIo method\), 77](#)  
[load\\_model\(\) \(glotaran.builtin.io.folder.folder\\_plugin.LegacyProjectIo method\), 81](#)  
[load\\_model\(\) \(glotaran.builtin.io.pandas.csv.CsvProjectIo method\), 88](#)  
[load\\_model\(\) \(glotaran.builtin.io.pandas.tsv.TsvProjectIo method\), 92](#)  
[load\\_model\(\) \(glotaran.builtin.io.pandas.xlsx.ExcelProjectIo method\), 96](#)  
[load\\_model\(\) \(glotaran.builtin.io.yml.yml.YmlProjectIo method\), 104](#)  
[load\\_model\(\) \(glotaran.io.interface.ProjectIoInterface method\), 239](#)  
[load\\_model\(\) \(glotaran.project.project.Project method\), 498](#)  
[load\\_model\(\) \(in module glotaran.plugin\\_system.project\\_io\\_registration\), 465](#)  
[load\\_model\\_file\(\) \(in module glotaran.cli.commands.util\), 216](#)  
[load\\_parameter\\_file\(\) \(in module glotaran.cli.commands.util\), 217](#)  
[load\\_parameters\(\) \(glotaran.builtin.io.folder.folder\\_plugin.FolderProjectIo method\), 77](#)  
[load\\_parameters\(\) \(glotaran.builtin.io.folder.folder\\_plugin.LegacyProjectIo method\), 81](#)  
[load\\_parameters\(\) \(glotaran.builtin.io.pandas.csv.CsvProjectIo method\), 88](#)  
[load\\_parameters\(\) \(glotaran.builtin.io.pandas.tsv.TsvProjectIo method\), 92](#)  
[load\\_parameters\(\) \(glotaran.builtin.io.pandas.xlsx.ExcelProjectIo method\), 96](#)  
[load\\_parameters\(\) \(glotaran.builtin.io.yml.yml.YmlProjectIo method\), 104](#)

[load\\_parameters\(\) \(glotaran.io.interface.ProjectIoInterface method\), 239](#)  
[load\\_parameters\(\) \(glotaran.project.project.Project method\), 498](#)  
[load\\_parameters\(\) \(in module glotaran.plugin\\_system.project\\_io\\_registration\), 465](#)  
[load\\_plugins\(\) \(in module glotaran.plugin\\_system.base\\_registry\), 446](#)  
[load\\_result\(\) \(glotaran.builtin.io.folder.folder\\_plugin.FolderProjectIo method\), 77](#)  
[load\\_result\(\) \(glotaran.builtin.io.folder.folder\\_plugin.LegacyProjectIo method\), 82](#)  
[load\\_result\(\) \(glotaran.builtin.io.pandas.csv.CsvProjectIo method\), 88](#)  
[load\\_result\(\) \(glotaran.builtin.io.pandas.tsv.TsvProjectIo method\), 92](#)  
[load\\_result\(\) \(glotaran.builtin.io.pandas.xlsx.ExcelProjectIo method\), 96](#)  
[load\\_result\(\) \(glotaran.builtin.io.yml.yml.YmlProjectIo method\), 104](#)  
[load\\_result\(\) \(glotaran.io.interface.ProjectIoInterface method\), 239](#)  
[load\\_result\(\) \(glotaran.project.project.Project method\), 498](#)  
[load\\_result\(\) \(in module glotaran.plugin\\_system.project\\_io\\_registration\), 466](#)  
[load\\_scheme\(\) \(glotaran.builtin.io.folder.folder\\_plugin.FolderProjectIo method\), 77](#)  
[load\\_scheme\(\) \(glotaran.builtin.io.folder.folder\\_plugin.LegacyProjectIo method\), 82](#)  
[load\\_scheme\(\) \(glotaran.builtin.io.pandas.csv.CsvProjectIo method\), 88](#)  
[load\\_scheme\(\) \(glotaran.builtin.io.pandas.tsv.TsvProjectIo method\), 92](#)  
[load\\_scheme\(\) \(glotaran.builtin.io.pandas.xlsx.ExcelProjectIo method\), 97](#)  
[load\\_scheme\(\) \(glotaran.builtin.io.yml.yml.YmlProjectIo method\), 105](#)  
[load\\_scheme\(\) \(glotaran.io.interface.ProjectIoInterface method\), 240](#)  
[load\\_scheme\(\) \(in module glotaran.plugin\\_system.project\\_io\\_registration\), 466](#)  
[load\\_scheme\\_file\(\) \(in module glotaran.cli.commands.util\), 217](#)  
[loader \(glotaran.typing.protocols.FileLoadableProtocol attribute\), 550](#)  
[loader\(\) \(glotaran.model.model.Model method\), 335](#)  
[loader\(\) \(glotaran.optimization.optimization\\_history.OptimizationHistory class method\), 410](#)  
[loader\(\) \(glotaran.parameter.parameter\\_history.ParameterHistory class method\), 428](#)  
[loader\(\) \(glotaran.parameter.parameter.Parameters method\), 439](#)  
[loader\(\) \(glotaran.project.result.Result method\), 535](#)  
[loader\(\) \(glotaran.project.scheme.Scheme method\), 541](#)  
[loader\(\) \(glotaran.utils.io.DatasetMapping class method\), 559](#)  
[location \(glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeG attribute\), 196](#)  
[location \(glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeSH attribute\), 203](#)  
[lower\(\) \(glotaran.utils.ipython.MarkdownStr method\), 569](#)  
[lstrip\(\) \(glotaran.utils.ipython.MarkdownStr method\), 569](#)

