PCP_cleaned_data

April 24, 2021

1 PCP case study

1.1 No chl prime

1.1.1 Notebook helper function for pretty output

[1]:
```
from IPython.display import Markdown, display


def print_md(markdown_printable):
    """Pretty render Markdown."""
    display(Markdown(str(markdown_printable)))


def print_yaml_file(file_path):
    """Pretty render yaml file."""
    with open(file_path) as f:
        print_md(f"```yaml\n{f.read()}\n```")
```

1.1.2 Plotting functions (pyglotaran_extras + matplotlib)

[2]:
```
import matplotlib.pyplot as plt
from pyglotaran_extras.plotting.plot_overview import plot_overview
from pyglotaran_extras.plotting.plot_svd import plot_svd
from pyglotaran_extras.plotting.style import PlotStyle

plot_style = PlotStyle()
plt.rc("axes", prop_cycle=plot_style.cycler)
plt.rcParams["figure.figsize"] = (10, 7)
```

1.1.3 Analysis functions

[3]:
```
from glotaran.analysis.optimize import optimize
from glotaran.io import load_dataset, load_model, load_parameters
from glotaran.project.scheme import Scheme
```

1.1.4 Read data

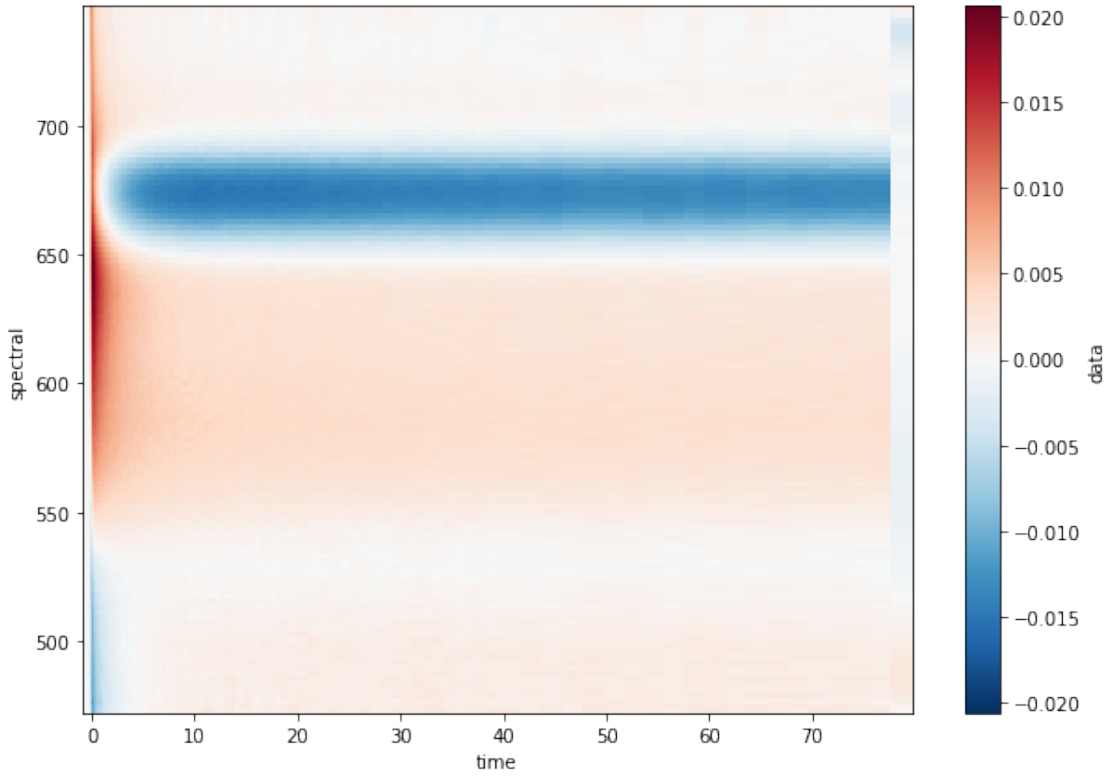
[4]:
```
dataset = load_dataset("PCP490.ascii")
dataset
```

[4]:
```
<xarray.Dataset>
Dimensions:                       (left_singular_value_index: 127,
right_singular_value_index: 127, singular_value_index: 127, spectral: 127, time:
198)
Coordinates:
  * time                          (time) float64 -0.825 -0.725 … 76.7 78.7
  * spectral                      (spectral) float64 473.2 475.3 … 743.2 745.3
Dimensions without coordinates: left_singular_value_index,
right_singular_value_index, singular_value_index
Data variables:
    data                          (time, spectral) float64 0.0002336 … 0.000…
    data_left_singular_vectors    (time, left_singular_value_index) float64 -0…
    data_singular_values          (singular_value_index) float64 0.7069 … 0…
    data_right_singular_vectors   (right_singular_value_index, spectral) float64
…
```

