User Documentation

ChroMo

Contents

[Input Data 2](#_Toc110362035)

[Excel File 2](#_Toc110362036)

[Excel Settings Requirements 2](#_Toc110362037)

[Program Execution 2](#_Toc110362038)

[Results 3](#_Toc110362039)

# Input Data

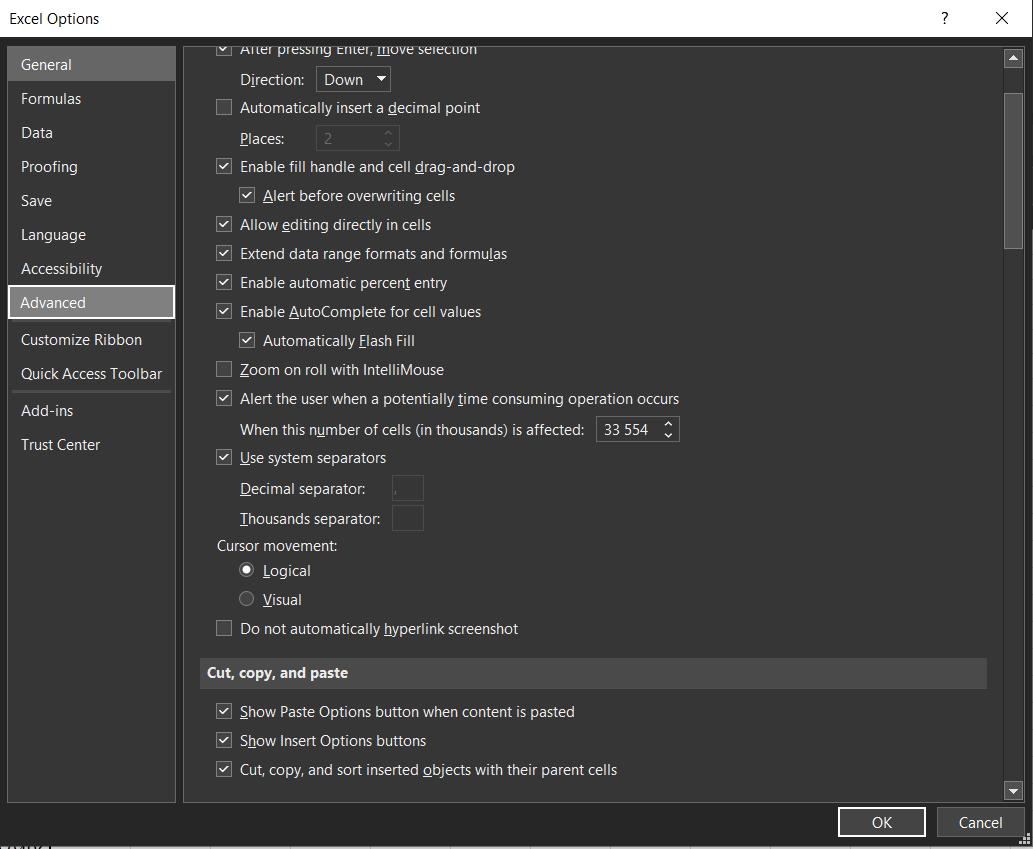
Input data needs to be in one folder which contains any amount of .xlsx files.

## Excel File

MS Excel files (.xlsx) needs to be in the form of template contained in the program folder.

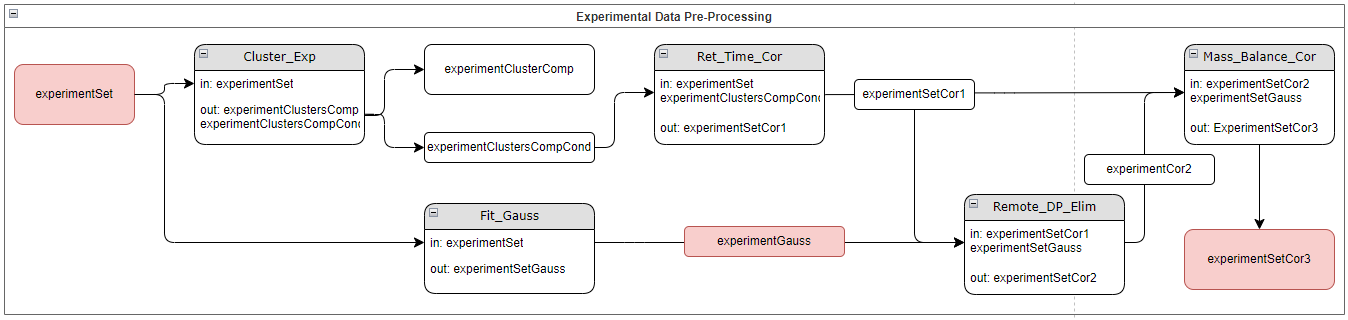
### Excel Settings Requirements

Thousand separator needs to be set as empty input. Otherwise, it can cause problems with values reading.



# Program Execution

## Data pre-processing

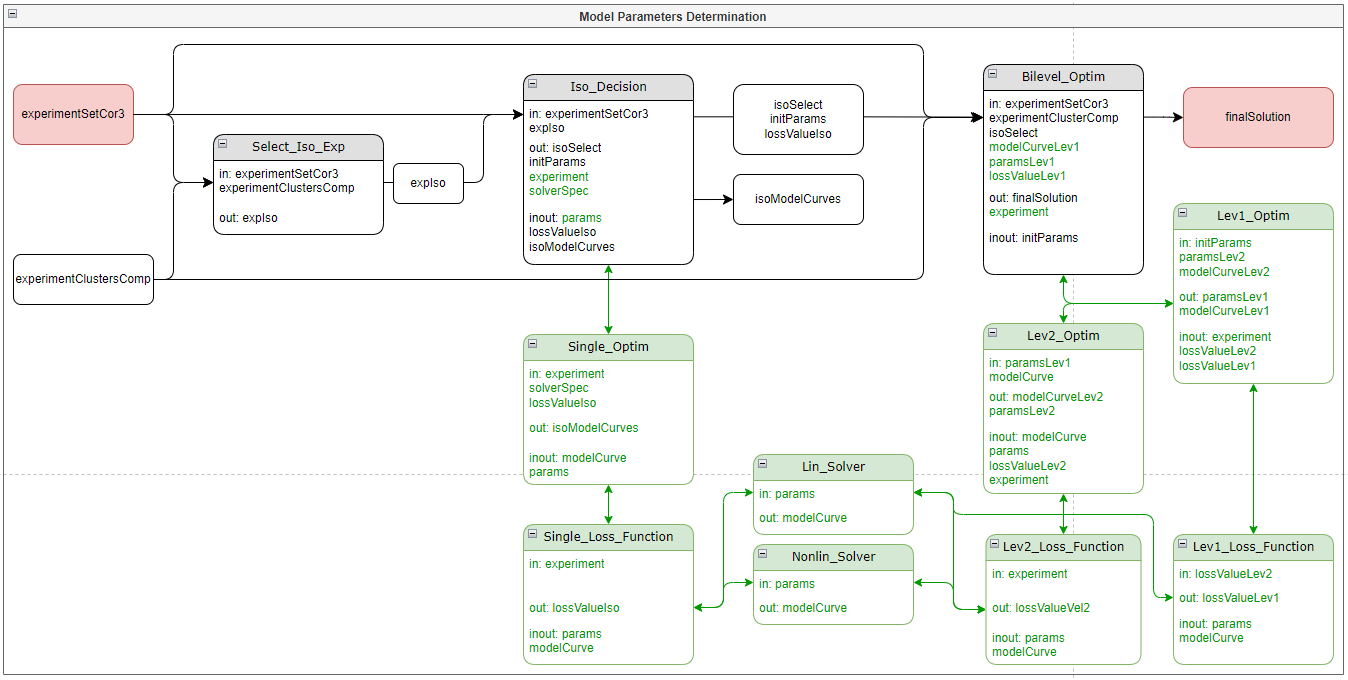


runtime question to user: Use fitted Gauss curve directly as a corrected datapoints? Y/N

if Y, then we skip Remote\_DP\_Elim.py and use the output of Fit\_Gauss.py directly as an input to Mass\_Balance\_Cor.py

if N, then we stick to our original plan (use Gaussian just for elimination of remote points base on threshold)

## Model fitting



# Results

# Functions and Objects Documentation

## Objects

### ExperimentSet

*class objects.ExperimentSet*

#### Description

Class representing group of experiments

#### Attributes

*experiments(list[Experiment])* - List of Experiment objects

*metadata(Metadata)* - Information about experiment set

### Experiment

*class objects.Experiment*

#### Description

Class representing experiment

#### Attributes

*experimentCondition(ExperimentCondition)* - Conditions of the experiment

*experimentComponents(list[ExperimentComponent])* - Components of the experiment

*metadata(Metadata)* - Information about experiment

### ExperimentCondition

*class objects.ExperimentCondition*

#### Description

Class representing conditions of an experiment

#### Attributes

*columnDiameter(float)* - Diameter of column used in experiment

*columnLength(float)* - Length of column used in experiment

*feedVolume(float)* - Volume of feed in experiment

*feedTime(float)* - Time of feed in experiment

*flowRate(float)* - low rate in experiment

### ExperimentComponent

*class objects.ExperimentComponent*

#### Description

Class representing component in an experiment

#### Attributes

*concentrationTime(pandas.DataFrame)* - Table of times and concentrations measured in experiment

*feedConcentration(float)* - Concentration of component in feed

*name(str)* - Name of the component

*experiment(Experiment)* - Reference to experiment with this component

### ExperimentClusters

*class objects.ExperimentClusters*

#### Description

Class representing clusters of components from multiple experiments

#### Attributes

*clusters(dict[str:list[ExperimentComponents]])* - Dictionary of component clusters

*metadata(Metadata)* - Information about clusters

### Metadata

*class objects.Metadata*

#### Description

Class representing information about other objects

#### Attributes

*date(str)* - Relevant date

*description(str)* - Description of the object

*path(str)* - Path to the source experiment(s)

### Operator

*class objects.Operator*

#### Description

Class managing the workflow of the program and comunication with the user

#### Methods

##### Start

*method objects.Operator.Start()*

###### Description

Starts and manages the workflow

##### Setting\_Parameters

*method objects.Operator.Setting\_Parameters()*

###### Description

Asks user to input relevant parameters

###### Returns

List of parameters (*list[float]*)

##### Load\_Experiment\_Set

*method objects.Operator.Load\_Experiment\_Set(path)*

###### Description

Creates an ExperimentSet object from excel files

###### Parameters

*path(str)* - Path to the files

###### Returns

ExperimentSet object (*ExperimentSet*)

##### Cluster\_By\_Component

*method objects.Operator.Cluster\_By\_Component(experimentSet)*

###### Description

Creates a ExperimentClusters object based on component

###### Parameters

*experimentSet(ExperimentSet)* - Experiment set to cluster

###### Returns

ExperimentClusters object (*ExperimentClusters*)

##### Cluster\_By\_Condition

*method objects.Operator.Cluster\_By\_Condition(experimentSet)*

###### Description

Creates a ExperimentClusters object based on component and condition of the experiment

###### Parameters

*experimentSet(ExperimentSet)* - Experiment set to cluster

###### Returns

ExperimentClusters object (*ExperimentClusters*)

##### Cluster\_By\_Condition2

*method objects.Operator.Cluster\_By\_Condition2(experimentSet)*

###### Description

Similar as Cluster\_By\_Condition, different cluster implementation

###### Parameters

*experimentSet(ExperimentSet)* - Experiment set to cluster

###### Returns

ExperimentClusters object (*ExperimentClusters*)

##### Cluster\_Match

*method objects.Operator.Cluster\_Match(comp1, comp2, tolerance = 0.05)*

###### Description

Method helping clustering methods to decide if two components are similar enough

###### Parameters

*comp1(ExperimentComponent)* - First component for comparison

*comp2(ExperimentComponent)* - Second component for comparison

*tolerance(float)* - Relative tolerance for similarity decision

###### Returns

Whether the two components are similar(*bool*)

##### Create\_Key

*method objects.Operator.Create\_Key(comp)*

###### Description

Method helping clustering methods to create key for dictionary

###### Parameters

*comp(ExperimentComponent)* - Component from which the key is created from

###### Returns

Key for clusters dictionary (*str*)

## Preprocessing functions

### Ret\_Time\_Cor

*function functions.Ret\_Time\_Cor(experimentSet, experimentClustersCompCond)*

#### *Description*

Shifts time of peaks to match for each cluster.

#### *Parameters*

*experimentSet(ExperimentSet)* – Experiment set for correction.

*experimentClustersCompCond(ExperimentClusters)* – Clusters of components to match peaks.

#### *Returns*

Experiment set with matched peaks (*ExperimentSet*)

### Fit\_Gauss

*function functions.Fit\_Gauss(experimentSet)*

#### *Descriptions*

Fits a asymmetric gaussian curve to each component.

#### Parameters

*experimentSet(ExperimentSet)* – Experiment set for gaussian cruve fitting.

#### Returns

Experiment set with fitted gaussian curve in each component (*ExperimentSet*)

### Remote\_DP\_Elim

*function functions.Remote\_DP\_Elim(experimentSetCor1, experimentSetGauss, absTolerance = 0.1)*

#### Description

Compares *experimentSetCor1* to *experimentSetGauss* with fitted gaussian curve and removes points from the *experimentSetCor1* that differ too much based on *absTolerance.*

#### Parameters

*experimentSetCor1(ExperimentSet)* – Experiment set for correction.

*experimentSetGauss(ExperimentSet)* – Experiment set with gauss curves for comparison.

*absTolerance(float)* – Absolute tolerance.

#### Returns

New experiment set with removed points that differ too much (*ExperimentSet*)

### Mass\_Balance\_Cor

*function functions.Mass\_Balance\_Cor(experimentSetCor2, experimentSetGauss)*

#### Description

Compares output mass of *experimentSetGauss* and feed mass of *experimentSetCor2* and changes feed time of *experimentSetCor2* to minimize the difference.

#### Parameters

*experimentSetCor2(ExperimentSet)* – Experiment set for correction.

*experimentSetGauss(ExperimentSet)* – Experiment set with gauss curves for comparison.

#### Returns

New experiment set with corrected feed time (*ExperimentSet*)

## Parameters Determination functions

### Select\_Iso\_Exp

*function functions.Select\_Iso\_Exp(experimentSetCor3, experimentClustersComp)*

#### Description

For each cluster selects the component with the highest concentration in peak.

#### Parameters

*experimentSetCor2(ExperimentSet)* – Experiment set from which cluster was created.

*experimentClustersComp(ExperimentClusters)* – Experiment clusters based on component.

#### Returns

New Experiment clusters with single component with the highest concentration (*ExperimentClusters*)

### Iso\_Decision

*function functions.Iso\_Decision(expIso, params, lossFunc = 'Simple')*

#### Description

For each components selects solver which has the best loss function value for given component.

#### Parameters

*expIso(ExperimentClusters)* – Experiment clusters with single component, output of Select\_Iso\_Exp.

*params()* – Parameters for loss function.

*lossFunc(str)* – Loss function selector.

#### Returns

Dictionary with keys being names of components and values being chosen solvers (*dict(str:str)*)

### Bilevel\_Optim

*function functions.Bilevel\_Optim(experimentSetCor3, experimentClustersComp)*

#### Description

Starts the bilevel optimalization process. Uses global variables for loss function parameters.

#### Parameters

*experimentSetCor3(ExperimentSet)* – Experiment set to optimize

*experimentClustersComp(ExperimentClusters)* – Experiment clusters based on component.

#### Returns

Solution with the best parameters (*Solution*)

### Lev1\_Optim

*function functions.Lev1\_Optim(experimentClustersComp)*

#### Description

Level 1 optimalization. Uses global variables for loss function parameters.

#### Parameters

*experimentClustersComp(ExperimentClusters)* – Experiment clusters to optimize

#### Returns

Value of the Lev1\_Loss\_Function after optimalization (*float*)

### Lev1\_Loss\_Function

*function functions.Lev1\_Loss\_Function(porosity, experimentClustersComp)*

#### Description

Loss function for Level 1 optimalization.

#### Parameters

*porosity(float)* – Porosity to optimize.

*experimentClustersComp* – Clusters for which we optimize.

#### Returns

Sum of Level 2 optimalizations for given porosity (*float*)

### Lev2\_Optim

*function functions.Lev2\_Optim(porosity, experimentCluster, key)*Description

Level 2 optimalization. Uses global variables for loss function parameters.

#### Parameters

*porosity(float)* – Porosity parameter.

*experimentCluster(list(ExperimentComponent))* – List of components for which we optimize.

*key(str)* – Parameter selector for given components.

#### Returns

Value of the Lev2\_Loss\_Function after optimalization (*float*)

### Lev2\_Loss\_Function

*function functions.Lev2\_Loss\_Function(params, experimentCluster, porosity, lossFunction = 'Simple')*

#### Description

Loss function for Level 2 optimalization.

#### Parameters

*params()* – Parameters for loss function.

*experimentCluster(list(ExperimentComponent))* – List of components for which we optimize.

*lossFunction(str)* – Loss function selector.

#### Returns

Sum of results of single loss functions for given parameters (*float*)

### Single\_Loss\_Function\_Choice

*function functions.singleLossFunctions*.*Single\_Loss\_Function\_Choice(choice, params, experimentComp)*

#### Description

Function with option to choose which loss function to use.

#### Parameters

*choice(str)* – Loss function selector.

*params()* – Parameters for loss function.

*experimentComp(ExperimentComponent)* – Component for loss function calculation.

#### Loss Function Options

*‘Simple’* - *Single\_Loss\_Function\_Simple* – Calculates error as an absolute value of difference

*‘Squares’* - *Single\_Loss\_Function\_Squares* – Calculates error as a square value of difference

*‘LogSimple’* - *Single\_Loss\_Function\_LogSimple* – Calculates error as a natural logarithm of absolute value of difference

*‘LogSquares’* - *Single\_Loss\_Function\_LogSquares* – Calculates error as a natural logarithm of square value of difference

#### Raises

If choice parameter doesn’t match any option, function will raise an Exception.

#### Returns

Result of chosen single loss function (*float*)

### Solver\_Choice

*function functions.solvers.* *Solver\_Choice(choice, params, experimentComp)*

#### Description

Function with option to choose a solver

#### Parameters

*choice(str)* – Solver selector.

*params()* – Parameters for solver.

*experimentComp(ExperimentComponent)* – Component for solver calculation.

#### Solver options

*‘Lin’* - *Lin\_Solver* – Linear solver

#### Raises

If choice parameter doesn’t match any option, function will raise an Exception.

#### Returns

Result of chosen solver (*numpy.array*)

## Analysis and Debugging functions

### Compare\_ExperimentSets

*function functions.Compare\_ExperimentSets(experimentSet1, experimentSet2)*

#### Description

Compares two experiment sets and allows user to find differences.

#### Usage

The function goes in order through experiments and their components and compares their measured concentrations. When the function finds a difference, it will print which two components are not matching and whether user wants to show the difference. If user inputs yes, the function will print side to side timeseries and two graphs of both timeseries, in which user can find the difference.

#### Parameters

*experimentSet1(ExperimentSet)* - First experiment set to compare.

*experimentSet2(ExperimentSet)* - Second experiment set to compare.

### Loss\_Function\_Analysis

*function functions.Loss\_Function\_Analysis(experimentClusterComp, component='Sac', xstart = 0, ystart = 0, xend = 5002, yend = 5002, xstep = 50, ystep = 50, porosityStart = 0.2, porosotyEnd = 1, porosityStep = 0.1)*

#### Description

Allows user to analyze results of Lev2\_Loss\_Function with specified input values.

#### Usage

After the initial call, the function will calculate and show graph of loss function value for Henry constant and Dispersion coefficient values given by parameters (xstart, ystart, xend…). After that, the function will ask user if he wants to print closeup (Print closeup?[Y - yes, N - no, E - exit]). If user inputs yes, the function will then ask for new parameters for Henry constant and Dispersion coefficient interval. The function will then calculate a new graph with the new parameters and asks for closeup again. This can be repeated indefinitely. Once user inputs no, the function will move on to the next porosity value with the original parameters and the process is repeated for all porosity values given by porosity parameters (porosityStart, porosityEnd, porosityStep).

#### Parameters

*experimentClusterComp(ExperimentCluster)* Cluster of components to analyze.

*component(str)* - Specified name of component to analyze.

*xstart(int)* - Start of interval of Henry constant values for analysis.

*ystart(int)* - Start of interval of Dispersion coeficient values for analysis.

*xend(int)* - End of interval of Henry constant values for analysis.

*yend(int)* - End of interval of Dispersion coeficient values for analysis.

*xstep(int)* - Step of Henry constant values from interval for analysis.

*ystep(int)* - Step of Dispersion coeficient values from interval for analysis.

*porositystart(float)* - Start of interval of porosity values for analysis.

*porosityend(float)* - End of interval of porosity values for analysis.

*porositystep(float)* - Step of porosity values from interval for analysis.