## M

[main \(in module glotaran.cli\), 222](#)  
[make\\_path\\_absolute\\_if\\_relative\(\) \(in module glotaran.utils.io\), 555](#)  
[maketrans\(\) \(glotaran.utils.ipython.MarkdownStr method\), 569](#)  
[markdown\(\) \(glotaran.model.model.Model method\), 336](#)  
[markdown\(\) \(glotaran.parameter.parameter.Parameter method\), 424](#)  
[markdown\(\) \(glotaran.parameter.parameters.Parameters method\), 439](#)  
[markdown\(\) \(glotaran.project.project.Project method\), 498](#)  
[markdown\(\) \(glotaran.project.project\\_data\\_registry.ProjectDataRegistry method\), 505](#)  
[markdown\(\) \(glotaran.project.project\\_model\\_registry.ProjectModelRegistry method\), 509](#)  
[markdown\(\) \(glotaran.project.project\\_parameter\\_registry.ProjectParameterRegistry method\), 513](#)  
[markdown\(\) \(glotaran.project.project\\_registry.ProjectRegistry method\), 518](#)  
[markdown\(\) \(glotaran.project.project\\_result\\_registry.ProjectResultRegistry method\), 524](#)  
[markdown\(\) \(glotaran.project.result.Result method\), 535](#)  
[markdown\(\) \(glotaran.project.scheme.Scheme method\), 542](#)  
[MarkdownStr \(class in glotaran.utils.ipython\), 561](#)  
[matrix \(glotaran.builtin.megacomplexes.decay.k\\_matrix.KMatrix attribute\), 186](#)  
[matrix \(glotaran.optimization.matrix\\_provider.MatrixContainer attribute\), 378](#)  
[matrix\\_as\\_markdown\(\) \(glotaran.builtin.megacomplexes.decay.k\\_matrix.KMatrix method\), 186](#)  
[MatrixContainer \(class in glotaran.optimization.matrix\\_provider\), 375](#)  
[MatrixProvider \(class in glotaran.optimization.matrix\\_provider\), 378](#)

MatrixProviderLinked (class in `glotaran.optimization.matrix_provider`), 384  
 MatrixProviderUnlinked (class in `glotaran.optimization.matrix_provider`), 393  
 maximum (`glotaran.parameter.parameter.Parameter` attribute), 424  
 maximum\_number\_function\_evaluations (`glotaran.project.scheme.Scheme` attribute), 542  
 megacomplex (`glotaran.builtin.megacomplexes.decay.decay_megacomplex` attribute), 136  
 megacomplex (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex` attribute), 144  
 megacomplex (`glotaran.builtin.megacomplexes.spectral.spectral_megacomplex` attribute), 209  
 megacomplex (`glotaran.model.dataset_model.DatasetModel` attribute), 325  
 megacomplex (`glotaran.model.model.Model` attribute), 336  
 megacomplex() (in module `glotaran.model`), 328  
 megacomplex\_plugin\_table() (in module `glotaran.plugin_system.megacomplex_registration`), 461  
 megacomplex\_scale (`glotaran.builtin.megacomplexes.decay.decay_megacomplex_scale` attribute), 136  
 megacomplex\_scale (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplex_scale` attribute), 144  
 megacomplex\_scale (`glotaran.builtin.megacomplexes.spectral.spectral_megacomplex_scale` attribute), 210  
 megacomplex\_scale (`glotaran.model.dataset_model.DatasetModel` attribute), 325  
 methods\_differ\_from\_baseclass() (in module `glotaran.plugin_system.base_registry`), 446  
 methods\_differ\_from\_baseclass\_table() (in module `glotaran.plugin_system.base_registry`), 447  
 minimum (`glotaran.parameter.parameter.Parameter` attribute), 424  
 Model (class in `glotaran.model.model`), 329  
 model (`glotaran.model.dataset_group.DatasetGroup` attribute), 318  
 model (`glotaran.project.result.Result` property), 535  
 model (`glotaran.project.scheme.Scheme` attribute), 542  
 model\_computed\_fields (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` method), 284  
 model\_computed\_fields (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` property), 269  
 model\_computed\_fields (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` property), 283  
 model\_computed\_fields (`glotaran.io.preprocessor.preprocessor.PreProcessor` property), 297  
 model\_config (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` attribute), 254  
 model\_config (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` attribute), 269  
 model\_config (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` attribute), 283  
 model\_config (`glotaran.io.preprocessor.preprocessor.PreProcessor` attribute), 297  
 model\_construct() (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` method), 255  
 model\_construct() (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` method), 269  
 model\_construct() (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` method), 283  
 model\_construct() (`glotaran.io.preprocessor.preprocessor.PreProcessor` method), 297  
 model\_copy() (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` method), 255  
 model\_copy() (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` method), 269  
 model\_copy() (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` method), 283  
 model\_copy() (`glotaran.io.preprocessor.preprocessor.PreProcessor` method), 297  
 model\_dispersion\_with\_wavenumber (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian` attribute), 173  
 model\_dispersion\_with\_wavenumber (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian` attribute), 180  
 model\_dump() (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` method), 255  
 model\_dump() (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` method), 269  
 model\_dump() (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` method), 283  
 model\_dump() (`glotaran.io.preprocessor.preprocessor.PreProcessor` method), 297  
 model\_dump\_json() (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` method), 255  
 model\_dump\_json() (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` method), 270  
 model\_dump\_json() (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` method), 284  
 model\_dump\_json() (`glotaran.io.preprocessor.preprocessor.PreProcessor` method), 298  
 model\_extra (`glotaran.io.preprocessor.pipeline.PreProcessingPipeline` property), 256  
 model\_extra (`glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage` property), 270  
 model\_extra (`glotaran.io.preprocessor.preprocessor.CorrectBaselineValue` property), 284  
 model\_extra (`glotaran.io.preprocessor.preprocessor.PreProcessor` property), 298



`model_fields` (glotaran.io.preprocessor.pipeline.PreProcessingPipeline attribute), 256  
`model_fields` (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage class method), 270  
`model_fields` (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue class method), 284  
`model_fields` (glotaran.io.preprocessor.preprocessor.PreProcessor class attribute), 298  
`model_fields_set` (glotaran.io.preprocessor.pipeline.PreProcessingPipeline property), 256  
`model_fields_set` (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage property), 270  
`model_fields_set` (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue property), 284  
`model_fields_set` (glotaran.io.preprocessor.preprocessor.PreProcessor class property), 298  
`model_interval` (glotaran.model.weight.Weight attribute), 338  
`model_json_schema` (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 256  
`model_json_schema` (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage class method), 270  
`model_json_schema` (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue class method), 284  
`model_json_schema` (glotaran.io.preprocessor.preprocessor.PreProcessor class method), 298  
`model_parametrized_name` (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 256  
`model_parametrized_name` (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage class method), 271  
`model_parametrized_name` (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue class method), 285  
`model_parametrized_name` (glotaran.io.preprocessor.preprocessor.PreProcessor class method), 299  
`model_post_init` (glotaran.io.preprocessor.pipeline.PreProcessingPipeline method), 256  
`model_post_init` (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage method), 271  
`model_post_init` (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue method), 285  
`model_post_init` (glotaran.io.preprocessor.preprocessor.PreProcessor method), 299  
`model_rebuild` (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 257  
`model_rebuild` (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage class method), 271