1.2 Investigating the original data

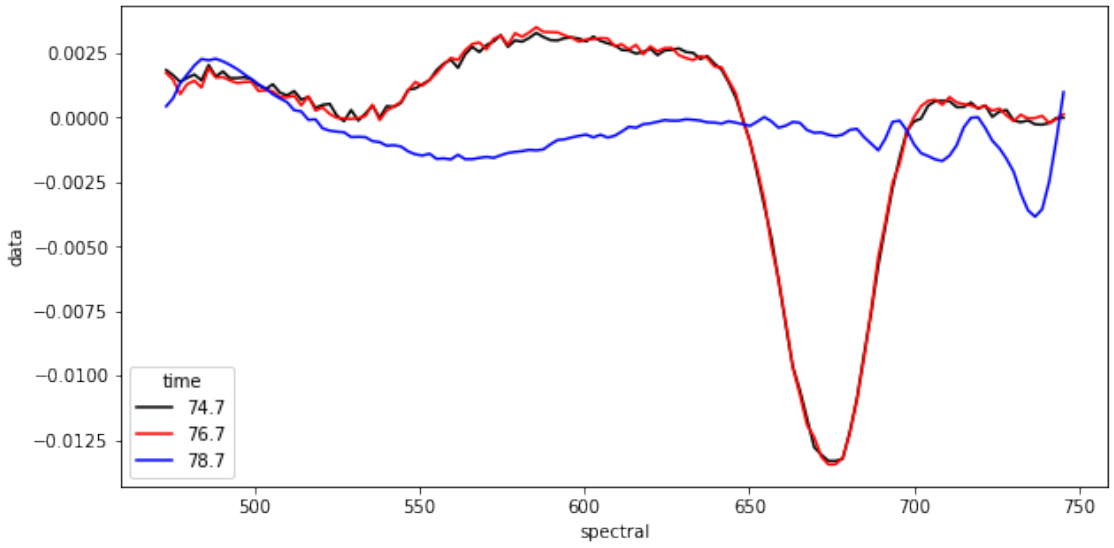
[5]:
```
dataset.data.plot(x="time", y="spectral")
```

[5]:
```
<matplotlib.collections.QuadMesh at 0x1dd3a52ad60>
```



[6]:
```
last_traces = dataset.data.isel(
    time=slice(-3, None),
)
last_traces.plot.line(x="spectral", aspect=2, size=5)
```

[6]:
```
[<matplotlib.lines.Line2D at 0x1dd3a7ed2b0>,
 <matplotlib.lines.Line2D at 0x1dd3a7ed370>,
 <matplotlib.lines.Line2D at 0x1dd3a7ed400>]
```



As can be seen in the two plots above, the last spectral value appears to be erroneous and thus should be removed from the dataset.

1.3 Removing data

To remove data in an index based manner the `isel` method together with `slice` can be used on the `Dataset` to select only the wanted data and the resulting `Dataset` can be assigned to a new variable. The keywords in `isel` are the names of the axes that the selection should be applied on and `slice` is the index based selection interval.

1.3.1 `slice` usage examples

`slice(-i)`

will select all but the last `i` values

`slice(i)`

will select all values up to `i`

`slice(i, None)`

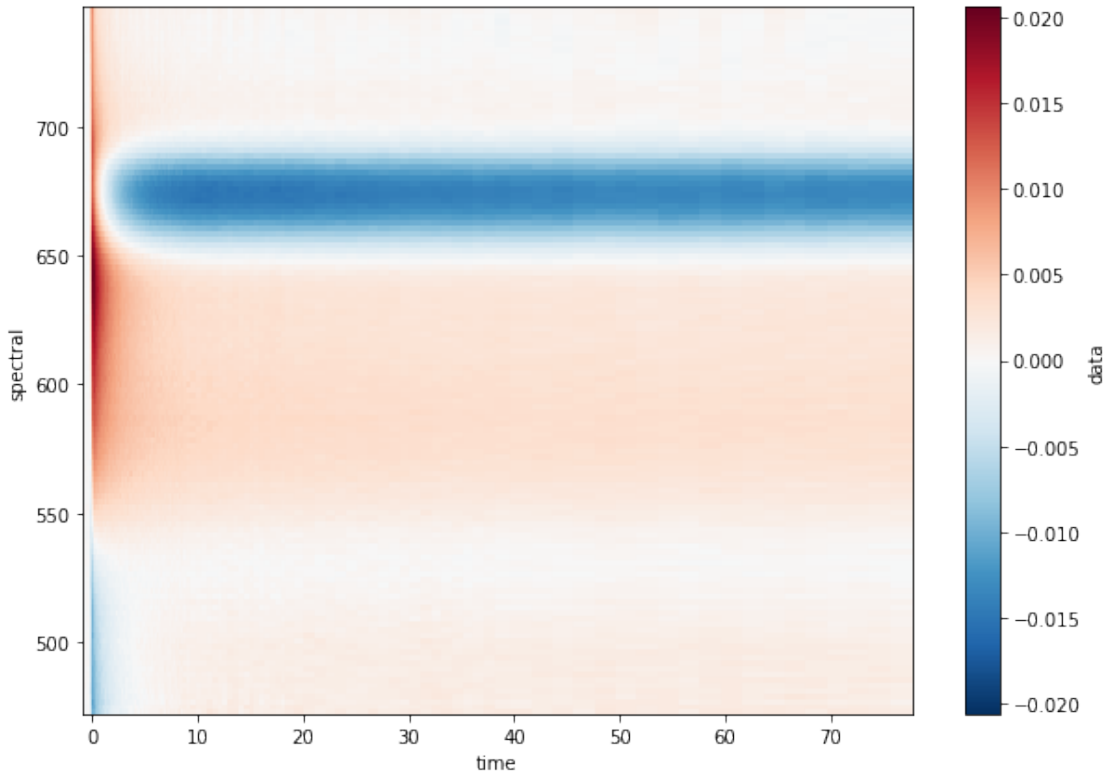
will select all values from index `i` until the end

