The model framework analysis

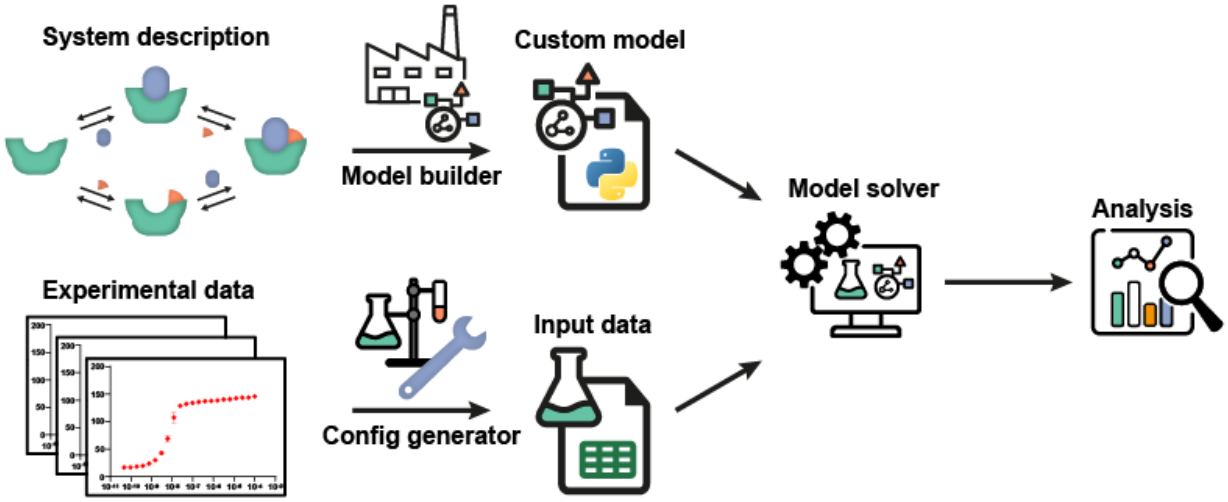
Geertjens, Nick H.J.

Mar 17, 2022

CONTENTS

Model-framework is a framework for chemical equilibrium models that utilizes a general derivation method capable of generating custom models for complex molecular systems, based on the simple, reversible reactions describing these systems. Some case studies illustrating the framework are described in the prepublished paper with details on the procedure in the accompanying Supplementary Material. The *framework* is also available from Github.

This documentation serves as overview and reference to the available analysis functions included in the base distribution of the *framework*.



CHAPTER

ONE

CONTENTS

1.1 Analysis methods overview

All analysis functions distributed with the base version of the *framework* can be found below.

1.1.1 Plot system data

src.post_process.**plot_model**(*system*, *normalise=False*, *model_plot_min=None*, *model_plot_max=None*, ***kwargs*)

Itterates over the conditions associated with the provided system and plots system.data and model solution (as a line) for each. If no solution is determined, the current parameter values are tried instead.

Parameters

- **system** (*System object*) – System objects to plot the data and model prediction for.
- **normalise** (*boolean*) – Wheter to normalise all datapoint between 0-1.
- **model_plot_min** (*float*) – The minimum titrate concentration to plot
- **model_plot_max** (*float*) – The maximum titrate concentration to plot

Warning: Modifies system.state

Returns

- *Matplotlib plot* – The plotted data
- *pandas.dataframe* – Dataframe containing the raw numbers plotted.

1.1.2 Plotting concentrations graph

src.post_process.**plot_concentrations**(*system*, ***, *plot_species='all'*, *concentrations_start=None*, *concentrations_stop=None*, ***kwargs*)

Plot the concentrations of species in the system for a single condition. The free titrate concentration is never plotted but is added to the output file.

It is possible to select which species with the plot_species argument: -'all': plots all species except for free titrate. -'components': plot all species marked in the model as 'independent', the components of the system. -'complexes': plot all species marked in the model as 'dependent', the complexes formed from the components. -['R', 'S', ...]: a list of species to plot. Species need to be defined in the model and lookup is case-sensitive.