glotaran.builtin.io.pandas.csv, 85  
 glotaran.builtin.io.pandas.tsv, 89  
 glotaran.builtin.io.pandas.xlsx, 94  
 glotaran.builtin.io.sdt, 98  
 glotaran.builtin.io.sdt.sdt\_file\_reader, 98  
 glotaran.builtin.io.yml, 100  
 glotaran.builtin.io.yml.utils, 100  
 glotaran.builtin.io.yml.yml, 101  
 glotaran.builtin.megacomplexes, 106  
 glotaran.builtin.megacomplexes.baseline, 106  
 glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex, 106  
 glotaran.builtin.megacomplexes.clp\_guide, 110  
 glotaran.builtin.megacomplexes.clp\_guide.clp\_guide\_megacomplex\_prepare\_dataset, 110  
 glotaran.builtin.megacomplexes.coherent\_artifact, 115  
 glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact\_megacomplex, 115  
 glotaran.builtin.megacomplexes.damped\_oscillation, 120  
 glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation\_megacomplex, 120  
 glotaran.builtin.megacomplexes.decay, 129  
 glotaran.builtin.megacomplexes.decay.decay\_matrix\_gaussian\_delinear, 129  
 glotaran.builtin.megacomplexes.decay.decay\_megacomplex, 131  
 glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex, 140  
 glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex, 150  
 glotaran.builtin.megacomplexes.decay.initial\_concentration, 155  
 glotaran.builtin.megacomplexes.decay.irf, 158  
 glotaran.builtin.megacomplexes.decay.k\_matrix, 180  
 glotaran.builtin.megacomplexes.decay.util, 186  
 glotaran.builtin.megacomplexes.spectral, 190  
 glotaran.builtin.megacomplexes.spectral.shape, 190  
 glotaran.builtin.megacomplexes.spectral.spectral\_megacomplex, 205  
 glotaran.cli, 214  
 glotaran.cli.commands, 214  
 glotaran.cli.commands.explore, 214  
 glotaran.cli.commands.export, 215  
 glotaran.cli.commands.optimize, 215  
 glotaran.cli.commands.pluginlist, 215  
 glotaran.cli.commands.print, 215  
 glotaran.cli.commands.util, 216  
 glotaran.cli.commands.validate, 222  
 glotaran.deprecation, 222  
 glotaran.deprecation.deprecation\_utils, 223  
 glotaran.deprecation.modules, 233  
 glotaran.deprecation.modules.builtin\_io\_yaml, 233  
 glotaran.deprecation.modules.examples, 234  
 glotaran.deprecation.modules.examples.sequential, 234  
 glotaran.io, 234  
 glotaran.io.interface, 235  
 glotaran.io.preprocessor, 243  
 glotaran.io.preprocessor.pipeline, 243  
 glotaran.io.preprocessor.preprocessor, 243  
 glotaran.model, 300  
 glotaran.model.clp\_constraint, 301  
 glotaran.model.clp\_penalties, 309  
 glotaran.model.damped\_oscillation\_megacomplex, 309  
 glotaran.model.dataset\_group, 316  
 glotaran.model.dataset\_model, 319  
 glotaran.model.gaussian\_delinear\_interval\_item, 327  
 glotaran.model.model, 329  
 glotaran.model.model.weight, 337  
 glotaran.optimization, 339  
 glotaran.optimization.data\_provider, 339  
 glotaran.optimization.estimation\_provider, 341  
 glotaran.optimization.matrix\_provider, 341  
 glotaran.optimization.nnls, 403  
 glotaran.optimization.optimization\_group, 404  
 glotaran.optimization.optimization\_history, 407  
 glotaran.optimization.optimize, 411  
 glotaran.optimization.optimizer, 411  
 glotaran.optimization.variable\_projection, 416  
 glotaran.parameter, 417  
 glotaran.parameter.parameter, 417  
 glotaran.parameter.parameter\_history, 425  
 glotaran.parameter.parameters, 430  
 glotaran.plugin\_system, 441  
 glotaran.plugin\_system.base\_registry, 441  
 glotaran.plugin\_system.data\_io\_registration, 450

glotaran.plugin\_system.io\_plugin\_utils, 456  
 glotaran.plugin\_system.megacomplex\_registration, 459  
 glotaran.plugin\_system.project\_io\_registration, 462  
 glotaran.project, 472  
 glotaran.project.dataclass\_helpers, 472  
 glotaran.project.generators, 475  
 glotaran.project.generators.generator, 475  
 glotaran.project.project, 482  
 glotaran.project.project\_data\_registry, 501  
 glotaran.project.project\_model\_registry, 505  
 glotaran.project.project\_parameter\_registry, 509  
 glotaran.project.project\_registry, 514  
 glotaran.project.project\_result\_registry, 520  
 glotaran.project.result, 525  
 glotaran.project.scheme, 537  
 glotaran.simulation, 543  
 glotaran.simulation.simulation, 543  
 glotaran.testing, 545  
 glotaran.testing.plugin\_system, 545  
 glotaran.testing.simulated\_data, 548  
 glotaran.testing.simulated\_data.parallel\_spectral\_decay, 549  
 glotaran.testing.simulated\_data.sequential\_spectral\_decay, 549  
 glotaran.testing.simulated\_data.shared\_decay, 549  
 glotaran.typing, 549  
 glotaran.typing.protocols, 549  
 glotaran.typing.types, 550  
 glotaran.utils, 550  
 glotaran.utils.attrs\_helper, 551  
 glotaran.utils.helpers, 551  
 glotaran.utils.io, 552  
 glotaran.utils.ipython, 560  
 glotaran.utils.regex, 570  
 glotaran.utils.sanitize, 573  
 glotaran.utils.tee, 577  
 module\_attribute() (in module glotaran.deprecation.deprecation\_utils), 230  
 monkeypatch\_plugin\_registry() (in module glotaran.testing.plugin\_system), 545  
 monkeypatch\_plugin\_registry\_data\_io() (in module glotaran.testing.plugin\_system), 546  
 monkeypatch\_plugin\_registry\_megacomplex() (in module glotaran.testing.plugin\_system), 547  
 monkeypatch\_plugin\_registry\_project\_io() (in module glotaran.testing.plugin\_system), 548  
 name, (glotaran.cli.commands.util.ValOrRangeOrList attribute), 221  
 nan\_or\_equal() (in module glotaran.utils.helpers), 552  
 NetCDFDataIo (class in glotaran.builtin.io.netCDF.netCDF), 84  
 no\_default\_vals\_in\_repr() (in module glotaran.utils.attrs\_helper), 551  
 non\_negative (glotaran.parameter.parameter.Parameter attribute), 424  
 normalize (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 163  
 normalize (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 168  
 normalize (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian attribute), 174  
 normalize (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute), 180  
 normalized() (glotaran.builtin.megacomplexes.decay.initial\_concentration method), 157  
 not\_implemented\_to\_value\_error() (in module glotaran.plugin\_system.io\_plugin\_utils), 458  
 nr\_compartments (glotaran.project.generators.generator.GeneratorArgument attribute), 482  
 number (glotaran.utils.regex.RegexPattern attribute), 572  
 number\_of\_clps (glotaran.optimization.matrix\_provider.MatrixProvider property), 384  
 number\_of\_clps (glotaran.optimization.matrix\_provider.MatrixProviderL property), 393  
 number\_of\_clps (glotaran.optimization.matrix\_provider.MatrixProviderU property), 403  
 number\_of\_clps (glotaran.optimization.optimization\_group.Optimization property), 407  
 number\_of\_clps (glotaran.project.result.Result attribute), 535  
 number\_of\_data\_points (glotaran.project.result.Result property), 535  
 number\_of\_free\_parameters (glotaran.project.result.Result attribute), 536  
 number\_of\_function\_evaluations (glotaran.project.result.Result attribute), 536  
 number\_of\_jacobian\_evaluations (glotaran.project.result.Result attribute), 536  
 number\_of\_parameters (glotaran.project.result.Result property), 536  
 number\_of\_records (glotaran.parameter.parameter\_history.ParameterHistory property), 429