`slice(i, j)`

will select all values from index `i` until index `j`

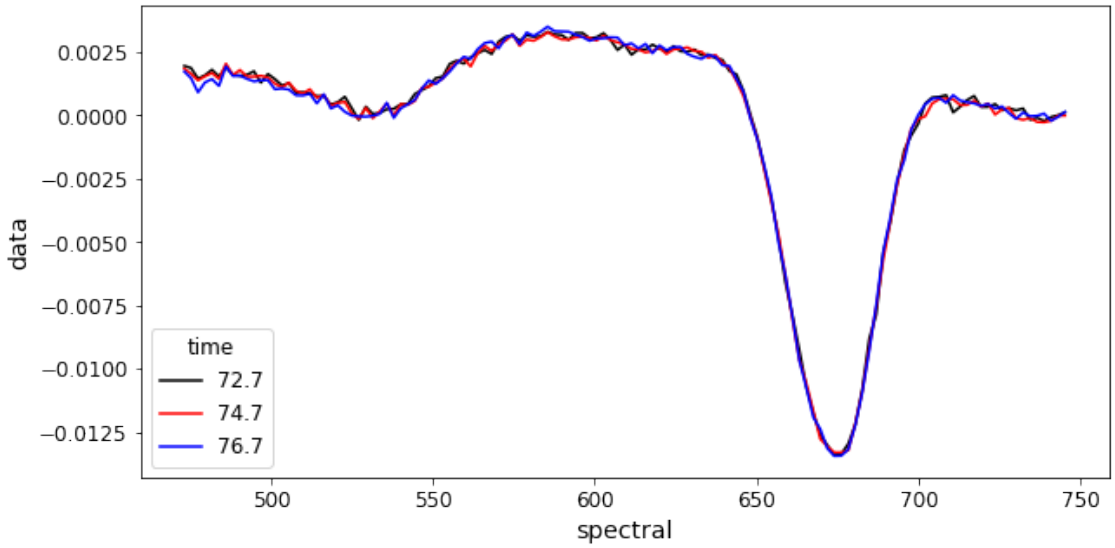
[7]:
```
dataset_cleaned = dataset.isel(time=slice(-1))  # drop the last spectral values
dataset_cleaned.data.plot(x="time", y="spectral")
```

[7]:
```
<matplotlib.collections.QuadMesh at 0x1dd3a639790>
```



[24]:
```
last_traces = dataset_cleaned.data.isel(
    time=slice(-3, None),
)
last_traces.plot.line(x="spectral", aspect=2, size=5)
```

[24]:
```
[<matplotlib.lines.Line2D at 0x1dd3deffc40>,
 <matplotlib.lines.Line2D at 0x1dd3deff610>,
 <matplotlib.lines.Line2D at 0x1dd3deffa00>]
```



