This section is intended as a guide to the software. By going through the analysis steps for some simple simulation data, the reader can get to know the layout and major functionalities of the program.

The reader should note that the functioning of some widgets may not be explicitly declared as they are sometimes deemed self-explanatory. Also, though far from claiming to have covered everything, the program is equipped with many verbose tooltips and error messages, which should, hopefully, enable the user to navigate the program if something is unclear or goes wrong.

0.1. The simulated data

The simulated data is intended as a rough approximation of the resulting data as recorded in a pump-probe-spectroscopy measurement of a non-descript material as introduced in section **??**. Thus the simulation computes an initial "excitation" whose magnitude at increasing times after the excitation is "measured". A positive (negative) magnitude could be interpreted as a higher (lower) transmission of the excited material at a certain wavelength when compared to the unexcited material.

The initial "excitation" consists of a number of Gaussians, i.e. a number of spectral "components". Added to the pure Gaussians is some pseudo-random noise, which is intended to imitate (to some degree) the fluctuations in the excitation process:

$$\text{excitation}(\lambda) = \sum_i^2 \text{comp}_i(\lambda) = \sum_i^2 \left(a_i \cdot \exp\left(-\frac{(\lambda - \mu_i)^2}{2 \cdot \sigma_i^2} \right) + \text{p}_{\text{overall}} \cdot \text{p}_{\text{pump}} \cdot X \sim \mathcal{N}(0,\, 2) \right) \tag{1}$$

Here the a_i, μ_i and σ_i represent the amplitudes, peak position and standard deviations of the Gaussians. $p_{overall}$ and p_{pump} are used as parameters to configure the scale of the noise contribution in the component Gaussian. (There are two noise scale parameters included, as this facilitates the configuration of multiple simulations, which is useful for the following sections.) The pseudo-random noise is drawn from a Normal distribution.

The component $comp_i$ then represents an array of intensities at each wavelength λ for which it is computed. It decays exponentially with decay constant τ_i. Thus the "measured" magnitude m of the "excitation" at each wavelength λ and at time t is computed as:

$$m(\lambda, t) = \sum_i^2 \left(\text{comp}_i(\lambda) \cdot \exp\left(-\frac{t}{\tau_i} \right) + \text{p}_{\text{overall}} \cdot \text{p}_{\text{probe}} \cdot X \sim \mathcal{N}(0,\, 2) \right) \tag{2}$$

where again some pseudo-random noise is added (and scaled via $p_{overall}$ and p_{probe}) to represent the fluctuations in the measurement process.

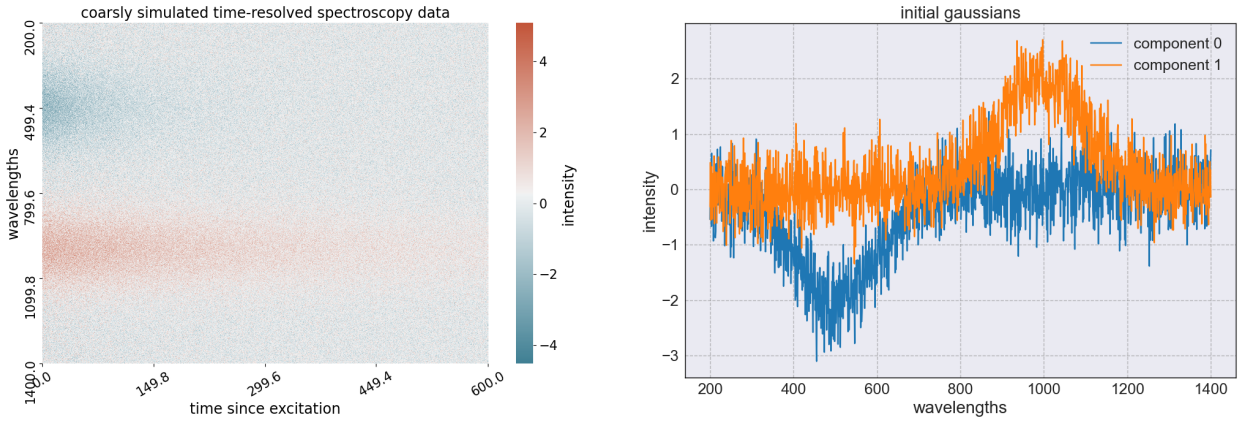
In order to somewhat more accurately replicate a pump-probe-spectroscopy measurement, the initial "excitation" is generated anew for each "measurement". I.e. the noise on the initial Gaussians is different for each "measurement".

For this introductory chapter, the simulation contains two components with the following parameters:

- amplitudes a_0, a_1: -2, 2
- peak positions μ_0, μ_1: 500, 1000
- standard deviations σ_i: 100
- decay constants τ_0, τ_1: 100, 400

- p_{pump}, p_{probe}: 2.0
- $p_{overall}$: 0.1

A graphical representation of the simulated data can be seen in fig. 0.1



(i) **(ii)**

Figure 0.1 – Simulated idealised time-resolved spectrum and an example of some distorted initial Gaussians as used in the simulation.

The simulation is implemented in the `SimulateData\sim_MultiExcitation.py` file, the configuration is done in `..\configFiles\wavelength_overlap\overlap_0.ini`.

0.2. Data file format

The program expects a data file to be in the format depicted in table 1. The cell marked with x is the first value in the file and it will not be used in any computation. It should be filled with a placeholder value like e.g. 0.

At the moment only `.txt` or `.csv` files are supported. In the `.txt` file the values should be delimited with some white-space characters, as the program splits the values on white-space, if the `.txt` file type is detected. (In `.csv` files the values should of course be delimited by commas.)

For convenience sake, data files should be stored in the `DataFiles` folder, which already exists in the GitHub repository (i.e. which will be copied when downloading the code). Other folders can also be accessed from within the program, this just requires more clicks.

x	*wavelengths*				
time	data	data	data	data	data
	data	data	data	data	data
	data	data	data	data	data
	data	data	data	data	data
	data	data	data	data	data

Table 1 – The value in the cell marked with x will not be used in any computation. It should be filled with a placeholder value. One row of data represents the intensities measured at each wavelength at the row's corresponding time (delay).

0.3. Inspection of the data and its SVD in the GUI

As already mentioned, the package can be downloaded from the *GitHub* repository: `https://github.com/claudioHerger/TA_analyis_GUI`. Once in it's destination folder, the main program (`TA_analysis_GUI.py`) can be started from a terminal via the OS specific *python* command. (*python .\TA_analyis_GUI.py* in Windows when the current working directory is the mentioned destination folder.)

Once the program has loaded the window seen in fig. 0.2 is displayed. Use button set ⓪ in fig. 0.2 to choose and inspect a data file. The data can be displayed by clicking the *show original data*-button. (If no file has been selected via the *change data file*-button, a corresponding file dialog window is opened.)

Figure 0.2 – ⓪ Display data via the *show original data*-button. Change the data file via the *change data file*-button. Change the data matrix dimensions via the *change matrix bounds*-button.
① Select SVD components for construction of the reduced matrix Ψ_{SVD} (equation (**??**)) via *show SVD components data*-button. Ψ_{SVD} and its difference from the original data matrix will be displayed as heat maps. Selected components will also be used for the fit.
② Start a fit using the selected components via the *show SVD fit data*-button. Once a fit has been successfully conducted, the Ψ_{fit} matrix (equation (**??**)) and its difference from the original data matrix will be displayed. Use the *fit results for tabnr*-entry to display a window containing exhaustive information on the fit results. Use the *DAS for tabnr*-entry to inspect the resulting DAS.
③ Define a fit function via the corresponding button. Once it has been successfully parsed, check the checkbox next to the button to use the defined function in the fit. (Checkbox checked: user defined fit function will be used. Checkbox unchecked: default fit function will be used.) Use the *Set initial fit parameter values*-button to do just that. If a fit does not converge, it can be helpful to set different start values. (The parsed user defined fit function will be stored to a file, so too will be the initial fit parameter values.)
ⓐ The *set colormaps for heat maps*-button can be used to change the color schemes for the heat map of the data matrices.
ⓑ Select the fit method from a drop-down menu.

The simulation data used in this tutorial is in the file `wavelength_overlap\overlap0.txt`. After selecting the file in the file dialog, a heat map depicting the data gets displayed on the GUI. By comparing fig. 0.1i

and fig. 0.3 one can see, that the complete data matrix is displayed. This could be changed using the *change matrix bounds*-button, which allows the user to set specific bounds for the wavelengths and time delays. This way the data matrix can be downsized in the wavelength and or time step dimension.[1] By again employing the *show original data*-button, a heat map of the downsized data matrix would be produced. Until new bounds are set (or until the data file is changed, in which case the bounds are reset, so that by default the complete matrix is used again), this downsized data matrix is also used for any subsequent analysis.

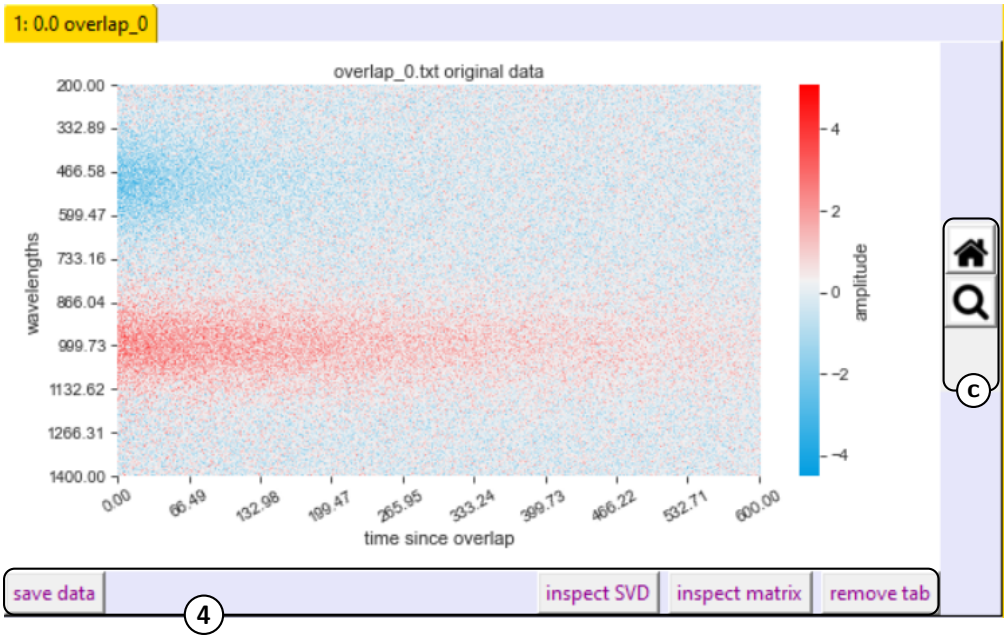


Figure 0.3 – ④ Save the figure and its corresponding matrix data to the `ResultData`-folder (within the `DataFiles`-folder) via the *save data*-button. Open a window to inspect the SVD of the data matrix via the *inspect SVD*-button. Inspect the data matrix row-by-row and column-by-column in a window via the *inspect matrix*-button. Remove the tab via the *remove tab*-button.
ⓒ These buttons can be used to zoom into the heatmap and then restore the initial heatmap again.

By clicking the *inspect SVD*-button (fig. 0.3), a window is opened to inspect the singular values and vectors of the data matrix (corresponding to the set start time value). In this window the first ten (weighted) singular vectors, and the first 50 singular values can be plotted. These numbers are chosen as they should suffice to differentiate between signal and noise components in most cases.

For the simulated data, the first two singular values are the largest (see fig. 0.4). Also, the first two singular vectors (both left and right) are the ones displaying most structure. Thus these two components are selected for the further analysis steps.

[1] This might be useful if one wants to cut some wavelengths or time steps from the data matrix.

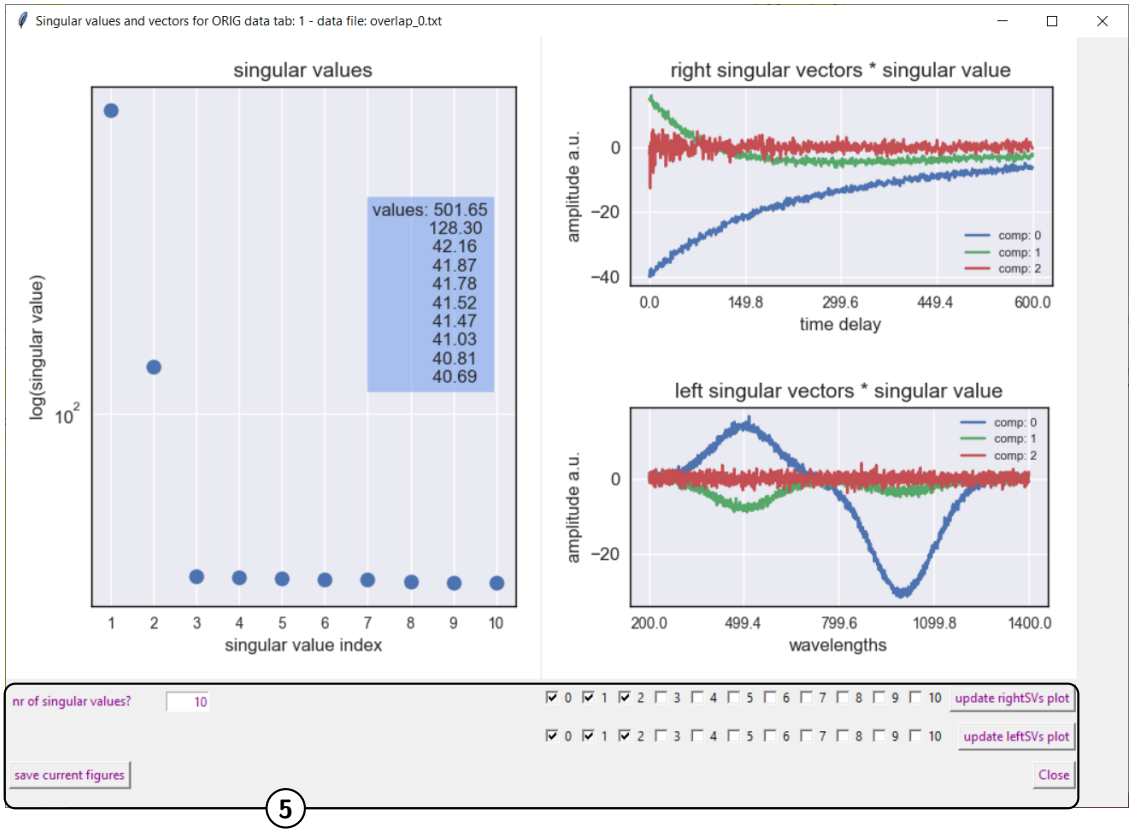
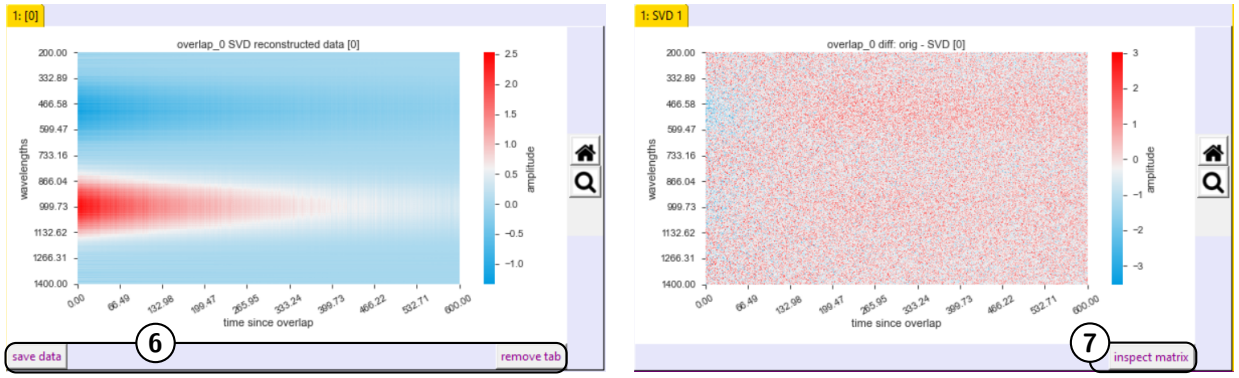


Figure 0.4 – ⑤ Save currently displayed figures via the *save current figures*-button (figures are saved in the same folder as the data saved via *save data* button in fig. 0.3). Set the number of displayed singular values via the *nr of singular values?*-entry (hit Return key to update singular values plot). Choose a set of singular vectors to inspect and display them via the *update rightSVs plot*-button and*update leftSVs plot*-button.

0.4. Comparing different SVD component sets

Using the checkboxes in ① from fig. 0.2, a set of SVD components can be selected. By clicking the *show SVD component data*-button the reduced matrix Ψ_{SVD} (equation (**??**)) can then be viewed. To facilitate comparison of Ψ_{SVD} with the original data, the difference matrix $\Delta\Psi_{SVD}$ is also displayed.

For illustrations sake, the Ψ_{SVD} and $\Delta\Psi_{SVD}$ of the simulated data using only the $0th$ component are displayed in fig. 0.5.

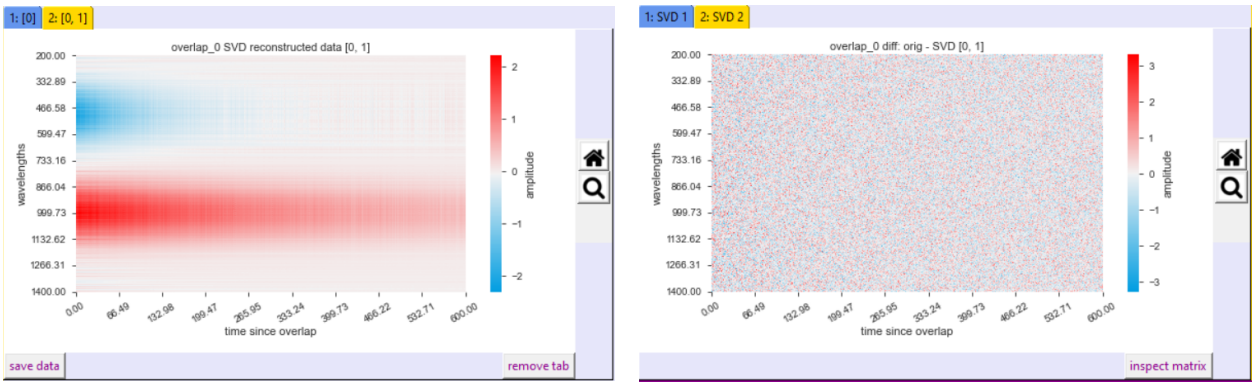


(i) Ψ_{SVD} **(ii)** $\Delta\Psi_{SVD}$

Figure 0.5 – The reduced matrix using only the $0th$ component of the simulated data in fig. 0.3.
⑥ Save this plot and corresponding data to file or remove the tab from the GUI, which will also remove the tab displaying $\Delta\Psi_{SVD}$.
⑦ Same as in fig. 0.3 but this time the $\Delta\Psi_{SVD}$ can be inspected instead of the actual data matrix Ψ.

Although difficult to see, the $\Delta\Psi_{SVD}$ matrix shown in fig. 0.5ii still displays some structure around the mean wavelengths of both simulated Gaussians. (Using the *inspect matrix*-button in ⑦, the $\Delta\Psi_{SVD}$ could further be inspected row-by-row (i.e. wavelength-by-wavelength) and column-by-column.)

In contrast, when selecting both the $0th$ and $1st$ SVD component for the construction of Ψ_{SVD} and $\Delta\Psi_{SVD}$, this small remaining structure disappears, as can be seen in fig. 0.6.
Thus, fitting the data with those two components selected seems appropriate.

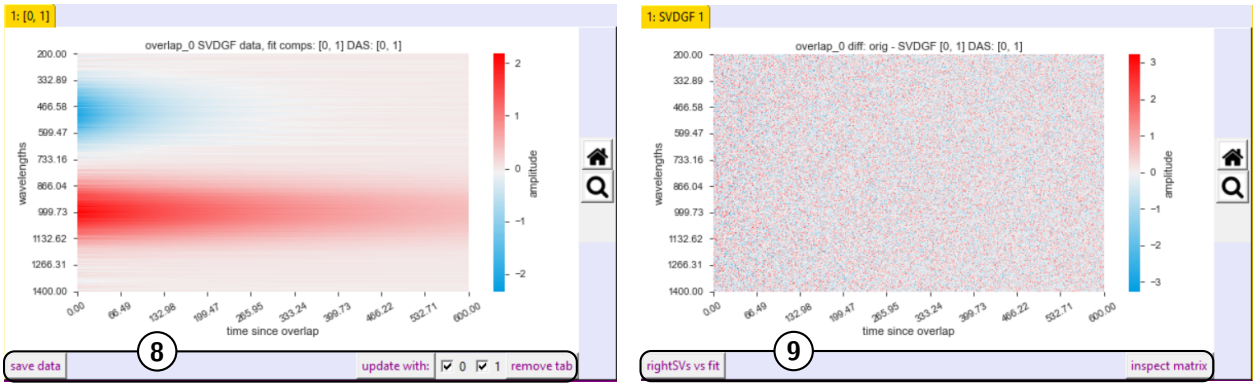


(i) Ψ_{SVD} **(ii)** $\Delta\Psi_{SVD}$

Figure 0.6 – The reduced matrix using the $0th$ and $1st$ component of the simulated data in fig. 0.3.

0.5. Performing a fit

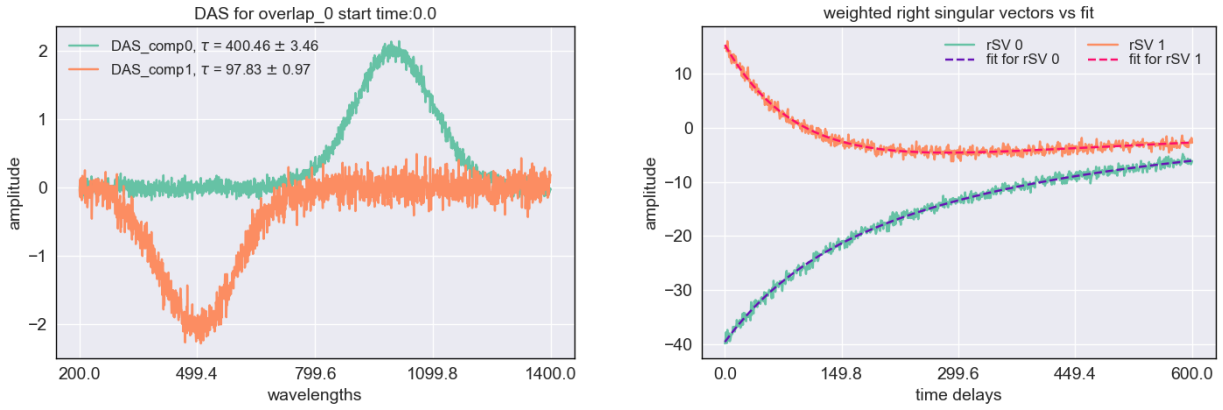
Using the *show SVD fit data*-button in ② a fit can be conducted. (Note that in this section the default initial fit parameter values were used, i.e. all $\tau_i = 50$ and all $a_{ij} = 0.7$.) By default generic fit functions are applied to the weighted right singular vectors (equations (**??**) and (**??**)). Again, two heat maps are produced: one for Ψ_{fit} the other depicting $\Delta\Psi_{fit}$.
As one can see in fig. 0.7, the simulated data is well reproduced by the fit, as no clear structure remains in $\Delta\Psi_{fit}$.



(i) Ψ_{fit} **(ii)** $\Delta\Psi_{fit}$

Figure 0.7 – The matrix resulting from the fit procedure using the $0th$ and $1st$ component of the simulated data in fig. 0.3.
⑧ Save this plot and corresponding data to file or remove the tab from the GUI, which will also remove the tab displaying $\Delta\Psi_{fit}$. Additionally, using the checkboxes and the *update with*-button, the user can update this plot and the one displaying $\Delta\Psi_{fit}$ by including only a selection of *DAS*.
⑨ use the *rightSVs vs fit*- button to open a window within which the weighted right singular vectors are plotted together with their reconstruction via the resulting fit parameters (see fig. 0.8ii for such a plot).

The resulting *DAS* can be inspected using the *DAS for tabnr?*-entry ②: simply enter the number of the tab that displays Ψ_{fit} and press the Return key. Also useful may be to check how the weighted right singular vectors are reconstructed by the resulting fit parameters. This can be done with the *rightSVs vs fit*-button in ⑨. In fig. 0.8 these fit results can be seen.
By comparing fig. 0.8i with fig. 0.1ii one can see that the *DAS* well reproduce the initial Gaussians used in the simulation. Additionally, fig. 0.8ii shows that the right singular vectors are also well reproduced by the fit.
All the fit results and further information can be inspected in the GUI by using the *fit results for tabnr?*-entry in the widget-set ②. The displayed text file would also be stored when saving the fit data using the *save data*-button in ⑧.



(i) **(ii)**

Figure 0.8 – The resulting *DAS* and the weighted right singular vectors as well as their reconstructions using the resulting fit parameters in equation (**??**).

0.6. Further information on additional GUI widgets

0.6.1. Changing the data matrix dimensions

Using the *change matrix bounds*-button in ⓪ with the data file `wavelength_overlap\overlap0.txt` selected, the window depicted in fig. 0.9 is opened. There the minimum and maximum values for the time delays and or wavelengths can be set. By clicking the *use these bounds*-button, the entered values are evaluated and if valid they are applied from now on to downsize the data matrix. Whenever the data file is changed, the bounds are reset so that by default the complete data matrix is used until bounds for this data file are entered. When opening the window, the displayed values in the entries correspond to the maximum and minimum time delay/wavelength values of the corresponding data file.

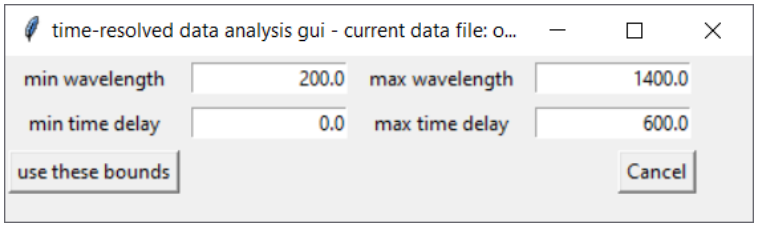


Figure 0.9 – Window to change the matrix dimensions.

0.6.2. Setting initial values for the fit parameters

Using the *Set initial fit parameter values*-button in ③, the window seen in fig. 0.10 can be opened. There the user may define a set of initial values for the fit parameters. The fit procedure (in `FunctionsUsedByPlotClasses/get_SVDGFit_parameters.py`) initialises n_{comps} decay constants τ_j and $n_{comps} \cdot n_{comps}$ amplitude parameters a_{ij} (n_{comps} = the number of components selected in ①). This is true if the generic fit function is used (**??**) as well as when any user defined fit function is applied[2]. Thus the initial parameter values set must at least contain that number of τ_j and a_{ij}. If it does not, the fit procedure will inform the user about that error and initialise the fit parameters to default values ($\tau_j = 50$, $a_{ij} = 0.7$).
The values entered in the text fields should be formatted as shown in fig. 0.10 and any value must be

[2]While all those parameters are used in the former, for the latter, any τ_j can be left out and any a_{ij} can be suppressed.

convertible to a float. There are some measures taken to ensure correct parsing of the user entered values, but no guarantee for completeness is given.
If correctly entered, the set of initial values is saved to a file formatted as a python dictionary (`configFiles/Initial_fit_parameter_values.txt`). Upon loading the program, this file is evaluated. If the evaluation is successful, the initial values used in the last session can be reused, else, the program will fallback to default values and the contents of the file will be overwritten once the user defines new values.

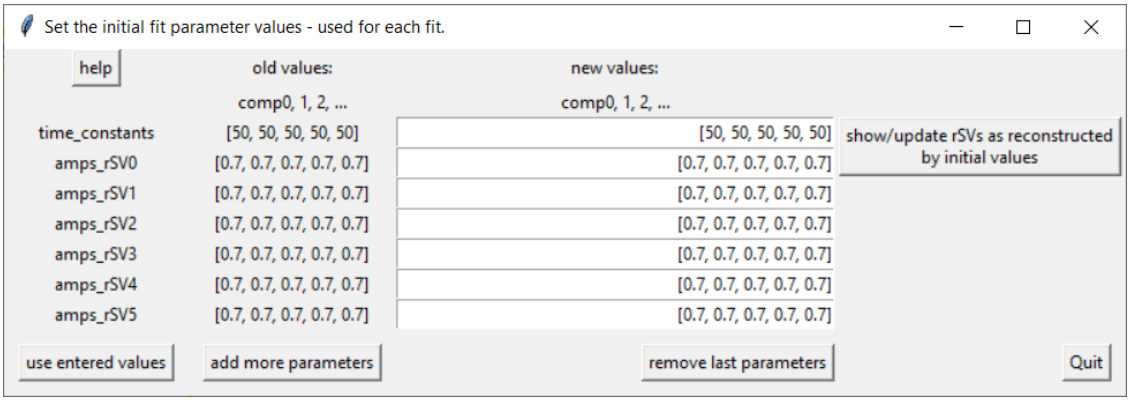


Figure 0.10 – Window to set initial values for the parameters used in the fit procedure. Using the *show/update rSV as reconstructed by initial values*-button a window similar to fig. 0.8ii can be opened to check whether the initial values are very far off from a possible solution.

0.6.3. Defining and using a target model

The *Define fit function*-button in ③ opens a window similar to the one in fig. 0.11. Each fit function f_i (equation (**??**)) will still be a sum over the selected components (therefore the *summand_comp* labels in the window), but the form of these summands can be set by entering a (somewhat) mathematical statement into the entries. To evaluate the summands in the fit procedure, they need to be parsed to valid python code that refers only to variables that are known in the scope of a specific function within the fit procedure. Thus certain variables need to have a very particular name. Consequently some rules need to be followed when creating the summands:

- the decay constants τ_j are to be referred to as $k\#$, where $\#$ represents the number of the component. I.e. for τ_2 write $k2$ in the entry.
- the time (e.g. in an exponential decay) needs to be referred to as t. Thus any t in the entry string will be parsed to *time_steps*.
- an exponential function must be referred to as *exp*(). It will be parsed to *np.exp*(), i.e. *numpy.exp()*.
- Use of any other brackets than "()" will likely result in an error when evaluating the parsed code.

Depending on which components are checked in ①, the window to define the fit function will contain more summand labels and entries.
The parsing of the entry strings can be checked in the window via the *check parsing from entries*-button. The summands entered in fig. 0.11 (the entries are deactivated while checking the parsing) evaluate to the generic fit function using the 0-*th* and 1*st* component.
As written on the window, the parsed strings will be evaluated via the *asteval* (Newville 2021) python module. This is safer than using the python *eval* function, though care should still be taken to avoid any mishaps.
As with the initial fit parameter values, the parsed fit function is saved to a file (`configFiles/target_model_summands.txt`). The summand labels *summand_comp#* are the keys and the entry strings are the

values of a python dictionary.
In order to employ the user defined fit function, the checkbox next to the *Define fit function*-button must be checked. This can only be done once a fit function has been defined.

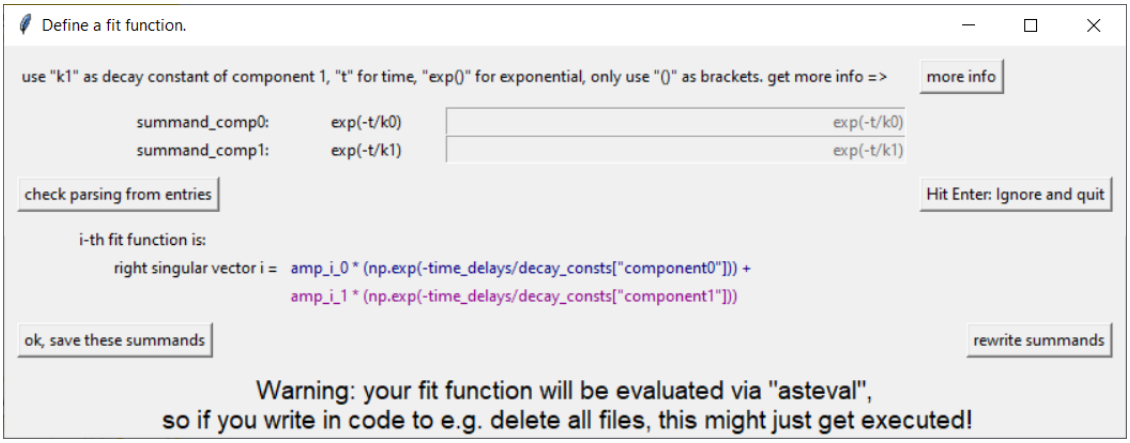


Figure 0.11 – Window to define a fit function.

0.6.4. Inspecting a matrix row-by-row and column-by-column

As already mentioned above, the GUI makes it possible to investigate the Ψ, $\Delta\Psi_{SVD}$ and $\Delta\Psi_{fit}$ matrices row-by-row and column-by-column in a separate window. This window can be opened via the *inspect matrix*-button in either ④, ⑦ and ⑨. The opened window to investigate Ψ depicted in 0.3 can be seen in fig. 0.12.
As established before, the data matrices resulting from time-resolved-spectroscopy are in the dimensions wavelength and time, i.e. $\Psi = \Psi(\lambda, t)$. Thus inspecting the matrix row-by-row actually means looking at the intensities measured at every timestep for a certain wavelength. In fig. 0.12 this can be done in the plot on the left. Using the slider below, the wavelength can be changed and the plot is redrawn. Correspondingly, inspection column-by-column means looking at the intensities measured at every wavelength for a certain timestep. Using the slider on the right, the timestep can be changed and the plot is redrawn.

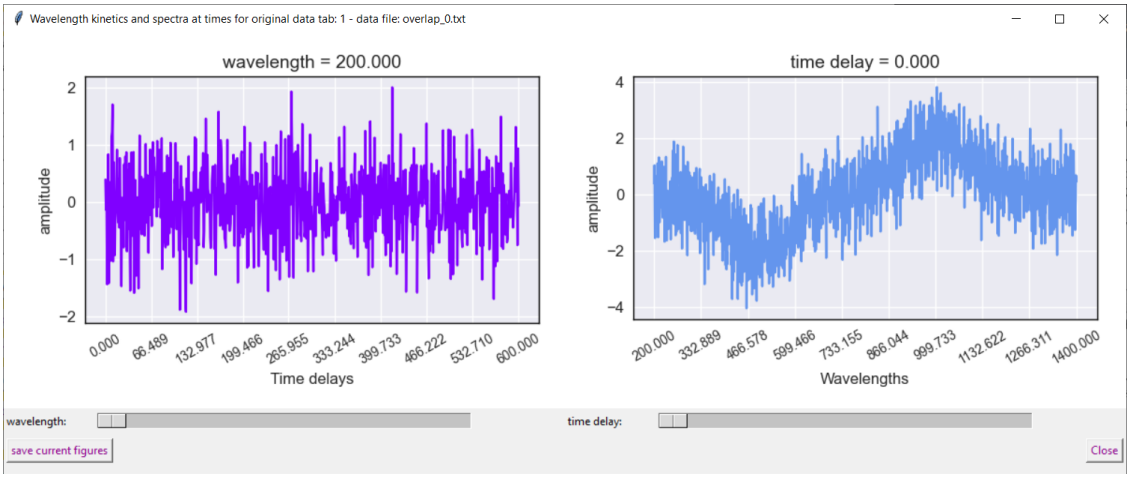


Figure 0.12 – Window to investigate a matrix row-by-row or column-by-column, i.e. wavelength-by-wavelength (on the left) and timestep-by-timestep (on the right).

0.6.5. Redrawing Ψ_{fit} with a selection of DAS only

As mentioned before, for every component used in the fit, a DAS_j is summarised from the amplitudes a_{ij} and the left singular vectors U_i (equation (**??**)). To the DAS_j corresponds the fitted decay constant τ_j. Using these, the Ψ_{fit} is computed as a sum of exponential decays (equation (**??**)).

By default the matrix Ψ_{fit} is computed as a sum over all DAS. However, in some cases it might be useful to see how Ψ_{fit} looks with only a selection of the DAS_j used in equation (**??**). This is possible via the checkboxes and *update with:*-button in ⑧. There the user can make a selection of DAS to use in equation (**??**) via the checkboxes. Then the *update with:*-button opens a dialog window similar to the one in fig. 0.13.

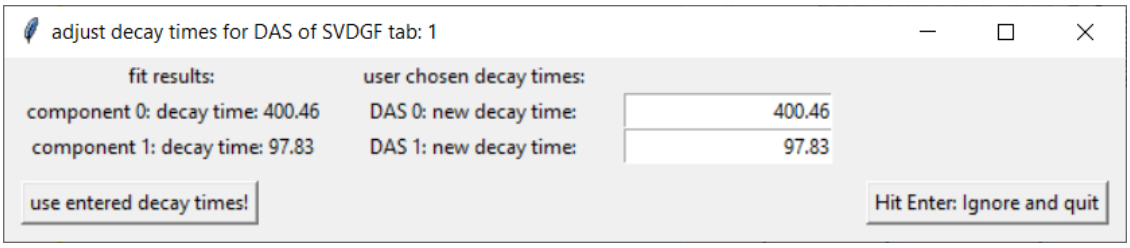
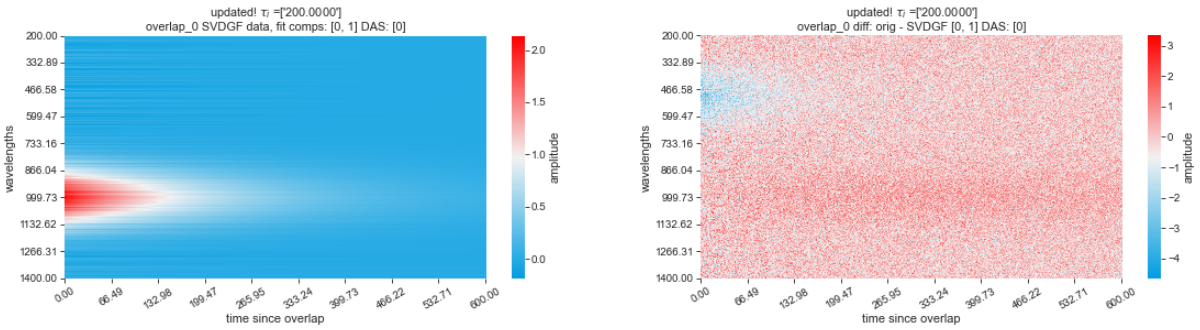


Figure 0.13 – Window to set decay constants τ_j to use in computation of Ψ_{fit}.

The dialog window displays the fitted decay constants τ_j and allows the user to enter new values for them. If the user wants to use the fitted τ_j values for the construction of Ψ_{fit}, pressing the Return key or the *Hit Enter: Ignore and quit*-button will close the window and redraw the heat map in fig. 0.7 with the Ψ_{fit} using the defined DAS selection with their original decay constants. Clicking the *use entered decay times!*-button redraws the Ψ_{fit} heat map using the defined DAS selection and lets the DAS decay with the user entered decay constants.

For convenience sake, whenever the Ψ_{fit} heat map is redrawn, so too is the heat map depicting $\Delta\Psi_{fit}$. The title string of these heat maps is rewritten to express the fact that these heat maps/data matrices have been tampered with, i.e. to express that the depicted data matrices are not direct fit results. This can be seen in fig. 0.14.



(i) Ψ_{fit} **(ii)** $\Delta\Psi_{fit}$

Figure 0.14 – The Ψ_{fit} and the $\Delta\Psi_{fit}$ with only the $0th$ DAS applied and with τ_0 changed from ≈ 400 to ≈ 200. The original figures can be seen in fig. 0.7

0.6.6. Changing the color-scheme for the heat maps

Via the *set colormaps for heatmaps*-button in ⓐ the window in fig. 0.15 is opened. In this window a colormap can be chosen by clicking on its name in the listbox. The selected colormap then gets displayed in the window. If the user likes a colormap, they can decide to which heat maps it should be applied to via

the checkboxes labelled *ORIG, SVD, Fit, SVD_diff and Fit_diff* and then clicking the *apply*-button. From then on, any newly created heatmap (whose checkbox was ticked) will be drawn with that colormap.
Below the checkboxes the current selection is displayed. Again, the current selection is saved to file (`configFiles/colormaps_for_heatmaps.txt`) and applied when restarting the program.

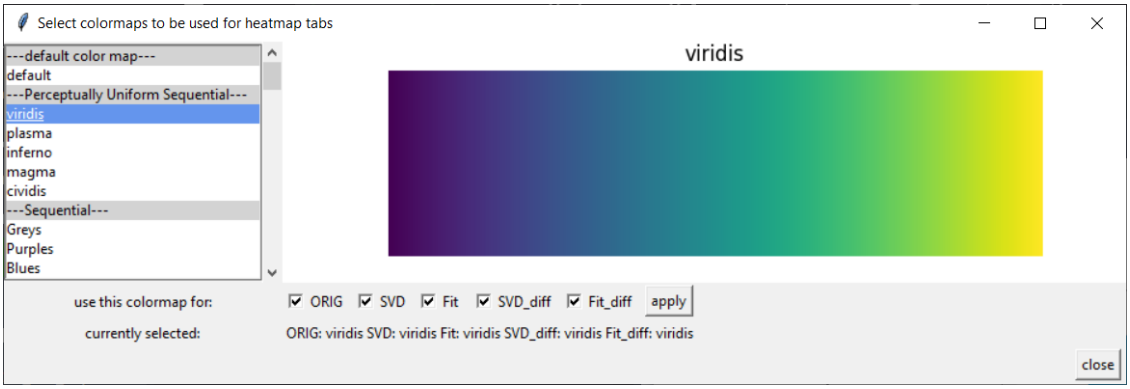


Figure 0.15 – Window to select the colormaps to be used for the heat maps of the data matrices.

0.7. Index: When will what data be saved to file

If some terminology is unclear, see other sections. Except for the configuration files, all saved data will be organised into sub-directories in `DataFiles/ResultData/...` by data file name, date, type (e.g. does it belong to original data heatmap tab or to SVD data heatmap tab etc.), selected components (where that applies) and then by the used start-time value (i.e. the minimum decay time of the complete or downsized data matrix), tab index, hour and minute of the creation of the data/plots etc.

0.7.1. Configuration files

The configuration files defining the colormaps to be used, the initial fit parameter values and the parsed target model summands will be stored to file whenever they are updated as described above. The folder is `configFiles/...` (See above.)

0.7.2. Original data heatmap tab

Clicking the *save data*-button in ④ will store

- a logfile (with date and time, data filename and the applied start-time value as well as the matrix bounds (if empty {} are indicated for the matrix bounds this means that the complete data matrix was used)),
- the data matrix and the corresponding heat map plot,
- the time delays (starting from applied start-time value) and wavelengths

to file.
When opening the SVD inspection window (*inspect SVD*-button in ④) and therein clicking the *save current figures*-button (⑤), the currently displayed singular values and vectors plots will be saved to file.
When opening the inspect matrix window (*inspect matrix*-button in ④) and therein clicking the *save current figures*-button, the currently displayed time trace/spectrum plots will be saved to file.

0.7.3. SVD components heatmap tabs
Clicking the *save data*-button in ⑥ will store

- a logfile (with date and time, data filename, the applied start-time value, the matrix bounds (if empty {} are indicated for the matrix bounds this means that the complete data matrix was used)) and the selected components),
- the Ψ_{SVD} and the $\Delta\Psi_{SVD}$ matrices with corresponding heat map plots,
- the data matrix,
- the time delays (starting from applied start-time value) and wavelengths,
- the full U and VT matrices,
- all the singular values,
- as well as three files with only the retained singular values and left and right singular vectors corresponding to the selected components

to file.
When opening the inspect matrix window (*inspect matrix*-button in ⑦) and therein clicking the *save current figures*-button, the currently displayed time trace/spectrum plots will be saved to file.

0.7.4. Fit heatmap tabs
Clicking the *save data*-button in ⑧ will store

- a logfile (with date and time, data filename the applied start-time value, the matrix bounds (if empty {} are indicated for the matrix bounds this means that the complete data matrix was used)), the selected components and a boolean to know whether or not a user defined target model was applied),
- the Ψ_{fit} and the $\Delta\Psi_{fit}$ matrices with corresponding heat map plots,
- a complete fit report as given by the *lmfit* package (also details the used initial fit parameter values),
- the data matrix,
- the time delays (starting from applied start-time value) and wavelengths,
- the retained singular values and left and right singular vectors,
- the computed DAS,
- if a user entered target model was used: the parsed summands,
- the $\Delta\Psi_{fit}$ as computed with the selected DAS (differs from $\Delta\Psi_{fit}$ if the plots were updated (via *update with*-button in ⑧)

to file.
When opening the inspect matrix window (*inspect matrix*-button in ⑨) and therein clicking the *save current figures*-button, the currently displayed time trace/spectrum plots will be saved to file.
When opening the window to inspect the fit vs right singular vectors (*rightSVs vs fit*-button in ⑨) and therein clicking the *save current figure*-button, the currently displayed figure will be saved to file.

0.7.5. Set initial fit parameters window
In this window one can configure the initial fit parameter values which will be stored separately, see above. But one can also inspect the fit functions as evaluated using the initial values as parameter values. (*show/update rSVs as reconstructed by initial values*-button) When saving the plot in the pop-up window,

- this plot,
- the entered initial fit parameter values,
- the right singular vectors of the selected components and their corresponding singular values

will be stored to file.
As it is possible to quickly update this plot, the final path will include the seconds of when the plot was saved.