`number_of_residuals` (*glotaran.project.result.Result* attribute), 536

`number_scientific` (*glotaran.utils.regex.RegexPattern* attribute), 572

## O

`objective_function()`  
(*glotaran.optimization.optimizer.Optimizer* method), 415

`OnlyConstraint` (class in *glotaran.model.clp\_constraint*), 304

`open()` (*glotaran.project.project.Project* class method), 499

`optimality` (*glotaran.project.result.Result* attribute), 536

`optimization_history` (*glotaran.project.result.Result* attribute), 536

`optimization_method`  
(*glotaran.project.scheme.Scheme* attribute), 542

`optimization_stdout`  
(*glotaran.utils.regex.RegexPattern* attribute), 572

`OptimizationGroup` (class in *glotaran.optimization.optimization\_group*), 404

`OptimizationHistory` (class in *glotaran.optimization.optimization\_history*), 408

`optimize()` (*glotaran.optimization.optimizer.Optimizer* method), 415

`optimize()` (*glotaran.project.project.Project* method), 499

`optimize()` (in module *glotaran.optimization.optimize*), 411

`optimize_cmd()` (in module *glotaran.cli.commands.optimize*), 215

`optimized_parameters` (*glotaran.project.result.Result* attribute), 536

`Optimizer` (class in *glotaran.optimization.optimizer*), 412

`order` (*glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact\_megacomplex.CoherentArtifactMegacomplex* attribute), 120

`OscillationParameterIssue` (class in *glotaran.builtin.megacomplexes.damped\_oscillation\_megacomplex.DampedOscillationMegacomplex*), 128

## P

`param_dict_to_markdown()` (in module *glotaran.parameter.parameters*), 430

`Parameter` (class in *glotaran.parameter.parameter*), 419

`parameter` (*glotaran.model.clp\_penalties.EqualAreaPenalty* attribute), 313

`parameter` (*glotaran.model.clp\_relation.ClpRelation* attribute), 316

`parameter()` (*glotaran.builtin.megacomplexes.decay.irf.IrfGaussian* method), 163

`parameter()` (*glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian* method), 168

`parameter()` (*glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian* method), 174

`parameter()` (*glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian* method), 180

`parameter_format` (*glotaran.io.interface.SavingOptions* attribute), 242

`parameter_history` (*glotaran.project.result.Result* attribute), 536

`parameter_labels` (*glotaran.parameter.parameter\_history.ParameterHistory* property), 429

`ParameterHistory` (class in *glotaran.parameter.parameter\_history*), 425

`Parameters` (class in *glotaran.parameter.parameters*), 431

`parameters` (*glotaran.builtin.megacomplexes.decay.initial\_concentration.InitialConcentrationDecay* attribute), 157

`parameters` (*glotaran.model.dataset\_group.DatasetGroup* attribute), 318

`parameters` (*glotaran.parameter.parameter\_history.ParameterHistory* property), 429

`parameters` (*glotaran.project.project.Project* property), 499

`parameters` (*glotaran.project.scheme.Scheme* attribute), 542

`parse_file()` (*glotaran.io.preprocessor.pipeline.PreProcessingPipeline* class method), 257

`parse_file()` (*glotaran.io.preprocessor.preprocessor.CorrectBaselineAveragePreProcessor* class method), 272

`parse_file()` (*glotaran.io.preprocessor.preprocessor.CorrectBaselineValuePreProcessor* class method), 286

`parse_file()` (*glotaran.io.preprocessor.preprocessor.PreProcessor* class method), 300

`parse_obj()` (*glotaran.io.preprocessor.pipeline.PreProcessingPipeline* class method), 258

`parse_obj()` (*glotaran.io.preprocessor.preprocessor.CorrectBaselineAveragePreProcessor* class method), 272

`parse_obj()` (*glotaran.io.preprocessor.preprocessor.CorrectBaselineValuePreProcessor* class method), 286

`parse_obj()` (*glotaran.io.preprocessor.preprocessor.PreProcessor* class method), 300

`parse_raw()` (*glotaran.io.preprocessor.pipeline.PreProcessingPipeline* class method), 258

`parse_raw()` (*glotaran.io.preprocessor.preprocessor.CorrectBaselineAveragePreProcessor* class method), 272

`parse_raw()` (*glotaran.io.preprocessor.preprocessor.CorrectBaselineValuePreProcessor* class method), 286