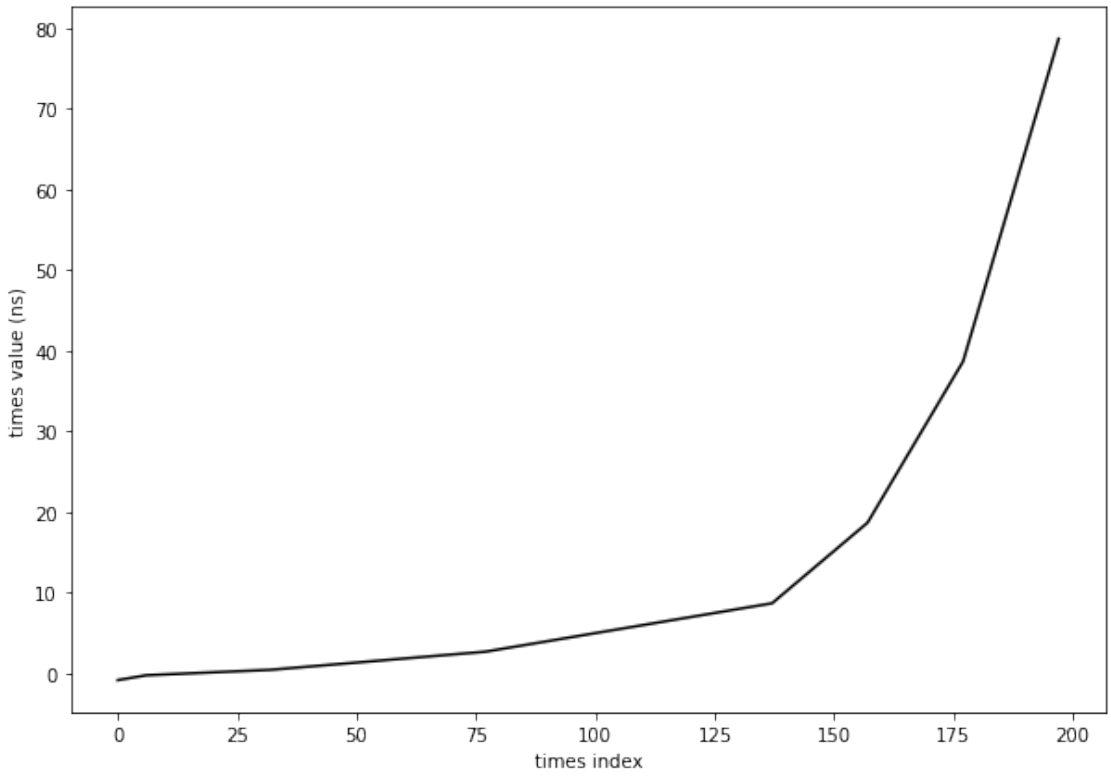
1.4 None linear time axis

To understand why the heatmap of the data appears to be unevenly spaced we can have a look at the time axis's values over their index.

[9]:
```
import numpy as np

plt.plot(np.arange(len(dataset.time)), dataset.time)
plt.xlabel("times index")
plt.ylabel("times value (ns)")
```

[9]:
```
Text(0, 0.5, 'times value (ns)')
```



The exponential values of the times long their index explains the "mistery" of the unevenly spaced heatmap.

1.4.1 Load model and parameters

[10]:
```
PCP_model = load_model("models/PCP-model.yml")
PCP_parameters = load_parameters("models/PCP-parameters.yml")
print_md(PCP_model.validate(parameters=PCP_parameters))
```

Your model is valid.

[11]:
```
print_md(PCP_model)
```

2 Model

Type: kinetic-spectrum

2.1 Initial Concentration

- **input1**:
  - *Label*: input1
  - *Compartments*: ['s1', 's2', 's3', 's4', 's5']
  - *Parameters*: [input.1, input.0, input.0, input.0, input.0]

- *Exclude From Normalize*: []

2.2 K Matrix

- **km1**:
  - *Label*: km1
  - *Matrix*:
    - *('s1', 's1')*: kinetic.9
    - *('s2', 's1')*: kinetic.1
    - *('s5', 's1')*: kinetic.2
    - *('s2', 's2')*: kinetic.9
    - *('s3', 's2')*: kinetic.3
    - *('s4', 's2')*: kinetic.4
    - *('s5', 's2')*: kinetic.8
    - *('s3', 's3')*: kinetic.9
    - *('s5', 's3')*: kinetic.5
    - *('s4', 's4')*: kinetic.9
    - *('s5', 's4')*: kinetic.6
    - *('s5', 's5')*: kinetic.7

2.3 Irf

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

2.4 Dataset

- **dataset1**:
  - *Label*: dataset1
  - *Megacomplex*: ['mc1']
  - *Initial Concentration*: input1
  - *Irf*: irf1

2.5 Megacomplex

- **mc1**:
  - *Label*: mc1
  - *K Matrix*: ['km1']

2.5.1 Create scheme and optimize it

[12]:
```
PCP_scheme = Scheme(PCP_model, PCP_parameters, {"dataset1": dataset_cleaned})
PCP_result = optimize(PCP_scheme)
```

```
   Iteration     Total nfev        Cost      Cost reduction    Step norm
Optimality
       0              1         9.2376e-04
3.59e-02
       1              2         5.5506e-04      3.69e-04       1.57e-02
4.15e-03
       2              3         5.4022e-04      1.48e-05       6.52e-03
8.85e-04
       3              4         5.3898e-04      1.23e-06       2.03e-03
2.55e-04
       4              5         5.3889e-04      9.12e-08       5.50e-04
6.87e-05
       5              6         5.3889e-04      6.64e-09       1.49e-04
1.85e-05
       6              7         5.3889e-04      4.81e-10       4.00e-05
4.98e-06
       7              8         5.3889e-04      3.48e-11       1.08e-05
1.34e-06
       8              9         5.3889e-04      2.52e-12       2.89e-06
3.60e-07
`ftol` termination condition is satisfied.
Function evaluations 9, initial cost 9.2376e-04, final cost 5.3889e-04, first-
order optimality 3.60e-07.
```