Parameters

- **system** (*System object*) – The system to analyse
- **plot_species** (*{'all', 'components', 'complexes'} or list of strings, optional*) – Which species to plot, see above for options. The default is 'all'.
- **concentrations_start** (*float, optional*) – The lowest titrate concentration to plot. The default is equal to the lowest concentration in the system.data.
- **concentrations_stop** (*float, optional*) – The highest titrate concentration to plot. The default is equal to the highest concentration in the system.data.

Raises **ValueError** – If the system has more than one condition associated.

Warning: Modifies system.state

Returns

- *matplotlib plot* – line plot showing the concentrations of the selected species as a function of the total titrate concentration present.
- *pandas.dataframe* – Dataframe containing the numbers plotted.

1.1.3 Alternative range of initial guess values solver

src.post_process.**range_solver**(*system*, ***, *range_fit_parameters=None*, *range_n=10*, *log_scale=True*, ***kwargs*)

Solve the system from a range of initial guess values, using a latin hypercube approach to determine combinations. Displays all determined values sorted by the squared error.

All parameters not included in range_fit_parameters are assumed to be fixed at their current system values.

Parameters

- **system** (*System object*) – The system to analyse
- **range_fit_parameters** (*Dictionary of string:tuple pairs*) – Each key represents a fit_parameter, the tuple should contain the min and max expected values of that parameter.
- **range_n** (*int*) – The number of strata to create within each range. The default is 10.
- **log_scale** (*Boolean*) – Whether to use a log based scale instead of a linear scale to create the strata. The default is True.

Warning: Modifies system parameters for the keys in range_fit_parameters Modifies system.solution

Returns **result** – All the determined parameter values with associated squared error.

Return type dataframe

1.1.4 Mean squared error landscape

src.post_process.**landscape**(*system*, *, *landscape_parameters=None*, ***kwargs*)

Landscape post process selection tool. Will call the appropriate landscape function based on the number of landscape_parameters. If no landscape_parameters are given, the landscape function will try to use the fit_parameters instead. If more than two parameters are given only the first two will be plotted.

Parameters

- **system** (*System object*) – The system to analyse
- **landscape_parameters** (*List of Strings*) – The parameters that should be varied in the landscape. If none are given, instead the fit_parameters will be used.
- ****kwargs** (*Optional parameters*) – See specific landscape functions for details on the optional parameters that can be suplied.

Raises **ValueError** – If len(landscape_parameters) < 1 or if no landscape_parameters are given and there are no fit_parameters

Warning: Modifies System.condition.state objects concentration

Returns

- *Matplotlib plot* – Appropriate landscape plot for the number of parameters.
- *Pandas dataframe* – Contains the raw values used in the plot

1D landscape

src.post_process.**landscape_1d**(*system*, *, *landscape_parameters*, *landscape_1_range=None*, ***kwargs*)

Plot curve of MSE values based on variation in a single fit parameter.

Parameters

- **system** (*System object*) – The system to plot parameter fits for.
- **landscape_parameters** (*List of strings*) – Note that a list is expected here. This is the parameter that will be plotted. If more than one is given, only the first one is used.
- **landscape_1_range** (*1darray like, optional*) – Range of values to fit for the parameter. Use of np.geomspace is recommended. If none are given the landscape parameter current value times 0.01 and 100 are used as min and max value respectively.

Returns

- *Matplotlib plot* – Line plot of the MSE values for the range of parameter values.
- *pandas.dataframe* – Dataframe containing the numbers plotted.

2D landscape

src.post_process.**landscape_2d**(*system*, *, *landscape_parameters*, *landscape_1_range=None*, *landscape_2_range=None*, ***kwargs*)

Creates a landschap of MSE values based on 2 fit parameters.

Parameters

- **system** (*System object*) – The system to plot parameter fits for.
- **landscape_parameters** (*List of strings*) – The parameters that will be changed. If more than two are given, the others are ignored.
- **landscape_1_range** (*1darray like, optional*) – Range of values to fit for the first parameter. Use of np.geomspace is recommended. If none are given the landscape parameter current value times 0.01 and 100 are used as min and max value respectively.
- **landscape_2_range** (*1darray like, optional*) – Range of values to fit for the second parameter. Use of np.geomspace is recommended. If none are given the landscape parameter current value times 0.01 and 100 are used as min and max value respectively.

Returns

- *Matplotlib plot* – 2D plot showing the MSE values for the range of parameter values as contour plot.
- *pandas.dataframe* – Dataframe containing the raw numbers plotted.

Raises **ValueError** – if len(landscape_parameters) < 2

1.1.5 Parameter confidence intervals

src.post_process.**confidence_interval**(*system*, *, *confidence_method='bootstrap'*, *confidence_repeats=10*, *bias_acceleration=True*, ***kwargs*)

Determines a confidence interval for the parameters based on the given approach. Currently only the bootstrap method is supported.

The bootstrap method is based on random draw, so a large number of repeats is recommended to get accurate results.

Parameters

- **system** (*System object*) – The system to perform the analysis on.
- **confidence_method** (*string, optional*) – The method to use to determine the interval. The default is 'bootstrap'.
- **confidence_repeats** (*integer, optional*) – The number of bootstrap sets to generate. A 1000 samples will result in reasonable predictions in most cases. The default is 10 to give an indication of the total duration required.

Warning: Modifies system.state

Returns **result** – Dataframe containing the determined point estimates for all generated samples.

Return type Pandas dataframe

1.1.6 Parameter sensitivity analysis

src.post_process.**parameter_sensitivity**(*system*, *, *sensitivity_perturbation=0.5*, *sensitivity_state='current'*, *sensitivity_parameters=None*, *sensitivity_m_function=<function _squared_error_m_function>*, ***kwargs*)

Performs local parameter sensitivity analysis on the system.

The parameters that are tested should be system constants. This means that the parameter has the same value across all associated conditions.

This method is based on the description in van Riel, BRIEFINGS IN BIOINFORMATICS Vol 7 (2006). The M(y) value is the model output value to track upon change in parameter value y. By default M is defined as the squared error of the model compared to the measurement data.

Parameters

- **system** (*System object*) – The system to analyse
- **sensitivity_perturbation** (*float, optional*) – The change in parameter value as a fraction increase. Negative values will result in a decrease. The default is 0.5, a 50% increase.
- **sensitivity_state** (*{'solution', 'current'}, optional*) – Whether to use the current parameter values or the values determined during system.solve. The default is 'solution', the solution values.
- **sensitivity_parameters** (*list of strings*) – List of parameters to test for sensitivity. By default the fit_parameters are used as sensitivity_parameters
- **sensitivity_m_function** (*python function, optional*) – The function to use to determine the M value, see above. By default the system.error_function is used. The function should take only a single argument 'system'.

Warning: Modifies system.state

Raises **ValueError** – If any of the sensitivity_parameters is not a system constant, see above.

Returns

- *Matplot plot* – Barplot showing the sensitivities of the different parameters.
- *pandas Dataframe* – The absolute values used to plot the data.

1.1.7 Plot residuals

src.post_process.**residuals**(*system*, ***kwargs*)

Gets the residuals from system and plots them against the titrate concentrations.

Parameters **system** (*System object*) – The system to plot the residuals for.

Returns

- *Matplotlib plot* – Scatterplot displaying the residuals between model prediction and measured values.
- *pandas.dataframe* – Dataframe containing the numbers plotted.

1.1.8 Automated input outlier detection

src.post_process.**outlier**(*system*, *, *write_output=True*, *skip_warnings=False*, ***kwargs*)

Uses a doornbos procedure to detect outliers in data associated with system. Will only detect suspects, removal needs to be performed manually after conformation. See src.analysis.outliers for details on the method.

Parameters

- **system** (*system object*) – The system to analyse for outliers.
- **write_output** (*Boolean, optional*) – Optional flag for directing the output to a file instead of system.out. The default is False.
- **skip_warnings** (*Boolean, optional*) – Optional flag to not print warnings. The default is false.

Returns Prints any outliers found to the system.out or to file depending on parameters.

Return type None

INDEX