`parse_raw()` (*glotaran.io.preprocessor.preprocessor.PreProcessor* class method), 300



parse\_version() (in module `glotaran.deprecation.deprecation_utils`), 230  
 partition() (`glotaran.utils.ipython.MarkdownStr` method), 569  
 plugin\_list\_cmd() (in module `glotaran.cli.commands.pluginlist`), 215  
 pop() (`glotaran.project.generators.generator.GeneratorArguments` method), 482  
 pop() (`glotaran.utils.io.DatasetMapping` method), 559  
 popitem() (`glotaran.project.generators.generator.GeneratorArguments` method), 482  
 popitem() (`glotaran.utils.io.DatasetMapping` method), 559  
 prepare\_time\_trace\_dataset() (in module `glotaran.io.prepare_dataset`), 243  
 PreProcessingPipeline (class in `glotaran.io.preprocessor.pipeline`), 244  
 PreProcessor (class in `glotaran.io.preprocessor.preprocessor`), 286  
 PreProcessor.Config (class in `glotaran.io.preprocessor.preprocessor`), 296  
 pretty\_format\_numerical() (in module `glotaran.utils.sanitize`), 574  
 previous\_result\_paths() (`glotaran.project.project_result_registry.ProjectResultRegistry` method), 524  
 print\_cmd() (in module `glotaran.cli.commands.print`), 216  
 Project (class in `glotaran.project.project`), 483  
 project\_io\_list\_supporting\_plugins() (in module `glotaran.cli.commands.util`), 217  
 project\_io\_plugin\_table() (in module `glotaran.plugin_system.project_io_registration`), 467  
 ProjectDataRegistry (class in `glotaran.project.project_data_registry`), 501  
 ProjectIoInterface (class in `glotaran.io.interface`), 236  
 ProjectModelRegistry (class in `glotaran.project.project_model_registry`), 505  
 ProjectParameterRegistry (class in `glotaran.project.project_parameter_registry`), 510  
 ProjectRegistry (class in `glotaran.project.project_registry`), 515  
 ProjectResultRegistry (class in `glotaran.project.project_result_registry`), 520  
 protect\_from\_overwrite() (in module `glotaran.plugin_system.io_plugin_utils`), 459

## R

raise\_deprecation\_error() (in module `glotaran.deprecation.deprecation_utils`), 230  
 rates (`glotaran.builtin.megacomplexes.damped_oscillation.damped_oscillation` attribute), 128  
 rates (`glotaran.builtin.megacomplexes.decay.decay_parallel_megacomplexes` attribute), 149  
 rates (`glotaran.builtin.megacomplexes.decay.decay_sequential_megacomplexes` attribute), 155  
 rates() (`glotaran.builtin.megacomplexes.decay.k_matrix.KMatrix` method), 186  
 read() (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile` method), 67  
 read() (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.TimeExplicitFile` method), 70  
 read() (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.WavelengthTimeExplicitFile` method), 73  
 read() (`glotaran.utils.tee.TeeContext` method), 578  
 recreate() (`glotaran.project.result.Result` method), 536  
 reduce\_matrix() (`glotaran.optimization.matrix_provider.MatrixProvider` method), 384  
 reduce\_matrix() (`glotaran.optimization.matrix_provider.MatrixProvider` method), 393  
 reduce\_matrix() (`glotaran.optimization.matrix_provider.MatrixProvider` method), 403  
 reduced() (`glotaran.builtin.megacomplexes.decay.k_matrix.KMatrix` method), 186  
 reduced\_chi\_square (`glotaran.project.result.Result` attribute), 536  
 RegexPattern (class in `glotaran.utils.regex`), 570  
 register\_data\_io() (in module `glotaran.plugin_system.data_io_registration`), 454  
 register\_megacomplex() (in module `glotaran.plugin_system.megacomplex_registration`), 461  
 register\_project\_io() (in module `glotaran.plugin_system.project_io_registration`), 467  
 registered\_plugins() (in module `glotaran.plugin_system.base_registry`), 448  
 relative\_posix\_path() (in module `glotaran.utils.io`), 555  
 removeprefix() (`glotaran.utils.ipython.MarkdownStr` method), 569  
 removesuffix() (`glotaran.utils.ipython.MarkdownStr` method), 569  
 replace() (`glotaran.utils.ipython.MarkdownStr` method), 569  
 report (`glotaran.io.interface.SavingOptions` attribute), 242  
 residual\_function (`glotaran.model.dataset_group.DatasetGroup` attribute), 318

`residual_function` (`glotaran.model.dataset_group.DatasetGroupDict` attribute), 319  
`residual_nnls`() (in module `glotaran.optimization.nnls`), 403  
`residual_variable_projection`() (in module `glotaran.optimization.variable_projection`), 416  
`Result` (class in `glotaran.project.result`), 525  
`result_path` (`glotaran.project.scheme.Scheme` attribute), 542  
`result_pattern` (`glotaran.project.project_result_registry.ProjectResultRegistry` attribute), 524  
`results` (`glotaran.project.project.Project` property), 500  
`retrieve_clps`() (`glotaran.optimization.termination_providers.termination_providers` method), 364  
`retrieve_clps`() (`glotaran.optimization.termination_providers.termination_providers` method), 368  
`retrieve_clps`() (`glotaran.optimization.termination_providers.termination_providers` method), 373  
`retrieve_decay_associated_data`() (in module `glotaran.builtin.megacomplexes.decay.util`), 189  
`retrieve_initial_concentration`() (in module `glotaran.builtin.megacomplexes.decay.util`), 189  
`retrieve_irf`() (in module `glotaran.builtin.megacomplexes.decay.util`), 190  
`retrieve_species_associated_data`() (in module `glotaran.builtin.megacomplexes.decay.util`), 190  
`rfind`() (`glotaran.utils.ipython.MarkdownStr` method), 569  
`rindex`() (`glotaran.utils.ipython.MarkdownStr` method), 569  
`rjust`() (`glotaran.utils.ipython.MarkdownStr` method), 570  
`root_mean_square_error` (`glotaran.project.result.Result` attribute), 536  
`rpartition`() (`glotaran.utils.ipython.MarkdownStr` method), 570  
`rsplit`() (`glotaran.utils.ipython.MarkdownStr` method), 570  
`rstrip`() (`glotaran.utils.ipython.MarkdownStr` method), 570  