[13]:
```
PCP_result.data["dataset1"]
```

[13]:
```
<xarray.Dataset>
Dimensions:                                  (clp_label: 5, component: 5,
from_species: 5, left_singular_value_index: 127, right_singular_value_index:
127, singular_value_index: 127, species: 5, spectral: 127, time: 197,
to_species: 5)
Coordinates:
  * time                                     (time) float64 -0.825 … 76.7
  * spectral                                 (spectral) float64 473.2 … 745.3
  * clp_label                                (clp_label) <U2 's1' 's2' … 's5'
  * species                                  (species) <U2 's1' 's2' … 's5'
    rate                                     (component) float64 -15.14 … …
    lifetime                                 (component) float64 -0.06605 …
  * to_species                               (to_species) <U2 's1' … 's5'
  * from_species                             (from_species) <U2 's1' … 's5'
Dimensions without coordinates: component, left_singular_value_index,
right_singular_value_index, singular_value_index
Data variables: (12/24)
```

```
    data                                        (time, spectral) float64 0.0002…
    data_left_singular_vectors                  (time, left_singular_value_index)
float64 …
    data_singular_values                        (singular_value_index) float64 …
    data_right_singular_vectors                 (right_singular_value_index,
spectral) float64 …
    matrix                                      (time, clp_label) float64 1.259…
    clp                                         (spectral, clp_label) float64 -…
    …                                          …
    a_matrix                                    (component, species) float64 1…
    k_matrix                                    (to_species, from_species) float64
…
    k_matrix_reduced                            (to_species, from_species) float64
…
    irf_center                                  float64 0.009177
    irf_width                                   float64 0.06631
    irf                                         (time) float64 4.309e-35 … 0.0
Attributes:
    root_mean_square_error:           0.00020075526312040566
    weighted_root_mean_square_error:  0.00020075526312040566
```

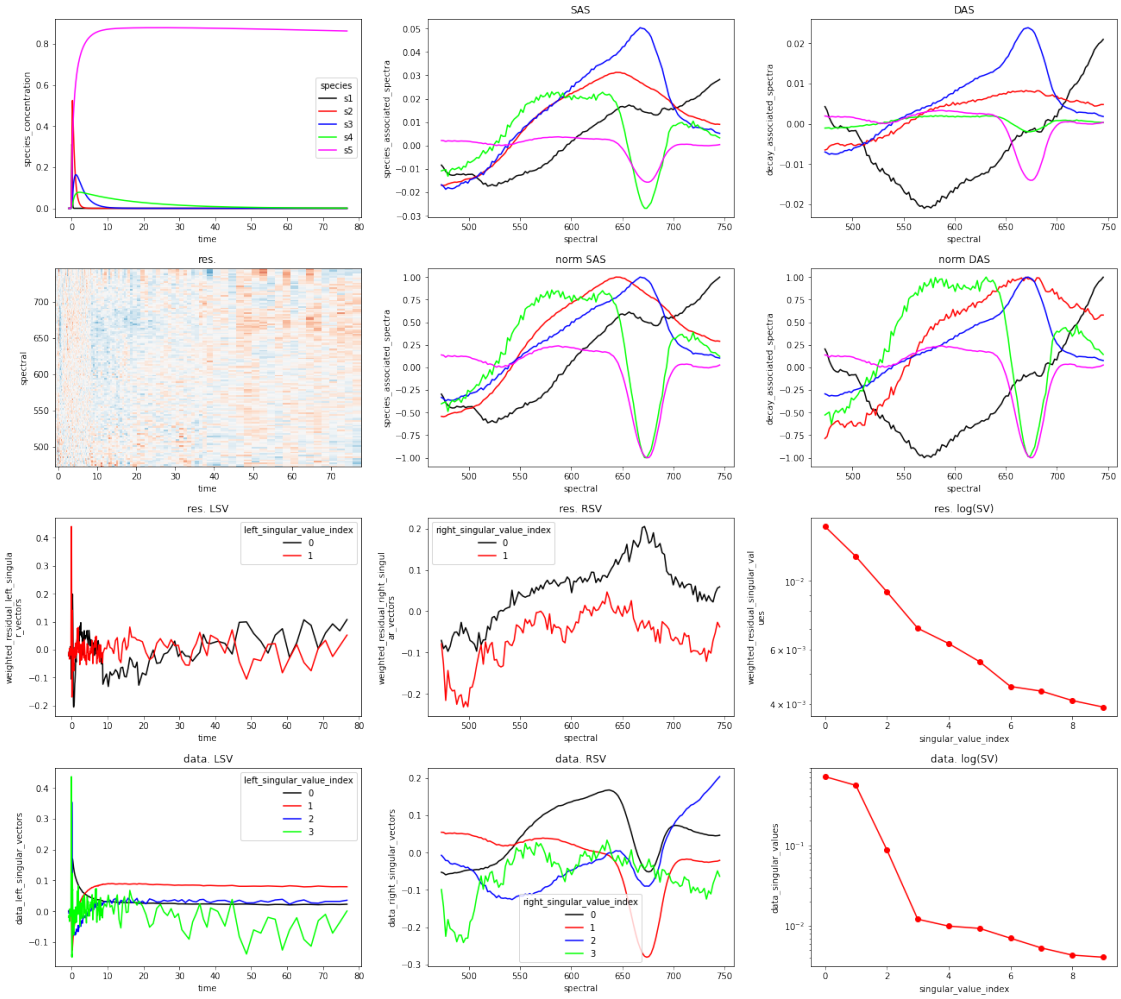
2.5.2 Result plots

[14]:
```
PCP_result.data["dataset1"]
```

[14]:
```
<xarray.Dataset>
Dimensions:                                     (clp_label: 5, component: 5,
from_species: 5, left_singular_value_index: 127, right_singular_value_index:
127, singular_value_index: 127, species: 5, spectral: 127, time: 197,
to_species: 5)
Coordinates:
  * time                                        (time) float64 -0.825 … 76.7
  * spectral                                    (spectral) float64 473.2 … 745.3
  * clp_label                                   (clp_label) <U2 's1' 's2' … 's5'
  * species                                     (species) <U2 's1' 's2' … 's5'
    rate                                        (component) float64 -15.14 … …
    lifetime                                    (component) float64 -0.06605 …
  * to_species                                  (to_species) <U2 's1' … 's5'
  * from_species                                (from_species) <U2 's1' … 's5'
Dimensions without coordinates: component, left_singular_value_index,
right_singular_value_index, singular_value_index
Data variables: (12/24)
    data                                        (time, spectral) float64 0.0002…
    data_left_singular_vectors                  (time, left_singular_value_index)
float64 …
    data_singular_values                        (singular_value_index) float64 …
    data_right_singular_vectors                 (right_singular_value_index,
```

```
spectral) float64 …
    matrix                                  (time, clp_label) float64 1.259…
    clp                                     (spectral, clp_label) float64 -…
    …                                      …
    a_matrix                                (component, species) float64 1…
    k_matrix                                (to_species, from_species) float64
…
    k_matrix_reduced                        (to_species, from_species) float64
…
    irf_center                              float64 0.009177
    irf_width                               float64 0.06631
    irf                                     (time) float64 4.309e-35 … 0.0
Attributes:
    root_mean_square_error:            0.00020075526312040566
    weighted_root_mean_square_error:   0.00020075526312040566
```

[15]:
```
fig = plot_overview(PCP_result.data["dataset1"], linlog=False)
```



As can be seen in the above plot (compared to the original one), cleaning the dataset only influenced the residual and SVD's of the residual and data, but not the SAS and DAS.

[16]:
```
print_md(PCP_result.optimized_parameters)
```

- **input**:

Label	*Value*	*StdErr*	*Min*	*Max*	*Vary*	*Non-Negative*	*Expr*
1	1	0	-inf	inf	False	False	None
0	0	0	-inf	inf	False	False	None

- **irf**:

Label	*Value*	*StdErr*	*Min*	*Max*	*Vary*	*Non-Negative*	*Expr*
center	0.00917684	1.31584	-inf	inf	True	False	None
width	0.0663091	0.94223	-inf	inf	True	False	None

- **kinetic**:

Label	*Value*	*StdErr*	*Min*	*Max*	*Vary*	*Non-Negative*	*Expr*
1	11	0	-inf	inf	False	False	None
2	4.1	0	-inf	inf	False	False	None
3	0.6	0	-inf	inf	False	False	None
4	0.2	0	-inf	inf	False	False	None
5	0.4	0	-inf	inf	False	False	None
6	0.02	0	-inf	inf	False	False	None
7	0.0005	0	-inf	inf	False	False	None
8	0.8	0	-inf	inf	False	False	None
9	0.04	0	-inf	inf	False	False	None

2.6 chl prime

2.6.1 Load model and parameters

Compared to the full data with the erroneous values for `time==78.7` the scaling of the flow to chl prime needed to be changed from

```
scaling:
  - [0.75, {vary: false}]
  - [0.25, {vary: false}]
```

to

```
scaling:
  - [0.80, {vary: false}]
  - [0.20, {vary: false}]
```

in order to counter the bleaching.

[17]:
```
PCP_chl_prime_model = load_model("models/PCP-chl-prime-model.yml")
PCP_chl_prime_parameters = load_parameters(
    "models/PCP-chl-prime-parameters-sliced-data.yml"
)
print_md(PCP_model.validate(parameters=PCP_chl_prime_parameters))
```

Your model is valid.

[18]:
```
print_md(PCP_chl_prime_model)
```

3 Model

Type: kinetic-spectrum

3.1 Initial Concentration

- **input1**:
  - *Label*: input1
  - *Compartments*: ['s1', 's2', 's3', 's4', 's5', 's6']
  - *Parameters*: [input.1, input.0, input.0, input.0, input.0, input.0]
  - *Exclude From Normalize*: []

3.2 K Matrix

- **km1**:
  - *Label*: km1
  - *Matrix*:
    - *('s1', 's1')*: kinetic.9
    - *('s2', 's1')*: kinetic.1
    - *('s5', 's1')*: kinetic.11
    - *('s6', 's1')*: kinetic.15
    - *('s2', 's2')*: kinetic.9
    - *('s3', 's2')*: kinetic.3
    - *('s4', 's2')*: kinetic.4
    - *('s5', 's2')*: kinetic.12
    - *('s6', 's2')*: kinetic.16
    - *('s3', 's3')*: kinetic.9
    - *('s5', 's3')*: kinetic.13
    - *('s6', 's3')*: kinetic.17
    - *('s4', 's4')*: kinetic.9
    - *('s5', 's4')*: kinetic.14
    - *('s6', 's4')*: kinetic.18
    - *('s5', 's5')*: kinetic.7
    - *('s6', 's6')*: kinetic.10