## S

`safe_dataframe_fillna`() (in module `glotaran.utils.io`), 555  
`safe_dataframe_replace`() (in module `glotaran.utils.io`), 556  
`sanitize_dict_keys`() (in module `glotaran.utils.sanitize`), 574  
`sanitize_dict_values`() (in module `glotaran.utils.sanitize`), 575  
`sanitize_list_with_broken_tuples`() (in module `glotaran.utils.sanitize`), 575  
`sanitize_parameter_list`() (in module `glotaran.utils.sanitize`), 575  
`sanitize_yaml`() (in module `glotaran.utils.sanitize`), 576  
`sanity_scientific_notation_conversion`() (in module `glotaran.utils.sanitize`), 576  
`save`() (`glotaran.project.project_result_registry.ProjectResultRegistry` method), 525  
`save`() (`glotaran.project.result.Result` method), 536  
`save_dataset`() (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.AsciiWavelengthTimeExplicitFile` method), 65  
`save_dataset`() (`glotaran.builtin.io.netCDF.netCDF.NetCDFDataIo` method), 84  
`save_dataset`() (`glotaran.builtin.io.sdt.sdt_file_reader.SdtDataIo` method), 100  
`save_dataset`() (`glotaran.io.interface.DataIoInterface` method), 236  
`save_dataset`() (in module `glotaran.plugin_system.data_io_registration`), 454  
`save_model`() (`glotaran.builtin.io.folder.folder_plugin.FolderProjectIo` method), 77  
`save_model`() (`glotaran.builtin.io.folder.folder_plugin.LegacyProjectIo` method), 82  
`save_model`() (`glotaran.builtin.io.pandas.csv.CsvProjectIo` method), 88  
`save_model`() (`glotaran.builtin.io.pandas.tsv.TsvProjectIo` method), 93  
`save_model`() (`glotaran.builtin.io.pandas.xlsx.ExcelProjectIo` method), 97  
`save_model`() (`glotaran.builtin.io.yaml.yaml.YmlProjectIo` method), 105  
`save_model`() (`glotaran.io.interface.ProjectIoInterface` method), 240  
`save_model`() (in module `glotaran.plugin_system.project_io_registration`), 468  
`save_parameters`() (`glotaran.builtin.io.folder.folder_plugin.FolderProjectIo` method), 77  
`save_parameters`() (`glotaran.builtin.io.folder.folder_plugin.LegacyProjectIo` method), 82  
`save_parameters`() (`glotaran.builtin.io.pandas.csv.CsvProjectIo` method), 89  
`save_parameters`() (`glotaran.builtin.io.pandas.tsv.TsvProjectIo` method), 93  
`save_parameters`() (`glotaran.builtin.io.pandas.xlsx.ExcelProjectIo` method), 97  
`save_parameters`() (`glotaran.builtin.io.yaml.yaml.YmlProjectIo` method), 105  
`save_parameters`() (`glotaran.io.interface.ProjectIoInterface` method), 240

method), 240

save\_parameters() (in module glotaran.plugin\_system.project\_io\_registration), 468

save\_result() (glotaran.builtin.io.folder.folder\_plugin.FolderProjectIo class method), 78

save\_result() (glotaran.builtin.io.folder.folder\_plugin.LegacyProjectIo class method), 82

save\_result() (glotaran.builtin.io.pandas.csv.CsvProjectIo class method), 89

save\_result() (glotaran.builtin.io.pandas.tsv.TsvProjectIo class method), 93

save\_result() (glotaran.builtin.io.pandas.xlsx.ExcelProjectIo class method), 97

save\_result() (glotaran.builtin.io.yml.yml.YmlProjectIo class method), 105

save\_result() (glotaran.io.interface.ProjectIoInterface class method), 240

save\_result() (in module glotaran.plugin\_system.project\_io\_registration), 469

save\_scheme() (glotaran.builtin.io.folder.folder\_plugin.FolderProjectIo class method), 78

save\_scheme() (glotaran.builtin.io.folder.folder\_plugin.LegacyProjectIo class method), 83

save\_scheme() (glotaran.builtin.io.pandas.csv.CsvProjectIo class method), 89

save\_scheme() (glotaran.builtin.io.pandas.tsv.TsvProjectIo class method), 93

save\_scheme() (glotaran.builtin.io.pandas.xlsx.ExcelProjectIo class method), 97

save\_scheme() (glotaran.builtin.io.yml.yml.YmlProjectIo class method), 105

save\_scheme() (glotaran.io.interface.ProjectIoInterface class method), 240

save\_scheme() (in module glotaran.plugin\_system.project\_io\_registration), 470

SavingOptions (class in glotaran.io.interface), 241

scale (glotaran.builtin.megacomplexes.decay.decay\_megacomplex\_model attribute), 136

scale (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex\_model attribute), 144

scale (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 163

scale (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 168

scale (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian attribute), 174

scale (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute), 180

scale (glotaran.builtin.megacomplexes.spectral.spectral\_megacomplex\_model attribute), 210

scale (glotaran.model.dataset\_model.DatasetModel attribute), 240

tribute), 325

schema() (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 258

schema() (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage class method), 272

schema() (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue class method), 286

schema() (glotaran.io.preprocessor.preprocessor.PreProcessor class method), 300

schema\_json() (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 258

schema\_json() (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage class method), 272

schema\_json() (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue class method), 286

schema\_json() (glotaran.io.preprocessor.preprocessor.PreProcessor class method), 300

Scheme (class in glotaran.project.scheme), 538

scheme (glotaran.project.result.Result attribute), 537

SCHEME\_FILE

glotaran-optimize command line option, 50

glotaran-print command line option, 50

glotaran-validate command line option, 51

SdtDataIo (class in glotaran.builtin.io.sdt.sdt\_file\_reader), 98

select (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage attribute), 272

select\_data() (in module glotaran.cli.commands.util), 217

select\_name() (in module glotaran.cli.commands.util), 217

serialize\_options() (in module glotaran.parameter.parameter), 418

set\_data\_plugin() (in module glotaran.plugin\_system.data\_io\_registration), 455

set\_explicit\_axis() (glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file.ExplicitFileModel method), 67

set\_explicit\_axis() (glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file.TimeExplicitFileModel method), 73

set\_explicit\_axis() (glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file.WavelengthTimeExplicitFileModel method), 73

set\_from\_history() (glotaran.parameter.parameters.Parameters class method), 439

set\_from\_label\_and\_value\_arrays() (glotaran.parameter.parameters.Parameters class method), 440

set\_megacomplex\_plugin() (in module glotaran.plugin\_system.megacomplex\_registration), 462

set\_parameters() (glotaran.model.dataset\_group.DatasetGroup

method), 318

set\_plugin() (in module `glotaran.plugin_system.base_registry`), 448

set\_project\_plugin() (in module `glotaran.plugin_system.project_io_registration`), 470

set\_transformed\_expression() (in module `glotaran.parameter.parameter`), 418

set\_value\_from\_optimization() (`glotaran.parameter.parameter.Parameter` method), 424

setdefault() (`glotaran.project.generators.generator.Generator` method), 482

setdefault() (`glotaran.utils.io.DatasetMapping` method), 559

shape (`glotaran.builtin.megacomplexes.spectral.spectral_megacomplex` attribute), 213

shell\_complete() (`glotaran.cli.commands.util.ValOrRangeOrList` method), 221

shift (`glotaran.builtin.megacomplexes.decay.irf.IrfGaussian` attribute), 163

shift (`glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian` attribute), 168

shift (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian` attribute), 174

shift (`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian` attribute), 180

show\_data\_io\_method\_help() (in module `glotaran.plugin_system.data_io_registration`), 455

show\_method\_help() (in module `glotaran.plugin_system.base_registry`), 449

show\_model\_definition() (`glotaran.project.project.Project` method), 500

show\_parameters\_definition() (`glotaran.project.project.Project` method), 500

show\_project\_io\_method\_help() (in module `glotaran.plugin_system.project_io_registration`), 471

signature\_analysis() (in module `glotaran.cli.commands.util`), 217

simulate() (in module `glotaran.simulation.simulation`), 543

simulate\_from\_clp() (in module `glotaran.simulation.simulation`), 544

simulate\_full\_model() (in module `glotaran.simulation.simulation`), 544

skewness (`glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeSkewedGaussian` attribute), 203

source (`glotaran.model.clp_penalties.EqualAreaPenalty` attribute), 313

source (`glotaran.model.clp_relation.ClpRelation` attribute), 316

source\_intervals (`glotaran.model.clp_penalties.EqualAreaPenalty` attribute), 313

source\_path (`glotaran.model.model.Model` attribute), 336

source\_path (`glotaran.project.result.Result` attribute), 537

source\_path (`glotaran.project.scheme.Scheme` attribute), 542

source\_path (`glotaran.typing.protocols.FileLoadableProtocol` attribute), 550

source\_path (`glotaran.utils.io.DatasetMapping` property), 560

spectral\_axis\_inverted (`glotaran.builtin.megacomplexes.spectral.spectral_megacomplex` attribute), 210

spectral\_axis\_scale (`glotaran.builtin.megacomplexes.spectral.spectral_megacomplex` attribute), 210

SpectralDatasetModel (class in `glotaran.builtin.megacomplexes.spectral.spectral_megacomplex`), 206

SpectralMegacomplex (class in `glotaran.builtin.megacomplexes.spectral.spectral_megacomplex`), 210

SpectralShape (class in `glotaran.builtin.megacomplexes.spectral.shape`), 191

SpectralShapeGaussian (class in `glotaran.builtin.megacomplexes.spectral.shape`), 193

SpectralShapeOne (class in `glotaran.builtin.megacomplexes.spectral.shape`), 196

SpectralShapeSkewedGaussian (class in `glotaran.builtin.megacomplexes.spectral.shape`), 198

SpectralShapeZero (class in `glotaran.builtin.megacomplexes.spectral.shape`), 203

split() (`glotaran.utils.ipython.MarkdownStr` method), 570

split\_envvar\_value() (`glotaran.cli.commands.util.ValOrRangeOrList` method), 221

splitlines() (`glotaran.utils.ipython.MarkdownStr` method), 570

standard\_error (`glotaran.parameter.parameter.Parameter` attribute), 424

startswith() (`glotaran.utils.ipython.MarkdownStr` method), 570

string\_to\_tuple() (in module `glotaran.utils.sanitize`), 576

strip() (`glotaran.utils.ipython.MarkdownStr` method),



- 570
- success (glotaran.project.result.Result attribute), 537
- supported\_file\_extensions() (in module glotaran.plugin\_system.base\_registry), 449
- supported\_file\_extensions\_data\_io() (in module glotaran.plugin\_system.data\_io\_registration), 456
- supported\_file\_extensions\_project\_io() (in module glotaran.plugin\_system.project\_io\_registration), 471
- swapcase() (glotaran.utils.ipython.MarkdownStr method), 570
- ## T
- target (glotaran.builtin.megacomplexes.clp\_guide.clp\_guide\_megacomplex.ClpGuideMegacomplex attribute), 114
- target (glotaran.model.clp\_constraint.ClpConstraint attribute), 303
- target (glotaran.model.clp\_constraint.OnlyConstraint attribute), 306
- target (glotaran.model.clp\_constraint.ZeroConstraint attribute), 309
- target (glotaran.model.clp\_penalties.EqualAreaPenalty attribute), 313
- target (glotaran.model.clp\_relation.ClpRelation attribute), 316
- target\_intervals (glotaran.model.clp\_penalties.EqualAreaPenalty attribute), 313
- TeeContext (class in glotaran.utils.tee), 577
- termination\_reason (glotaran.project.result.Result attribute), 537
- time\_explicit (glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file\_type.Type attribute), 65
- TimeExplicitFile (class in glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file\_type.Type attribute), 68
- times() (glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file\_type.Type method), 73
- title() (glotaran.utils.ipython.MarkdownStr method), 570
- to\_csv() (glotaran.optimization.optimization\_history.OptimizationHistory method), 410
- to\_csv() (glotaran.parameter.parameter\_history.ParameterHistory method), 429
- to\_dataframe() (glotaran.parameter.parameter\_history.ParameterHistory method), 429
- to\_dataframe() (glotaran.parameter.parameters.Parameters method), 440
- to\_info\_dict() (glotaran.cli.commands.util.ValOrRangeOfVals method), 221
- to\_parameter\_dict\_list() (glotaran.parameter.parameters.Parameters method), 440
- to\_parameter\_dict\_or\_list() (glotaran.parameter.parameters.Parameters method), 440
- to\_string() (glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation method), 128
- to\_string() (glotaran.model.dataset\_model.ExclusiveMegacomplexIssue method), 326
- to\_string() (glotaran.model.dataset\_model.UniqueMegacomplexIssue method), 326
- transformed\_expression (glotaran.parameter.parameter.Parameter attribute), 424
- translate() (glotaran.utils.ipython.MarkdownStr method), 570
- TsvProjectIo (class in glotaran.builtin.io.pandas.tsv), 90
- tuple\_number (glotaran.utils.regex.RegexPattern attribute), 573
- tuple\_word (glotaran.utils.regex.RegexPattern attribute), 573
- type (glotaran.builtin.megacomplexes.baseline.baseline\_megacomplex.BaselineMegacomplex attribute), 110
- type (glotaran.builtin.megacomplexes.clp\_guide.clp\_guide\_megacomplex.ClpGuideMegacomplex attribute), 114
- type (glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact\_type.Type attribute), 120
- type (glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation\_type.Type attribute), 128
- type (glotaran.builtin.megacomplexes.decay.decay\_megacomplex.DecayMegacomplex attribute), 140
- type (glotaran.builtin.megacomplexes.decay.decay\_parallel\_megacomplex.DecayParallelMegacomplex attribute), 140
- type (glotaran.builtin.megacomplexes.decay.decay\_sequential\_megacomplex.DecaySequentialMegacomplex attribute), 155
- type (glotaran.builtin.megacomplexes.decay.irf.Irf attribute), 160
- type (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 163
- type (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 168
- type (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian attribute), 174
- type (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian attribute), 180
- type (glotaran.builtin.megacomplexes.spectral.shape.SpectralShape attribute), 192
- type (glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian attribute), 196
- type (glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeOne attribute), 198
- type (glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeSkewed attribute), 203
- type (glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeZero attribute), 205

type (glotaran.builtin.megacomplexes.spectral.spectral\_megacomplexes.validate\_spec(glotaran.project.project.scheme.Scheme method), attribute), 213  
 type (glotaran.model.clp\_constraint.ClpConstraint attribute), 303  
 type (glotaran.model.clp\_constraint.OnlyConstraint attribute), 306  
 type (glotaran.model.clp\_constraint.ZeroConstraint attribute), 309  
 type (glotaran.model.clp\_penalties.ClpPenalty attribute), 311  
 type (glotaran.model.clp\_penalties.EqualAreaPenalty attribute), 313

## U

UniqueMegaComplexIssue (class in glotaran.model.dataset\_model), 326  
 update() (glotaran.project.generators.generator.GeneratorArguments method), 482  
 update() (glotaran.utils.io.DatasetMapping method), 560  
 update\_forward\_refs() (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 258  
 update\_forward\_refs() (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage class method), 272  
 update\_forward\_refs() (glotaran.io.preprocessor.preprocessor.CorrectBaseline class method), 286  
 update\_forward\_refs() (glotaran.io.preprocessor.preprocessor.PreProcessingPipeline class method), 300  
 update\_parameter\_expression() (glotaran.parameter.parameters.Parameters method), 440  
 upper() (glotaran.utils.ipython.MarkdownStr method), 570

## V

valid() (glotaran.model.model.Model method), 336  
 valid() (glotaran.project.scheme.Scheme method), 542  
 valid\_label() (in module glotaran.parameter.parameter), 418  
 validate() (glotaran.io.preprocessor.pipeline.PreProcessingPipeline class method), 258  
 validate() (glotaran.io.preprocessor.preprocessor.CorrectBaselineAverage class method), 272  
 validate() (glotaran.io.preprocessor.preprocessor.CorrectBaseline class method), 286  
 validate() (glotaran.io.preprocessor.preprocessor.PreProcessingPipeline class method), 300  
 validate() (glotaran.model.model.Model method), 336  
 validate() (glotaran.project.project.Project method), 500

validate\_spec(glotaran.project.project.scheme.Scheme method), 542  
 validate\_cmd() (in module glotaran.cli.commands.validate), 222  
 validate\_global\_megacomplexes() (in module glotaran.model.dataset\_model), 322  
 validate\_megacomplexes() (in module glotaran.model.dataset\_model), 322  
 validate\_oscillation\_parameter() (in module glotaran.builtin.megacomplexes.damped\_oscillation.damped\_oscillation), 123  
 ValOrRangeOrList (class in glotaran.cli.commands.util), 218  
 value (glotaran.io.preprocessor.preprocessor.CorrectBaselineValue attribute), 286  
 value (glotaran.model.weight.Weight attribute), 338  
 value (glotaran.parameter.parameter.Parameter attribute), 424  
 values() (glotaran.project.generators.generator.GeneratorArguments method), 482  
 values() (glotaran.project.project\_registry.ItemMapping method), 515  
 values() (glotaran.utils.io.DatasetMapping method), 560  
 verify (glotaran.parameter.parameter.Parameter attribute), 424  
 verify() (glotaran.project.result.Result method), 537  
 version (glotaran.project.project.Project attribute), 501

## W

warn\_deprecated() (in module glotaran.deprecation.deprecation\_utils), 231  
 wavelength\_explicit (glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file.DataFileT attribute), 65  
 WavelengthExplicitFile (class in glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file), 70  
 wavelengths() (glotaran.builtin.io.ascii.wavelength\_time\_explicit\_file.WavelengthExplicitFile method), 73  
 Weight (class in glotaran.model.weight), 337  
 weight (glotaran.model.clp\_penalties.EqualAreaPenalty attribute), 313  
 weights (glotaran.model.model.Model attribute), 337  
 wlsfit (glotaran.builtin.megacomplexes.coherent\_artifact.coherent\_artifact attribute), 120  
 wlsfit (glotaran.builtin.megacomplexes.decay.irf.IrfGaussian attribute), 163  
 width (glotaran.builtin.megacomplexes.decay.irf.IrfMultiGaussian attribute), 168  
 width (glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian attribute), 174

`width`(`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian`  
`attribute`), 180  
`width`(`glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeGaussian`  
`attribute`), 196  
`width`(`glotaran.builtin.megacomplexes.spectral.shape.SpectralShapeSkewedGaussian`  
`attribute`), 203  
`width_dispersion_coefficients`  
(`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralGaussian`  
`attribute`), 174  
`width_dispersion_coefficients`  
(`glotaran.builtin.megacomplexes.decay.irf.IrfSpectralMultiGaussian`  
`attribute`), 180  
`word` (`glotaran.utils.regex.RegexPattern` `attribute`), 573  
`write()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.ExplicitFile`  
`method`), 67  
`write()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.TimeExplicitFile`  
`method`), 70  
`write()` (`glotaran.builtin.io.ascii.wavelength_time_explicit_file.WavelengthExplicitFile`  
`method`), 73  
`write()` (`glotaran.utils.tee.TeeContext` `method`), 578  
`write_data()` (in module `glotaran.cli.commands.util`),  
217  
`write_dict()` (in module `glotaran.builtin.io.yml.utils`),  
101

## X

`xtol` (`glotaran.project.scheme.Scheme` `attribute`), 542

## Y

`YmlProjectIo` (class in `glotaran.builtin.io.yml.yml`), 102

## Z

`ZeroConstraint` (class in  
`glotaran.model.clp_constraint`), 306  
`zfill()` (`glotaran.utils.ipython.MarkdownStr` `method`),  
570