3.3 Irf

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

3.4 Dataset

- **dataset1**:
  - *Label*: dataset1
  - *Megacomplex*: [‘mc1’]
  - *Initial Concentration*: input1
  - *Irf*: irf1

3.5 Megacomplex

- **mc1**:
  - *Label*: mc1
  - *K Matrix*: [‘km1’]

3.6 Spectral Relations

-
  - *Compartment*: s5
  - *Target*: s6
  - *Parameter*: rel.r1
  - *Interval*: [[0, 1000]]

3.6.1 Create sceme and optimize it

[19]:
```
PCP_chl_prime_scheme = Scheme(
    PCP_chl_prime_model, PCP_chl_prime_parameters, {"dataset1": dataset_cleaned}
)
PCP_chl_prime_result = optimize(PCP_chl_prime_scheme)
```

```
   Iteration     Total nfev        Cost      Cost reduction    Step norm
Optimality
       0              1         9.3012e-04
3.56e-02
       1              2         5.7945e-04      3.51e-04       1.51e-02
3.84e-03
       2              3         5.6709e-04      1.24e-05       5.71e-03
9.06e-04
       3              4         5.6583e-04      1.26e-06       1.96e-03
2.96e-04
       4              5         5.6571e-04      1.19e-07       6.01e-04
```

```
9.05e-05
       5               6         5.6570e-04      1.11e-08       1.84e-04
2.77e-05
       6               7         5.6570e-04      1.04e-09       5.61e-05
8.43e-06
       7               8         5.6570e-04      9.63e-11       1.71e-05
2.57e-06
       8               9         5.6570e-04      8.94e-12       5.21e-06
7.83e-07
       9              10         5.6570e-04      8.30e-13       1.59e-06
2.38e-07
`ftol` termination condition is satisfied.
Function evaluations 10, initial cost 9.3012e-04, final cost 5.6570e-04, first-
order optimality 2.38e-07.
```

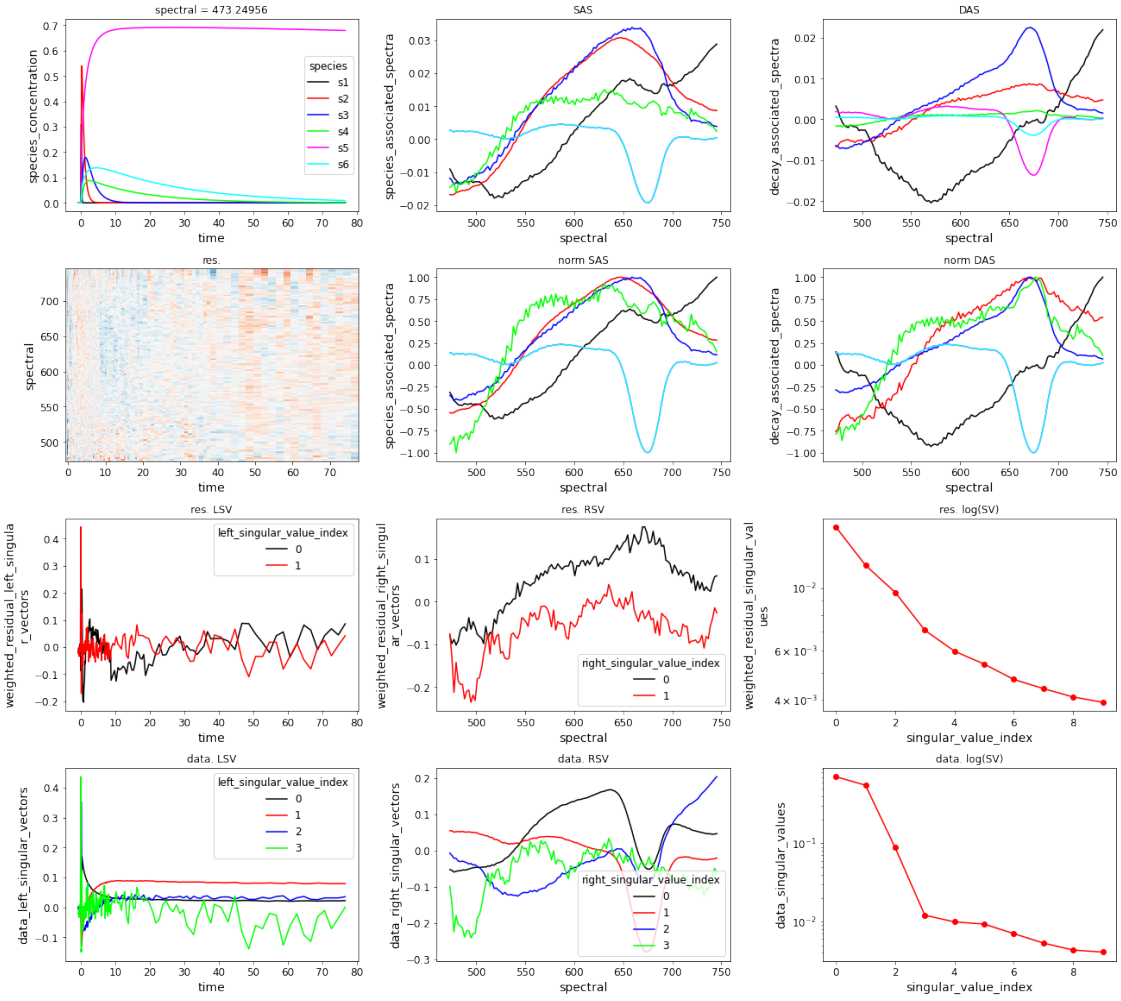
[20]:
```
PCP_chl_prime_result.data["dataset1"]
```

[20]:
```
<xarray.Dataset>
Dimensions:                                    (clp_label: 6, component: 6,
from_species: 6, left_singular_value_index: 127, right_singular_value_index:
127, singular_value_index: 127, species: 6, spectral: 127, time: 197,
to_species: 6)
Coordinates:
  * time                                       (time) float64 -0.825 … 76.7
  * spectral                                   (spectral) float64 473.2 … 745.3
  * clp_label                                  (clp_label) <U2 's1' 's2' … 's6'
  * species                                    (species) <U2 's1' 's2' … 's6'
    rate                                       (component) float64 -15.14 … …
    lifetime                                   (component) float64 -0.06605 …
  * to_species                                 (to_species) <U2 's1' … 's6'
  * from_species                               (from_species) <U2 's1' … 's6'
Dimensions without coordinates: component, left_singular_value_index,
right_singular_value_index, singular_value_index
Data variables: (12/24)
    data                                       (time, spectral) float64 0.0002…
    data_left_singular_vectors                 (time, left_singular_value_index)
float64 …
    data_singular_values                       (singular_value_index) float64 …
    data_right_singular_vectors                (right_singular_value_index,
spectral) float64 …
    matrix                                     (spectral, time, clp_label)
float64 …
    clp                                        (spectral, clp_label) float64 -…
    …                                         …
    a_matrix                                   (component, species) float64 1…
    k_matrix                                   (to_species, from_species) float64
…
```

```
    k_matrix_reduced                          (to_species, from_species) float64
…
    irf_center                                float64 0.011
    irf_width                                 float64 0.06682
    irf                                       (time) float64 1.017e-34 … 0.0
Attributes:
    root_mean_square_error:           0.00021265320925715141
    weighted_root_mean_square_error:  0.00021265320925715141
```

3.6.2 Result plots

[21]:
```
fig = plot_overview(PCP_chl_prime_result.data["dataset1"], linlog=False)
```



[22]:
```
print_md(PCP_chl_prime_result.optimized_parameters)
```

- **input**:

Label	*Value*	*StdErr*	*Min*	*Max*	*Vary*	*Non-Negative*	*Expr*
1	1	0	-inf	inf	False	False	None
0	0	0	-inf	inf	False	False	None

- **irf**:

Label	*Value*	*StdErr*	*Min*	*Max*	*Vary*	*Non-Negative*	*Expr*
center	0.0109996	1.26728	-inf	inf	True	False	None
width	0.0668175	0.932177	-inf	inf	True	False	None

- **kinetic**:

Label	*Value*	*StdErr*	*Min*	*Max*	*Vary*	*Non-Negative*	*Expr*
1	11	0	-inf	inf	False	False	None
2	4.1	0	-inf	inf	False	False	None
3	0.6	0	-inf	inf	False	False	None
4	0.2	0	-inf	inf	False	False	None
5	0.4	0	-inf	inf	False	False	None
6	0.02	0	-inf	inf	False	False	None
7	0.0005	0	-inf	inf	False	False	None
8	0.6	0	-inf	inf	False	False	None
9	0.04	0	-inf	inf	False	False	None
10	0.04	0	-inf	inf	False	False	None
11	3.28	0	-inf	inf	False	False	$kinetic.2 * $scaling.1
12	0.48	0	-inf	inf	False	False	$kinetic.8 * $scaling.1
13	0.32	0	-inf	inf	False	False	$kinetic.5 * $scaling.1
14	0.016	0	-inf	inf	False	False	$kinetic.6 * $scaling.1
15	0.82	0	-inf	inf	False	False	$kinetic.2 * $scaling.2
16	0.12	0	-inf	inf	False	False	$kinetic.8 * $scaling.2
17	0.08	0	-inf	inf	False	False	$kinetic.5 * $scaling.2
18	0.004	0	-inf	inf	False	False	$kinetic.6 * $scaling.2

- **rel**:

Label	*Value*	*StdErr*	*Min*	*Max*	*Vary*	*Non-Negative*	*Expr*
r1	1	0	-inf	inf	False	False	None

- **scaling**:

Label	*Value*	*StdErr*	*Min*	*Max*	*Vary*	*Non-Negative*	*Expr*
1	0.8	0	-inf	inf	False	False	None
2	0.2	0	-inf	inf	False	False	None

[ ]: