

---

# **McSAS Documentation**

***Release 1.3***

**Brian R. Pauw**

**May 16, 2017**

<b>1</b>	<b>McSAS Quick Usage guide</b>	<b>1</b>
1.1	Introduction	1
1.2	-1. Starting McSAS	1
1.3	0. Loading test data	2
1.4	1. Configuring the algorithm	2
1.5	2. Configuring the Model	2
1.6	3. Configuring the Post-fit Analysis	2
1.7	4. Running the fit	3
1.8	5. The result	3
1.9	5. Getting peak parameters	3
1.10	6. Further results	4
1.11	7. What's next?	4
1.12	8. SASFit test data settings:	4
<b>2</b>	<b>The Math Behind</b>	<b>5</b>
2.1	SAXS	5
<b>3</b>	<b>Source Code Documentation</b>	<b>7</b>
3.1	mcsas.mcsas package	7
3.2	mcsas.models package	16
3.3	mcsas.gui package	24
3.4	mcsas.log package	52
3.5	mcsas.utils package	54
3.6	mcsas.datafile package	69
3.7	mcsas.dataobj package	78
3.8	mcsas.bases package	91
3.9	mcsas.main module	109
3.10	McSAS	110
3.11	mcsas.cxfreeze module	110
3.12	mcsas.autobuild module	113
3.13	mcsas.mcsas_test module	113
3.14	Indices and tables	114
<b>4</b>	<b>How to generate the documentation</b>	<b>115</b>
4.1	Requirements	115
4.2	Generate a PDF document	115
4.3	Generate HTML pages	115

4.4	Update Source Code Documentation . . . . .	115
<b>5</b>	<b>Indices and tables</b>	<b>117</b>
	<b>Bibliography</b>	<b>118</b>
	<b>Python Module Index</b>	<b>119</b>

### Introduction

This guide is intended as an aid to getting the first fits using McSAS.

For comprehensive details of what goes on under the hood, please refer as a baseline to the available publications. Additionally, the code is open source, and provides the best “documentation” of what actually takes place.

When publishing results using this code, the user is requested to cite either or both of the following works:

**Bressler, I, Pauw, B. R, and Thuenemann, A., submitted to J. Appl. Cryst., arXiv:1412.1900**

**Pauw, B. R., Pedersen, J. S., Tardif, S., Takata, M. and Iversen, B. B.,**

10. Appl. Cryst. 46 (2013), 365—371.

### Scope of the code capabilities

The McSAS code at the moment can:

1. Fit supplied data to a variety of models, with absolute unit support.
2. Graphically show the distributions of selected parameters and associated parameter ranges.
3. Graphical output includes distribution population modes with uncertainties.
4. Output the fit, data, settings, and distributions for further processing.
5. Can be used with or without user interface, using command-line arguments.

### -1. Starting McSAS

McSAS can be used on both Linux / Unix systems (including MacOS X) as well as personal computers running Windows.

**On Unix- and Unix-like computers, McSAS can be started from any terminal by typing:** \$  
/path/to/mcsas/main.py

On Windows, the compiled executable can be double-clicked to start.

## 0. Loading test data

A demonstration dataset has been simulated using SASFit, and output in a three-column file: “test-data/quickstartdemo1.csv”.

The dataset can be loaded by right-clicking in the empty list in the “Data Files”-tab. When loaded, the list shows some information of the data: its length, content, q-limits, and scatterer radius limits calculated using the Q limits (theoretically, the spacing between Q-points also dictates the size limits, but the tendency of users to use much too narrowly-spaced Q-points would result in unworkable estimates).

Lastly, the “log”-window shows how many datapoints had an uncertainty estimate below 1% of the intensity. These datapoint uncertainty values have been adjusted to 1%, as that is a practically demonstrated limit of SAS data accuracy.

## 1. Configuring the algorithm

The algorithm uses several internal parameters. Some of these can be adjusted in the “Algorithm”-tab of the user interface. They are:

1. The convergence criterion. If the uncertainty estimates provided with the data are not accurate, or if the fitting model chosen is unsuitable or incompletely descriptive, the algorithm may not arrive at a final solution (a convergence criterion of 1 or below).
2. Number of repetitions from which uncertainties are determined. Set to 10 for the quick start.
3. Number of contributions. This can be adjusted to minimise the optimization time. The average optimisation time can be found in the “Timing” line on the graphical output.
4. Background (checkbox). When set, it will add a flat background contribution to the fit

If necessary, the other internal algorithm parameters can be changed through careful editing of the “mcsasparameters.json” parameter dictionary. This should not be needed for common use.

## 2. Configuring the Model

For the quick start, the model selected should be the “Sphere” model. After selecting this model, verify that the sphere radius is “Active”, then move back to the “Data files”-tab, and double-click the “quickstartdemo1”-line to copy the sphere size estimates to the model.

## 3. Configuring the Post-fit Analysis

Select the already filled lines in the “Post-fit Analysis”-tab, and select “remove” from the right-click menu. Then add a new entry to the list by selecting “add range” from the right-click menu. In the emerging window, change the histogram X-axis scaling from “lin” to “log” and click “add”.

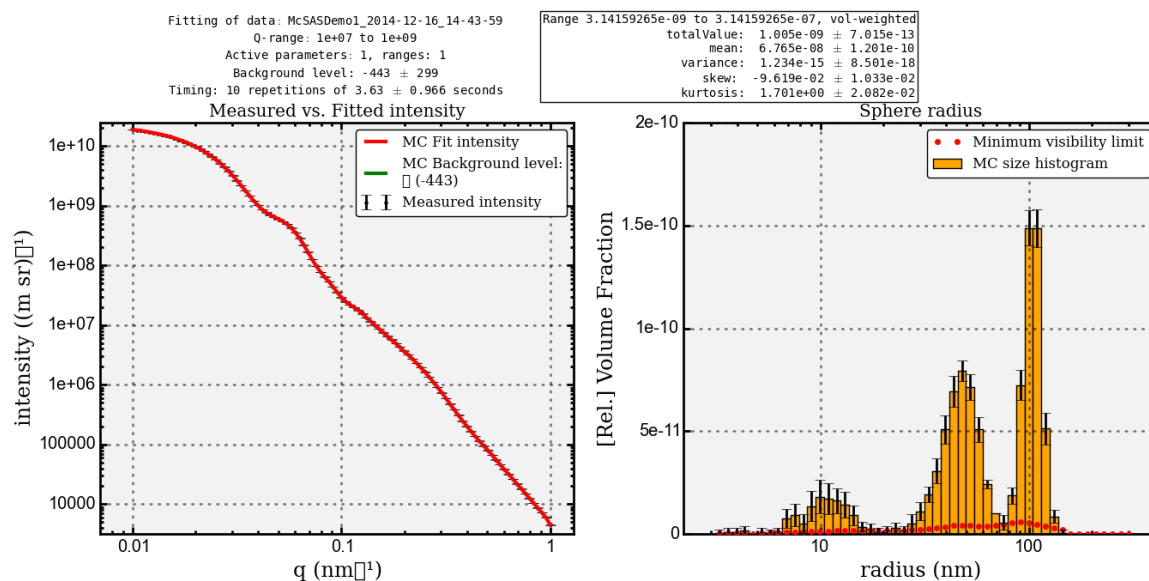
The settings are now complete.

## 4. Running the fit

Click the “start”-button. Optimisation takes 36 seconds on a 3.4 GHz intel i7 iMac (2012 model).

## 5. The result

The result window should pop up automatically and resemble the plot shown in *quickstartdemo1.pdf*



The left-hand plot shows the data in black with error bars, the MC fit in red, and a green line indicating the fitted background level (not shown as it approaches zero). The background value is furthermore indicated in the legend.

The right-hand plot shows the resulting volume-weighted size histogram, with uncertainties on the bars, and the red dashed line indicating the minimum level required for each bin to contribute a measurable amount to the scattering pattern (i.e. more than the uncertainty).

As is clear from the vertical axis on that plot, the partial volume fractions are unrealistic if the scattering contrast has not been set. One may also want to rebin the plot in fewer bins to reduce the relative uncertainties on the bins.

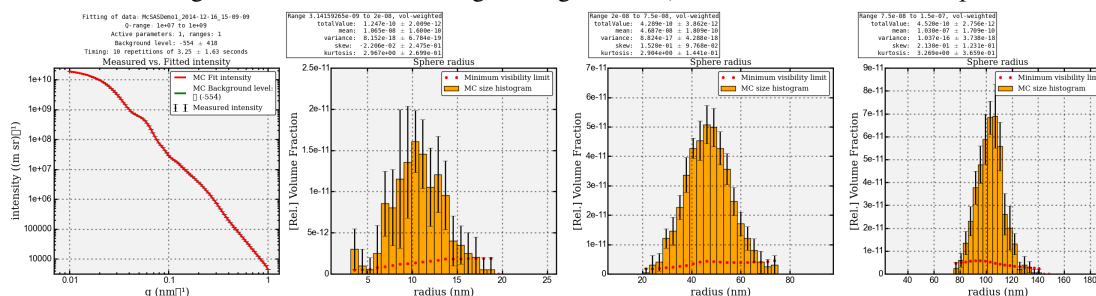
## 5. Getting peak parameters

Above the size distribution, the population statistics can be found such as the mean, the variance, skew and kurtosis (each with their own uncertainty estimates). These are valid for the entire range, and are calculated from the individual contributions (not from the histogram). If we want to find the values for the individual populations, we need to set the correct ranges in the Post-fit Analysis list:

1. 3.14 - 20, (binning e.g. 20 bins, linearly spaced, volume-weighted)
2. 20 - 75, (binning for example as above)
3. 75 - 150, (ibid.)

And then pressing “start” again. (This restarts the fitting procedure. A “Re-analyze”-option is in the works but was not finished at the time of writing)

This should give the following figure (as shown in *quickstartdemo1a.pdf*):



The mean of each population is slightly shifted upward as compared to the simulation parameters (given in paragraph 8). Note, however, that the simulation parameters define the *numeric* mean, whereas the ones shown in the histogram are the *volumetric* mean. Therefore, the latter are shifted up.

## 6. Further results

In the directory that the “quickstartdemo1.csv”-file was located, you will find several more files after a successful fit. These are the automatically saved results. They all contain a timestamp on when the fit was performed. Files with an asterisk are only present if the fit was successful. They are:

1. quickstartdemo1\_[timestamp].pdf\*: the autosaved plot
2. quickstartdemo1\_[timestamp].log.txt: the fitting log window.
3. quickstartdemo1\_[timestamp].contributions.pickle\*: A pickled list of raw contributions.
4. quickstartdemo1\_[timestamp].hist[parameter].csv\*: The histogram information of every range.
5. quickstartdemo1\_[timestamp].stats[parameter].csv\*: The statistics information of every range
6. quickstartdemo1\_[timestamp].settings.cfg: The settings used in the fit.

## 7. What's next?

If you have the ability and interest in improving the code, please consider joining the development effort, which will work on including more shapes and adding slit-smearing options.

If you have more questions that are not answered in either 1) the paper, 2) the code, and 3) this document, feel free to send me an e-mail which you can find on the papers or <http://mcsas.net/>.

Good luck!

## 8. SASFit test data settings:

SASfit version 0.94.5, precompiled Mac OS X version. quickstartdemo1.csv:

1. Spheres, gaussian dist,  $N = 0.00105$ ,  $S = 10$ ,  $X_0 = 100$ ,  $\eta = 1$ .
2. Spheres, gaussian dist,  $N = 0.0135$ ,  $S = 10$ ,  $X_0 = 40$ ,  $\eta = 1$ .
3. Spheres, gaussian dist,  $N = 0.4$ ,  $S = 3$ ,  $X_0 = 8$ ,  $\eta = 1$ .

### SAXS

In `models.sphere` the form factor is defined as:

$$ff_{sph}(q, r) = \frac{3 \sin(qr) - qr \cos(qr)}{(qr)^3}$$

$$v_{sph}(r) = \frac{4}{3} \pi r^3$$

$$v_{sph,abs}(r, \Delta\rho) = \Delta\rho^2 v_{sph}(r)$$

**Where  $q$  is** the scattering vector loaded from the data file and possibly preprocessed, respectively filtered by defining min/max  $q$  or masking invalid values equal or below zero.

**$r$  denotes** the radius of the sphere set in the user interface (UI) or varied during optimization.

**$\Delta\rho$  denotes** the scattering length density difference constant of the model against the solution which is defined in the UI.

`Sphere.formfactor (dataset)`

Calculates the form factor of a sphere defined by:

$$F(q, r) = \frac{3 \sin(qr) - qr \cdot \cos(qr)}{(qr)^3}$$

`Sphere.volume ()`

Calculates the volume of a sphere defined by:

$$v(r) = \frac{4\pi}{3} r^3$$

`Sphere.absVolume ()`

Calculates the volume of a sphere taking the scattering length density difference  $\Delta\rho$  into account:

$$v_{abs}(r, \Delta\rho) = v_{sph}(r) \cdot \Delta\rho^2$$

`SASModel.weight ()`

Calculates an intensity weighting used during fitting. It is based on the scatterers volume. It can be modified by a user-defined compensation exponent  $c$ . The default value is  $c = \frac{2}{3}$



$$w(r) = v(r)^{2c}$$

`SASModel.calcIntensity` (*data*, *compensationExponent=None*)

Returns the intensity  $I$ , the volume  $v_{abs}$  and the intensity weights  $w$  for a single parameter contribution over all  $q$ :

$$I(q, r) = F^2(q, r) \cdot w(r)$$

Contents:

## mcsas.mcsas package

### Submodules

#### mcsas.mcsas.backgroundscalingfit module

class **BackgroundScalingFit** (*findBackground, \*args*)

Bases: object

Chi-squared convergence calculation happens here. Optimizes the scaling and background factor to match *intCalc* closest to *intObs*. Returns an array with scaling factors. Input initial guess *sc* has to be a two-element array with the scaling and background.

#### Input arguments:

##### Parameters

- **intObs** – An array of *measured*, respectively *observed* intensities
- **intCalc** – An array of intensities which should be scaled to match *intObs*
- **intError** – An array of uncertainties to match *intObs*
- **sc** – A 2-element array of initial guesses for scaling factor and background
- **ver** – (*optional*) Can be set to 1 for old version, more robust but slow, default 2 for new version, 10x faster than version 1, requires decent starting values
- **outputIntensity** – (*optional*) Return the scaled intensity as third output argument, default: False
- **background** – (*optional*) Enables a flat background contribution, default: True

**Returns** (*sc, conval*): A tuple of an array containing the intensity scaling factor and background and the reduced chi-squared value.

**static aGoFsAlpha** (*dataMeas, dataErr, dataCalc*)

The alternative Goodness-of-Fit value without alpha, i.e. multiplied by alpha, according to [Henn 2016] (<http://dx.doi.org/10.1107/S2053273316013206>).

**calc** (*data, modelData, sc, ver=2*)

**Warning:** method 'mcsas.backgroundscalingfit.BackgroundScalingFit.calc' undocumented

**static chi** (*sc, dataMeas, dataErr, dataCalc*)

Chi calculation, difference of measured and calculated signal.

**static chiNoBg** (*sc, dataMeas, dataErr, dataCalc*)

Chi calculation, difference of measured and calculated signal, scaling only, no background.

**static chiSqr** (*dataMeas, dataErr, dataCalc*)

Reduced Chi-squared calculation, size of parameter-space not taken into account; for data with known intError.

**dataScaled** (*data, sc*)

Returns the input data scaled by the provided factor and background level applied if requested.

**fitLM** (*dataMeas, dataErr, dataCalc, sc*)

**Warning:** method 'mcsas.backgroundscalingfit.BackgroundScalingFit.fitLM' undocumented

**fitSimplex** (*dataMeas, dataErr, dataCalc, sc*)

**Warning:** method 'mcsas.backgroundscalingfit.BackgroundScalingFit.fitSimplex' undocumented

## mcsas.mcsas.mcsas module

### class McSAS

Bases: *bases.algorithm.algorithmbase.AlgorithmBase*

Main class containing all functions required to do Monte Carlo fitting.

#### Required:

- data:** The dataset to fit. Has to be an instance of :py:class:SASData
- model:** The scattering model object to assume. It has to be an instance of ScatteringModel.

For more settings, see mcsas/mcsasparameters.json

#### Returns:

A McSAS object with the following Results stored in the *result* member attribute. These can be extracted using `McSAS.result[<parameterIndexNumber>][<Keyword>]` where the *parameterIndexNumber* indicates which

shape parameter information is requested. E.g. an ellipsoid has 3: width, height and orientation. (Some information is only stored in *parameterIndexNumber* = 0 (default)).

**Keyword** may be one of the following:

***fitMeasValMean***: 1D array (*common result*) The fitted measVal, given as the mean of all numReps Results.

***fitX0***: 1D array (*common result*) Corresponding q values (may be different than the input q if *X0Bounds* was used).

***fitMeasValStd***: array (*common result*) Standard deviation of the fitted I(q), calculated as the standard deviation of all numReps results.

***contribs***: size array (numContribs x numReps) (*common result*) Collection of numContribs contributions fitted to best represent the provided I(q) data. Contains the Results of each of *numReps* iterations. This can be used for rebinning without having to re-optimize.

***scalingFactors***: size array (2 x numReps) (*common result*) Scaling and background values for each repetition. Used to display background level in data and fit plot.

***histogramXLowerEdge***: array histogram bin left edge position (x-axis in histogram).

***histogramXMean***: array Center positions for the size histogram bins (x-axis in histogram, used for errorbars).

***histogramXWidth***: array histogram bin width (x-axis in histogram, defines bar plot bar widths).

***volumeHistogramYMean***: array Volume-weighted particle size distribution values for all numReps Results (y-axis bar height).

***numberHistogramYMean***: array Number-weighted analogue of the above *volumeHistogramYMean*.

***volumeHistogramRepetitionsY***: size array (self.histogramBins x numReps) Volume-weighted particle size distribution bin values for each fit repetition (the mean of which is *volumeHistogramYMean*, and the sample standard deviation is *volumeHistogramYStd*).

***numberHistogramRepetitionsY***: size array (self.histogramBins x numReps) Number-weighted particle size distribution bin values for each MC fit repetition.

***volumeHistogramYStd***: array Standard deviations of the corresponding volume-weighted size distribution bins, calculated from *numReps* repetitions of the model fitting function.

***numberHistogramYStd***: array Standard deviation for the number-weighted distribution.

***volumeFraction***: size array (numContribs x numReps) Volume fractions for each of numContribs contributions in each of *numReps* iterations.

***numberFraction***: size array (numContribs x numReps) Number fraction for each contribution.

***totalVolumeFraction***: size array (numReps) Total scatterer volume fraction for each of the *numReps* iterations.

***totalNumberFraction***: size array (numReps) Total number fraction.

***minimumRequiredVolume***: size array (numContribs x numReps) Minimum required volume fraction for each contribution to become statistically significant.

***minimumRequiredNumber***: size array (numContribs x numReps) Number-weighted analogue to *minimumRequiredVolume*.

***volumeHistogramMinimumRequired***: size array (histogramXMean) Array with the minimum required volume fraction per bin to become statistically significant. Used to display minimum required level in histogram.

***numberHistogramMinimumRequired***: size array (**histogramXMean**) Number-weighted analogue to *volumeHistogramMinimumRequired*.

***scalingFactors***: size array (**2 x numReps**) Scaling and background values for each repetition. Used to display background level in data and fit plot.

***totalVolumeFraction***: size array (**numReps**) Total scatterer volume fraction for each of the *numReps* iterations.

***minimumRequiredVolume***: size array (**numContribs x numReps**) Minimum required volume fraction for each contribution to become statistically significant.

***volumeHistogramMinimumRequired***: size array (**histogramXMean**) Array with the minimum required volume fraction per bin to become statistically significant. Used to display minimum required level in histogram.

Creates instances from defined parameters and replaces the class attributes accordingly.

**analyse** ()

This function runs the Monte Carlo optimisation a multitude (*numReps*) of times. If convergence is not achieved, it will try again for a maximum of *maxRetries* attempts.

**calc** (\*\*kwargs)

**Warning:** method 'mcsas.mcsas.McSAS.calc' undocumented

**data** = None

**classmethod factory** ()

**Warning:** method 'mcsas.mcsas.McSAS.factory' undocumented

**gen2DMeasVal** ()

This function is optionally run after the histogram procedure for anisotropic images, and will calculate the MC fit measVal in image form

**histogram** (contribs=None)

Takes the *contribs* result from the *McSAS.analyse()* function and calculates the corresponding volume- and number fractions for each contribution as well as the minimum observability limits. It will subsequently bin the Result across the range for histogramming purposes.

While the volume-weighted distribution will be in absolute units (providing volume fractions of material within a given size range), the number distributions have been normalized to 1.

Output a list of dictionaries with one dictionary per shape parameter:

***histogramXLowerEdge***: array histogram bin left edge position (x-axis in histogram)

***histogramXMean***: array Center positions for the size histogram bins (x-axis in histogram, used for errorbars)

***histogramXWidth***: array histogram bin width (x-axis in histogram, defines bar plot bar widths)

***volumeHistogramYMean***: array Volume-weighted particle size distribution values for all *numReps* Results (y-axis bar height)

***numberHistogramYMean***: **array** Number-weighted analogue of the above *volumeHistogramYMean*

***volumeHistogramRepetitionsY***: **size (histogramBins x numReps) array** Volume-weighted particle size distribution bin values for each MC fit repetition (whose mean is *volumeHistogramYMean*, and whose sample standard deviation is *volumeHistogramYStd*)

***numberHistogramRepetitionsY***: **size (histogramBins x numReps) array** Number-weighted particle size distribution bin values for each MC fit repetition

***volumeHistogramYStd***: **array** Standard deviations of the corresponding volume-weighted size distribution bins, calculated from *numReps* repetitions of the model fitting function

***numberHistogramYStd***: **array** Standard deviation for the number-weighted distribution

***volumeFraction***: **size (numContribs x numReps) array** Volume fractions for each of numContribs contributions in each of numReps iterations

***numberFraction***: **size (numContribs x numReps) array** Number fraction for each contribution

***totalVolumeFraction***: **size (numReps) array** Total scatterer volume fraction for each of the *numReps* iterations

***totalNumberFraction***: **size (numReps) array** Total number fraction

***minimumRequiredVolume***: **size (numContribs x numReps) array** minimum required volume fraction for each contribution to become statistically significant.

***minimumRequiredNumber***: **size (numContribs x numReps) array** number-weighted analogue to *minimumRequiredVolume*

***volumeHistogramMinimumRequired***: **size (histogramXMean) array** array with the minimum required volume fraction per bin to become statistically significant. Used to display minimum required level in histogram.

***numberHistogramMinimumRequired***: **size (histogramXMean) array** number-weighted analogue to *volumeHistogramMinimumRequired*

***scalingFactors***: **size (2 x numReps) array** Scaling and background values for each repetition. Used to display background level in data and fit plot.

**mcFit** (*numContribs*, *minConvergence*, *outputMeasVal=False*, *outputDetails=False*, *nRun=None*)  
Object-oriented, shape-flexible core of the Monte Carlo procedure. Takes optional arguments:

***outputMeasVal***: Returns the fitted *measVal* besides the *Result*

***outputDetails***: details of the fitting procedure, number of iterations and so on

***nRun***: “serial number” of run. Used to store results in *parameters* in the right place

**model** = None

**plot** (*axisMargin=0.3*, *outputFilename=None*, *autoClose=False*)  
Expects *outputFilename* to be of type *gui.calc.OutputFilename*.

**result** = None

**array** (*value*)

**Warning:** function ‘*mcsas.mcsas.array*’ undocumented

**mcsas.mcsas.mcsasdefaultcfg module**

default for settings and info used for a McSAS run used by McSASCfg

**class ExtendedEncoder** (*skipkeys=False, ensure\_ascii=True, check\_circular=True, allow\_nan=True, sort\_keys=False, indent=None, separators=None, default=None*)  
 Bases: `json.encoder.JSONEncoder`

JSON encoder extended to deal with Unicode, arrays and cls descriptions

Constructor for JSONEncoder, with sensible defaults.

If `skipkeys` is false, then it is a `TypeError` to attempt encoding of keys that are not str, int, float or None. If `skipkeys` is True, such items are simply skipped.

If `ensure_ascii` is true, the output is guaranteed to be str objects with all incoming non-ASCII characters escaped. If `ensure_ascii` is false, the output can contain non-ASCII characters.

If `check_circular` is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an `OverflowError`). Otherwise, no such check takes place.

If `allow_nan` is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a `ValueError` to encode such floats.

If `sort_keys` is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If `indent` is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, `separators` should be an (item\_separator, key\_separator) tuple. The default is (' ', ': ') if `indent` is None and (';', ': ') otherwise. To get the most compact JSON representation, you should specify (';', ':') to eliminate whitespace.

If specified, `default` is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a `TypeError`.

**default** (*obj*)

**Warning:** method 'mcsas.mcsasdefaultcfg.ExtendedEncoder.default' undocumented

**Parameter** (*\*args, \*\*kwargs*)

**Warning:** function 'mcsas.mcsasdefaultcfg.Parameter' undocumented

**class cInfo** (*\*\*kwargs*)

Bases: `object`

This class contains all the information required to read, verify and write configuration parameters files.

initialise the defaults and populate the database with values where appropriate default parameter file can be provided using kwarg: `paramDefFile = 'path/to/file'` `mcsasparameters.json` should be in the same directory as this function

`getPar (key)`

**Warning:** method ‘mcsas.mcsasdefaultcfg.cInfo.getPar’ undocumented

`getParVal (par)`

shortcut method for getting the value of a parameter

`loadParams (fname=None)`

writes the default definitions and bounds for the configuration parameters to self.parameters Can also be used to update existing parameters from supplied filename

`parameterNames = []`

`parameters = None`

`parseConfig ()`

Runs through the entire settings, raising warnings where necessary

`setParVal (par, value)`

shortcut method for setting the value of a parameter only

`writeConfig (fname)`

writes the configuration to a settings file. Required input parameter is the filename to write to.

### mcsas.mcsas.mcsasparameters module

`class McSASParameters (paramDefFile=None)`

Bases: `utils.propertynames.PropertyNames`

**Defines the static parameters used for the fitting procedure:**

- *model*: an instance of McSAS.model defining the fitting model
- *contribParamBounds*: Bounds of the active (fitting) parameter
- *qBounds*: limits in *q* between which the fit is applied
- *psiBounds*: **limits in azimuthal angle (2D patterns only) to which** the fit is applied
- *qMagnitude*: indicates the multiplier to scale *q* to  $m^{-1}$
- *iMagnitude*: indicates the multiplier to scale *I* to  $(m\ sr)^{-1}$
- *histogramBins*: **number of bins to use for size distribution** determination. Does not affect fit
- *histogramXScale*: can be “log” or “linear”, sets the horizontal axis scaling. Does not affect fit
- *histogramWeighting*: can be “volume”(recommended) or “number”. Determines whether to plot the volume-weighted size distribution (which is closest to what is measured in a scattering measurement) or a number-weighted size distribution.
- *deltaRhoSquared* **the value (in  $m^{-4}$ ) of the squared electron** density contrast. Typically on the order of  $10^{25}$  to  $10^{30}$  for X-ray scattering measurements. To get a correct volume fraction, the intensity must be in reciprocal meters, as must *q* and the object size parameters must be adjusted accordingly.
- *startFromMinimum*: **may be depreciated: starts with an initial guess** consisting of minimum size values rather than random. For testing purposes only.
- *maxRetries*: **if a single optimisation fails, it will be retried** this integer of times.



- ***maskNegativeInt***: may be depreciated due to overlap with similar functionality in the McSAS.data module. Setting this will ignore datapoints with intensity values < 0
- ***maskZeroInt***: similar to above, but for intensity values = 0

Most of them should be moved to McSAS as dynamic parameters of type *Parameter* which allows them to be configurable in the GUI. Some, esp. *histogramBins* and *histogramXScale* should be moved to a custom *FitParameter* class along with the *active* flag. (WIP)

Extension in progress: the class can take keyword-value pairs to overwrite (some of) the parameter values. Additionally, a default configuration file (json-style) can be provided using kwarg: `paramDefFile = 'path/to/file'` A (limited) set of custom parameters can be supplied in another configuration file that overwrite the defaults: `paramFile = 'path/to/file'`

The order is:

- default file, whose values are superseded by
- custom file, whose values are superseded by
- keyword-value pairs, which only set parameter *values*, or
- keyword-dict pairs, which sets parameter attributes as defined in the supplied dictionary

initialise the defaults and populate the database with values where appropriate default parameter file can be provided using kwarg: `paramDefFile = 'path/to/file'` relative to application root dir

**contribParamBounds** = ()

**loadParameters** (*filename*)

**Warning:** method 'mcsas.mcsasparameters.McSASParameters.loadParameters' undocumented

**model** = None

**paramDefFile** = 'mcsas/mcsasparameters.json'

**parameters** = []

**pickUnit** (*unitClass=None, displayUnit=None*)

returns a unit object instance of the right class and displayUnit

**Parameter** (\*args, \*\*kwargs)

**Warning:** function 'mcsas.mcsasparameters.Parameter' undocumented

### mcsas.mcsas.plotting module

Defines the format of the final report on success of an MC fit.

**class CoordinateFormat** (*xname, xunit, yname, yunit*)

Bases: object

A Function object which sets up the particular formatting with axis name and associated unit at initialization time and formats plot coordinates given as (x,y) pair.

**class PlotResults** (*allRes, dataset, axisMargin=0.3, outputFilename=None, modelData=None, auto-Close=False, logToFile=False, queue=None*)

Bases: object

This function plots the output of the Monte-Carlo procedure in two windows, with the left window the measured signal versus the fitted measVal (on double-log scale), and the righthand window the size distribution.

**figInit** (*nHists, figureTitle, nR=1*)

initialize figure and initialise axes using GridSpec. Each rangeinfo (nR) contains two rows and nHists + 1 columns. the top row axes are for placing text objects: settings and stats. The bottom row axes are for plotting the fits and the histograms TODO: add settings to window title? (next to figure\_xy)

**formatAlgoInfo** ()

Preformats the algorithm information ready for printing the colons are surrounded by string-marks, to force LaTeX rendering

**formatRangeInfo** (*parHist, RI, weighti=0*)

Preformats the rangeInfo results ready for printing

**plot1D** (*dataset, fitX0, fitMeasVal, qAxis*)

plots 1D data and fit

**classmethod plotGrid** (*ax*)

**Warning:** method 'mcsas.plotting.PlotResults.plotGrid' undocumented

**plotHist** (*plotPar, parHist, hAxis, rangei*)

histogram plot

**plotInfo** (*InfoAxis*)

plots the range statistics in the small info axes above plots

**plotPartial** (*fitX0, fitMeasVal, fitSTD, qAxis, label='MC partial measVal'*)

plots 1D data and fit

**plotStats** (*parHist, rangei, fig, InfoAxis*)

plots the range statistics in the small info axes above plots

**setAxis** (*ah*)

**Warning:** method 'mcsas.plotting.PlotResults.setAxis' undocumented

**class PlotSeriesStats**

Bases: object

Simple 1D plotting of series statistics.

**plot** (*stats*)

**Warning:** method 'mcsas.plotting.PlotSeriesStats.plot' undocumented

**show** ()

**Warning:** method ‘mcsas.plotting.PlotSeriesStats.show’ undocumented

**getTextSize** (*fig, fontProps*)

Returns the width and height of a character for the given font setup. This can be different on each platform.

## Module contents

McSAS Core

## mcsas.models package

### Submodules

#### mcsas.models.cylindersisotropic module

**class** **CylindersIsotropic**

Bases: *bases.model.sasmodel.SASModel*

Form factor of cylinders previous version (length-fixed) checked against SASfit

**absVolume** ()

**Warning:** method ‘models.cylindersisotropic.CylindersIsotropic.absVolume’ undocumented

**formfactor** (*dataset*)

**Warning:** method ‘models.cylindersisotropic.CylindersIsotropic.formfactor’ undocumented

**parameters** = (None, None, None, None, None, None, None)

**shortName** = ‘Isotropic Cylinders’

**volume** ()

**Warning:** method ‘models.cylindersisotropic.CylindersIsotropic.volume’ undocumented

**Parameter** (*\*args, \*\*kwargs*)

**Warning:** function ‘models.cylindersisotropic.Parameter’ undocumented

**mcsas.models.cylindersisotropicaspect module****class CylindersIsotropic**Bases: *bases.model.sasmodel.SASModel*

Form factor of cylinders which are radially isotropic (so not spherically isotropic!) !!!completed but not verified!!!

**formfactor** (*dataset*)

**Warning:** method 'models.cylindersisotropicaspect.CylindersIsotropic.formfactor' undocumented

**parameters** = (None, None, None, None)**shortName** = 'Cylinders defined by aspect ratio'**volume** ()

**Warning:** method 'models.cylindersisotropicaspect.CylindersIsotropic.volume' undocumented

**Parameter** (\*args, \*\*kwargs)

**Warning:** function 'models.cylindersisotropicaspect.Parameter' undocumented

**mcsas.models.cylindersradiallyisotropic module****class CylindersRadiallyIsotropic**Bases: *bases.model.sasmodel.SASModel*

Form factor of cylinders which are radially isotropic (so not spherically isotropic!) !!!completed but not verified!!!

**absVolume** ()

**Warning:** method 'models.cylindersradiallyisotropic.CylindersRadiallyIsotropic.absVolume' undocumented

**formfactor** (*dataset*)

**Warning:** method 'models.cylindersradiallyisotropic.CylindersRadiallyIsotropic.formfactor' undocumented

**parameters** = (None, None, None, None, None)**shortName** = 'Radially (in-plane) isotropic cylinders'

**volume** ()

**Warning:** method ‘models.cylindersradiallyisotropic.CylindersRadiallyIsotropic.volume’ undocumented

**Parameter** (\*args, \*\*kwargs)

**Warning:** function ‘models.cylindersradiallyisotropic.Parameter’ undocumented

### mcsas.models.cylindersradiallyisotropictilted module

This model is a special case of the radially isotropic cylinders, where the cylinders are also slightly tilted out of the plane parallel to the detector. This out of plane tilt is described using a Gaussian. The integration over this tilt angle is done over several segments of the Gaussian PDF, with each segment occupying an equal cumulative probability. The centroid value used for the integration is the mass-weighted centre.

**class CylindersRadiallyIsotropicTilted**

Bases: *bases.model.sasmodel.SASModel*

Form factor of cylinders *UNFINISHED* which are radially isotropic (so not spherically isotropic!)

**formfactor** (dataset)

**Warning:** method ‘models.cylindersradiallyisotropictilted.CylindersRadiallyIsotropicTilted.formfactor’ undocumented

**parameters** = (None, None, None, None, None, None)

**shortName** = ‘Cylinders defined by aspect ratio’

**volume** ()

**Warning:** method ‘models.cylindersradiallyisotropictilted.CylindersRadiallyIsotropicTilted.volume’ undocumented

**Parameter** (\*args, \*\*kwargs)

**Warning:** function ‘models.cylindersradiallyisotropictilted.Parameter’ undocumented

### mcsas.models.ellipsoidalcoreshell module

**class EllipsoidalCoreShell**

Bases: *bases.model.sasmodel.SASModel*

Form factor for an ellipsoidal core shell structure as defined in the SASfit manual (par. 3.2.3) Tested 2014-01-21 against SASfit function with good agreement.

**absVolume** ()

**Warning:** method 'models.ellipsoidalcoreshell.EllipsoidalCoreShell.absVolume' undocumented

**formfactor** (*dataset*)

**Warning:** method 'models.ellipsoidalcoreshell.EllipsoidalCoreShell.formfactor' undocumented

**parameters** = (None, None, None, None, None, None, None)

**shortName** = 'Core-Shell Ellipsoid'

**volume** ()

**Warning:** method 'models.ellipsoidalcoreshell.EllipsoidalCoreShell.volume' undocumented

**Parameter** (\*args, \*\*kwargs)

**Warning:** function 'models.ellipsoidalcoreshell.Parameter' undocumented

### mcsas.models.ellipsoidsisotropic module

**class EllipsoidsIsotropic**

Bases: *bases.model.sasmodel.SASModel*

Form factor for a spheroidal structure with semi-axes a = b, c. c can be set to be an aspect ratio with respect to a tested with Ellipsoid II from SASfit 20140626

**absVolume** ()

**Warning:** method 'models.ellipsoidsisotropic.EllipsoidsIsotropic.absVolume' undocumented

**formfactor** (*dataset*)

**Warning:** method 'models.ellipsoidsisotropic.EllipsoidsIsotropic.formfactor' undocumented

**parameters** = (None, None, None, None, None, None)

**shortName** = 'Isotropic Ellipsoids'

**volume** ()

**Warning:** method ‘models.ellipsoidsisotropic.EllipsoidsIsotropic.volume’ undocumented

**Parameter** (\*args, \*\*kwargs)

**Warning:** function ‘models.ellipsoidsisotropic.Parameter’ undocumented

## mcsas.models.gaussianchain module

**class GaussianChain**

Bases: *bases.model.sasmodel.SASModel*

Form factor of flexible polymer chains which are not selfavoiding and obey Gaussian statistics after [Debye47]

See also: [http://sasfit.sf.net/manual/Gaussian\\_Chain#Gauss\\_2](http://sasfit.sf.net/manual/Gaussian_Chain#Gauss_2)

$L_0 = (b_p - (k * R_g^2) * \eta_s)^2$  with  $k = 1 \text{ nm}$ .  $k * R_g^2 = \text{volume approximation}$

**fixTestParams** = **functools.partial**(<function GaussianChain.fixTestParams>, <class ‘models.gaussianchain.GaussianC

**formfactor** (dataset)

**Warning:** method ‘models.gaussianchain.GaussianChain.formfactor’ undocumented

**parameters** = (None, None, None, None)

**shortName** = ‘Gaussian Chain’

**volume** ()

**Warning:** method ‘models.gaussianchain.GaussianChain.volume’ undocumented

**Parameter** (\*args, \*\*kwargs)

**Warning:** function ‘models.gaussianchain.Parameter’ undocumented

**test** ()

**Warning:** function ‘models.gaussianchain.test’ undocumented

**mcsas.models.kholodenko module****class Kholodenko**Bases: *bases.model.sasmodel.SASModel*Form factor of a worm-like structure after [*Kholodenko93*]**formfactor** (*dataset*)**Warning:** method 'models.kholodenko.Kholodenko.formfactor' undocumented**parameters** = (None, None, None)**shortName** = 'Kholodenko Worm'**volume** ()**Warning:** method 'models.kholodenko.Kholodenko.volume' undocumented**calcPcs** (*u*)**Warning:** function 'models.kholodenko.calcPcs' undocumented**core** (*z, qValue, kuhnLength, x*)**Warning:** function 'models.kholodenko.core' undocumented**coreIntegral** (*qValue, kuhnLength, x*)**Warning:** function 'models.kholodenko.coreIntegral' undocumented**test** ()**Warning:** function 'models.kholodenko.test' undocumented**mcsas.models.lmadensesphere module****class LMA DenseSphere**Bases: *bases.model.sasmodel.SASModel*

Form factor of a sphere convoluted with a structure factor, equations 15-17 from Pedersen, J. Appl. Cryst. 27 (1994), 595–608. Correct eqn given in Kinning and Thomas, Macromolecules 17 (1984) 1712. Internally set



parameters are volume fraction of the hard spheres, and the multiplication factor /mf/ for an additional stand-off distance between the hard spheres:  $R_h = mf \cdot R$  where  $R_h$  is the hard-sphere radius (“interaction radius”) used in the structure factor,  $R$  is the radius of the sphere, and  $mf$  is the multiplication factor.

**absVolume** ()

**Warning:** method ‘models.lmadensesphere.LMADenseSphere.absVolume’ undocumented

**canSmear** = True

**formfactor** (dataset)

**Warning:** method ‘models.lmadensesphere.LMADenseSphere.formfactor’ undocumented

**parameters** = (None, None, None, None)

**shortName** = ‘LMADenseSphere’

**volume** ()

**Warning:** method ‘models.lmadensesphere.LMADenseSphere.volume’ undocumented

**Parameter** (\*args, \*\*kwargs)

**Warning:** function ‘models.lmadensesphere.Parameter’ undocumented

## mcsas.models.sphere module

**Parameter** (\*args, \*\*kwargs)

**Warning:** function ‘models.sphere.Parameter’ undocumented

## class Sphere

Bases: *bases.model.sasmodel.SASModel*

Form factor of a sphere

**absVolume** ()

Calculates the volume of a sphere taking the scattering length density difference  $\Delta\rho$  into account:

$$v_{abs}(r, \Delta\rho) = v_{sph}(r) \cdot \Delta\rho^2$$

**canSmear** = True

**formfactor** (*dataset*)

Calculates the form factor of a sphere defined by:

$$F(q, r) = \frac{3 \sin(qr) - qr \cdot \cos(qr)}{(qr)^3}$$

**parameters** = (None, None)

**shortName** = 'Sphere'

**surface** ()

Calculates the surface of a sphere defined by:

$$s(r) = 4\pi r^2$$

**volume** ()

Calculates the volume of a sphere defined by:

$$v(r) = \frac{4\pi}{3} r^3$$

**test** ()

**Warning:** function 'models.sphere.test' undocumented

## mcsas.models.sphericalcoreshell module

**Parameter** (\*args, \*\*kwargs)

**Warning:** function 'models.sphericalcoreshell.Parameter' undocumented

**class SphericalCoreShell**

Bases: *bases.model.sasmodel.SASModel*

Form factor for a spherical core shell structure as defined in the SASfit manual (par. 3.1.4, Spherical Shell III). One modification is the ability to specify SLD for core, shell and solvent, identical to the notation used in the Core-shell ellipsoid. Compared with a SASfit-generated model (both with and without distribution)

**absVolume** ()

**Warning:** method 'models.sphericalcoreshell.SphericalCoreShell.absVolume' undocumented

**formfactor** (*dataset*)

**Warning:** method 'models.sphericalcoreshell.SphericalCoreShell.formfactor' undocumented

**parameters** = (None, None, None, None, None)

**shortName** = 'Core-Shell Sphere'

`volume()`

**Warning:** method ‘models.sphericalcoreshell.SphericalCoreShell.volume’ undocumented

## Module contents

### mcsas.gui package

#### Subpackages

mcsas.gui.bases package

#### Subpackages

mcsas.gui.bases.mainwindow package

#### Submodules

mcsas.gui.bases.mainwindow.mainwindow module

**class** `MainWindow` (*appversion*, *parent=None*)

Bases: `QMainWindow`, `gui.bases.mainwindow.ui_mainwindow.Ui_MainWindow`, `gui.bases.mixins.appsettings.AppSettings`

Main window base class.

Provides functionality for storing and loading application settings, managing widgets.

**closeEvent** (*event*)

**Warning:** method ‘gui.bases.mainwindow.mainwindow.MainWindow.closeEvent’ undocumented

**getCommandLineArguments** ()

Get command line arguments, if any.

**onStartupSignal**

**restoreSettings** ()

Load defaults for settings if missing and available.

**show** ()

**Warning:** method ‘gui.bases.mainwindow.mainwindow.MainWindow.show’ undocumented

**storeSettings** ()

**Warning:** method ‘gui.bases.mainwindow.mainwindow.MainWindow.storeSettings’ undocumented

### mcsas.gui.bases.mainwindow.mainwindow\_rc module

**qCleanupResources** ()

**Warning:** function ‘gui.bases.mainwindow.mainwindow\_rc.qCleanupResources’ undocumented

**qInitResources** ()

**Warning:** function ‘gui.bases.mainwindow.mainwindow\_rc.qInitResources’ undocumented

### mcsas.gui.bases.mainwindow.ui\_mainwindow module

**class Ui\_MainWindow**

Bases: object

**retranslateUi** (*MainWindow*)

**Warning:** method ‘gui.bases.mainwindow.ui\_mainwindow.Ui\_MainWindow.retranslateUi’ undocumented

**setupUi** (*MainWindow*)

**Warning:** method ‘gui.bases.mainwindow.ui\_mainwindow.Ui\_MainWindow.setupUi’ undocumented

## Module contents

### mcsas.gui.bases.mixins package

#### Submodules

### mcsas.gui.bases.mixins.appsettings module

**class AppSettings**

Bases: object

**appSettings**

**setRootGroup()**

Resets any QSettings group(s) currently set.

### mcsas.gui.bases.mixins.contextmenuwidget module

**class ContextMenuWidget**

Bases: `object`

Menu states are mutual exclusive states of available context menu actions.

Menu states map to boolean methods of this object. The names of a state equal names of methods of this object otherwise the menu state is useless. `updateMenu()` tests the states, i.e. boolean methods for a true return value. The first one specifies the menu to be build. If none matches, the 'default' state will be used.

Menu states are defined by `updateMenuState()`. The provided list of entry names has to be a partial set of those returned by `menuEntries()`. Entries may be defined by `addMenuEntry()`.

TODO: Instead of multiple inheritance, convert it to a standalone object. Problems: need to store bound methods as callbacks, to get state on `update()` The widget this menu applies to needs to be stored only once at init.

**action** (*name*)

**Warning:** method 'gui.bases.mixins.contextmenuwidget.ContextMenuWidget.action' undocumented

**addMenuEntry** (*name=None, text=None, toolTip=None, shortCut=None, checkable=False, checked=False, callbacks=None, menuStates=None*)

Argument 'callbacks' is supposed to be a list of methods which will be connected to the `QAction.triggered` signal.

**addMenuEntryAction** (*name, action, menuStates=None*)

Common menu states: default and all. 'all' includes the default state. Everything else expects a method of the same name which evaluates to True or False. It is used to show or hide the respective context menu action. Actions bound to the same state appear grouped in the menu. The underlying ordered dict preserves the order of states.

**addMenuSeparator** (*menuStates=None*)

**Warning:** method 'gui.bases.mixins.contextmenuwidget.ContextMenuWidget.addMenuSeparator' undocumented

**addToMenuState** (*stateName, \*entryNames*)

**Warning:** method 'gui.bases.mixins.contextmenuwidget.ContextMenuWidget.addToMenuState' undocumented

**allMenuStates** = '\*'

**menuEntries** (*stateName*)

**Warning:** method ‘gui.bases.mixins.contextmenuwidget.ContextMenuWidget.menuEntries’ undocumented

**removeMenuEntries** (*stateName*)

**Warning:** method ‘gui.bases.mixins.contextmenuwidget.ContextMenuWidget.removeMenuEntries’ undocumented

**updateMenu** (*widget=None*)

**Warning:** method ‘gui.bases.mixins.contextmenuwidget.ContextMenuWidget.updateMenu’ undocumented

**escapeAmp** (*text*)

**Warning:** function ‘gui.bases.mixins.contextmenuwidget.escapeAmp’ undocumented

### mcsas.gui.bases.mixins.dropwidget module

**class DropWidget**

Bases: `object`

Drag&Drop support for widgets which inherit from this.

**dragEnterEvent** (*ev*)

**Warning:** method ‘gui.bases.mixins.dropwidget.DropWidget.dragEnterEvent’ undocumented

**dropEvent** (*ev*)

**Warning:** method ‘gui.bases.mixins.dropwidget.DropWidget.dropEvent’ undocumented

### mcsas.gui.bases.mixins.titlehandler module

**class TitleHandler** (*title*)

Bases: `bases.dataset.titlemixin.TitleMixin`

**registerUpdateFunc** (*func*)

**Warning:** method ‘gui.bases.mixins.titlehandler.TitleHandler.registerUpdateFunc’ undocumented

**classmethod** **setup** (*parent*, *title*)

Gets a title and the widget this title belongs to.

**update** (*obj*)

**Warning:** method ‘gui.bases.mixins.titlehandler.TitleHandler.update’ undocumented

## Module contents

### Submodules

#### mcsas.gui.bases.datalist module

**class** **DataItem** (*data*)

Bases: `QTreeWidgetItem`

Generates a `QTreeWidgetItem` from arbitrary python objects. Storing those objects separately.

**data** (*\*args*)

**Warning:** method ‘gui.bases.datalist.DataItem.data’ undocumented

**dataId** ()

**Warning:** method ‘gui.bases.datalist.DataItem.dataId’ undocumented

**getItemProperty** (*value*)

For a value, returns this items getter/setter methods according to value type.

**static** **hash32** (*data*)

Avoids `OverflowError` at `setData()` with PySide on MacOS.

**isRemovable**

**isTopLevelItem** ()

**Warning:** method ‘gui.bases.datalist.DataItem.isTopLevelItem’ undocumented

**listIndex** ()

Index of this items top most parent in the treewidget.

**remove** ()

Removes the item from its treewidget or parent item.

**setAlignment** (*alignment*)

**Warning:** method 'gui.bases.datalist.DataItem.setAlignment' undocumented

**setChanged** (*column*)

**Warning:** method 'gui.bases.datalist.DataItem.setChanged' undocumented

**setClicked** (*column*)

**Warning:** method 'gui.bases.datalist.DataItem.setClicked' undocumented

**update** ()

Updates this item according to eventually changed data object

**wasClickedAndChanged** ()

Tests if this item was previously clicked and changed in the UI.

**class DataList** (*parent=None, title=None, withBtn=True, nestedItems=True*)

Bases: `QWidget`, `gui.bases.mixins.dropwidget.DropWidget`, `gui.bases.mixins.contextmenuwidget.ContextMenuWidget`

Manages all loaded spectra.

```
>>> from utilsgui import DialogInteraction, DisplayException
>>> from spectralist import SpectraList
>>> sl = DialogInteraction.instance(SpectraList)
```

Test available actions >>> [str(action.text()) for action in sl.listWidget.actions()] ['load spectra', 'remove', '', 'save matrices', 'select all'] >>> sl.listWidget.count() 0

Test methods on empty list >>> sl.updateSpectra() >>> sl.removeSelectedSpectra() >>> [sl.getMatrix(i) for i in -1,0,1] [None, None, None] >>> DialogInteraction.query(DisplayException, sl.saveMatrix, ... slot = 'accept') >>> sl.selectionChangedSlot()

**add** (*data*)

**Warning:** method 'gui.bases.datalist.DataList.add' undocumented

**clear** ()

**Warning:** method 'gui.bases.datalist.DataList.clear' undocumented

**currentSelection** ()



**Warning:** method 'gui.bases.datalist.DataList.currentSelection' undocumented

**data** (*indexOrItem=None, selectedOnly=False*)

Returns the list of data for a given list index or list widget item. If none is specified return the data of all items or the data of selected items only, if desired.

**expandAll** ()

**Warning:** method 'gui.bases.datalist.DataList.expandAll' undocumented

**fitColumnsToContents** (\*args)

**Warning:** method 'gui.bases.datalist.DataList.fitColumnsToContents' undocumented

**hasSelection** ()

**Warning:** method 'gui.bases.datalist.DataList.hasSelection' undocumented

**isEmpty** ()

**Warning:** method 'gui.bases.datalist.DataList.isEmpty' undocumented

**isNotEmpty** ()

**Warning:** method 'gui.bases.datalist.DataList.isNotEmpty' undocumented

**isRemovableSelected** ()

True, if there is at least one item selected which may be removed

**itemDoubleClicked** (*item, column*)

**Warning:** method 'gui.bases.datalist.DataList.itemDoubleClicked' undocumented

**itemUpdate** (*item, column*)

Reimplement to update item if changed by user in GUI

**itemsHaveChildren** ()

**Warning:** method 'gui.bases.datalist.DataList.itemsHaveChildren' undocumented

**leaveEvent** (*event*)

**Warning:** method 'gui.bases.datalist.DataList.leaveEvent' undocumented

**loadData** (*sourceList=None, processSourceFunc=None, showProgress=True, alignment=None, \*\*kwargs*)

Loads a list of data source items.

*processSourceFunc* is expected to be a function which gets individual elements of *sourceList* as argument. It returns an arbitrary data item which is then added to this data list widget.

Reimplement it in child classes and it will be called on load button and add action signal.

This method handles exceptions and progress indication.

Test loading a single spectra >>> import utils >>> from tests import TestData >>> from utils-gui import DialogInteraction, UiSettings, fileDialogType >>> from chemsettings import ChemSettings >>> from datafiltersgui import DataFiltersGui >>> from spectralist import SpectraList >>> cs = DialogInteraction.instance(ChemSettings) >>> dfg = DialogInteraction.instance(DataFiltersGui) >>> sl = DialogInteraction.instance(SpectraList, settings = cs) >>> utils.LastPath.path = TestData.spectra(0) >>> DialogInteraction.query(fileDialogType(), sl.loadData, ... slot = 'accept') >>> sl.updateSpectra() >>> utils.LastPath.path = utils.getTempFileName() >>> matrixfiles = DialogInteraction.query(fileDialogType(), sl.saveMatrix, ... slot = 'accept') >>> len(matrixfiles) 1 >>> matrixfiles

Verify written matrix data with existent matrix export >>> TestData.verifyMatrix(TestData.spectra(0), ... matrixfiles[0]) True

**removeItems** (*indexList*)

Deletes items specified in the given list of indices.

**removeSelected** ()

**Warning:** method 'gui.bases.datalist.DataList.removeSelected' undocumented

**reraiseLast** ()

Reraise the last error if any and display an error message dialog.

**selectAll** ()

Selects all items in the list if not all are selected. Clears the selection if all items in the list already are selected.

**selectionChanged** ()

**Warning:** method 'gui.bases.datalist.DataList.selectionChanged' undocumented

**setCurrentIndex** (*index*)

**Warning:** method 'gui.bases.datalist.DataList.setCurrentIndex' undocumented

**setHeader** (*labels=None*)

**Warning:** method 'gui.bases.datalist.DataList.setHeader' undocumented

**setupUi** ()

Reimplement this in child classes for custom UI configuration.

**sigEditingFinished**

**sigEmpty**

**sigReceivedUrls**

**sigRemovedData**

**sigSelectedData**

**sigUpdatedData**

**topLevelItems** ()

**Warning:** method 'gui.bases.datalist.DataList.topLevelItems' undocumented

**updateData** (*selectedOnly=False, showProgress=True, updateFunc=None, prepareFunc=None, stopFunc=None, \*\*kwargs*)

Calls the provided function on all data items.

The object returned by prepareFunc() is forwarded as optional argument to updateFunc(dataItem, optionalArguments = None).

**updateItems** ()

**Warning:** method 'gui.bases.datalist.DataList.updateItems' undocumented

### mcsas.gui.bases.dockwidget module

**class DockWidget** (*parent, childWidgetType, \*args, \*\*kwargs*)

Bases: QDockWidget

Widget for docking in a QMainWindow environment.

**child**

**closeEvent** (*event*)

**Warning:** method ‘gui.bases.dockwidget.DockWidget.closeEvent’ undocumented

**onVisibilityChange** (*visible*)

**Warning:** method ‘gui.bases.dockwidget.DockWidget.onVisibilityChange’ undocumented

### mcsas.gui.bases.logwidget module

**class LogWidget** (*parent=None, appversion=None*)

Bases: QTextBrowser, *gui.bases.mixins.contextmenuwidget.ContextMenuWidget*

Simple TextEdit which can save its contents to file.

Fill it with text. >>> from gui.utils.dialoginteraction import DialogInteraction >>> from gui.widgets.logwidget import LogWidget >>> testdata = ‘blablubbanblubb’ >>> te = DialogInteraction.instance(LogWidget) >>> te.append(testdata) >>> te.contents() == testdata True

Call save to file dialog and save. >>> fn = str(DialogInteraction.query(fileDialogType(), te.saveToFile, ... slot = ‘accept’))

Verify written data and remove test file. >>> import os >>> data = None >>> if fn is not None: ... data = open(fn).read() ... os.remove(fn) >>> data == testdata True

PlainTextEdit Tests: Simple TextEdit which can save its contents to file.

Fill it with text. >>> from utilsgui import DialogInteraction, fileDialogType >>> from plaintextedit import PlainTextEdit >>> testdata = ‘blablubbanblubb’ >>> te = DialogInteraction.instance(PlainTextEdit) >>> te.appendPlainText(testdata) >>> te.toPlainText() == testdata True

Call save to file dialog and save. >>> fn = str(DialogInteraction.query(fileDialogType(), te.saveToFile, ... slot = ‘accept’))

Verify written data and remove test file. >>> import os >>> data = None >>> if fn is not None: ... data = open(fn).read() ... os.remove(fn) >>> data == testdata True

**addWatchDir** (*path*)

Adds a directory to check for *plotlog* messages from multiprocessing.

**append** (*text*)

Appends a new line of text.

**appversion**

**clear** ()

**Warning:** method ‘gui.bases.logwidget.LogWidget.clear’ undocumented

**contents** ()

**Warning:** method ‘gui.bases.logwidget.LogWidget.contents’ undocumented

**isCopyAvailable()**

**Warning:** method 'gui.bases.logwidget.LogWidget.isCopyAvailable' undocumented

**isEmpty()**

**Warning:** method 'gui.bases.logwidget.LogWidget.isEmpty' undocumented

**onCloseSlot()**

Workaround to exit a program during calculation. Logging a message processes Qt events (see `append()`). It makes sure this slot is called if connected to a application close signal. For example, emitted by the main window on `closeEvent()`.

**saveToFile** (*filename=None*)

**Warning:** method 'gui.bases.logwidget.LogWidget.saveToFile' undocumented

**scrollToBottom()**

**Warning:** method 'gui.bases.logwidget.LogWidget.scrollToBottom' undocumented

**scrollToTop()**

**Warning:** method 'gui.bases.logwidget.LogWidget.scrollToTop' undocumented

**setCopyAvailable** (*yes*)

**Warning:** method 'gui.bases.logwidget.LogWidget.setCopyAvailable' undocumented

**title = None**

**url2href** (*text*)

[http://daringfireball.net/2010/07/improved\\_regex\\_for\\_matching\\_urls](http://daringfireball.net/2010/07/improved_regex_for_matching_urls)

### mcsas.gui.bases.settingswidget module

**class SettingsWidget** (*parent=None*)

Bases: `QWidget`

Provides access to user provided application settings.

Call `get(<objectname>)` to get an input widget value.

**connectInputWidgets** (*widget*)

**Warning:** method ‘gui.bases.settingswidget.SettingsWidget.connectInputWidgets’ undocumented

**get** (*key, defaultValue=None*)

Retrieves the values for a given key name.

**static getEditingFinishedSignal** (*widget*)

**Warning:** method ‘gui.bases.settingswidget.SettingsWidget.getEditingFinishedSignal’ undocumented

**getInputWidget** (*datatype*)

**Warning:** method ‘gui.bases.settingswidget.SettingsWidget.getInputWidget’ undocumented

**getValue** (*widget, defaultValue=None*)

Retrieves the values for a given widget.

**getWidget** (*key*)

**Warning:** method ‘gui.bases.settingswidget.SettingsWidget.getWidget’ undocumented

**set** (*key, value*)

**Warning:** method ‘gui.bases.settingswidget.SettingsWidget.set’ undocumented

**setValue** (*widget, value*)

**Warning:** method ‘gui.bases.settingswidget.SettingsWidget.setValue’ undocumented

**sigValueChanged**

**sigValuesChanged**

### mcsas.gui.bases.settingswidget\_test module

**class TestSettings** (*parent=None*)

Bases: *gui.bases.settingswidget.SettingsWidget*

**setupUi** (*dummy*)

**Warning:** method ‘gui.bases.settingswidget\_test.TestSettings.setupUi’ undocumented

**testFloatingPointInputBox** ()

**Warning:** function ‘gui.bases.settingswidget\_test.testFloatingPointInputBox’ undocumented

**testIntegerInputBox** ()

**Warning:** function ‘gui.bases.settingswidget\_test.testIntegerInputBox’ undocumented

**testTextualInputBox** ()

**Warning:** function ‘gui.bases.settingswidget\_test.testTextualInputBox’ undocumented

**w**

## Module contents

**mcsas.gui.utils package**

### Subpackages

**mcsas.gui.utils.appversion package**

### Submodules

**mcsas.gui.utils.appversion.appversion module**

**class AppVersion** (*programName=None, versionNumber=None, organizationName=None, organization-Domain=None, defaultSettings=None*)

Bases: object

Stores version meta data.

**defaultSettings** ()

**Warning:** method ‘gui.utils.appversion.appversion.AppVersion.defaultSettings’ undocumented

**classmethod** `isValid(other)`

**Warning:** method ‘gui.utils.appversion.appversion.AppVersion.isValid’ undocumented

**name** (`()`)

**Warning:** method ‘gui.utils.appversion.appversion.AppVersion.name’ undocumented

**number** (`()`)

**Warning:** method ‘gui.utils.appversion.appversion.AppVersion.number’ undocumented

**organizationDomain** (`()`)

**Warning:** method ‘gui.utils.appversion.appversion.AppVersion.organizationDomain’ undocumented

**organizationName** (`()`)

**Warning:** method ‘gui.utils.appversion.appversion.AppVersion.organizationName’ undocumented

**static** `updateFile(module, newversion)`

Updates the version string within a given module. Replaces the version string in the source code textually. Assumes there is an AppVersion constructor call in the module.

### mcsas.gui.utils.appversion.qappversion module

**class** `QAppVersion(*args, **kwargs)`

Bases: `gui.utils.appversion.appversion.AppVersion`

Set QApplication properties based on version meta data.

**settingsKey** (`()`)

Version dependent settings key.

### Module contents

### Submodules



**mcsas.gui.utils.dialoginteraction module****class DialogInteraction** (*config*)Bases: `QThread`

Simulates user interaction on blocking widgets.

Calls a specified slot on all widgets of a given type after a given delay. Please see tests in other classes for examples.

*config*: A dict containing argument/value pairs. For valid arguments, please see `_propertyNames`.**classmethod application** ()**Warning:** method 'gui.utils.dialoginteraction.DialogInteraction.application' undocumented**classmethod instance** (*instanceType*, \*args, \*\*kwargs)**Warning:** method 'gui.utils.dialoginteraction.DialogInteraction.instance' undocumented**classmethod query** (*queryType*, *blockingFunc*, \*args, \*\*kwargs)**Warning:** method 'gui.utils.dialoginteraction.DialogInteraction.query' undocumented**classmethod queryInstance** (*instanceType*, \*args, \*\*kwargs)**Warning:** method 'gui.utils.dialoginteraction.DialogInteraction.queryInstance' undocumented**run** ()**Warning:** method 'gui.utils.dialoginteraction.DialogInteraction.run' undocumented**mcsas.gui.utils.displayexception module****class DisplayException** (*exception*, *level*='critical', *fmt*=None)Bases: `QMessageBox`Displays a message box with the text of a provided exception. *level*: one of 'question', 'info', 'warning', 'critical'

```
>>> from utilsgui import DisplayException, DialogInteraction
>>> from utils import AppError
>>> de = DialogInteraction.queryInstance(DisplayException,
```

```

...                               AppError("test"),
...                               button = 'default')
>>> type(de).__name__
'DisplayException'
>>> de.result() == 0
True

```

**classmethod classAndMethodName()**

Retrieves some meta info to describe the error further.

### mcsas.gui.utils.filedialog module

File dialogs and convenience functions.

**fileDialog** (*parent, labeltext, path, directory=False, readOnly=True*)

Opens a dialog to select one or more files for reading.

Alternative to native file dialogs.

**fileDialogType** ()

**Warning:** function 'gui.utils.filedialog.fileDialogType' undocumented

**getOpenFiles** (*parent, labeltext, path, filefilter=None, multiple=True*)

**Warning:** function 'gui.utils.filedialog.getOpenFiles' undocumented

**getSaveDirectory** (*parent, labeltext, path*)

**Warning:** function 'gui.utils.filedialog.getSaveDirectory' undocumented

**getSaveFile** (*parent, labeltext, path, filefilter*)

**Warning:** function 'gui.utils.filedialog.getSaveFile' undocumented

**makeFilter** (*filterList*)

**Warning:** function 'gui.utils.filedialog.makeFilter' undocumented

### mcsas.gui.utils.progressdialog module

**class ProgressDialog** (*parent=None, title=None, count=0*)

Bases: `QProgressDialog`

A progress dialog to visualize calculation status of the program.

Tests the ProgressDialog with 10 steps, cancels in between. `>>> import time >>> from utilsgui import ProgressDialog, DialogInteraction >>> pd = DialogInteraction.queryInstance(ProgressDialog, ... slot = 'cancel', ... count = 11) >>> for i in range(0, pd.maximum()): ... dummy = pd.update() ... time.sleep(0.2) >>> pd.wasCanceled()` True

Tests again, without cancel this time. `>>> pd.reset() >>> for i in range(0, pd.maximum()): ... dummy = pd.update() ... time.sleep(0.2) >>> pd.wasCanceled()` False

**update** ()

Updates progress status and returns True if canceled, False otherwise.

### mcsas.gui.utils.signal module

**tryDisconnect** (*sig, slot*)

Tries to disconnect signal *sig* from slot *slot*.

### mcsas.gui.utils.translate module

**tr** (*s*)

**Warning:** function 'gui.utils.translate.tr' undocumented

### Module contents

**processEventLoop** ()

**Warning:** function 'gui.utils.processEventLoop' undocumented

## Submodules

### mcsas.gui.algorithmwidget module

**class AlgorithmWidget** (*parent, algorithm, appSettings*)

Bases: `gui.bases.settingswidget.SettingsWidget`, `gui.bases.mixins.appsettings.AppSettings`

**algorithm**

Retrieves AlgorithmBase object containing all parameters for this settings.

**blockSigValueChanged()**

**Warning:** method 'gui.algorithmwidget.AlgorithmWidget.blockSigValueChanged' undocumented

**static clearLayout** (*layout*, *newParent=None*)

Removes all widgets from the given layout and reparents them if *newParent* is a sub class of QWidget

**inputWidgets**

Returns all existing input names (for store/restore).

**keys**

**makeSetting** (*param*, *activeBtns=False*)

Creates an input widget for the provided Parameter and configures it appropriately.

**makeWidgets** (\**args*)

**Warning:** method 'gui.algorithmwidget.AlgorithmWidget.makeWidgets' undocumented

**onBackendUpdate()**

**Warning:** method 'gui.algorithmwidget.AlgorithmWidget.onBackendUpdate' undocumented

**static removeWidgets** (*widget*)

Removes all widgets from the layout of the given widget.

**resizeEvent** (*resizeEvent*)

Resizes widget based on available width.

**resizeWidgets** (*targetWidth*)

**Warning:** method 'gui.algorithmwidget.AlgorithmWidget.resizeWidgets' undocumented

**restoreSession** (*section=None*)

**Warning:** method 'gui.algorithmwidget.AlgorithmWidget.restoreSession' undocumented

**sigBackendUpdated**

**storeSession** (*section=None*)

Stores current UI configuration to persistent application settings.

**uiWidgets**

May return a list of input widgets compatible but not associated to a parameter, e.g. for UI configuration.  
To be overridden in subclasses.

**unlockSigValueChanged()**

**Warning:** method 'gui.algorithmwidget.AlgorithmWidget.unlockSigValueChanged' undocumented

**updateAll()**

Called in MainWindow on calculation start.

**updateParam** (*param*, *emitBackendUpdated=True*)

Write UI settings back to the algorithm. Processes all input widgets which belong to a certain parameter.

**updateUi()**

Update input widgets according to possibly changed backend data.

**updateWidget** (*widget*, *emitBackendUpdated=True*)

Write UI settings back to the algorithm. Gets the parameter associated with a certain widget and processes all related widgets as well.

**class SettingsGridWidget** (*parent*, *algorithm*, *appSettings=None*, *showParams=None*)

Bases: *gui.algorithmwidget.AlgorithmWidget*

Base class for displaying simple input boxes of various settings arranged on a grid dynamically based on the width of the window.

Additional arguments: *showParams* is a list of parameter names to show in this widget. If not specified it shows all available parameters by default.

**resizeWidgets** (*targetWidth*)

Creates a new layout with appropriate row/column count.

**isNotNone** (*lst*)

**Warning:** function 'gui.algorithmwidget.isNotNone' undocumented

**rearrangeWidgets** (*layout*, *widgets*, *targetWidth*)

**Warning:** function 'gui.algorithmwidget.rearrangeWidgets' undocumented

### mcsas.gui.calc module

**class Calculator**

Bases: *utils.hdf.HDFMixin*

**algo**

**hdfLoad** (*filehandle*)

load a calculator configuration

**hdfWrite** (*hdf*)

write a calculator configuration.

**indent** = ''

**isStopped()**

**Warning:** method 'gui.calc.Calculator.isStopped' undocumented

**model**

**modelActiveParams()**

**Warning:** method 'gui.calc.Calculator.modelActiveParams' undocumented

**modelParams()**

**Warning:** method 'gui.calc.Calculator.modelParams' undocumented

**nolog = False**

**postProcess()**

**Warning:** method 'gui.calc.Calculator.postProcess' undocumented

**prepare()**

Resets series data. Supposed to be called before each run of multiple `__call__()` invocations.

**stop()**

**Warning:** method 'gui.calc.Calculator.stop' undocumented

**class OutputFilename** (*dataset, createDir=True*)

Bases: `object`

Generates output filenames with a common time stamp and logs appropriate messages.

**basename**

**filename** (*kind=None, extension='.txt'*)

Creates a file name from data base name, its directory and the current timestamp. It's created once so that all output files have the same base name and timestamp.

**filenameVerbose** (*kind, descr, extension='.txt'*)

Returns the file name as in `filename()` and logs a descriptive message containing the full file name which is usually click-able.

**outDir**

**timestamp**

**cfgwrite** (*self*, *fp*)

Write an .ini-format representation of the configuration state.

### mcsas.gui.datawidget module

**class DataWidget** (*parent*, *appSettings*)

Bases: `QWidget`, `gui.bases.mixins.appsettings.AppSettings`

**buildUi** (*dataobj*)

**Warning:** method 'gui.datawidget.DataWidget.buildUi' undocumented

**clearUi** ()

**Warning:** method 'gui.datawidget.DataWidget.clearUi' undocumented

**makeConfigUi** (*config*)

**Warning:** method 'gui.datawidget.DataWidget.makeConfigUi' undocumented

**onBackendUpdate** ()

**Warning:** method 'gui.datawidget.DataWidget.onBackendUpdate' undocumented

**onDataSelected** (*dataobj*)

**Warning:** method 'gui.datawidget.DataWidget.onDataSelected' undocumented

**onEmptyDataList** ()

Forgets which data settings were already restored.

**restoreSession** ()

**Warning:** method 'gui.datawidget.DataWidget.restoreSession' undocumented

**sigConfig**

**storeSession** ()

**Warning:** method 'gui.datawidget.DataWidget.storeSession' undocumented

### mcsas.gui.filelist module

**class FileList** (*parent=None, title=None, withBtn=True, nestedItems=True*)

Bases: *gui.bases.datalist.DataList*

**configFromLast** ()

Get the data config of the last item in the list.

**itemDoubleClicked** (*item, column*)

**Warning:** method 'gui.filelist.FileList.itemDoubleClicked' undocumented

**loadData** (*fileList=None*)

**Warning:** method 'gui.filelist.FileList.loadData' undocumented

**setDataConfig** (*dataConfig*)

Propagates the given DataConfig to all DataObj in the list. Makes sure that all data sets have the same configuration finally. Disable this in order to have individual per-data-set configuration.

**setupUi** ()

**Warning:** method 'gui.filelist.FileList.setupUi' undocumented

**sigSphericalSizeRange**

### mcsas.gui.liststyle module

**makeAlternatingRowColorsTransparent** (*widget*)

**Warning:** function 'gui.liststyle.makeAlternatingRowColorsTransparent' undocumented

**setBackgroundStyleSheet** (*widget, imgpath*)

**Warning:** function 'gui.liststyle.setBackgroundStyleSheet' undocumented



**mcsas.gui.mainwindow module****class MainWindow** (*parent=None, args=None*)Bases: *gui.bases.mainwindow.mainwindow.MainWindow***calc** ()**Warning:** method 'gui.mainwindow.MainWindow.calc' undocumented**calculator**

Returns a calculator object.

**closeEvent** (*closeEvent*)**Warning:** method 'gui.mainwindow.MainWindow.closeEvent' undocumented**fileDialog** ()**Warning:** method 'gui.mainwindow.MainWindow.fileDialog' undocumented**initUi** ()**Warning:** method 'gui.mainwindow.MainWindow.initUi' undocumented**keyPressEvent** (*keyEvent*)**Warning:** method 'gui.mainwindow.MainWindow.keyPressEvent' undocumented**onCloseSignal****onStartStopClick** (*checked*)**Warning:** method 'gui.mainwindow.MainWindow.onStartStopClick' undocumented**restoreSettings** ()**Warning:** method 'gui.mainwindow.MainWindow.restoreSettings' undocumented

**setupUi** (\*args)

**Warning:** method 'gui.mainwindow.MainWindow.setupUi' undocumented

**storeSettings** ()

**Warning:** method 'gui.mainwindow.MainWindow.storeSettings' undocumented

**class ToolBox** (\*args, \*\*kwargs)

Bases: `QToolBox`

`QToolBox` containing the widgets for user settings. Used to propagate resize events to child widgets to enable responsive behaviour. On MacOS, fixes failed detection of size changes in child widget due to scroll area.

**resizeEvent** (event)

**Warning:** method 'gui.mainwindow.ToolBox.resizeEvent' undocumented

**eventLoop** (args)

Starts the UI event loop and get command line parser arguments.

### mcsas.gui.modelwidget module

**class ModelWidget** (parent, calculator, \*args)

Bases: `gui.algorithmwidget.AlgorithmWidget`

**algorithm**

**model**

**onDataSelected** (dataobj)

Gets the data which is currently selected in the UI and rebuilds the model selection box based on compatible models.

**restoreSession** (model=None)

Load last known user settings from persistent app settings.

**selectModel** (model)

model: string containing the name of the model to select. Calls `_selectModelSlot()` via signal.

**setSphericalSizeRange** (minVal, maxVal)

**Warning:** method 'gui.modelwidget.ModelWidget.setSphericalSizeRange' undocumented

**setStatsWidget** (statsWidget)

Sets the statistics widget to use for updating ranges.

**storeSession** (*section=None*)

**Warning:** method ‘gui.modelwidget.ModelWidget.storeSession’ undocumented

### mcsas.gui.optimizationwidget module

**class OptimizationWidget** (*\*args*)

Bases: *gui.algorithmwidget.AlgorithmWidget*

**onDataSelected** (*dataobj*)

Sets defaults for certain types of DataConfig selected, respectively fixes some values.

**resizeWidgets** (*targetWidth*)

Creates a new layout with appropriate row/column count.

**restoreSession** (*\*args, \*\*kwargs*)

**Warning:** method ‘gui.optimizationwidget.OptimizationWidget.restoreSession’ undocumented

**storeSession** (*\*args, \*\*kwargs*)

**Warning:** method ‘gui.optimizationwidget.OptimizationWidget.storeSession’ undocumented

**uiWidgets**

### mcsas.gui.qt module

### mcsas.gui.rangelist module

**class RangeDialog** (*parent=None, model=None*)

Bases: *QDialog*

Creates a modal dialog window to ask the user for a range to be added.

**output** ()

**Warning:** method ‘gui.rangelist.RangeDialog.output’ undocumented

**class RangeList** (*calculator=None, appSettings=None, \*\*kwargs*)

Bases: *gui.bases.datalist.DataList, gui.bases.mixins.appsettings.AppSettings*

**append** (*histList*)

**Warning:** method 'gui.rangelist.RangeList.append' undocumented

**editEntry** ()

**Warning:** method 'gui.rangelist.RangeList.editEntry' undocumented

**itemUpdate** (*item*, *column*)

**Warning:** method 'gui.rangelist.RangeList.itemUpdate' undocumented

**loadData** (*ranges=None*)

Overridden base class method for adding entries to the list.

**mouseDoubleClickEvent** (*event*)

Shows RangeDialog on double click.

**onRemoval** (*removedHistograms*)

**Warning:** method 'gui.rangelist.RangeList.onRemoval' undocumented

**recalc** ()

**Warning:** method 'gui.rangelist.RangeList.recalc' undocumented

**restoreSession** ()

Load last known user settings from persistent app settings.

**setupUi** ()

**Warning:** method 'gui.rangelist.RangeList.setupUi' undocumented

**storeSession** ()

**Warning:** method 'gui.rangelist.RangeList.storeSession' undocumented

**updateHistograms** ()

Called after UI update by sigBackendUpdated from an AlgorithmWidget.

**getItemIndex** (*comboBox*, *text*)

**Warning:** function ‘gui.rangelist.getItemIndex’ undocumented

### mcsas.gui.scientrybox module

class **SciEntryBox** (*parent=None*)

Bases: `QLineEdit`

**fmt**

**indicateCorrectness** (*isValid*)

**Warning:** method ‘gui.scientrybox.SciEntryBox.indicateCorrectness’ undocumented

**static numberFormat** (*decimals=None*)

**Warning:** method ‘gui.scientrybox.SciEntryBox.numberFormat’ undocumented

**setMaximum** (*value*)

Work around issues regarding round-off errors by the ‘g’ format type when comparing numbers from user input with calculation results.

**setMinimum** (*value*)

Work around issues regarding round-off errors by the ‘g’ format type when comparing numbers from user input with calculation results.

**setPrefix** (*value*)

**Warning:** method ‘gui.scientrybox.SciEntryBox.setPrefix’ undocumented

**setRange** (*lo, hi*)

**Warning:** method ‘gui.scientrybox.SciEntryBox.setRange’ undocumented

**setValue** (*value*)

**Warning:** method ‘gui.scientrybox.SciEntryBox.setValue’ undocumented

**toolTipFmt** = ‘A value between {lo} and {hi} (including).’

**classmethod updateToolTip** (*widget, decimals=None*)

**Warning:** method 'gui.scientrybox.SciEntryBox.updateToolTip' undocumented

**value** ()

**Warning:** method 'gui.scientrybox.SciEntryBox.value' undocumented

**class SciEntryValidator** (*parent=None*)

Bases: `QDoubleValidator`

Assumes the associated `QLineEdit` is provided as parent object in the constructor.

**fixup** (*input*)

Restricts the value to the valid range defined by `setTop()` and `setBottom()`. Limits the precision as well.

**validate** (*input, pos*)

**Warning:** method 'gui.scientrybox.SciEntryValidator.validate' undocumented

### mcsas.gui.settingsgroup module

**class AdvancedSettings** (*\*args, \*\*kwargs*)

Bases: `gui.settingsgroup.SettingsGroup`, `QGroupBox`

**showAdvanced** (*show=False*)

**Warning:** method 'gui.settingsgroup.AdvancedSettings.showAdvanced' undocumented

**updateWidgets** (*widget=None*)

**Warning:** method 'gui.settingsgroup.AdvancedSettings.updateWidgets' undocumented

**class DefaultSettings** (*\*args, \*\*kwargs*)

Bases: `gui.settingsgroup.SettingsGroup`, `QWidget`

**class SettingsGroup** (*\*args, \*\*kwargs*)

Bases: `object`

**rearrangeWidgets** (*targetWidth*)

**Warning:** method 'gui.settingsgroup.SettingsGroup.rearrangeWidgets' undocumented

## mcsas.gui.version module

## Module contents

## mcsas.log package

### Submodules

#### mcsas.log.log module

Interface and convenience methods for general logging.

**addHandler** (*handler*)

Set up a new handler and add it for logging.

**formatter** ()

Date and time format for logging, ISO 8601:2004

**removeHandler** (*handler*)

**Warning:** function 'log.log.removeHandler' undocumented

**replaceHandler** (*handler*)

**Warning:** function 'log.log.replaceHandler' undocumented

**replaceStdOutErr** (*sout=None, serr=None*)

Replaces stdout/err with calls to logging.info/error.

**timestamp** ()

Current local time in seconds.

**timestampFormat** ()

Format for current local time, suitable for file names. >>> timestampFormat() '%Y-%m-%d\_%H-%M-%S'

**timestampFormatted** (*ts=None*)

Current local time. >>> timestamp() == time.strftime("%Y-%m-%d\_%H-%M-%S") True

#### mcsas.log.sink module

Interface and convenience methods for general logging.

**class Sink**

Bases: object

**buf** = None

**flush** ()

**Warning:** method 'log.sink.Sink.flush' undocumented

**process** (*msg, func*)

**Warning:** method 'log.sink.Sink.process' undocumented

**class StdErrSink**

Bases: *log.sink.Sink*

**write** (*msg*)

**Warning:** method 'log.sink.StdErrSink.write' undocumented

**class StdOutSink**

Bases: *log.sink.Sink*

**write** (*msg*)

**Warning:** method 'log.sink.StdOutSink.write' undocumented

## mcsas.log.widgethandler module

**class WidgetHandler** (*widget*)

Bases: *logging.Handler*

A logging.Handler which appends messages to some widget.append().

```
>>> import logging
>>> from PyQt4.QtGui import QTextEdit
>>> from utilsgui import DialogInteraction
>>> from logdock import LogHandler
>>> pte = DialogInteraction.instance(QTextEdit)
>>> handler = LogHandler(pte)
>>> msg = "testtest"
>>> handler.emit(logging.LogRecord("", 0, "", 0, msg, (), None))
>>> str(pte.toPlainText()) == msg
True
```

**emit** (*record*)

**Warning:** method 'log.widgethandler.WidgetHandler.emit' undocumented

**widget**



## Module contents

### `getWidgetHandlers()`

Returns all active WidgetHandlers for logging.

## mcsas.utils package

### Submodules

#### mcsas.utils.binning module

- `binningArray()`: Can be used to do n-by-n pixel binning of 2D detector images. The returned uncertainty is the larger of either the binned uncertainty or the sample standard deviation in the bin.
- `binning1d()`: bins the data and propagates errors, or calculates errors if not initially provided
- `binningWeighted1d()`: Weighted binning, where the intensities of a pixel are divided between the two neighbouring bins depending on the distances to the centres. If error provided is empty, the standard deviation of the intensities in the bins are computed.

### `array(value)`

**Warning:** function 'utils.binning.array' undocumented

### `binning1d(q, intensity, error=None, numBins=200, stats='std')`

An unweighted binning routine. The intensities are sorted across bins of equal size. If provided error is empty, the standard deviation of the intensities in the bins are computed.

### `binningArray(q, psi, intensity, error, s=2)`

This function applies a simple s-by-s binning routine on images. It calculates new error based on old error superseded by standard deviation in a bin.

### `binningWeighted1d(q, intensity, error=None, numBins=200, stats='se')`

Implementation of the binning routine written in Matlab. The intensities are divided across the q-range in bins of equal size. The intensities of a pixel are divided between the two neighbouring bins depending on the distances to the centres. If error provided is empty, the standard deviation of the intensities in the bins are computed.

#### Usage:

```
qbin, ibin, ebin = binning_weighted_1d(q, intensity, error = [],
                                       numBins = 200, stats = 'se')
```

#### Optional input arguments:

**numBins:** integer indicating the number of bins to divide the intensity over. Alternatively, this can be an array of equidistant bin centres. If you go this route, depending on the range, not all intensity may be counted.

**stats:** Can be set to 'auto'. This takes the maximum error between supplied Poisson statistics error-based errors or the standard error.

Written by Brian R. Pauw, 2011, released under BSD open source license.

### mcsas.utils.classproperty module

#### class **classproperty**

Bases: `property`

Subclass `property` to make `classmethod` properties possible.

Use it like this:

```
@classproperty
@classmethod
def var(cls):
    <code>
```

Getters only, see <http://stackoverflow.com/questions/128573/using-property-on-classmethods>

### mcsas.utils.devtools module

Definitions used during development only. This module is supposed to not being required in a release package.

**DBG** (\*args, \*\*kwargs)

**Warning:** function 'utils.devtools.DBG' undocumented

**DBGF** (\*args)

**Warning:** function 'utils.devtools.DBGF' undocumented

### mcsas.utils.error module

Some Error classes.

**exception `AppError`** (msg='')

Bases: `Exception`

**classmethod `getMessage`** (msg='')

**Warning:** method 'utils.error.AppError.getMessage' undocumented

**msg = ''**

General error with descriptive message to be forwarded and shown to the user in a message box.

**exception `EmptySelection`** (msg='')

Bases: `utils.error.AppError`

**exception `FileError`** (msg, fn)

Bases: `utils.error.AppError`

**exception `InitError`** (msg='')

Bases: `utils.error.AppError`

**exception LoadError** (*msg*='')

Bases: *utils.error.AppError*

**exception VerboseError**

Bases: *utils.error.AppError*

Verbose Exception containing class and method where it was raised. A cache makes sure VerboseErrors with the same name can be matched.

**classmethod new** (*name*, *msg*='')

Creates an exception dynamically. Remembers previous ones for testing

**classmethod prefix** (*msg*)

Retrieves some meta info to describe the error further.

### **mcsas.utils.findmodels module**

For use in gui/modelwidget.py, to help find valid calculation models

**class FindModels** (*\*searchPaths*)

Bases: *object*

Finds all methods of type *ScatteringModel* in the subdirectories starting from *searchPath*. *searchPath* defaults to the root mcsas pwd + "models". returns a list of full paths, and a list of associated model names

**classmethod candidateFiles** (*\*searchPaths*)

**Warning:** method 'utils.findmodels.FindModels.candidateFiles' undocumented

**classmethod getSearchPaths** ()

**Warning:** method 'utils.findmodels.FindModels.getSearchPaths' undocumented

**classmethod libraryPath** ()

**Warning:** method 'utils.findmodels.FindModels.libraryPath' undocumented

**classmethod rootName** ()

**Warning:** method 'utils.findmodels.FindModels.rootName' undocumented

**classproperty** (*func*)

**Warning:** function 'utils.findmodels.classproperty' undocumented

**findFiles** (*searchPath, extension*)

generator for files ending in .py code from: <http://stackoverflow.com/questions/2186525>

**reorder** (*indata, priorityKeys*)

**Warning:** function ‘utils.findmodels.reorder’ undocumented

## mcsas.utils.hdf module

Helper functions for HDF5 functionality

**HDFCleanup** (*infile*)

Unused space is reclaimed by making a copy of the contents in the current hdf 5 file object, and moves the copy in place of the original. if the input argument supplied is a file name instead of an HDF5 object, the method returns nothing. Else, the method returns the new HDF5 object

**class HDFMixin**

Bases: object

**classmethod hdfLoad** ()

Restores an instance of this type from a given HDF file location or group.

**hdfStore** (*filename, rootLocation=None*)

Writes itself to an HDF file at the given position or group.

**hdfWrite** (*hdf*)

To be overridden by sub classes to store themselves in an HDF structure. *hdf*: a HDFWriter instance.

**class HDFWriter** (*hdfHandle, rootLocation=None*)

Bases: object

Represents an open HDF file location in memory and keeps track of the current address/name for reading or writing. Once this object loses scope, its data is actually written to file.

**location**

**log** (*msg*)

**Warning:** method ‘utils.hdf.HDFWriter.log’ undocumented

**classmethod open** (*filename, rootLocation=None*)

**Warning:** method ‘utils.hdf.HDFWriter.open’ undocumented

**writeAttribute** (*key, value*)

**Warning:** method ‘utils.hdf.HDFWriter.writeAttribute’ undocumented

**writeAttributes** (*\*\*kwargs*)

**Warning:** method 'utils.hdf.HDFWriter.writeAttributes' undocumented

**writeDataset** (*name, data*)

**Warning:** method 'utils.hdf.HDFWriter.writeDataset' undocumented

**writeMember** (*obj, memberName*)

**Warning:** method 'utils.hdf.HDFWriter.writeMember' undocumented

**writeMembers** (*obj, \*members*)

**Warning:** method 'utils.hdf.HDFWriter.writeMembers' undocumented

**getCallerInfo** (*referenceType=None, stackOffset=0*)

*referenceType*: Stop the search for a frame when this type for a local 'self' is found. *stackOffset*: grab that frame counted from the last instead of search

## mcsas.utils.lastpath module

General utilities without GUI dependencies.

**class LastPath**

Bases: object

Stores a file system path for use in file open dialogs.

How to test this platform independent? >>> from utils import LastPath, getHomeDir >>> LastPath.path == getHomeDir() True >>> LastPath.path = '.' >>> LastPath.path == '.' True

**classmethod get** ()

**Warning:** method 'utils.lastpath.LastPath.get' undocumented

**classmethod set** (*lastpath*)

Accepts a directory path or a file path. Determines the directory itself in the latter case.

**getHomeDir** ()

**Warning:** function ‘utils.lastpath.getHomeDir’ undocumented

### mcsas.utils.loadstore module

- `pickleLoad()`: Reads in pickled data from a file (by filename)
- `pickleStore()`: write a block or dictionary to a file (by filename)

**pickleLoad** (*filename*)

Loads data from a pickle file

**pickleStore** (*filename, somedata*)

Writes python object to a file.

### mcsas.utils.mixedmethod module

This module implements a `mixedmethod()` decorator for class definitions. Basic idea found here: <http://www.daniweb.com/software-development/python/code/406393/mixedmethod-mixes-class-and-instance-method-into-one>

Usually, a class method works for an instance only. If decorated with `@classmethod` it works for the class/type and for the instance but ignores all instance data and works with the class data only. Methods in a class decorated with `@mixedmethod` work with class/type and instances in the same way BUT if the underlying object is a class/type it works on the class/type data. If the underlying object is an instance it works on the instance data. Thus, for instances modifications apply to that individual instance only. For classes modifications apply on the class level which applies changes to all instances created from that.

**class mixedmethod** (*func*)

Bases: `object`

This decorator mutates a function defined in a class into a ‘mixed’ class and instance method. Usage:

```
class Spam:
    @mixedmethod
    def egg(selforcls, *args, **kwargs):
        # selforcls is the instance when called on an instance
        # selforcls is the class when called on the class
        pass
```

### mcsas.utils.parameter module

**FitParameter** (*\*args, \*\*kwargs*)

**Warning:** function ‘utils.parameter.FitParameter’ undocumented

**class FitParameterBase**

Bases: `bases.algorithm.parameter.ParameterBase`

Deriving parameters for curve fitting from `bases.algorithm.parameter` to introduce more specific fit related attributes.

**activeRange** = `functools.partial(<function FitParameterBase.activeRange>, <class ‘utils.parameter.FitParameterBase’>)`

**activeVal** = `functools.partial(<function FitParameterBase.activeVal>, <class ‘utils.parameter.FitParameterBase’>)`

```

activeValues = functools.partial(<function _makeGetter.<locals>.getter>, <class 'utils.parameter.FitParameterBase'>)
displayActiveRange = functools.partial(<function FitParameterBase.displayActiveRange>, <class 'utils.parameter.F
hdfStoreAsMember ()

```

**Warning:** method 'utils.parameter.FitParameterBase.hdfStoreAsMember' undocumented

```

histograms = functools.partial(<function _makeGetter.<locals>.getter>, <class 'utils.parameter.FitParameterBase'>)
isActive = functools.partial(<function _makeGetter.<locals>.getter>, <class 'utils.parameter.FitParameterBase'>)
setActive = functools.partial(<function FitParameterBase.setIsActive>, <class 'utils.parameter.FitParameterBase'>)
setActiveRange = functools.partial(<function FitParameterBase.setActiveRange>, <class 'utils.parameter.FitParameterBase'>)
setActiveVal = functools.partial(<function FitParameterBase.setActiveVal>, <class 'utils.parameter.FitParameterBase'>)
setActiveValues = functools.partial(<function _makeSetter.<locals>.setter>, <class 'utils.parameter.FitParameterBase'>)
setDisplayActiveRange = functools.partial(<function FitParameterBase.setDisplayActiveRange>, <class 'utils.parameter.FitParameterBase'>)
setHistograms = functools.partial(<function _makeSetter.<locals>.setter>, <class 'utils.parameter.FitParameterBase'>)
setIsActive = functools.partial(<function FitParameterBase.setIsActive>, <class 'utils.parameter.FitParameterBase'>)
setValueRange = functools.partial(<function FitParameterBase.setValueRange>, <class 'utils.parameter.FitParameterBase'>)

```

**class FitParameterBoolean**

Bases: *utils.parameter.FitParameterBase, bases.algorithm.parameter.ParameterBoolean*

**class FitParameterFloat**

Bases: *utils.parameter.FitParameterNumerical, bases.algorithm.parameter.ParameterFloat*

**class FitParameterLog**

Bases: *utils.parameter.FitParameterBase, bases.algorithm.parameter.ParameterLog*

**class FitParameterNumerical**

Bases: *utils.parameter.FitParameterBase, bases.algorithm.parameter.ParameterNumerical*

**generate** (*lower=None, upper=None, count=1*)

**Warning:** method 'utils.parameter.FitParameterNumerical.generate' undocumented

**class FitParameterString**

Bases: *utils.parameter.FitParameterBase, bases.algorithm.parameter.ParameterString*

**class Histogram** (*param, lower, upper, binCount=50, xscale=None, yweight=None, autoFollow=True*)

Bases: *bases.dataset.dataset.DataSet, bases.dataset.dataset.DisplayMixin*

Stores histogram related settings of a parameter. The results too, eventually(?). yes, please. Stores&calculates rangeInfo() results for all available weighting options.

Creates an histogram with default bin count, will be updated later.

**autoFollow****binCount****bins****calc** (*contribs, paramIndex, fractions*)**Warning:** method 'utils.parameter.Histogram.calc' undocumented**cdf****classmethod displayData** ()

Properties used for UI display.

**classmethod displayDataDescr** ()

Descriptive text of fields for UI display.

**hdfWrite** (*hdf*)**Warning:** method 'utils.parameter.Histogram.hdfWrite' undocumented**classmethod integralProps** ()

All properties needed to properly serialize and restore this histogram. Same order expected by the constructor.

**lower****lowerDisplay**

Lower limit in display units including the unit text.

**moments****observability****param****paramName****updateRange** ()

Restricts histogram range according to changed parameter range if needed. Checks histogram range against parameter limits.

**upper****upperDisplay**

Upper limit in display units including the unit text.

**xLowerEdge****xMean****xWidth****xrange****xscale**



**static xscaling** (*index=None*)

**Warning:** method 'utils.parameter.Histogram.xscaling' undocumented

**yweight**

**static yweighting** (*index=None*)

**Warning:** method 'utils.parameter.Histogram.yweighting' undocumented

**class Histograms**

Bases: list

Manages a set of user configured histograms for evaluation after monte-carlo run.

**append** (*value*)

**Warning:** method 'utils.parameter.Histograms.append' undocumented

**calc** (*\*args*)

**Warning:** method 'utils.parameter.Histograms.calc' undocumented

**clear** ()

**Warning:** method 'utils.parameter.Histograms.clear' undocumented

**hdfWrite** (*hdf*)

**Warning:** method 'utils.parameter.Histograms.hdfWrite' undocumented

**updateRanges** ()

Updates ranges of all histograms.

**class Moments** (*contribs, paramIndex, valueRange, fraction, algo=None*)

Bases: object

**static fieldNames** ()

Returns the field names in the same order as str()

**fields**

Tuple of member data incl. uncertainty for export.

```

intensity
kurtosis
mean
skew
total
varName = '_kurtosis'
variance

```

**Parameter** (*\*args, \*\*kwargs*)

**Warning:** function 'utils.parameter.Parameter' undocumented

**class VectorResult** (*vecResult*)

Bases: object

Stores multiple populations of a single result data vector. Calculates statistics at initialization.

```

full
mean
std

```

**classproperty** (*func*)

**Warning:** function 'utils.parameter.classproperty' undocumented

**isActiveFitParam** (*param*)

Checks any type of parameter for activeness. Shorter than that below or a try/except clause.

**isFitParam** (*param*)

**Warning:** function 'utils.parameter.isFitParam' undocumented

### mcsas.utils.pickleinstancemethods module

### mcsas.utils.propertynames module

**class PropertyNames**

Bases: object

**classmethod propNames** ()

**Warning:** method 'utils.propertynames.PropertyNames.propNames' undocumented

**classmethod `properties()`**

Returns all attributes configured in this class.

### **mcsas.utils.tests module**

Utils for testing something.

**assertName** (*newName, errorType, noWhitespace=False*)

**Warning:** function 'utils.tests.assertName' undocumented

**isCallable** (*obj*)

**Warning:** function 'utils.tests.isCallable' undocumented

**isFrozen** ()

**Warning:** function 'utils.tests.isFrozen' undocumented

**isInteger** (*obj*)

**Warning:** function 'utils.tests.isInteger' undocumented

**isLinux** ()

**Warning:** function 'utils.tests.isLinux' undocumented

**isList** (*obj*)

**Warning:** function 'utils.tests.isList' undocumented

**isMac** ()

**Warning:** function 'utils.tests.isMac' undocumented

**isMap** (*obj*)

**Warning:** function ‘utils.tests.isMap’ undocumented

**isEmptyString** (*obj*)

**Warning:** function ‘utils.tests.isEmptyString’ undocumented

**isNumber** (*obj*)

**Warning:** function ‘utils.tests.isNumber’ undocumented

**isSet** (*obj*)

**Warning:** function ‘utils.tests.isSet’ undocumented

**isString** (*obj*)

**Warning:** function ‘utils.tests.isString’ undocumented

**isWindows** ()

**Warning:** function ‘utils.tests.isWindows’ undocumented

**testfor** (*condition, exception, errorMessage=*‘‘)

**Warning:** function ‘utils.tests.testfor’ undocumented

### mcsas.utils.units module

Defines methods for using and manipulating units of variables. Some default magnitude-name dictionaries are provided, but the user can supply their own dictionary if required. Default unit to translate to must be set. Required keyword arguments:

- **magnitudedict**: a dictionary of magnitude - name pairs. Names must be unicode strings.
- **simagnitudename**: the si magnitude name.

Example usage:

```
>>> rUnit = Length("nm")
>>> rUnit.siMagnitudeName
u'm'
>>> rUnit.displayMagnitudeName
u'nm'
>>> rUnit.magnitudeConversion
1e-09
```

or:

```
>>> rUnit.toSi(32)
3.2e-08
```

Selecting a default:

```
>>> qUnit = ScatteringVector(u"cm-1")
>>> qUnit.magnitudeConversion
100.0
```

**class Angle** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class Area** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class DynamicViscosity** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class Fraction** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class Length** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class NoUnit** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class SLD** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class ScatteringIntensity** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class ScatteringVector** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class Temperature** (*magnitudeName=None*)

Bases: *utils.units.Unit*

test case for special conversions. Done by redefining toSI and toDisplay. Implemented units are given in \_magnitudeMap.

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**magnitudeConversion**

**toDisplay** (*value*)

**Warning:** method 'utils.units.Temperature.toDisplay' undocumented

**toSi** (*value*)

**Warning:** method 'utils.units.Temperature.toSi' undocumented

**class Time** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**class Unit** (*magnitudeName=None*)

Bases: *object*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

**availableMagnitudeNames**

**displayMagnitude**

**displayMagnitudeName**

**hdfWrite** (*hdf*)

**Warning:** method 'utils.units.Unit.hdfWrite' undocumented

**static invName** (*unitString*)

Adds an  $^{-1}$  sign or removes it if already present

**classmethod magnitude** (*name*)

Returns a (numerical) magnitude matching a magnitude name

**magnitudeConversion**

Scaling factor to move from display magnitude to si units. Required display argument:

*displaymagnitudename* : The name of the magnitude to convert from

Optional display argument:

*simagnitudename* [The name of the magnitude to convert to.] Defaults to self.siMagnitudeName

**Returns:** *float* : A scaling factor for display unit to scale to si unit.

**magnitudeMapping** = {}

**classmethod name** ()

**Warning:** method 'utils.units.Unit.name' undocumented

**siMagnitude** = ''

**siMagnitudeName** = ''

**toDisplay** (*value*)

**Warning:** method 'utils.units.Unit.toDisplay' undocumented

**toSi** (*value*)

**Warning:** method 'utils.units.Unit.toSi' undocumented

**class Volume** (*magnitudeName=None*)

Bases: *utils.units.Unit*

Set up a unit of measurement. The provided magnitude name is used by default in the UI. Using SI units if unspecified.

## Module contents

**classname** (*obj*)

**Warning:** function 'utils.classname' undocumented

**classproperty** (*func*)

**Warning:** function ‘utils.classproperty’ undocumented

**clip** (*value, minv, maxv*)

Expects a range tuple or list consisting of lower and upper limits.

**fixFilename** (*filename*)

Works around Windows file path length limitation of 260 chars.

**hashNumpyArray** (*arr*)

**Warning:** function ‘utils.hashNumpyArray’ undocumented

**mcopen** (*fn, mode, encoding='utf8'*)

**Warning:** function ‘utils.mcopen’ undocumented

## mcsas.datafile package

### Submodules

#### mcsas.datafile.arrayfile module

class **ArrayFile** (*filename, \*\*kwargs*)

Bases: `datafile.asciifile.AsciiFile`

A data file containing a single array of data, mostly.

classmethod **fileFilter** ()

**Warning:** method ‘datafile.arrayfile.ArrayFile.fileFilter’ undocumented

**getDataObj** ()

**Warning:** method ‘datafile.arrayfile.ArrayFile.getDataObj’ undocumented

**parseLines** (*asciiLines, \*\*kwargs*)

Parses lines of an ASCII file in order to extract a single array of numbers. Reimplement this in subclasses for different behaviour.

**rawArray**



**classproperty** (*func*)

**Warning:** function 'datafile.arrayfile.classproperty' undocumented

**np\_array** (*value*)

**Warning:** function 'datafile.arrayfile.np\_array' undocumented

### mcsas.datafile.asciifile module

**class AsciiFile** (*filename, \*\*kwargs*)

Bases: *datafile.datafile.DataFile*

A generic ascii data file.

**classmethod appendFile** (*filename, data, \*\*kwargs*)

like writeFile but appends data to an existing file

**classmethod appendHeaderLine** (*filename, header*)

writes a single-line header to a file consisting of a string or tuple of strings to be joined

**classmethod formatData** (*data, \*\*kwargs*)

**Warning:** method 'datafile.asciifile.AsciiFile.formatData' undocumented

**classmethod formatRow** (*row, \*\*kwargs*)

**Warning:** method 'datafile.asciifile.AsciiFile.formatRow' undocumented

**classmethod formatValue** (*value*)

**Warning:** method 'datafile.asciifile.AsciiFile.formatValue' undocumented

**newline** = '\n'

**parseLines** (*asciiLines, \*\*kwargs*)

Parses lines of an ASCII file in order to extract a single array of numbers. Reimplement this in subclasses for different behaviour.

**readArray** (*asciiLines, dataType=<class 'float'>, startLine=0, endLine=None, \*\*kwargs*)

Reads a numpy.array from a specified segment (startLine, endLine) of a line buffer given by asciiLines. Stops at lines incompatible to previous lines read due to different number of fields or incompatible data type. Returns the last line successfully parsed and the populated numpy.array.

**readFile** (*\*\*kwargs*)

**Warning:** method 'datafile.asciifile.AsciiFile.readFile' undocumented

**readTuple** (*fields, dataType=<class 'float'>, \*\*kwargs*)

Converts each field to the requested datatype. Raises an error if it is incompatible, the line is skipped in that case.

**separator** = ' '

**valueFormat** = '{0: 14.6E}'

**classmethod writeFile** (*filename, data, \*\*kwargs*)

**Warning:** method 'datafile.asciifile.AsciiFile.writeFile' undocumented

**classmethod writeHeaderLine** (*filename, header*)

writes a single-line header to a file consisting of a string or tuple of strings to be joined

**np\_array** (*value*)

**Warning:** function 'datafile.asciifile.np\_array' undocumented

### mcsas.datafile.datafile module

**class DataFile** (*filename, \*\*kwargs*)

Bases: object

Base class for handling data files. Can be initialized with a file (name) to read or with a data array.

Test error behaviour >>> from utils import DataFile, getTempFileName, getTempFile, FileNotFound >>> fn = getTempFileName() >>> try: DataFile.checkFilename(fn) ... except FileNotFound, e: str(e).find(fn) > 0 True

Prepare test data file >>> fd = getTempFile() >>> l = ['123 234n', '1,23 43.4rn', '2.3; 34,4n', ... '21.2 42 2n', '23,2 3.4 n'] >>> fd.writelines(l) >>> fd.close()

Test data parsing >>> df = DataFile() >>> df.loadFile(fd.name) [('123', '234'), ('1.23', '43.4'), ('2.3', '34.4'), ('21.2', '42', '2'), ('23.2', '3.4')]

Remove test data finally >>> import os >>> if os.path.isfile(fd.name): ... os.remove(fd.name)

**classmethod extensions** ()

**Warning:** method 'datafile.datafile.DataFile.extensions' undocumented

**fileFilter**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: def f(cls, arg1, arg2, ...): ... f = classmethod(f)
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

**filename**

Absolute path name of this file.

**getDataObj()**

Creates and returns the appropriate DataObj instance for this file type.

**name**

The plain name of the file with path and extension stripped.

**readFile(\*\*kwargs)**

Gets a proper file name and returns file data. May modify the instance. To be reimplemented.

**static sanitizeReadFilename(filename)**

Checks provided filename for plausibility and updates LastPath.

**classmethod sanitizeWriteFilename(filename)**

Checks and sets the file name to write to.

**setFilename(filename)**

Checks provided filename for plausibility and updates LastPath

**write(filename, \*\*kwargs)**

**Warning:** method 'datafile.datafile.DataFile.write' undocumented

**classmethod writeData(filename, data, \*\*kwargs)**

Convenience method to write data directly to file.

**classmethod writeFile(filename, data, \*\*kwargs)**

Gets a proper file name and numpy array and writes it to file. Reimplement this.

**classproperty(func)**

**Warning:** function 'datafile.datafile.classproperty' undocumented

### mcsas.datafile.nxcansasfile module

**class NXSHheader(dataCount, description=None)**

Bases: object

**line(index)**

Returns the specified line of the header as string.

**classmethod** `maxLines()`

**Warning:** method 'datafile.nxcansasfile.NXSHeader.maxLines' undocumented

**class** `NXcanSASFile(filename, **kwargs)`

Bases: `datafile.datafile.DataFile`

A NXcanSAS file, which is a NeXus-conform HDF5 file for storing corrected SAS data.

**dataRoot**

**classmethod** `fileFilter()`

**Warning:** method 'datafile.nxcansasfile.NXcanSASFile.fileFilter' undocumented

**classmethod** `getDataObj()`

**Warning:** method 'datafile.nxcansasfile.NXcanSASFile.getDataObj' undocumented

**rawArray**

**readFile** (*\*\*kwargs*)

**Warning:** method 'datafile.nxcansasfile.NXcanSASFile.readFile' undocumented

**readItem** (*element*)

**Warning:** method 'datafile.nxcansasfile.NXcanSASFile.readItem' undocumented

**classproperty** (*func*)

**Warning:** function 'datafile.nxcansasfile.classproperty' undocumented

### **mcsas.datafile.pdhfile module**

**class** `PDHFile(filename, **kwargs)`

Bases: `datafile.arrayfile.ArrayFile`

**classmethod** `fileFilter()`

**Warning:** method 'datafile.pdhfile.PDHFile.fileFilter' undocumented

**classmethod** **formatData** (*data*, *description=None*)

**Warning:** method 'datafile.pdhfile.PDHFile.formatData' undocumented

**parseLines** (*asciiLines*, *\*\*kwargs*)

**Warning:** method 'datafile.pdhfile.PDHFile.parseLines' undocumented

**class** **PDHHeader** (*dataCount*, *description=None*)

Bases: `object`

**line** (*index*)

Returns the specified line of the header as string.

**classmethod** **maxLines** ()

**Warning:** method 'datafile.pdhfile.PDHHeader.maxLines' undocumented

**classproperty** (*func*)

**Warning:** function 'datafile.pdhfile.classproperty' undocumented

## Module contents

**class** **DataFile** (*filename*, *\*\*kwargs*)

Bases: `object`

Base class for handling data files. Can be initialized with a file (name) to read or with a data array.

Test error behaviour >>> from utils import DataFile, getTempFileName, getTempFile, FileNotFound >>> fn = getTempFileName() >>> try: DataFile.checkFilename(fn) ... except FileNotFound, e: str(e).find(fn) > 0 True

Prepare test data file >>> fd = getTempFile() >>> l = ['123 234n', '1,23 43.4rn', '2.3; 34,4n', ... '21.2 42 2n', '23,2 3.4 n'] >>> fd.writelines(l) >>> fd.close()

Test data parsing >>> df = DataFile() >>> df.loadFile(fd.name) [(('123', '234'), ('1.23', '43.4'), ('2.3', '34.4'), ('21.2', '42', '2'), ('23.2', '3.4'))]

Remove test data finally >>> import os >>> if os.path.isfile(fd.name): ... os.remove(fd.name)

**classmethod** **extensions** ()

**Warning:** method 'datafile.DataFile.extensions' undocumented

### **fileFilter**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: def f(cls, arg1, arg2, ...): ... f = classmethod(f)
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

### **filename**

Absolute path name of this file.

### **getDataObj()**

Creates and returns the appropriate DataObj instance for this file type.

### **name**

The plain name of the file with path and extension stripped.

### **readFile(\*\*kwargs)**

Gets a proper file name and returns file data. May modify the instance. To be reimplemented.

### **static sanitizeReadFilename(filename)**

Checks provided filename for plausibility and updates LastPath.

### **classmethod sanitizeWriteFilename(filename)**

Checks and sets the file name to write to.

### **setFilename(filename)**

Checks provided filename for plausibility and updates LastPath

### **write(filename, \*\*kwargs)**

**Warning:** method 'datafile.DataFile.write' undocumented

### **classmethod writeData(filename, data, \*\*kwargs)**

Convenience method to write data directly to file.

### **classmethod writeFile(filename, data, \*\*kwargs)**

Gets a proper file name and numpy array and writes it to file. Reimplement this.

### **class AsciiFile(filename, \*\*kwargs)**

Bases: `datafile.datafile.DataFile`

A generic ascii data file.

### **classmethod appendFile(filename, data, \*\*kwargs)**

like writeFile but appends data to an existing file

**classmethod** `appendHeaderLine` (*filename*, *header*)

writes a single-line header to a file consisting of a string or tuple of strings to be joined

**classmethod** `formatData` (*data*, *\*\*kwargs*)

**Warning:** method 'datafile.AsciiFile.formatData' undocumented

**classmethod** `formatRow` (*row*, *\*\*kwargs*)

**Warning:** method 'datafile.AsciiFile.formatRow' undocumented

**classmethod** `formatValue` (*value*)

**Warning:** method 'datafile.AsciiFile.formatValue' undocumented

**newline** = '\n'

**parseLines** (*asciiLines*, *\*\*kwargs*)

Parses lines of an ASCII file in order to extract a single array of numbers. Reimplement this in subclasses for different behaviour.

**readArray** (*asciiLines*, *dataType*=<class 'float'>, *startLine*=0, *endLine*=None, *\*\*kwargs*)

Reads a numpy.array from a specified segment (*startLine*, *endLine*) of a line buffer given by *asciiLines*. Stops at lines incompatible to previous lines read due to different number of fields or incompatible data type. Returns the last line successfully parsed and the populated numpy.array.

**readFile** (*\*\*kwargs*)

**Warning:** method 'datafile.AsciiFile.readFile' undocumented

**readTuple** (*fields*, *dataType*=<class 'float'>, *\*\*kwargs*)

Converts each field to the requested datatype. Raises an error if it is incompatible, the line is skipped in that case.

**separator** = ','

**valueFormat** = '{0: 14.6E}'

**classmethod** `writeFile` (*filename*, *data*, *\*\*kwargs*)

**Warning:** method 'datafile.AsciiFile.writeFile' undocumented

**classmethod** `writeHeaderLine` (*filename*, *header*)

writes a single-line header to a file consisting of a string or tuple of strings to be joined

**class** **ArrayFile** (*filename*, *\*\*kwargs*)

Bases: *datafile.asciifile.AsciiFile*

A data file containing a single array of data, mostly.

**classmethod** **fileFilter** ()

**Warning:** method 'datafile.ArrayFile.fileFilter' undocumented

**getDataObj** ()

**Warning:** method 'datafile.ArrayFile.getDataObj' undocumented

**parseLines** (*asciiLines*, *\*\*kwargs*)

Parses lines of an ASCII file in order to extract a single array of numbers. Reimplement this in subclasses for different behaviour.

**rawArray**

**class** **PDHFile** (*filename*, *\*\*kwargs*)

Bases: *datafile.arrayfile.ArrayFile*

**classmethod** **fileFilter** ()

**Warning:** method 'datafile.PDHFile.fileFilter' undocumented

**classmethod** **formatData** (*data*, *description=None*)

**Warning:** method 'datafile.PDHFile.formatData' undocumented

**parseLines** (*asciiLines*, *\*\*kwargs*)

**Warning:** method 'datafile.PDHFile.parseLines' undocumented

**class** **PDHHeader** (*dataCount*, *description=None*)

Bases: *object*

**line** (*index*)

Returns the specified line of the header as string.

**classmethod** **maxLines** ()



**Warning:** method ‘datafile.PDHHeader.maxLines’ undocumented

**getFileFilter()**

Returns the file filter text of all available data file formats which can be used with a file selection dialog UI.

## mcsas.dataobj package

### Submodules

#### mcsas.dataobj.dataconfig module

**class CallbackRegistry**

Bases: object

**callback** (*what*, \**args*, \*\**kwargs*)

**Warning:** method ‘dataobj.dataconfig.CallbackRegistry.callback’ undocumented

**callbackSlots**

**register** (*what*, \**func*)

**Warning:** method ‘dataobj.dataconfig.CallbackRegistry.register’ undocumented

**class DataConfig**

Bases: *bases.algorithm.algorithmbase.AlgorithmBase*, *dataobj.dataconfig.CallbackRegistry*

**callbackSlots**

**hdfWrite** (*hdf*)

**Warning:** method ‘dataobj.dataconfig.DataConfig.hdfWrite’ undocumented

**is2d**

**onUpdatedX0** (*x0*)

Sets available range of loaded data.

**onUpdatedX1** (*x1*)

**Warning:** method ‘dataobj.dataconfig.DataConfig.onUpdatedX1’ undocumented

**parameters** = (None, None, None, None, None, None, None, None)

**sampleName****showParams****updateFMasks ()****Warning:** method 'dataobj.dataconfig.DataConfig.updateFMasks' undocumented**updateFuMin ()****Warning:** method 'dataobj.dataconfig.DataConfig.updateFuMin' undocumented**updateX0Limits ()****Warning:** method 'dataobj.dataconfig.DataConfig.updateX0Limits' undocumented**updateX0Unit (newUnit)**

Sets the unit of the x0 vector.

**updateX1Limits ()****Warning:** method 'dataobj.dataconfig.DataConfig.updateX1Limits' undocumented**updateX1Unit (newUnit)****Warning:** method 'dataobj.dataconfig.DataConfig.updateX1Unit' undocumented**Parameter (\*args, \*\*kwargs)****Warning:** function 'dataobj.dataconfig.Parameter' undocumented**funcNotInFuncList (f, flst)**

Custom predicate for comparing bounded methods: Duplicate only if instance ID and method name match.

**mcsas.dataobj.dataconfig\_test module****assertDefaults (dc)**

**Warning:** function ‘dataobj.dataconfig\_test.assertDefaults’ undocumented

**testCallbacks** ()

**Warning:** function ‘dataobj.dataconfig\_test.testCallbacks’ undocumented

**testLimits** ()

**Warning:** function ‘dataobj.dataconfig\_test.testLimits’ undocumented

**testSerialize** ()

**Warning:** function ‘dataobj.dataconfig\_test.testSerialize’ undocumented

**testSetter** ()

**Warning:** function ‘dataobj.dataconfig\_test.testSetter’ undocumented

### mcsas.dataobj.dataobj module

Represents input data associated with a measurement.

**class DataObj** (*\*\*kwargs*)

Bases: `abc.NewBase`

General container for data loaded from file. It offers specialised methods to derive information from the provided data.

**accumulate** (*others*)

**Warning:** method ‘dataobj.dataobj.DataObj.accumulate’ undocumented

**config**

**configType**

Returns a compatible DataConfig type.

**count**

**f**

The measurement vector.

**filename**

**hasUncertainties**

Returns True if this data set has an error bar for its intensities.

**hdfWrite** (*hdf*)

**Warning:** method 'dataobj.dataobj.DataObj.hdfWrite' undocumented

**initConfig** ()

Initializes a new data configuration and sets the sample name which is used to differentiate different data objects of the same type later on.

**is2d**

Returns true if this dataset contains two-dimensional data with psi information available.

**modelType**

Returns a compatible ScatteringModel type.

**sampleName****seriesKey**

The Name of the DataObj property to use as series key, hard-coded for now, assuming it exists. It allows to let the user chose from a generated list of properties (todo).

**seriesKeyName**

Returns the docstring of the property defined by self.seriesKeyProp.

**seriesKeyValue**

Returns the value of the property defined by self.seriesKeyProp.

**setConfig** (*config=None*)

Set the configuration of this data object if the type matches.

**setFilename** (*fn*)

Stores the absolute path to this data file. Should be reviewed when data sets can be created from several files.

**classmethod sourceName** ()

Returns the name of the measurement method.

**updateConfig** ()

Updates the config object based on this data set. All callbacks are run right after this method in setConfig().

**x0**

First sampling vector.

**x1**

Second sampling vector.

**x2**

Third sampling vector.

**classproperty** (*func*)

**Warning:** function 'dataobj.dataobj.classproperty' undocumented

**mcsas.dataobj.datavector module**

A class describing a vector with limits, units, mask and uncertainties

**class DataVector** (*name, raw, rawU=None, unit=None*)

Bases: `object`

a class for combining aspects of a particular vector of data. This is intended only as a storage container without additional functionality.

**binnedData**

**binnedDataU**

**hdfWrite** (*hdf*)

**Warning:** method 'dataobj.datavector.DataVector.hdfWrite' undocumented

**limit**

**limsString**

**name**

**rawData**

**rawDataU**

**sanitized**

**sanitizedU**

**siData**

**siDataU**

**unit**

**validIndices**

**mcsas.dataobj.sasconfig module**

**class GaussianSmearing**

Bases: `dataobj.sasconfig.SmearingConfig`

**inputValid** ()

**Warning:** method 'dataobj.sasconfig.GaussianSmearing.inputValid' undocumented

**parameters** = (None,)

**setIntPoints** (*q*)

Sets smearing profile integration points for trapezoidal slit. Top (umbra) of trapezoid has full width  $x_t$ , bottom of trapezoid (penumbra) has full width. Since the smearing function is assumed to be symmetrical, the integration parameters are calculated in the interval  $[0, x_b/2]$

**showParams**

**updatePLimits** (*pLimit*)

**Warning:** method 'dataobj.sasconfig.GaussianSmearing.updatePLimits' undocumented

**updatePUnit** (*newUnit*)

**Warning:** method 'dataobj.sasconfig.GaussianSmearing.updatePUnit' undocumented

**updateQLimits** (*qLimit*)

**Warning:** method 'dataobj.sasconfig.GaussianSmearing.updateQLimits' undocumented

**updateQUnit** (*newUnit*)

**Warning:** method 'dataobj.sasconfig.GaussianSmearing.updateQUnit' undocumented

**updateSmearingLimits** (*q*)

**Warning:** method 'dataobj.sasconfig.GaussianSmearing.updateSmearingLimits' undocumented

**Parameter** (*\*args, \*\*kwargs*)

**Warning:** function 'dataobj.sasconfig.Parameter' undocumented

**class SASConfig** (*\*args, \*\*kwargs*)

Bases: *dataobj.dataconfig.DataConfig*

**hdfWrite** (*hdf*)

**Warning:** method 'dataobj.sasconfig.SASConfig.hdfWrite' undocumented

**onUpdatedX0** (*x0*)

Sets available range of loaded data.

**onUpdatedX1** (*x1*)

**Warning:** method 'dataobj.sasconfig.SASConfig.onUpdatedX1' undocumented

**prepareSmearing** (*q*)

**Warning:** method 'dataobj.sasconfig.SASConfig.prepareSmearing' undocumented

**shortName** = 'SAS data configuration'

**showParams**

**smearing**

**updateX0Unit** (*newUnit*)

**Warning:** method 'dataobj.sasconfig.SASConfig.updateX0Unit' undocumented

**updateX1Unit** (*newUnit*)

**Warning:** method 'dataobj.sasconfig.SASConfig.updateX1Unit' undocumented

**class SmearingConfig**

Bases: *bases.algorithm.algorithmbase.AlgorithmBase*

Abstract base class, can't be instantiated.

Creates instances from defined parameters and replaces the class attributes accordingly.

**hdfWrite** (*hdf*)

**Warning:** method 'dataobj.sasconfig.SmearingConfig.hdfWrite' undocumented

**parameters** = (None, None, None)

**prepared**

**qOffset**

**shortName** = 'SAS smearing configuration'

**updatePLimits** (*pLimit*)

**Warning:** method 'dataobj.sasconfig.SmearingConfig.updatePLimits' undocumented

**updatePUnit** (*newUnit*)

**Warning:** method 'dataobj.sasconfig.SmearingConfig.updatePUnit' undocumented

**updateQLimits** (*qLimit*)

**Warning:** method 'dataobj.sasconfig.SmearingConfig.updateQLimits' undocumented

**updateQUnit** (*newUnit*)

**Warning:** method 'dataobj.sasconfig.SmearingConfig.updateQUnit' undocumented

**updateSmearingLimits** (*q*)

**Warning:** method 'dataobj.sasconfig.SmearingConfig.updateSmearingLimits' undocumented

**weights**

**class TrapezoidSmearing**

Bases: *dataobj.sasconfig.SmearingConfig*

**halfTrapzPDF** (*x, c, d*)

**Warning:** method 'dataobj.sasconfig.TrapezoidSmearing.halfTrapzPDF' undocumented

**inputValid** ()

**Warning:** method 'dataobj.sasconfig.TrapezoidSmearing.inputValid' undocumented

**onUmbraUpdate** ()

Value in umbra will not exceed available q.

**parameters** = (None, None)

**setIntPoints** (*q*)

sets smearing profile integration points for trapezoidal slit. Top (umbra) of trapezoid has full width *xt*, bottom of trapezoid (penumbra) has full width. Since the smearing function is assumed to be symmetrical, the integration parameters are calculated in the interval  $[0, x_b/2]$

**showParams**

**updatePLimits** (*pLimit*)



**Warning:** method ‘dataobj.sasconfig.TrapezoidSmearing.updatePLimits’ undocumented

**updatePUnit** (*newUnit*)

**Warning:** method ‘dataobj.sasconfig.TrapezoidSmearing.updatePUnit’ undocumented

**updateQLimits** (*qLimit*)

**Warning:** method ‘dataobj.sasconfig.TrapezoidSmearing.updateQLimits’ undocumented

**updateQUnit** (*newUnit*)

**Warning:** method ‘dataobj.sasconfig.TrapezoidSmearing.updateQUnit’ undocumented

**updateSmearingLimits** (*q*)

**Warning:** method ‘dataobj.sasconfig.TrapezoidSmearing.updateSmearingLimits’ undocumented

### mcsas.dataobj.sasconfig\_test module

**assertDefaults** (*sc*)

**Warning:** function ‘dataobj.sasconfig\_test.assertDefaults’ undocumented

**testSerialize** ()

**Warning:** function ‘dataobj.sasconfig\_test.testSerialize’ undocumented

### mcsas.dataobj.sasdata module

Represents data associated with a measurement by small angle scattering (SAS). Some examples and tests.

```
>>> import numpy
>>> testdata = numpy.random.rand(4,4)
>>> testtitle = "some title"
>>> from sasdata import SASData
```

Testing >>> first = SASData(testtitle, testdata) >>> first.title == testtitle True >>> numpy.all(first.rawArray == testdata) True

**class SASData** (*\*\*kwargs*)

Bases: *dataobj.dataobj.DataObj*

Represents one set of data from a unique source (a file, for example).

**configType**

**count**

**dataContent**

shows the content of the loaded data: Q, I, IErr, etc

**classmethod displayData** ()

**Warning:** method 'dataobj.sasdata.SASData.displayData' undocumented

**classmethod displayDataDescr** ()

**Warning:** method 'dataobj.sasdata.SASData.displayDataDescr' undocumented

**modelType**

**p**

Q-Vector at which the intensities are measured. Provided for convenience use within models.

**pLimsString**

Properly formatted q-limits for UI label text.

**q**

Q-Vector at which the intensities are measured. Provided for convenience use within models.

**qLimsString**

Properly formatted q-limits for UI label text.

**rUnit**

**shannonChannelEst** ()

**Warning:** method 'dataobj.sasdata.SASData.shannonChannelEst' undocumented

**shannonChannelEstText**

**classmethod sourceName** ()

The type of data source for UI label text.

**sphericalSizeEst** ()

**Warning:** method 'dataobj.sasdata.SASData.sphericalSizeEst' undocumented

**sphericalSizeEstText****updateConfig()****Warning:** method 'dataobj.sasdata.SASData.updateConfig' undocumented**classproperty** (*func*)**Warning:** function 'dataobj.sasdata.classproperty' undocumented

## Module contents

**class DataObj** (*\*\*kwargs*)Bases: `abc.NewBase`

General container for data loaded from file. It offers specialised methods to derive information from the provided data.

**accumulate** (*others*)**Warning:** method 'dataobj.DataObj.accumulate' undocumented**config****configType**

Returns a compatible DataConfig type.

**count****f**

The measurement vector.

**filename****hasUncertainties**

Returns True if this data set has an error bar for its intensities.

**hdfWrite** (*hdf*)**Warning:** method 'dataobj.DataObj.hdfWrite' undocumented**initConfig()**

Initializes a new data configuration and sets the sample name which is used to differentiate different data objects of the same type later on.

**is2d**

Returns true if this dataset contains two-dimensional data with psi information available.

**modelType**

Returns a compatible ScatteringModel type.

**sampleName****seriesKey**

The Name of the DataObj property to use as series key, hard-coded for now, assuming it exists. It allows to let the user chose from a generated list of properties (todo).

**seriesKeyName**

Returns the docstring of the property defined by self.seriesKeyProp.

**seriesKeyValue**

Returns the value of the property defined by self.seriesKeyProp.

**setConfig** (*config=None*)

Set the configuration of this data object if the type matches.

**setFilename** (*fn*)

Stores the absolute path to this data file. Should be reviewed when data sets can be created from several files.

**classmethod sourceName** ()

Returns the name of the measurement method.

**updateConfig** ()

Updates the config object based on this data set. All callbacks are run right after this method in setConfig().

**x0**

First sampling vector.

**x1**

Second sampling vector.

**x2**

Third sampling vector.

**class SASData** (*\*\*kwargs*)

Bases: *dataobj.dataobj.DataObj*

Represents one set of data from a unique source (a file, for example).

**configType****count****dataContent**

shows the content of the loaded data: Q, I, IErr, etc

**classmethod displayData** ()

**Warning:** method 'dataobj.SASData.displayData' undocumented

**classmethod displayDataDescr** ()

**Warning:** method 'dataobj.SASData.displayDataDescr' undocumented

**modelType**

**p**

Q-Vector at which the intensities are measured. Provided for convenience use within models.

**pLimsString**

Properly formatted q-limits for UI label text.

**q**

Q-Vector at which the intensities are measured. Provided for convenience use within models.

**qLimsString**

Properly formatted q-limits for UI label text.

**rUnit****shannonChannelEst** ()**Warning:** method 'dataobj.SASData.shannonChannelEst' undocumented**shannonChannelEstText****classmethod sourceName** ()

The type of data source for UI label text.

**sphericalSizeEst** ()**Warning:** method 'dataobj.SASData.sphericalSizeEst' undocumented**sphericalSizeEstText****updateConfig** ()**Warning:** method 'dataobj.SASData.updateConfig' undocumented**class DataConfig**Bases: *bases.algorithm.algorithmbase.AlgorithmBase*, *dataobj.dataconfig.CallbackRegistry***callbackSlots****hdfWrite** (*hdf*)**Warning:** method 'dataobj.DataConfig.hdfWrite' undocumented**is2d****onUpdatedX0** (*x0*)

Sets available range of loaded data.

**onUpdatedX1** (*x1*)

**Warning:** method ‘dataobj.DataConfig.onUpdatedX1’ undocumented

**parameters** = (None, None, None, None, None, None, None, None)

**sampleName**

**showParams**

**updateFMasks** ()

**Warning:** method ‘dataobj.DataConfig.updateFMasks’ undocumented

**updateFuMin** ()

**Warning:** method ‘dataobj.DataConfig.updateFuMin’ undocumented

**updateX0Limits** ()

**Warning:** method ‘dataobj.DataConfig.updateX0Limits’ undocumented

**updateX0Unit** (*newUnit*)

Sets the unit of the x0 vector.

**updateX1Limits** ()

**Warning:** method ‘dataobj.DataConfig.updateX1Limits’ undocumented

**updateX1Unit** (*newUnit*)

**Warning:** method ‘dataobj.DataConfig.updateX1Unit’ undocumented

## mcsas.bases package

### Subpackages

mcsas.bases.algorithm package

### Submodules

**mcsas.bases.algorithm.algorithmbase module****class AlgorithmBase**

Bases: object

Base class for all data filtering algorithms.

Creates instances from defined parameters and replaces the class attributes accordingly.

**factory** (\*args, \*\*kwargs)**Warning:** method 'bases.algorithm.algorithmbase.AlgorithmBase.factory' undocumented**hdfWrite** (hdf)**Warning:** method 'bases.algorithm.algorithmbase.AlgorithmBase.hdfWrite' undocumented**classmethod makeDefault** ()**Warning:** method 'bases.algorithm.algorithmbase.AlgorithmBase.makeDefault' undocumented**classmethod name** ()**Warning:** method 'bases.algorithm.algorithmbase.AlgorithmBase.name' undocumented**param** = functools.partial(<function AlgorithmBase.param>, <class 'bases.algorithm.algorithmbase.AlgorithmBase'>)**paramCount** = functools.partial(<function AlgorithmBase.paramCount>, <class 'bases.algorithm.algorithmbase.AlgorithmBase'>)**params** = functools.partial(<function AlgorithmBase.params>, <class 'bases.algorithm.algorithmbase.AlgorithmBase'>)**classmethod setName** (name)**Warning:** method 'bases.algorithm.algorithmbase.AlgorithmBase.setName' undocumented**setParam** = functools.partial(<function AlgorithmBase.setParam>, <class 'bases.algorithm.algorithmbase.AlgorithmBase'>)**classmethod setParams** (\*parameters)Expects a list of ParameterBase classes/types and sets them as class attributes to this class. They will become instances later, please see `__init__()`**showParams**

A list of parameter names which defines the parameters and their ordering shown in a UI. To be overridden in sub classes.

**update** (*other*)

Copy parameter values from another algorithm of the same type.

**exception AlgorithmError**

Bases: `Exception`

**exception AlgorithmNameError**

Bases: `bases.algorithm.algorithmbase.AlgorithmError`

**exception AlgorithmParameterError**

Bases: `bases.algorithm.algorithmbase.AlgorithmError`

**classproperty** (*func*)

**Warning:** function ‘bases.algorithm.algorithmbase.classproperty’ undocumented

### **mcsas.bases.algorithm.algorithmbase\_test module**

**class DummyAlgo**

Bases: `bases.algorithm.algorithmbase.AlgorithmBase`

**dummy** (*v*)

**Warning:** method ‘bases.algorithm.algorithmbase\_test.DummyAlgo.dummy’ undocumented

**parameters** = (None, None)

**shortName** = ‘Dummy’

**Parameter** (*\*args, \*\*kwargs*)

**Warning:** function ‘bases.algorithm.algorithmbase\_test.Parameter’ undocumented

**np\_array** (*value*)

**Warning:** function ‘bases.algorithm.algorithmbase\_test.np\_array’ undocumented

**testParam** ()

Algorithm without parameters allowed

**testParam1** ()

**Warning:** function ‘bases.algorithm.algorithmbase\_test.testParam1’ undocumented



`testSerialize()`

**Warning:** function ‘bases.algorithm.algorithmbase\_test.testSerialize’ undocumented

`testTypeVsInstance()`

**Warning:** function ‘bases.algorithm.algorithmbase\_test.testTypeVsInstance’ undocumented

### mcsas.bases.algorithm.numbergenerator module

**class** `NumberGenerator`

Bases: `object`

Base class for number generators. Generates numbers in the interval [0, 1]. Scaling is supposed to happen elsewhere.

**classmethod** `get` (*count=1*)

**Warning:** method ‘bases.algorithm.numbergenerator.NumberGenerator.get’ undocumented

**classmethod** `hdfWrite` (*hdf*)

**Warning:** method ‘bases.algorithm.numbergenerator.NumberGenerator.hdfWrite’ undocumented

**class** `RandomExponential`

Bases: `bases.algorithm.numbergenerator.NumberGenerator`

**classmethod** `get` (*count=1*)

**Warning:** method ‘bases.algorithm.numbergenerator.RandomExponential.get’ undocumented

`lower = 0.0`

`upper = 1.0`

**class** `RandomExponential1`

Bases: `bases.algorithm.numbergenerator.RandomExponential`

Alias class for `RandomExponential`

**class** `RandomExponential2`

Bases: `bases.algorithm.numbergenerator.RandomExponential`

Picks values with inverse logarithmic probability over  $(0, 1)$ , as if it were spanning two decades.

`upper = 2.0`

**class RandomExponential3**Bases: *bases.algorithm.numbergenerator.RandomExponential*Picks values with inverse logarithmic probability over  $(0, 1)$ , as if it were spanning three decades.**upper = 3.0****class RandomUniform**Bases: *bases.algorithm.numbergenerator.NumberGenerator***classmethod get** (*count=1*)**Warning:** method 'bases.algorithm.numbergenerator.RandomUniform.get' undocumented**class RandomXorShiftUniform**Bases: *bases.algorithm.numbergenerator.NumberGenerator*Implemented according to xorshift1024\* at <http://xorshift.di.unimi.it>

```
>>> from bases.algorithm.numbergenerator import RandomXorShiftUniform
>>> RandomXorShiftUniform.getSeed()
>>> RandomXorShiftUniform.setSeed()
>>> RandomXorShiftUniform.next()
>>> RandomXorShiftUniform.get()
>>> RandomXorShiftUniform.get(3)
```

**classmethod dtype** ()**Warning:** method 'bases.algorithm.numbergenerator.RandomXorShiftUniform.dtype' undocumented**classmethod get** (*count=1*)**Warning:** method 'bases.algorithm.numbergenerator.RandomXorShiftUniform.get' undocumented**classmethod getSeed** ()

Generate seed using numpy.

**classmethod next** ()**Warning:** method 'bases.algorithm.numbergenerator.RandomXorShiftUniform.next' undocumented**p = None****s = None****classmethod setSeed** (*seedData=None*)

**Warning:** method ‘bases.algorithm.numbergenerator.RandomXorShiftUniform.setSeed’ undocumented

**class RandomXorShiftUniformTest** (*methodName='runTest'*)

Bases: unittest.case.TestCase

Tests RandomXorShiftUniform output against reference C implementation. The full path to the executable has to be specified. Call it like this:

```
nosetests bases.algorithm.numbergenerator
```

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

**classmethod generateTests** ()

**Warning:** method ‘bases.algorithm.numbergenerator.RandomXorShiftUniformTest.generateTests’ undocumented

**getRef** ()

**Warning:** method ‘bases.algorithm.numbergenerator.RandomXorShiftUniformTest.getRef’ undocumented

**setUp** ()

**Warning:** method ‘bases.algorithm.numbergenerator.RandomXorShiftUniformTest.setUp’ undocumented

**tearDown** ()

**Warning:** method ‘bases.algorithm.numbergenerator.RandomXorShiftUniformTest.tearDown’ undocumented

**test0** ()

**Warning:** method ‘bases.algorithm.numbergenerator.RandomXorShiftUniformTest.test0’ undocumented

**test1** ()

**Warning:** method 'bases.algorithm.numbergenerator.RandomXorShiftUniformTest.test1' undocumented

**test2()**

**Warning:** method 'bases.algorithm.numbergenerator.RandomXorShiftUniformTest.test2' undocumented

**test3()**

**Warning:** method 'bases.algorithm.numbergenerator.RandomXorShiftUniformTest.test3' undocumented

**test4()**

**Warning:** method 'bases.algorithm.numbergenerator.RandomXorShiftUniformTest.test4' undocumented

**lshift**(*x*, *s*)

**Warning:** function 'bases.algorithm.numbergenerator.lshift' undocumented

**rshift**(*x*, *s*)

**Warning:** function 'bases.algorithm.numbergenerator.rshift' undocumented

### mcsas.bases.algorithm.parameter module

This module defines a generic parameter class for algorithms. It contains meta information which allows for automated UI building. Create sub classes by calling `factory()` in this module. It creates a new sub class type which inherits `ParameterBase`:

```
>>> from parameter import factory as paramFactory
>>> ParamType = paramFactory("radius", 1.3, valueRange = (0, 2))
```

Created a new type `RadiusParameter`:

```
>>> print(ParamType)
<class 'parameter.RadiusParameter'>
```

Using methods on instances work as usual:

```
>>> p = ParamType()
>>> p.name()
'radius'
>>> p.value()
1.3
```

Update the instance:

```
>>> p.setValue(2.4)
>>> p.value()
2.4
```

Changing class default: >>> ParamType.setValue(3.5) >>> ParamType.value() 3.5

Existing instance keep their values: >>> p.value() 2.4

New instances get the updated defaults: >>> q = ParamType() >>> q.value() 3.5

Parameter attributes are accessible on type/class as well as on the instance. Updating an attribute of an instance changes just that individual instance whereas updating an attribute of the type changes that attribute in general for all new instances to be created which behaves like a default value.

#### exception **DecimalsError**

Bases: *bases.algorithm.parameter.ParameterError*

#### exception **DefaultValueError**

Bases: *bases.algorithm.parameter.ParameterError*

#### exception **DisplayValuesError**

Bases: *bases.algorithm.parameter.ParameterError*

#### class **ParameterBase**

Bases: object

Base class for algorithm parameters providing additional information to ease automated GUI building.

**classmethod addAttributes** (*dictionary, \*names, \*\*namesAndValues*)

Sets an *ordered* list of attributes. Initializes the private variable to None and sets a default getter method for each name provided. Additionally, sets *attributeNames* to return all attribute names.

**classmethod attributeNames** ()

Returns an ordered list of attribute names considering multiple inheritance and maintaining its order.

**attributes** = *functools.partial*(<function **ParameterBase.attributes**>, <class 'bases.algorithm.parameter.ParameterBase'>)

**copy** ()

**Warning:** method 'bases.algorithm.parameter.ParameterBase.copy' undocumented

**displayName** = *functools.partial*(<function **\_makeGetter.<locals>.getter**>, <class 'bases.algorithm.parameter.ParameterBase'>)

**displayValue** = *functools.partial*(<function **ParameterBase.displayValue**>, <class 'bases.algorithm.parameter.ParameterBase'>)

**classmethod dtype** ()

**Warning:** method 'bases.algorithm.parameter.ParameterBase.dtype' undocumented

**formatDisplayName** = `functools.partial(<function ParameterBase.formatDisplayName>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**generate** (*lower=None, upper=None, count=1*)

Returns a list of valid parameter values within given bounds. Accepts vectors of individual bounds for lower and upper limit. This allows for inequality parameter constraints. *lower*, *upper*: arrays for lower and upper bounds

**classmethod get** (*key, default=None*)

metagetter to get an attribute parameter

**hdfStoreAsMember** ()

**Warning:** method 'bases.algorithm.parameter.ParameterBase.hdfStoreAsMember' undocumented

**hdfWrite** (*hdf*)

**Warning:** method 'bases.algorithm.parameter.ParameterBase.hdfWrite' undocumented

**classmethod isDataType** (*value*)

**Warning:** method 'bases.algorithm.parameter.ParameterBase.isDataType' undocumented

**name** = `functools.partial(<function _makeGetter.<locals>.getter>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**onValueUpdate** = `functools.partial(<function _makeGetter.<locals>.getter>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**classmethod set** (*key, value*)

metasetter to set an attribute value

**setAttributes** = `functools.partial(<function ParameterBase.setAttributes>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**setDisplay\_name** = `functools.partial(<function ParameterBase.setDisplayName>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**setDisplayValue** = `functools.partial(<function ParameterBase.setDisplayValue>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**classmethod setName** (*name*)

Changing the name is allowed for the class/type only, not for instances.

**setOnValueUpdate** = `functools.partial(<function _makeSetter.<locals>.setter>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**setValue** = `functools.partial(<function ParameterBase.setValue>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**value** = `functools.partial(<function _makeGetter.<locals>.getter>, <class 'bases.algorithm.parameter.ParameterBase'>)`

**class ParameterBoolean**

Bases: `bases.algorithm.parameter.ParameterBase`

**classmethod dtype** ()

**Warning:** method 'bases.algorithm.parameter.ParameterBoolean.dtype' undocumented

**exception ParameterError**

Bases: `Exception`

**class ParameterFloat**

Bases: `bases.algorithm.parameter.ParameterNumerical`

**decimals** = `functools.partial(<function _makeGetter.<locals>.getter>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**displayMagnitudeName** = `functools.partial(<function ParameterFloat.displayMagnitudeName>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**displayValue** = `functools.partial(<function ParameterFloat.displayValue>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**displayValueRange** = `functools.partial(<function ParameterFloat.displayValueRange>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**classmethod dtype** ()

**Warning:** method 'bases.algorithm.parameter.ParameterFloat.dtype' undocumented

**hdfStoreAsMember** ()

**Warning:** method 'bases.algorithm.parameter.ParameterFloat.hdfStoreAsMember' undocumented

**classmethod isDataType** (*value*)

**Warning:** method 'bases.algorithm.parameter.ParameterFloat.isDataType' undocumented

**setDecimals** = `functools.partial(<function ParameterFloat.setDecimals>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**setDisplayValue** = `functools.partial(<function ParameterFloat.setDisplayValue>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**setSuffix** = `functools.partial(<function ParameterFloat.setSuffix>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**setUnit** = `functools.partial(<function _makeSetter.<locals>.setter>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**suffix** = `functools.partial(<function ParameterFloat.suffix>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**toDisplay** = `functools.partial(<function ParameterFloat.toDisplay>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**toSi** = `functools.partial(<function ParameterFloat.toSi>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**unit** = `functools.partial(<function _makeGetter.<locals>.getter>, <class 'bases.algorithm.parameter.ParameterFloat'>)`

**exception ParameterGeneratorError**

Bases: `bases.algorithm.parameter.ParameterError`

**class ParameterLog**

Bases: `bases.algorithm.parameter.ParameterFloat`

Used to select an UI input widget with logarithmic behaviour.

**exception ParameterNameError**

Bases: `bases.algorithm.parameter.ParameterError`

**class ParameterNumerical**

Bases: `bases.algorithm.parameter.ParameterBase`

```
clip = functools.partial(<function ParameterNumerical.clip>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
displayValues = functools.partial(<function ParameterNumerical.displayValues>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
classmethod dtype ()
```

**Warning:** method 'bases.algorithm.parameter.ParameterNumerical.dtype' undocumented

```
generate (lower=None, upper=None, count=1)
```

**Warning:** method 'bases.algorithm.parameter.ParameterNumerical.generate' undocumented

```
generator = functools.partial(<function _makeGetter.<locals>.getter>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
hdfStoreAsMember ()
```

**Warning:** method 'bases.algorithm.parameter.ParameterNumerical.hdfStoreAsMember' undocumented

```
classmethod isDataType (value)
```

ParameterNumerical is a fallback for all number not being float.

```
max = functools.partial(<function ParameterNumerical.max>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
min = functools.partial(<function ParameterNumerical.min>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
setDisplayValues = functools.partial(<function ParameterNumerical.setDisplayValues>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
setGenerator = functools.partial(<function ParameterNumerical.setGenerator>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
setStepping = functools.partial(<function ParameterNumerical.setStepping>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
setSuffix = functools.partial(<function ParameterNumerical.setSuffix>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
setValue = functools.partial(<function ParameterNumerical.setValue>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
setValueRange = functools.partial(<function ParameterNumerical.setValueRange>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
stepping = functools.partial(<function _makeGetter.<locals>.getter>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
suffix = functools.partial(<function _makeGetter.<locals>.getter>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
valueRange = functools.partial(<function ParameterNumerical.valueRange>, <class 'bases.algorithm.parameter.ParameterNumerical'>)
```

```
class ParameterString
```

Bases: `bases.algorithm.parameter.ParameterBase`

String-based parameter class. The default value should be the first item in the `_valueRange` list.

```
classmethod dtype ()
```

**Warning:** method 'bases.algorithm.parameter.ParameterString.dtype' undocumented



**classmethod** `isDataType` (*value*)

**Warning:** method ‘bases.algorithm.parameter.ParameterString.isDataType’ undocumented

**classmethod** `setValueRange` (*newRange*)

**Warning:** method ‘bases.algorithm.parameter.ParameterString.setValueRange’ undocumented

**valueRange** = `functools.partial`(<function `ParameterString.valueRange`>, <class ‘bases.algorithm.parameter.ParameterString’>)

**exception** `SteppingError`

Bases: `bases.algorithm.parameter.ParameterError`

**exception** `SuffixError`

Bases: `bases.algorithm.parameter.ParameterError`

**exception** `ValueRangeError`

Bases: `bases.algorithm.parameter.ParameterError`

**classproperty** (*func*)

**Warning:** function ‘bases.algorithm.parameter.classproperty’ undocumented

**factory** (*name*, *value*, *paramTypes=None*, *\*\*kwargs*)

Generates a new Parameter type derived from one of the predefined base classes chosen by the supplied value: Providing a string value results in a type derived from `ParameterBase`, providing an integer value produces a `ParameterNumerical` type and a float value results in a `ParameterFloat` type. Alternatively, a class type `cls` can be provided which is used as base class for the resulting Parameter class type. Make sure in this case, all attributes mandatory for this base type are provided too.

- name*: short name of the new parameter without spaces
- value*: default value from which the type is derived if `cls` is not given

Optional arguments:

- paramTypes*: tuple of available parameter types instead of the default
- cls*: forces a certain Parameter type.
- description*: Updates the `__doc__` attribute. May be displayed in the UI somewhere.

**generateValues** (*numberGenerator*, *defaultRange*, *lower*, *upper*, *count*)

**Warning:** function ‘bases.algorithm.parameter.generateValues’ undocumented

**mcsas.bases.algorithm.parameter\_test module****class Dummy**

Bases: object

**dummyFunc** (*value*)**Warning:** method 'bases.algorithm.parameter\_test.Dummy.dummyFunc' undocumented**Parameter** (*\*args, \*\*kwargs*)**Warning:** function 'bases.algorithm.parameter\_test.Parameter' undocumented**testParameterBaseCopy** ()**Warning:** function 'bases.algorithm.parameter\_test.testParameterBaseCopy' undocumented**testParameterCompare** ()**Warning:** function 'bases.algorithm.parameter\_test.testParameterCompare' undocumented**testParameterFloat** ()**Warning:** function 'bases.algorithm.parameter\_test.testParameterFloat' undocumented**testParameterFloatCopy** ()**Warning:** function 'bases.algorithm.parameter\_test.testParameterFloatCopy' undocumented**testParameterName** ()**Warning:** function 'bases.algorithm.parameter\_test.testParameterName' undocumented**testParameterNumerical** ()

**Warning:** function ‘bases.algorithm.parameter\_test.testParameterNumerical’ undocumented

`testParameterNumericalCopy()`

**Warning:** function ‘bases.algorithm.parameter\_test.testParameterNumericalCopy’ undocumented

`testParameterNumericalDisplayValues()`

**Warning:** function ‘bases.algorithm.parameter\_test.testParameterNumericalDisplayValues’ undocumented

`testParameterNumericalStepping()`

**Warning:** function ‘bases.algorithm.parameter\_test.testParameterNumericalStepping’ undocumented

`testParameterNumericalSuffix()`

**Warning:** function ‘bases.algorithm.parameter\_test.testParameterNumericalSuffix’ undocumented

`testParameterNumericalValueRange()`

**Warning:** function ‘bases.algorithm.parameter\_test.testParameterNumericalValueRange’ undocumented

`testParameterSerialize()`

**Warning:** function ‘bases.algorithm.parameter\_test.testParameterSerialize’ undocumented

## Module contents

`Parameter(*args, **kwargs)`

**Warning:** function ‘bases.algorithm.Parameter’ undocumented

## mcsas.bases.dataset package

## Submodules

### mcsas.bases.dataset.dataset module

**class DataSet** (\*\*kwargs)

Bases: `bases.dataset.titlemixin.TitleMixin`, `bases.dataset.rawarraymixin.RawArrayMixin`

Container base class for all kinds of data to be passed around in the UI. Knows its originally loaded (raw) data array and its title to be shown in a UI.

**class DisplayMixin** (\*\*kwargs)

Bases: `object`

Provides additional data to display in a list or tree view.

**displayData**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: def f(cls, arg1, arg2, ...): ... f = classmethod(f)
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

**displayDataDescr**

classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: def f(cls, arg1, arg2, ...): ... f = classmethod(f)
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the `staticmethod` builtin.

**isRemovable**

Returns if this data set may be removed (e.g. from data lists in a GUI)

**class ResultMixin** (\*\*kwargs)

Bases: `object`

**result**

Supposed to return a list of *Result* types.

**classproperty** (func)

**Warning:** function ‘bases.dataset.dataset.classproperty’ undocumented

### mcsas.bases.dataset.rawarraymixin module

**class RawArrayMixin** (*rawArray=None, \*\*kwargs*)

Bases: object

Memorizes the original data before processing it eventually.

**classmethod isValidInput** (*rawArray*)

**Warning:** method ‘bases.dataset.rawarraymixin.RawArrayMixin.isValidInput’ undocumented

**rawArray**

**setRawArray** (*rawArray*)

**Warning:** method ‘bases.dataset.rawarraymixin.RawArrayMixin.setRawArray’ undocumented

**valid**

### mcsas.bases.dataset.titlemixin module

**class TitleMixin** (*title=None, \*\*kwargs*)

Bases: object

Manages a title of this object for display in a GUI.

**title**

Data object title

## Module contents

### mcsas.bases.model package

#### Submodules

### mcsas.bases.model.modeldata module

**class ModelData** (*cumInt, vset, wset, sset, numParams*)

Bases: object

**chisqrInt**

Make the model intensity comparable to the measured intensity. The difference of both will be calculated in BackgroundScalingFit in order to perform the chi-square test.

**cumInt**

Returns the cumulated model intensity or signal.

**hdfWrite** (*hdf*)

**Warning:** method ‘bases.model.modeldata.ModelData.hdfWrite’ undocumented

**numParams**

Returns the number of active (fitted) parameters.

**sset**

Returns the associated set of surfaces.

**volumeFraction** (*scaling*)

Returns the volume fraction based on the provided scaling factor to match this model data to the measured data. Assumes that the weights ‘self.wset’ contain the scatterer volume squared.

**vset**

Returns the associated set of volumes.

**wset**

Returns the associated set of weights.

**class SASModelData** (*cumInt, vset, wset, sset, numParams*)

Bases: *bases.model.modeldata.ModelData*

## mcsas.bases.model.sasmodel module

**class SASModel**

Bases: *bases.model.scatteringmodel.ScatteringModel*

**calcIntensity** (*data, compensationExponent=None*)

Returns the intensity  $I$ , the volume  $v_{abs}$  and the intensity weights  $w$  for a single parameter contribution over all  $q$ :

$$I(q, r) = F^2(q, r) \cdot w(r)$$

**canSmear = False**

**getQ** (*dataset*)

This is a function that returns  $Q$ . In case of smearing, dataset itself is a 2D matrix of  $Q$ -values. When smearing is not enabled, dataset.q contains a 1D vector of  $q$ .

I do realize that this is not a good way of doing things. This should be replaced at a given point in time by a better solution within sasdata.

**modelDataType** ()

**Warning:** method ‘bases.model.sasmodel.SASModel.modelDataType’ undocumented

**weight** ()

Calculates an intensity weighting used during fitting. It is based on the scatterers volume. It can be modified by a user-defined compensation exponent  $c$ . The default value is  $c = \frac{2}{3}$

$$w(r) = v(r)^{2c}$$

**mcsas.bases.model.scatteringmodel module****class ScatteringModel**

Bases: *bases.algorithm.algorithmbase.AlgorithmBase*

Creates instances from defined parameters and replaces the class attributes accordingly.

**absVolume()**

Forwarding to usual volume() by default. Can be overridden to include SLD.

**activeParamCount** = `functools.partial(<function ScatteringModel.activeParamCount>, <class 'bases.model.scatteringmodel.ScatteringModel'>)`

**activeParamNames** = `functools.partial(<function ScatteringModel.activeParamNames>, <class 'bases.model.scatteringmodel.ScatteringModel'>)`

**activeParams** = `functools.partial(<function ScatteringModel.activeParams>, <class 'bases.model.scatteringmodel.ScatteringModel'>)`

**calc** (*data*, *pset*, *compensationExponent=None*)

Calculates the total intensity and scatterer volume contributions using the current model. *pset* number columns equals the number of active parameters. Returns a ModelData object for a certain type of measurement.

**calcIntensity** (*data*, *compensationExponent=None*)

Calculates the model intensity which is later compared to the data. Returns a tuple containing an array of the calculated intensities for the grid provided with the data and the volume of a single particle based on the model parameters. Has to be implemented in derived classes specific to a certain type of measurement.

**fixTestParams** = `functools.partial(<function ScatteringModel.fixTestParams>, <class 'bases.model.scatteringmodel.ScatteringModel'>)`

**formfactor** (*dataset*)

Calculates the Rayleigh function of this model. Reimplement this for new models.

**generateParameters** (*count=1*)

Generates a set of parameters for this model using the predefined Parameter.generator. Allows for different random number distributions.

**getModelData** (*cumInt*, *vset*, *wset*, *sset*)

**Warning:** method 'bases.model.scatteringmodel.ScatteringModel.getModelData' undocumented

**classmethod getParametersFromFilename** (*filename*)

Derives model parameters for testing from reference data file.

**hdfWrite** (*hdf*)

**Warning:** method 'bases.model.scatteringmodel.ScatteringModel.hdfWrite' undocumented

**modelDataType** ()

Returns the appropriate ModelData class for this type of model.

**surface** ()

Returns the surface area of a single scatterer. Used for the surface weighted distribution histogram. Returns 0 by default. Reimplement this for a model.

**classmethod test** (*filename*)

Regression test of a scattering model. File names are expected to contain the parameter values which produce the provided intensity. Otherwise implement fixTestParams() for the particular model.

- filename*: Name of the file in `cls.testDataDir` to test against.

- cls.testRelErr*: Acceptable mean of relative error against reference intensity. Default: 1e-5

- cls.testVolExp*: Volume compensation exponent, sets the amount of volume contribution the intensity is scaled by.

- cls.testDataDir*: Directory of test data relative to program dir. Default: "testdata"

`update = functools.partial(<function ScatteringModel.update>, <class 'bases.model.scatteringmodel.ScatteringModel'>)`

`updateParamBounds (bounds)`

**Warning:** method 'bases.model.scatteringmodel.ScatteringModel.updateParamBounds' undocumented

`volume ()`

Calculates the volume of this model, taking `compensationExponent` into account from input or preset parameters. Reimplement this for new models.

`weight ()`

A weighting function for the form factor. With SAXS, it is usually the volume squared.

## mcsas.bases.model.scatteringmodel\_test module

### Module contents

### Module contents

## mcsas.main module

`getScriptPath ()`

Returns the full path to the current script file which calls this function.

`main (argv=None)`

**Warning:** function 'main.main' undocumented

`makeAbsolutePath (relpath)`

**Warning:** function 'main.makeAbsolutePath' undocumented



# McSAS

## mcsas.cxfreeze module

### Overview

Creates a standalone program package for a particular platform to be run by restricted users without installing any additional packages.

This script is executable and has to be run on the platform for which a package shall be created. Please follow the instructions below for each particular platform.

Common Package Dependencies:

- [Python 2.7](#)
- [Qt 4.8 + PySide](#)
- **NumPy and SciPy** In order to work around freeze failures with newer versions it is recommended to stick with Numpy 1.7 and SciPy 1.12 which was tested successfully.
- [matplotlib](#)

In addition to the dependencies of the MCSAS package listed above the [cx\\_Freeze package](#) is used for freezing the python source code structure into a standalone package.

### Working with Source Code Repositories

In order to download the latest source code repositories of packages such as MCSAS or [cx\\_Freeze](#) a client to [Git](#) and [Mercurial](#) is required. There are several available, for both Mac OS X and Windows the [SourceTree](#) program is recommended.

### Windows

A self-contained archive consisting of `MCSAS.exe` and all necessary libraries and files is created by the following command executed in the MCSAS folder:

```
> python cxfreeze.py build_exe
```

### Requirements

On a fresh installation of Windows 7 the following packages are required:

- [Python 2.7.9](#)
- [PySide 1.2.1](#)
- [NumPy 1.7.1](#)
- [SciPy 0.12.0](#)
- [matplotlib 1.4.2](#) and its requirements:
  - **Six 1.9.0** Install it on the command line by:

```
pip install six-1.9.0-py2.py3-none-any.whl
```

- dateutil 2.4.0
- pyparsing 2.0.3
- cx\_Freeze 4.3.4
- pywin32 219
- h5py HDF5 support, install one of the precompiled Windows packages, such as `h5py-2.4.0.win32-py2.7.exe`

## Mac OS X

After installing the required packages below a disk image file (.dmg) consisting of the application bundle is created by:

```
$ /usr/local/bin/python2 cxfreeze.py bdist_dmg
```

Alternatively, for testing purposes the bundle can be created without packaging into a disk image by:

```
$ /usr/local/bin/python2 cxfreeze.py bdist_mac
```

## Requirements

On a fresh installation of OS X 10.8 the following packages are required:

- **Xcode command line tools: for build essentials such as a compiler** ( `xcode461_cltools_10_86938245a.dmg` )
- Python 2.7.9
- Qt 4.8.6
- PySide 1.2.1 / Qt 4.8
- NumPy 1.7.1
- SciPy 0.12.0
- **matplotlib 1.4.2** Install it on the command line by:

```
$ /usr/local/bin/pip install matplotlib-1.4.2-*.whl
```

- h5py HDF5 support, install HDF5 from source first:

```
$ cd hdf5-src
$ ./configure --prefix=/usr/local
$ make && sudo make install
$
$ pip2 install h5py
```

- a modified `cx_Freeze 4.3.4` with local modifications for successful app freezing on OS X

Download the source and install it on the command line by:

```
$ hg clone https://bitbucket.org/ibressler/cx_freeze
$ cd cx_freeze
$ hg co 4.x
$ /usr/local/bin/python2 setup.py install
```

## Ubuntu/Linux

Similar to the procedure on Windows a self-contained archive containing all necessary libraries and files is created by:

```
$ python cxfreeze.py build_exe
```

## Requirements

On a fresh installation of Ubuntu Linux 14.04 LTS the following packages are required:

- apt-get install git build-essential python-setuptools python-dev liblapack-dev libfreetype6-dev tk-dev
- PySide 1.2.4
- NumPy 1.7.2
- SciPy 0.12.1
- matplotlib 1.4.2
- cx\_Freeze 4.3.4
- future 0.16.1
- h5py 2.2.1

## Internals

### class **Archiver**

Bases: object

#### **archive** (*targetPath*)

Creates an archive from the given absolute target directory path. The archive file will have the base name of the last directory of the given path.

#### **execName**

#### **getLogFilename** ()

**Warning:** method 'cxfreeze.Archiver.getLogFilename' undocumented

### class **Archiver7z** (*filetype='7z'*)

Bases: *cxfreeze.Archiver*

#### **archive** (*targetPath*)

**Warning:** method ‘cxfreeze.Archiver7z.archive’ undocumented

**class ArchiverZip**

Bases: `cxfreeze.Archiver`

**archive** (*targetPath*)

Expects an absolute target directory path

**includeModels** (*includeFilesLst*)

**Warning:** function ‘cxfreeze.includeModels’ undocumented

**sanitizeVersionNumber** (*number*)

Removes non-digits to be compatible with pywin32

## mcsas.autobuild module

## mcsas.mcsas\_test module

**getExpectedData** (*filename*)

**Warning:** function ‘mcsas\_test.getExpectedData’ undocumented

**getSettings** (*testfn, expectedfn*)

Test settings for mcsas routine. Using number of repetitions and contributions from expected test data to improve comparability.

**getTestData** (*filename*)

**Warning:** function ‘mcsas\_test.getTestData’ undocumented

**isEqualFloat** (*a, b, tol=1e-16*)

Return TRUE if both float arrays can be considered as equal. *tol* Tolerance in relative mean difference Supposed to be symmetrical.

**storeResultData** (*filename, result*)

**Warning:** function ‘mcsas\_test.storeResultData’ undocumented

**test** ()

Testing the algorithm in 1D. Atm, we just test as much as possible. Testing post-processing routines should be separated later as it’s deterministic and though easier to test.

## Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

---

### How to generate the documentation

---

#### Requirements

- Python, of course
- Sphinx package
- For Latex/PDF generation, there should be a latex environment installed

#### Generate a PDF document

```
cd <mcsas>/doc
make latexpdf
```

The resulting `McSAS.pdf` can be found in `<mcsas>/doc/_build/latex/`.

#### Generate HTML pages

```
cd <mcsas>/doc
make html
```

The entry point `index.html` can be found in `<mcsas>/doc/_build/html/`.

#### Update Source Code Documentation

```
cd <mcsas>/doc
make clean
make apidoc
```

This command automatically generates sphinx documentation files for all source code files in the directory. It assumes the current working directory is the McSAS root directory.

## CHAPTER 5

---

### Indices and tables

---

- `genindex`
- `search`



---

## Bibliography

---

- [Debye47] P. Debye, Molecular-weight determination by light scattering, *Journal of Physical and Colloid Chemistry*, 51:18–32, 1947.
- [Kholodenko93] A. L. Kholodenko. Analytical calculation of the scattering function for polymers of arbitrary flexibility using the dirac propagator. *Macromolecules*, 26:4179–4183, 1993.

### b

bases, 109  
bases.algorithm, 104  
bases.algorithm.algorithmbase, 92  
bases.algorithm.algorithmbase\_test, 93  
bases.algorithm.numbergenerator, 94  
bases.algorithm.parameter, 97  
bases.algorithm.parameter\_test, 103  
bases.dataset, 106  
bases.dataset.dataset, 105  
bases.dataset.rawarraymixin, 106  
bases.dataset.titlemixin, 106  
bases.model, 109  
bases.model.modeldata, 106  
bases.model.sasmodel, 107  
bases.model.scatteringmodel, 108  
mcsas.backgroundscalingfit, 7

### c

cxfreeze, 110

### d

datafile, 74  
datafile.arrayfile, 69  
datafile.asciifile, 70  
datafile.datafile, 71  
datafile.nxcansasfile, 72  
datafile.pdhfile, 73  
dataobj, 88  
dataobj.dataconfig, 78  
dataobj.dataconfig\_test, 79  
dataobj.dataobj, 80  
dataobj.datavector, 82  
dataobj.sasconfig, 82  
dataobj.sasconfig\_test, 86  
dataobj.sasdata, 86

### g

gui, 52

gui.algorithmwidget, 40  
gui.bases, 36  
gui.bases.datalist, 28  
gui.bases.dockwidget, 32  
gui.bases.logwidget, 33  
gui.bases.mainwindow, 25  
gui.bases.mainwindow.mainwindow, 24  
gui.bases.mainwindow.mainwindow\_rc, 25  
gui.bases.mainwindow.ui\_mainwindow, 25  
gui.bases.mixins, 28  
gui.bases.mixins.appsettings, 25  
gui.bases.mixins.contextmenuwidget, 26  
gui.bases.mixins.dropwidget, 27  
gui.bases.mixins.titlehandler, 27  
gui.bases.settingswidget, 34  
gui.bases.settingswidget\_test, 35  
gui.calc, 42  
gui.datawidget, 44  
gui.filelist, 45  
gui.liststyle, 45  
gui.mainwindow, 46  
gui.modelwidget, 47  
gui.optimizationwidget, 48  
gui.qt, 48  
gui.rangelist, 48  
gui.scientrybox, 50  
gui.settingsgroup, 51  
gui.utils, 40  
gui.utils.appversion, 37  
gui.utils.appversion.appversion, 36  
gui.utils.appversion.qappversion, 37  
gui.utils.dialoginteraction, 38  
gui.utils.displayexception, 38  
gui.utils.filedialog, 39  
gui.utils.progressdialog, 40  
gui.utils.signal, 40  
gui.utils.translate, 40  
gui.version, 52

## l

`log`, 54  
`log.log`, 52  
`log.sink`, 52  
`log.widgethandler`, 53

## m

`main`, 109  
`mcsas`, 16  
`mcsas.mcsas`, 8  
`mcsas.mcsasdefaultcfg`, 12  
`mcsas.mcsasparameters`, 13  
`mcsas_test`, 113  
`models`, 24  
`models.cylindersisotropic`, 16  
`models.cylindersisotropicaspect`, 17  
`models.cylindersradiallyisotropic`, 17  
`models.cylindersradiallyisotropictilted`, 18  
`models.ellipsoidalcoreshell`, 18  
`models.ellipsoidsisotropic`, 19  
`models.gaussianchain`, 20  
`models.kholodenko`, 21  
`models.lmadensesphere`, 21  
`models.sphere`, 22  
`models.sphericalcoreshell`, 23

## p

`mcsas.plotting`, 14

## u

`utils`, 68  
`utils.binning`, 54  
`utils.classproperty`, 55  
`utils.devtools`, 55  
`utils.error`, 55  
`utils.findmodels`, 56  
`utils.hdf`, 57  
`utils.lastpath`, 58  
`utils.loadstore`, 59  
`utils.mixedmethod`, 59  
`utils.parameter`, 59  
`utils.pickleinstancemethods`, 63  
`utils.propertynames`, 63  
`utils.tests`, 64  
`utils.units`, 65

## A

- absVolume() (CylindersIsotropic method), 16
- absVolume() (CylindersRadiallyIsotropic method), 17
- absVolume() (EllipsoidalCoreShell method), 19
- absVolume() (EllipsoidsIsotropic method), 19
- absVolume() (LMADenseSphere method), 22
- absVolume() (ScatteringModel method), 108
- absVolume() (Sphere method), 22
- absVolume() (SphericalCoreShell method), 23
- accumulate() (DataObj method), 80, 88
- action() (ContextMenuWidget method), 26
- activeParamCount (ScatteringModel attribute), 108
- activeParamNames (ScatteringModel attribute), 108
- activeParams (ScatteringModel attribute), 108
- activeRange (FitParameterBase attribute), 59
- activeVal (FitParameterBase attribute), 59
- activeValues (FitParameterBase attribute), 59
- add() (DataList method), 29
- addAttributes() (bases.algorithm.parameter.ParameterBase class method), 98
- addHandler() (in module log.log), 52
- addMenuEntry() (ContextMenuWidget method), 26
- addMenuEntryAction() (ContextMenuWidget method), 26
- addMenuSeparator() (ContextMenuWidget method), 26
- addToMenuState() (ContextMenuWidget method), 26
- addWatchDir() (LogWidget method), 33
- AdvancedSettings (class in gui.settingsgroup), 51
- aGoFsAlpha() (BackgroundScalingFit static method), 8
- algo (Calculator attribute), 42
- algorithm (AlgorithmWidget attribute), 40
- algorithm (ModelWidget attribute), 47
- AlgorithmBase (class in bases.algorithm.algorithmbase), 92
- AlgorithmError, 93
- AlgorithmNameError, 93
- AlgorithmParameterError, 93
- AlgorithmWidget (class in gui.algorithmwidget), 40
- allMenuStates (ContextMenuWidget attribute), 26
- analyse() (McSAS method), 10
- Angle (class in utils.units), 66
- append() (Histograms method), 62
- append() (LogWidget method), 33
- append() (RangeList method), 48
- appendFile() (datafile.AsciiFile class method), 75
- appendFile() (datafile.asciifile.AsciiFile class method), 70
- appendHeaderLine() (datafile.AsciiFile class method), 75
- appendHeaderLine() (datafile.asciifile.AsciiFile class method), 70
- AppError, 55
- application() (gui.utils.dialoginteraction.DialogInteraction class method), 38
- appSettings (AppSettings attribute), 25
- AppSettings (class in gui.bases.mixins.appsettings), 25
- AppVersion (class in gui.utils.appversion.appversion), 36
- appversion (LogWidget attribute), 33
- archive() (Archiver method), 112
- archive() (Archiver7z method), 112
- archive() (ArchiverZip method), 113
- Archiver (class in cxfreeze), 112
- Archiver7z (class in cxfreeze), 112
- ArchiverZip (class in cxfreeze), 113
- Area (class in utils.units), 66
- array() (in module mcsas.mcsas), 11
- array() (in module utils.binning), 54
- ArrayFile (class in datafile), 76
- ArrayFile (class in datafile.arrayfile), 69
- AsciiFile (class in datafile), 75
- AsciiFile (class in datafile.asciifile), 70
- assertDefaults() (in module dataobj.dataconfig\_test), 79
- assertDefaults() (in module dataobj.sasconfig\_test), 86
- assertName() (in module utils.tests), 64
- attributeNames() (bases.algorithm.parameter.ParameterBase class method), 98
- attributes (ParameterBase attribute), 98
- autoFollow (Histogram attribute), 60
- availableMagnitudeNames (Unit attribute), 67

## B

BackgroundScalingFit (class in mc-  
sas.backgroundscalingfit), 7  
 basename (OutputFilename attribute), 43  
 bases (module), 109  
 bases.algorithm (module), 104  
 bases.algorithm.algorithmbase (module), 92  
 bases.algorithm.algorithmbase\_test (module), 93  
 bases.algorithm.numbergenerator (module), 94  
 bases.algorithm.parameter (module), 97  
 bases.algorithm.parameter\_test (module), 103  
 bases.dataset (module), 106  
 bases.dataset.dataset (module), 105  
 bases.dataset.rawarraymixin (module), 106  
 bases.dataset.titlemixin (module), 106  
 bases.model (module), 109  
 bases.model.modeldata (module), 106  
 bases.model.sasmodel (module), 107  
 bases.model.scatteringmodel (module), 108  
 binCount (Histogram attribute), 61  
 binnedData (DataVector attribute), 82  
 binnedDataU (DataVector attribute), 82  
 binning1d() (in module utils.binning), 54  
 binningArray() (in module utils.binning), 54  
 binningWeighted1d() (in module utils.binning), 54  
 bins (Histogram attribute), 61  
 blockSigValueChanged() (AlgorithmWidget method), 40  
 buf (Sink attribute), 52  
 buildUi() (DataWidget method), 44

## C

calc() (BackgroundScalingFit method), 8  
 calc() (Histogram method), 61  
 calc() (Histograms method), 62  
 calc() (MainWindow method), 46  
 calc() (McSAS method), 10  
 calc() (ScatteringModel method), 108  
 calcIntensity() (SASModel method), 107  
 calcIntensity() (ScatteringModel method), 108  
 calcPcs() (in module models.kholodenko), 21  
 Calculator (class in gui.calc), 42  
 calculator (MainWindow attribute), 46  
 callback() (CallbackRegistry method), 78  
 CallbackRegistry (class in dataobj.dataconfig), 78  
 callbackSlots (CallbackRegistry attribute), 78  
 callbackSlots (DataConfig attribute), 78, 90  
 candidateFiles() (utils.findmodels.FindModels  
method), 56  
 canSmear (LMADenseSphere attribute), 22  
 canSmear (SASModel attribute), 107  
 canSmear (Sphere attribute), 22  
 cdf (Histogram attribute), 61  
 cfgwrite() (in module gui.calc), 43  
 chi() (BackgroundScalingFit static method), 8

child (DockWidget attribute), 32  
 chiNoBg() (BackgroundScalingFit static method), 8  
 chiSqr() (BackgroundScalingFit static method), 8  
 chisqrInt (ModelData attribute), 106  
 cInfo (class in mcsas.mcsasdefaultcfg), 12  
 classAndMethodName() (gui.utils.displayexception.DisplayException  
class method), 39  
 classname() (in module utils), 68  
 classproperty (class in utils.classproperty), 55  
 classproperty() (in module  
bases.algorithm.algorithmbase), 93  
 classproperty() (in module bases.algorithm.parameter),  
102  
 classproperty() (in module bases.dataset.dataset), 105  
 classproperty() (in module datafile.arrayfile), 69  
 classproperty() (in module datafile.datafile), 72  
 classproperty() (in module datafile.nxcansasfile), 73  
 classproperty() (in module datafile.pdhfile), 74  
 classproperty() (in module dataobj.dataobj), 81  
 classproperty() (in module dataobj.sasdata), 88  
 classproperty() (in module utils), 68  
 classproperty() (in module utils.findmodels), 56  
 classproperty() (in module utils.parameter), 63  
 clear() (DataList method), 29  
 clear() (Histograms method), 62  
 clear() (LogWidget method), 33  
 clearLayout() (AlgorithmWidget static method), 41  
 clearUi() (DataWidget method), 44  
 clip (ParameterNumerical attribute), 100  
 clip() (in module utils), 69  
 closeEvent() (DockWidget method), 32  
 closeEvent() (MainWindow method), 24, 46  
 config (DataObj attribute), 80, 88  
 configFromLast() (FileList method), 45  
 configType (DataObj attribute), 80, 88  
 configType (SASData attribute), 87, 89  
 connectInputWidgets() (SettingsWidget method), 35  
 contents() (LogWidget method), 33  
 ContextMenuWidget (class in  
gui.bases.mixins.contextmenuwidget), 26  
 contribParamBounds (McSASParameters attribute), 14  
 CoordinateFormat (class in mcsas.plotting), 14  
 copy() (ParameterBase method), 98  
 core() (in module models.kholodenko), 21  
 coreIntegral() (in module models.kholodenko), 21  
 count (DataObj attribute), 80, 88  
 count (SASData attribute), 87, 89  
 cumInt (ModelData attribute), 106  
 currentSelection() (DataList method), 29  
 cxfreeze (module), 110  
 CylindersIsotropic (class in models.cylindersisotropic),  
16  
 CylindersIsotropic (class in mod-  
els.cylindersisotropicaspect), 17

CylindersRadiallyIsotropic (class in models.cylindersradiallyisotropic), 17  
 CylindersRadiallyIsotropicTilted (class in models.cylindersradiallyisotropictilted), 18

## D

data (McSAS attribute), 10  
 data() (DataItem method), 28  
 data() (DataList method), 30  
 DataConfig (class in dataobj), 90  
 DataConfig (class in dataobj.dataconfig), 78  
 dataContent (SASData attribute), 87, 89  
 DataFile (class in datafile), 74  
 DataFile (class in datafile.datafile), 71  
 datafile (module), 74  
 datafile.arrayfile (module), 69  
 datafile.asciifile (module), 70  
 datafile.datafile (module), 71  
 datafile.nxcansasfile (module), 72  
 datafile.pdhfile (module), 73  
 dataId() (DataItem method), 28  
 DataItem (class in gui.bases.datalist), 28  
 DataList (class in gui.bases.datalist), 29  
 DataObj (class in dataobj), 88  
 DataObj (class in dataobj.dataobj), 80  
 dataobj (module), 88  
 dataobj.dataconfig (module), 78  
 dataobj.dataconfig\_test (module), 79  
 dataobj.dataobj (module), 80  
 dataobj.datavector (module), 82  
 dataobj.sasconfig (module), 82  
 dataobj.sasconfig\_test (module), 86  
 dataobj.sasdata (module), 86  
 dataRoot (NXcanSASFile attribute), 73  
 dataScaled() (BackgroundScalingFit method), 8  
 DataSet (class in bases.dataset.dataset), 105  
 DataVector (class in dataobj.datavector), 82  
 DataWidget (class in gui.datawidget), 44  
 DBG() (in module utils.devtools), 55  
 DBGF() (in module utils.devtools), 55  
 decimals (ParameterFloat attribute), 100  
 DecimalsError, 98  
 default() (ExtendedEncoder method), 12  
 DefaultSettings (class in gui.settingsgroup), 51  
 defaultSettings() (AppVersion method), 36  
 DefaultValueError, 98  
 DialogInteraction (class in gui.utils.dialoginteraction), 38  
 displayActiveRange (FitParameterBase attribute), 60  
 displayData (DisplayMixin attribute), 105  
 displayData() (dataobj.SASData class method), 89  
 displayData() (dataobj.sasdata.SASData class method), 87  
 displayData() (utils.parameter.Histogram class method), 61

displayDataDescr (DisplayMixin attribute), 105  
 displayDataDescr() (dataobj.SASData class method), 89  
 displayDataDescr() (dataobj.sasdata.SASData class method), 87  
 displayDataDescr() (utils.parameter.Histogram class method), 61  
 DisplayException (class in gui.utils.displayexception), 38  
 displayMagnitude (Unit attribute), 67  
 displayMagnitudeName (ParameterFloat attribute), 100  
 displayMagnitudeName (Unit attribute), 67  
 DisplayMixin (class in bases.dataset.dataset), 105  
 displayName (ParameterBase attribute), 98  
 displayValue (ParameterBase attribute), 98  
 displayValue (ParameterFloat attribute), 100  
 displayValueRange (ParameterFloat attribute), 100  
 displayValues (ParameterNumerical attribute), 101  
 DisplayValuesError, 98  
 DockWidget (class in gui.bases.dockwidget), 32  
 dragEnterEvent() (DropWidget method), 27  
 dropEvent() (DropWidget method), 27  
 DropWidget (class in gui.bases.mixins.dropwidget), 27  
 dtype() (bases.algorithm.numbergenerator.RandomXorShiftUniform class method), 95  
 dtype() (bases.algorithm.parameter.ParameterBase class method), 98  
 dtype() (bases.algorithm.parameter.ParameterBoolean class method), 99  
 dtype() (bases.algorithm.parameter.ParameterFloat class method), 100  
 dtype() (bases.algorithm.parameter.ParameterNumerical class method), 101  
 dtype() (bases.algorithm.parameter.ParameterString class method), 101  
 Dummy (class in bases.algorithm.parameter\_test), 103  
 dummy() (DummyAlgo method), 93  
 DummyAlgo (class in bases.algorithm.algorithmbase\_test), 93  
 dummyFunc() (Dummy method), 103  
 DynamicViscosity (class in utils.units), 66

## E

editEntry() (RangeList method), 49  
 EllipsoidalCoreShell (class in models.ellipsoidalcoreshell), 18  
 EllipsoidsIsotropic (class in models.ellipsoidsisotropic), 19  
 emit() (WidgetHandler method), 53  
 EmptySelection, 55  
 escapeAmp() (in gui.bases.mixins.contextmenuwidget), 27  
 eventLoop() (in module gui.mainwindow), 47  
 execName (Archiver attribute), 112  
 expandAll() (DataList method), 30  
 ExtendedEncoder (class in mcsas.mcsasdefaultcfg), 12



extensions() (datafile.DataFile class method), 74  
 extensions() (datafile.datafile.DataFile class method), 71

## F

f (DataObj attribute), 80, 88  
 factory() (AlgorithmBase method), 92  
 factory() (in module bases.algorithm.parameter), 102  
 factory() (mcsas.mcsas.McSAS class method), 10  
 fieldNames() (Moments static method), 62  
 fields (Moments attribute), 62  
 figInit() (PlotResults method), 15  
 fileDialog() (in module gui.utils.filedialog), 39  
 fileDialog() (MainWindow method), 46  
 fileDialogType() (in module gui.utils.filedialog), 39  
 FileNotFoundError, 55  
 fileFilter (DataFile attribute), 71, 75  
 fileFilter() (datafile.ArrayFile class method), 77  
 fileFilter() (datafile.arrayfile.ArrayFile class method), 69  
 fileFilter() (datafile.nxcansasfile.NXcanSASFile class method), 73  
 fileFilter() (datafile.PDHFile class method), 77  
 fileFilter() (datafile.pdhfile.PDHFile class method), 73  
 FileList (class in gui.filelist), 45  
 filename (DataFile attribute), 72, 75  
 filename (DataObj attribute), 80, 88  
 filename() (OutputFilename method), 43  
 filenameVerbose() (OutputFilename method), 43  
 findFiles() (in module utils.findmodels), 56  
 FindModels (class in utils.findmodels), 56  
 fitColumnsToContents() (DataList method), 30  
 fitLM() (BackgroundScalingFit method), 8  
 FitParameter() (in module utils.parameter), 59  
 FitParameterBase (class in utils.parameter), 59  
 FitParameterBoolean (class in utils.parameter), 60  
 FitParameterFloat (class in utils.parameter), 60  
 FitParameterLog (class in utils.parameter), 60  
 FitParameterNumerical (class in utils.parameter), 60  
 FitParameterString (class in utils.parameter), 60  
 fitSimplex() (BackgroundScalingFit method), 8  
 fixFilename() (in module utils), 69  
 fixTestParams (GaussianChain attribute), 20  
 fixTestParams (ScatteringModel attribute), 108  
 fixup() (SciEntryValidator method), 51  
 flush() (Sink method), 52  
 fmt (SciEntryBox attribute), 50  
 formatAlgoInfo() (PlotResults method), 15  
 formatData() (datafile.AsciiFile class method), 76  
 formatData() (datafile.asciifile.AsciiFile class method), 70  
 formatData() (datafile.PDHFile class method), 77  
 formatData() (datafile.pdhfile.PDHFile class method), 74  
 formatDisplayName (ParameterBase attribute), 98  
 formatRangeInfo() (PlotResults method), 15  
 formatRow() (datafile.AsciiFile class method), 76

formatRow() (datafile.asciifile.AsciiFile class method), 70  
 formatter() (in module log.log), 52  
 formatValue() (datafile.AsciiFile class method), 76  
 formatValue() (datafile.asciifile.AsciiFile class method), 70  
 formfactor() (CylindersIsotropic method), 16, 17  
 formfactor() (CylindersRadiallyIsotropic method), 17  
 formfactor() (CylindersRadiallyIsotropicTilted method), 18  
 formfactor() (EllipsoidalCoreShell method), 19  
 formfactor() (EllipsoidsIsotropic method), 19  
 formfactor() (GaussianChain method), 20  
 formfactor() (Kholodenko method), 21  
 formfactor() (LMADenseSphere method), 22  
 formfactor() (ScatteringModel method), 108  
 formfactor() (Sphere method), 22  
 formfactor() (SphericalCoreShell method), 23  
 Fraction (class in utils.units), 66  
 full (VectorResult attribute), 63  
 funcNotInFuncList() (in module dataobj.dataconfig), 79

## G

GaussianChain (class in models.gaussianchain), 20  
 GaussianSmearing (class in dataobj.sasconfig), 82  
 gen2DMeasVal() (McSAS method), 10  
 generate() (FitParameterNumerical method), 60  
 generate() (ParameterBase method), 99  
 generate() (ParameterNumerical method), 101  
 generateParameters() (ScatteringModel method), 108  
 generateTests() (bases.algorithm.numbergenerator.RandomXorShiftUniform class method), 96  
 generateValues() (in module bases.algorithm.parameter), 102  
 generator (ParameterNumerical attribute), 101  
 get() (bases.algorithm.numbergenerator.NumberGenerator class method), 94  
 get() (bases.algorithm.numbergenerator.RandomExponential class method), 94  
 get() (bases.algorithm.numbergenerator.RandomUniform class method), 95  
 get() (bases.algorithm.numbergenerator.RandomXorShiftUniform class method), 95  
 get() (bases.algorithm.parameter.ParameterBase class method), 99  
 get() (SettingsWidget method), 35  
 get() (utils.lastpath.LastPath class method), 58  
 getCallerInfo() (in module utils.hdf), 58  
 getCommandLineArguments() (MainWindow method), 24  
 getDataObj() (ArrayFile method), 69, 77  
 getDataObj() (DataFile method), 72, 75  
 getDataObj() (datafile.nxcansasfile.NXcanSASFile class method), 73

- getEditingFinishedSignal() (SettingsWidget method), 35
  - getExpectedData() (in module mcsas\_test), 113
  - getFileFilter() (in module datafile), 78
  - getHomeDir() (in module utils.lastpath), 58
  - getInputWidget() (SettingsWidget method), 35
  - getItemIndex() (in module gui.rangelist), 49
  - getItemProperty() (DataItem method), 28
  - getLogFilename() (Archiver method), 112
  - getMessage() (utils.error.AppError class method), 55
  - getModelData() (ScatteringModel method), 108
  - getOpenFiles() (in module gui.utils.filedialog), 39
  - getPar() (cInfo method), 12
  - getParametersFromFilename() (bases.model.scatteringmodel.ScatteringModel class method), 108
  - getParVal() (cInfo method), 13
  - getQ() (SASModel method), 107
  - getRef() (RandomXorShiftUniformTest method), 96
  - getSaveDirectory() (in module gui.utils.filedialog), 39
  - getSaveFile() (in module gui.utils.filedialog), 39
  - getScriptPath() (in module main), 109
  - getSearchPaths() (utils.findmodels.FindModels class method), 56
  - getSeed() (bases.algorithm.numbergenerator.RandomXorShiftUniform class method), 95
  - getSettings() (in module mcsas\_test), 113
  - getTestData() (in module mcsas\_test), 113
  - getTextSize() (in module mcsas.plotting), 16
  - getValue() (SettingsWidget method), 35
  - getWidget() (SettingsWidget method), 35
  - getWidgetHandlers() (in module log), 54
  - gui (module), 52
  - gui.algorithmwidget (module), 40
  - gui.bases (module), 36
  - gui.bases.datalist (module), 28
  - gui.bases.dockwidget (module), 32
  - gui.bases.logwidget (module), 33
  - gui.bases.mainwindow (module), 25
  - gui.bases.mainwindow.mainwindow (module), 24
  - gui.bases.mainwindow.mainwindow\_rc (module), 25
  - gui.bases.mainwindow.ui\_mainwindow (module), 25
  - gui.bases.mixins (module), 28
  - gui.bases.mixins.appsettings (module), 25
  - gui.bases.mixins.contextmenuwidget (module), 26
  - gui.bases.mixins.dropwidget (module), 27
  - gui.bases.mixins.titlehandler (module), 27
  - gui.bases.settingswidget (module), 34
  - gui.bases.settingswidget\_test (module), 35
  - gui.calc (module), 42
  - gui.datawidget (module), 44
  - gui.filelist (module), 45
  - gui.liststyle (module), 45
  - gui.mainwindow (module), 46
  - gui.modelwidget (module), 47
  - gui.optimizationwidget (module), 48
  - gui.qt (module), 48
  - gui.rangelist (module), 48
  - gui.scientrybox (module), 50
  - gui.settingsgroup (module), 51
  - gui.utils (module), 40
  - gui.utils.appversion (module), 37
  - gui.utils.appversion.appversion (module), 36
  - gui.utils.appversion.qappversion (module), 37
  - gui.utils.dialoginteraction (module), 38
  - gui.utils.displayexception (module), 38
  - gui.utils.filedialog (module), 39
  - gui.utils.progressdialog (module), 40
  - gui.utils.signal (module), 40
  - gui.utils.translate (module), 40
  - gui.version (module), 52
- ## H
- halfTrapzPDF() (TrapezoidSmearing method), 85
  - hash32() (DataItem static method), 28
  - hashNumpyArray() (in module utils), 69
  - hasSelection() (DataList method), 30
  - hasUncertainties (DataObj attribute), 80, 88
  - hdfCleanup() (in module utils.hdf), 57
  - hdfLoad() (Calculator method), 42
  - hdfLoad() (utils.hdf.HDFMixin class method), 57
  - HDFMixin (class in utils.hdf), 57
  - hdfStore() (HDFMixin method), 57
  - hdfStoreAsMember() (FitParameterBase method), 60
  - hdfStoreAsMember() (ParameterBase method), 99
  - hdfStoreAsMember() (ParameterFloat method), 100
  - hdfStoreAsMember() (ParameterNumerical method), 101
  - hdfWrite() (AlgorithmBase method), 92
  - hdfWrite() (bases.algorithm.numbergenerator.NumberGenerator class method), 94
  - hdfWrite() (Calculator method), 42
  - hdfWrite() (DataConfig method), 78, 90
  - hdfWrite() (DataObj method), 81, 88
  - hdfWrite() (DataVector method), 82
  - hdfWrite() (HDFMixin method), 57
  - hdfWrite() (Histogram method), 61
  - hdfWrite() (Histograms method), 62
  - hdfWrite() (ModelData method), 107
  - hdfWrite() (ParameterBase method), 99
  - hdfWrite() (SASConfig method), 83
  - hdfWrite() (ScatteringModel method), 108
  - hdfWrite() (SmearingConfig method), 84
  - hdfWrite() (Unit method), 67
  - HDFWriter (class in utils.hdf), 57
  - Histogram (class in utils.parameter), 60
  - histogram() (McSAS method), 10
  - Histograms (class in utils.parameter), 62
  - histograms (FitParameterBase attribute), 60



## I

includeModels() (in module cxfreeze), 113  
 indent (Calculator attribute), 42  
 indicateCorrectness() (SciEntryBox method), 50  
 initConfig() (DataObj method), 81, 88  
 InitError, 55  
 initUi() (MainWindow method), 46  
 inputValid() (GaussianSmearing method), 82  
 inputValid() (TrapezoidSmearing method), 85  
 inputWidgets (AlgorithmWidget attribute), 41  
 instance() (gui.utils.dialoginteraction.DialogInteraction class method), 38  
 integralProps() (utils.parameter.Histogram class method), 61  
 intensity (Moments attribute), 62  
 invName() (Unit static method), 67  
 is2d (DataConfig attribute), 78, 90  
 is2d (DataObj attribute), 81, 88  
 isActive (FitParameterBase attribute), 60  
 isActiveFitParam() (in module utils.parameter), 63  
 isCallable() (in module utils.tests), 64  
 isCopyAvailable() (LogWidget method), 33  
 isDataType() (bases.algorithm.parameter.ParameterBase class method), 99  
 isDataType() (bases.algorithm.parameter.ParameterFloat class method), 100  
 isDataType() (bases.algorithm.parameter.ParameterNumeric class method), 101  
 isDataType() (bases.algorithm.parameter.ParameterString class method), 101  
 isEmpty() (DataList method), 30  
 isEmpty() (LogWidget method), 34  
 isEqualFloat() (in module mcsas\_test), 113  
 isFitParam() (in module utils.parameter), 63  
 isFrozen() (in module utils.tests), 64  
 isInteger() (in module utils.tests), 64  
 isLinux() (in module utils.tests), 64  
 isList() (in module utils.tests), 64  
 isMac() (in module utils.tests), 64  
 isMap() (in module utils.tests), 64  
 isEmptyString() (in module utils.tests), 65  
 isEmpty() (DataList method), 30  
 isNotEmpty() (in module gui.algorithmwidget), 42  
 isNumber() (in module utils.tests), 65  
 isRemovable (DataItem attribute), 28  
 isRemovable (DisplayMixin attribute), 105  
 isRemovableSelected() (DataList method), 30  
 isSet() (in module utils.tests), 65  
 isStopped() (Calculator method), 42  
 isString() (in module utils.tests), 65  
 isTopLevelItem() (DataItem method), 28  
 isValid() (gui.utils.appversion.appversion.AppVersion class method), 36

isValidInput() (bases.dataset.rawarraymixin.RawArrayMixin class method), 106  
 isWindows() (in module utils.tests), 65  
 itemDoubleClicked() (DataList method), 30  
 itemDoubleClicked() (FileList method), 45  
 itemsHaveChildren() (DataList method), 30  
 itemUpdate() (DataList method), 30  
 itemUpdate() (RangeList method), 49

## K

keyPressEvent() (MainWindow method), 46  
 keys (AlgorithmWidget attribute), 41  
 Kholodenko (class in models.kholodenko), 21  
 kurtosis (Moments attribute), 63

## L

LastPath (class in utils.lastpath), 58  
 leaveEvent() (DataList method), 31  
 Length (class in utils.units), 66  
 libraryPath() (utils.findmodels.FindModels class method), 56  
 limit (DataVector attribute), 82  
 limsString (DataVector attribute), 82  
 line() (NXSHeader method), 72  
 line() (PDHHeader method), 74, 77  
 listIndex() (DataItem method), 28  
 LMADenseSphere (class in models.lmadensesphere), 21  
 loadData() (DataList method), 31  
 loadData() (FileList method), 45  
 loadData() (RangeList method), 49  
 LoadError, 55  
 loadParameters() (McSASParameters method), 14  
 loadParams() (cInfo method), 13  
 location (HDFWriter attribute), 57  
 log (module), 54  
 log() (HDFWriter method), 57  
 log.log (module), 52  
 log.sink (module), 52  
 log.widgethandler (module), 53  
 LogWidget (class in gui.bases.logwidget), 33  
 lower (Histogram attribute), 61  
 lower (RandomExponential attribute), 94  
 lowerDisplay (Histogram attribute), 61  
 lshift() (in module bases.algorithm.numbergenerator), 97

## M

magnitude() (utils.units.Unit class method), 68  
 magnitudeConversion (Temperature attribute), 67  
 magnitudeConversion (Unit attribute), 68  
 magnitudeMapping (Unit attribute), 68  
 main (module), 109  
 main() (in module main), 109  
 MainWindow (class in gui.bases.mainwindow.mainwindow), 24

- MainWindow (class in gui.mainwindow), 46  
 makeAbsolutePath() (in module main), 109  
 makeAlternatingRowColorsTransparent() (in module gui.liststyle), 45  
 makeConfigUi() (DataWidget method), 44  
 makeDefault() (bases.algorithm.algorithmbase.AlgorithmBase class method), 92  
 makeFilter() (in module gui.utils.filedialog), 39  
 makeSetting() (AlgorithmWidget method), 41  
 makeWidgets() (AlgorithmWidget method), 41  
 max (ParameterNumerical attribute), 101  
 maxLines() (datafile.nxcansasfile.NXSHeader class method), 72  
 maxLines() (datafile.pdhfile.PDHHeader class method), 74  
 maxLines() (datafile.PDHHeader class method), 77  
 mcFit() (McSAS method), 11  
 mcpopen() (in module utils), 69  
 McSAS (class in mcsas.mcsas), 8  
 mcsas (module), 16  
 mcsas.backgroundscalingfit (module), 7  
 mcsas.mcsas (module), 8  
 mcsas.mcsasdefaultcfg (module), 12  
 mcsas.mcsasparameters (module), 13  
 mcsas.plotting (module), 14  
 mcsas\_test (module), 113  
 McSASParameters (class in mcsas.mcsasparameters), 13  
 mean (Moments attribute), 63  
 mean (VectorResult attribute), 63  
 menuEntries() (ContextMenuWidget method), 26  
 min (ParameterNumerical attribute), 101  
 mixedmethod (class in utils.mixedmethod), 59  
 model (Calculator attribute), 43  
 model (McSAS attribute), 11  
 model (McSASParameters attribute), 14  
 model (ModelWidget attribute), 47  
 modelActiveParams() (Calculator method), 43  
 ModelData (class in bases.model.modeldata), 106  
 modelDataType() (SASModel method), 107  
 modelDataType() (ScatteringModel method), 108  
 modelParams() (Calculator method), 43  
 models (module), 24  
 models.cylindersisotropic (module), 16  
 models.cylindersisotropicaspect (module), 17  
 models.cylindersradiallyisotropic (module), 17  
 models.cylindersradiallyisotropicilted (module), 18  
 models.ellipsoidalcoreshell (module), 18  
 models.ellipsoidsisotropic (module), 19  
 models.gaussianchain (module), 20  
 models.kholodenko (module), 21  
 models.lmadensesphere (module), 21  
 models.sphere (module), 22  
 models.sphericalcoreshell (module), 23  
 modelType (DataObj attribute), 81, 88  
 modelType (SASData attribute), 87, 89  
 ModelWidget (class in gui.modelwidget), 47  
 Moments (class in utils.parameter), 62  
 moments (Histogram attribute), 61  
 mouseDoubleClickEvent() (RangeList method), 49  
 msg (AppError attribute), 55
- ## N
- name (DataFile attribute), 72, 75  
 name (DataVector attribute), 82  
 name (ParameterBase attribute), 99  
 name() (AppVersion method), 37  
 name() (bases.algorithm.algorithmbase.AlgorithmBase class method), 92  
 name() (utils.units.Unit class method), 68  
 new() (utils.error.VerboseError class method), 56  
 newline (AsciiFile attribute), 70, 76  
 next() (bases.algorithm.numbergenerator.RandomXorShiftUniform class method), 95  
 nolog (Calculator attribute), 43  
 NoUnit (class in utils.units), 66  
 np\_array() (in module bases.algorithm.algorithmbase\_test), 93  
 np\_array() (in module datafile.arrayfile), 70  
 np\_array() (in module datafile.asciifile), 71  
 number() (AppVersion method), 37  
 numberFormat() (SciEntryBox static method), 50  
 NumberGenerator (class in bases.algorithm.numbergenerator), 94  
 numParams (ModelData attribute), 107  
 NXcanSASFile (class in datafile.nxcansasfile), 73  
 NXSHheader (class in datafile.nxcansasfile), 72
- ## O
- observability (Histogram attribute), 61  
 onBackendUpdate() (AlgorithmWidget method), 41  
 onBackendUpdate() (DataWidget method), 44  
 onCloseSignal (MainWindow attribute), 46  
 onCloseSlot() (LogWidget method), 34  
 onDataSelected() (DataWidget method), 44  
 onDataSelected() (ModelWidget method), 47  
 onDataSelected() (OptimizationWidget method), 48  
 onEmptyDataList() (DataWidget method), 44  
 onRemoval() (RangeList method), 49  
 onStartStopClick() (MainWindow method), 46  
 onStartupSignal (MainWindow attribute), 24  
 onUmbraUpdate() (TrapezoidSmearing method), 85  
 onUpdatedX0() (DataConfig method), 78, 90  
 onUpdatedX0() (SASConfig method), 83  
 onUpdatedX1() (DataConfig method), 78, 90  
 onUpdatedX1() (SASConfig method), 83  
 onValueUpdate (ParameterBase attribute), 99  
 onVisibilityChange() (DockWidget method), 33  
 open() (utils.hdf.HDFWriter class method), 57

OptimizationWidget (class in gui.optimizationwidget), 48  
 organizationDomain() (AppVersion method), 37  
 organizationName() (AppVersion method), 37  
 outDir (OutputFilename attribute), 43  
 output() (RangeDialog method), 48  
 OutputFilename (class in gui.calc), 43

## P

p (RandomXorShiftUniform attribute), 95  
 p (SASData attribute), 87, 89  
 param (AlgorithmBase attribute), 92  
 param (Histogram attribute), 61  
 paramCount (AlgorithmBase attribute), 92  
 paramDefFile (McSASParameters attribute), 14  
 Parameter() (in module bases.algorithm), 104  
 Parameter() (in module bases.algorithm.algorithmbase\_test), 93  
 Parameter() (in module bases.algorithm.parameter\_test), 103  
 Parameter() (in module dataobj.dataconfig), 79  
 Parameter() (in module dataobj.sasconfig), 83  
 Parameter() (in module mcsas.mcsasdefaultcfg), 12  
 Parameter() (in module mcsas.mcsasparameters), 14  
 Parameter() (in module models.cylindersisotropic), 16  
 Parameter() (in module models.cylindersisotropicaspect), 17  
 Parameter() (in module models.cylindersradiallyisotropic), 18  
 Parameter() (in module models.cylindersradiallyisotropictilted), 18  
 Parameter() (in module models.ellipsoidalcoreshell), 19  
 Parameter() (in module models.ellipsoidsisotropic), 20  
 Parameter() (in module models.gaussianchain), 20  
 Parameter() (in module models.lmadensesphere), 22  
 Parameter() (in module models.sphere), 22  
 Parameter() (in module models.sphericalcoreshell), 23  
 Parameter() (in module utils.parameter), 63  
 ParameterBase (class in bases.algorithm.parameter), 98  
 ParameterBoolean (class in bases.algorithm.parameter), 99  
 ParameterError, 99  
 ParameterFloat (class in bases.algorithm.parameter), 100  
 ParameterGeneratorError, 100  
 ParameterLog (class in bases.algorithm.parameter), 100  
 ParameterNameError, 100  
 parameterNames (cInfo attribute), 13  
 ParameterNumerical (class in bases.algorithm.parameter), 100  
 parameters (cInfo attribute), 13  
 parameters (CylindersIsotropic attribute), 16, 17  
 parameters (CylindersRadiallyIsotropic attribute), 17  
 parameters (CylindersRadiallyIsotropictilted attribute), 18  
 parameters (DataConfig attribute), 78, 91  
 parameters (DummyAlgo attribute), 93  
 parameters (EllipsoidalCoreShell attribute), 19  
 parameters (EllipsoidsIsotropic attribute), 19  
 parameters (GaussianChain attribute), 20  
 parameters (GaussianSmearing attribute), 82  
 parameters (Kholodenko attribute), 21  
 parameters (LMADenseSphere attribute), 22  
 parameters (McSASParameters attribute), 14  
 parameters (SmearingConfig attribute), 84  
 parameters (Sphere attribute), 23  
 parameters (SphericalCoreShell attribute), 23  
 parameters (TrapezoidSmearing attribute), 85  
 ParameterString (class in bases.algorithm.parameter), 101  
 paramName (Histogram attribute), 61  
 params (AlgorithmBase attribute), 92  
 parseConfig() (cInfo method), 13  
 parseLines() (ArrayFile method), 69, 77  
 parseLines() (AsciiFile method), 70, 76  
 parseLines() (PDHFile method), 74, 77  
 PDHFile (class in datafile), 77  
 PDHFile (class in datafile.pdhfile), 73  
 PDHHeader (class in datafile), 77  
 PDHHeader (class in datafile.pdhfile), 74  
 pickleLoad() (in module utils.loadstore), 59  
 pickleStore() (in module utils.loadstore), 59  
 pickUnit() (McSASParameters method), 14  
 pLimsString (SASData attribute), 87, 90  
 plot() (McSAS method), 11  
 plot() (PlotSeriesStats method), 15  
 plot1D() (PlotResults method), 15  
 plotGrid() (mcsas.plotting.PlotResults class method), 15  
 plotHist() (PlotResults method), 15  
 plotInfo() (PlotResults method), 15  
 plotPartial() (PlotResults method), 15  
 PlotResults (class in mcsas.plotting), 14  
 PlotSeriesStats (class in mcsas.plotting), 15  
 plotStats() (PlotResults method), 15  
 postProcess() (Calculator method), 43  
 prefix() (utils.error.VerboseError class method), 56  
 prepare() (Calculator method), 43  
 prepared (SmearingConfig attribute), 84  
 prepareSmearing() (SASConfig method), 84  
 process() (Sink method), 53  
 processEventLoop() (in module gui.utils), 40  
 ProgressDialog (class in gui.utils.progressdialog), 40  
 properties() (utils.propertynames.PropertyNames class method), 63  
 PropertyNames (class in utils.propertynames), 63  
 propNames() (utils.propertynames.PropertyNames class method), 63

## Q

q (SASData attribute), 87, 90

- QAppVersion (class in gui.utils.appversion.qappversion), 37
- qCleanupResources() (in module gui.bases.mainwindow.mainwindow\_rc), 25
- qInitResources() (in module gui.bases.mainwindow.mainwindow\_rc), 25
- qLimsString (SASData attribute), 87, 90
- qOffset (SmearingConfig attribute), 84
- query() (gui.utils.dialoginteraction.DialogInteraction class method), 38
- queryInstance() (gui.utils.dialoginteraction.DialogInteraction class method), 38
- ## R
- RandomExponential (class bases.algorithm.numbergenerator), 94
- RandomExponential1 (class bases.algorithm.numbergenerator), 94
- RandomExponential2 (class bases.algorithm.numbergenerator), 94
- RandomExponential3 (class bases.algorithm.numbergenerator), 95
- RandomUniform (class bases.algorithm.numbergenerator), 95
- RandomXorShiftUniform (class bases.algorithm.numbergenerator), 95
- RandomXorShiftUniformTest (class bases.algorithm.numbergenerator), 96
- RangeDialog (class in gui.rangelist), 48
- RangeList (class in gui.rangelist), 48
- rawArray (ArrayFile attribute), 69, 77
- rawArray (NXcanSASFile attribute), 73
- rawArray (RawArrayMixin attribute), 106
- RawArrayMixin (class in bases.dataset.rawarraymixin), 106
- rawData (DataVector attribute), 82
- rawDataU (DataVector attribute), 82
- readArray() (AsciiFile method), 70, 76
- readFile() (AsciiFile method), 70, 76
- readFile() (DataFile method), 72, 75
- readFile() (NXcanSASFile method), 73
- readItem() (NXcanSASFile method), 73
- readTuple() (AsciiFile method), 71, 76
- rearrangeWidgets() (in module gui.algorithmwidget), 42
- rearrangeWidgets() (SettingsGroup method), 51
- recalc() (RangeList method), 49
- register() (CallbackRegistry method), 78
- registerUpdateFunc() (TitleHandler method), 27
- remove() (DataItem method), 28
- removeHandler() (in module log.log), 52
- removeItems() (DataList method), 31
- removeMenuEntries() (ContextMenuWidget method), 27
- removeSelected() (DataList method), 31
- removeWidgets() (AlgorithmWidget static method), 41
- reorder() (in module utils.findmodels), 57
- replaceHandler() (in module log.log), 52
- replaceStdOutErr() (in module log.log), 52
- reraiseLast() (DataList method), 31
- resizeEvent() (AlgorithmWidget method), 41
- resizeEvent() (ToolBox method), 47
- resizeWidgets() (AlgorithmWidget method), 41
- resizeWidgets() (OptimizationWidget method), 48
- resizeWidgets() (SettingsGridWidget method), 42
- restoreSession() (AlgorithmWidget method), 41
- restoreSession() (DataWidget method), 44
- restoreSession() (ModelWidget method), 47
- restoreSession() (OptimizationWidget method), 48
- restoreSession() (RangeList method), 49
- restoreSettings() (MainWindow method), 24, 46
- result (McSAS attribute), 11
- result (ResultMixin attribute), 105
- ResultMixin (class in bases.dataset.dataset), 105
- retranslateUi() (Ui\_MainWindow method), 25
- rootName() (utils.findmodels.FindModels class method), 56
- rshift() (in module bases.algorithm.numbergenerator), 97
- run() (DialogInteraction method), 38
- rUnit (SASData attribute), 87, 90
- ## S
- s (RandomXorShiftUniform attribute), 95
- sampleName (DataConfig attribute), 78, 91
- sampleName (DataObj attribute), 81, 89
- sanitized (DataVector attribute), 82
- sanitizedU (DataVector attribute), 82
- sanitizeReadFilename() (DataFile static method), 72, 75
- sanitizeVersionNumber() (in module cxfreeze), 113
- sanitizeWriteFilename() (datafile.DataFile class method), 75
- sanitizeWriteFilename() (datafile.datafile.DataFile class method), 72
- SASConfig (class in dataobj.sasconfig), 83
- SASData (class in dataobj), 89
- SASData (class in dataobj.sasdata), 87
- SASModel (class in bases.model.sasmodel), 107
- SASModelData (class in bases.model.modeldata), 107
- saveToFile() (LogWidget method), 34
- ScatteringIntensity (class in utils.units), 66
- ScatteringModel (class in bases.model.scatteringmodel), 108
- ScatteringVector (class in utils.units), 67
- SciEntryBox (class in gui.scientrybox), 50
- SciEntryValidator (class in gui.scientrybox), 51
- scrollToBottom() (LogWidget method), 34
- scrollToTop() (LogWidget method), 34
- selectAll() (DataList method), 31



- selectionChanged() (DataList method), 31
- selectModel() (ModelWidget method), 47
- separator (AsciiFile attribute), 71, 76
- seriesKey (DataObj attribute), 81, 89
- seriesKeyName (DataObj attribute), 81, 89
- seriesKeyValue (DataObj attribute), 81, 89
- set() (bases.algorithm.parameter.ParameterBase class method), 99
- set() (SettingsWidget method), 35
- set() (utils.lastpath.LastPath class method), 58
- setActive (FitParameterBase attribute), 60
- setActiveRange (FitParameterBase attribute), 60
- setActiveVal (FitParameterBase attribute), 60
- setActiveValues (FitParameterBase attribute), 60
- setAlignment() (DataItem method), 28
- setAttributes (ParameterBase attribute), 99
- setAxis() (PlotResults method), 15
- setBackgroundColorStyle() (in module gui.liststyle), 45
- setChanged() (DataItem method), 29
- setClicked() (DataItem method), 29
- setConfig() (DataObj method), 81, 89
- setCopyAvailable() (LogWidget method), 34
- setCurrentIndex() (DataList method), 31
- setDataConfig() (FileList method), 45
- setDecimals (ParameterFloat attribute), 100
- setDisplayActiveRange (FitParameterBase attribute), 60
- setDisplayNames (ParameterBase attribute), 99
- setDisplayValue (ParameterBase attribute), 99
- setDisplayValue (ParameterFloat attribute), 100
- setDisplayValues (ParameterNumerical attribute), 101
- setFilename() (DataFile method), 72, 75
- setFilename() (DataObj method), 81, 89
- setGenerator (ParameterNumerical attribute), 101
- setHeader() (DataList method), 32
- setHistograms (FitParameterBase attribute), 60
- setIntPoints() (GaussianSmearing method), 82
- setIntPoints() (TrapezoidSmearing method), 85
- setIsActive (FitParameterBase attribute), 60
- setMaximum() (SciEntryBox method), 50
- setMinimum() (SciEntryBox method), 50
- setName() (bases.algorithm.algorithmbase.AlgorithmBase class method), 92
- setName() (bases.algorithm.parameter.ParameterBase class method), 99
- setOnValueUpdate (ParameterBase attribute), 99
- setParam (AlgorithmBase attribute), 92
- setParams() (bases.algorithm.algorithmbase.AlgorithmBase class method), 92
- setParVal() (cInfo method), 13
- setPrefix() (SciEntryBox method), 50
- setRange() (SciEntryBox method), 50
- setRawArray() (RawArrayMixin method), 106
- setRootGroup() (AppSettings method), 25
- setSeed() (bases.algorithm.numbergenerator.RandomXorShiftUniform class method), 95
- setSphericalSizeRange() (ModelWidget method), 47
- setStatsWidget() (ModelWidget method), 47
- setStepping (ParameterNumerical attribute), 101
- setSuffix (ParameterFloat attribute), 100
- setSuffix (ParameterNumerical attribute), 101
- SettingsGridWidget (class in gui.algorithmwidget), 42
- SettingsGroup (class in gui.settingsgroup), 51
- settingsKey() (QAppVersion method), 37
- SettingsWidget (class in gui.bases.settingswidget), 34
- setUnit (ParameterFloat attribute), 100
- setup() (gui.bases.mixins.titlehandler.TitleHandler class method), 28
- setUp() (RandomXorShiftUniformTest method), 96
- setupUi() (DataList method), 32
- setupUi() (FileList method), 45
- setupUi() (MainWindow method), 46
- setupUi() (RangeList method), 49
- setupUi() (TestSettings method), 35
- setupUi() (Ui\_MainWindow method), 25
- setValue (ParameterBase attribute), 99
- setValue (ParameterNumerical attribute), 101
- setValue() (SciEntryBox method), 50
- setValue() (SettingsWidget method), 35
- setValueRange (FitParameterBase attribute), 60
- setValueRange (ParameterNumerical attribute), 101
- setValueRange() (bases.algorithm.parameter.ParameterString class method), 102
- shannonChannelEst() (SASData method), 87, 90
- shannonChannelEstText (SASData attribute), 87, 90
- shortName (CylindersIsotropic attribute), 16, 17
- shortName (CylindersRadiallyIsotropic attribute), 17
- shortName (CylindersRadiallyIsotropicTilted attribute), 18
- shortName (DummyAlgo attribute), 93
- shortName (EllipsoidalCoreShell attribute), 19
- shortName (EllipsoidsIsotropic attribute), 19
- shortName (GaussianChain attribute), 20
- shortName (Kholodenko attribute), 21
- shortName (LMADenseSphere attribute), 22
- shortName (SASConfig attribute), 84
- shortName (SmearingConfig attribute), 84
- shortName (Sphere attribute), 23
- shortName (SphericalCoreShell attribute), 23
- show() (MainWindow method), 24
- show() (PlotSeriesStats method), 15
- showAdvanced() (AdvancedSettings method), 51
- showParams (AlgorithmBase attribute), 92
- showParams (DataConfig attribute), 79, 91
- showParams (GaussianSmearing attribute), 82
- showParams (SASConfig attribute), 84
- showParams (TrapezoidSmearing attribute), 85
- siData (DataVector attribute), 82

- siDataU (DataVector attribute), 82
- sigBackendUpdated (AlgorithmWidget attribute), 41
- sigConfig (DataWidget attribute), 44
- sigEditingFinished (DataList attribute), 32
- sigEmpty (DataList attribute), 32
- sigReceivedUrls (DataList attribute), 32
- sigRemovedData (DataList attribute), 32
- sigSelectedData (DataList attribute), 32
- sigSphericalSizeRange (FileList attribute), 45
- sigUpdatedData (DataList attribute), 32
- sigValueChanged (SettingsWidget attribute), 35
- sigValuesChanged (SettingsWidget attribute), 35
- siMagnitude (Unit attribute), 68
- siMagnitudeName (Unit attribute), 68
- Sink (class in log.sink), 52
- skew (Moments attribute), 63
- SLD (class in utils.units), 66
- smearing (SASConfig attribute), 84
- SmearingConfig (class in dataobj.sasconfig), 84
- sourceName() (dataobj.DataObj class method), 89
- sourceName() (dataobj.dataobj.DataObj class method), 81
- sourceName() (dataobj.SASData class method), 90
- sourceName() (dataobj.sasdata.SASData class method), 87
- Sphere (class in models.sphere), 22
- SphericalCoreShell (class in models.sphericalcoreshell), 23
- sphericalSizeEst() (SASData method), 87, 90
- sphericalSizeEstText (SASData attribute), 87, 90
- sset (ModelData attribute), 107
- std (VectorResult attribute), 63
- StdErrSink (class in log.sink), 53
- StdOutSink (class in log.sink), 53
- stepping (ParameterNumerical attribute), 101
- SteppingError, 102
- stop() (Calculator method), 43
- storeResultData() (in module mcsas\_test), 113
- storeSession() (AlgorithmWidget method), 41
- storeSession() (DataWidget method), 44
- storeSession() (ModelWidget method), 47
- storeSession() (OptimizationWidget method), 48
- storeSession() (RangeList method), 49
- storeSettings() (MainWindow method), 24, 47
- suffix (ParameterFloat attribute), 100
- suffix (ParameterNumerical attribute), 101
- SuffixError, 102
- surface() (ScatteringModel method), 108
- surface() (Sphere method), 23
- test() (bases.model.scatteringmodel.ScatteringModel class method), 108
- test() (in module mcsas\_test), 113
- test() (in module models.gaussianchain), 20
- test() (in module models.kholodenko), 21
- test() (in module models.sphere), 23
- test0() (RandomXorShiftUniformTest method), 96
- test1() (RandomXorShiftUniformTest method), 96
- test2() (RandomXorShiftUniformTest method), 97
- test3() (RandomXorShiftUniformTest method), 97
- test4() (RandomXorShiftUniformTest method), 97
- testCallbacks() (in module dataobj.dataconfig\_test), 80
- testFloatingPointInputBox() (in module gui.bases.settingswidget\_test), 36
- testfor() (in module utils.tests), 65
- testIntegerInputBox() (in module gui.bases.settingswidget\_test), 36
- testLimits() (in module dataobj.dataconfig\_test), 80
- testParam() (in module bases.algorithm.algorithmbase\_test), 93
- testParam1() (in module bases.algorithm.algorithmbase\_test), 93
- testParameterBaseCopy() (in module bases.algorithm.parameter\_test), 103
- testParameterCompare() (in module bases.algorithm.parameter\_test), 103
- testParameterFloat() (in module bases.algorithm.parameter\_test), 103
- testParameterFloatCopy() (in module bases.algorithm.parameter\_test), 103
- testParameterName() (in module bases.algorithm.parameter\_test), 103
- testParameterNumerical() (in module bases.algorithm.parameter\_test), 103
- testParameterNumericalCopy() (in module bases.algorithm.parameter\_test), 104
- testParameterNumericalDisplayValues() (in module bases.algorithm.parameter\_test), 104
- testParameterNumericalStepping() (in module bases.algorithm.parameter\_test), 104
- testParameterNumericalSuffix() (in module bases.algorithm.parameter\_test), 104
- testParameterNumericalValueRange() (in module bases.algorithm.parameter\_test), 104
- testParameterSerialize() (in module bases.algorithm.parameter\_test), 104
- testSerialize() (in module bases.algorithm.algorithmbase\_test), 93
- testSerialize() (in module dataobj.dataconfig\_test), 80
- testSerialize() (in module dataobj.sasconfig\_test), 86
- testSetter() (in module dataobj.dataconfig\_test), 80
- TestSettings (class in gui.bases.settingswidget\_test), 35
- testTextualInputBox() (in module gui.bases.settingswidget\_test), 36

## T

- tearDown() (RandomXorShiftUniformTest method), 96
- Temperature (class in utils.units), 67

- testTypeVsInstance() (in module bases.algorithm.algorithmbase\_test), 94
- Time (class in utils.units), 67
- timestamp (OutputFilename attribute), 43
- timestamp() (in module log.log), 52
- timestampFormat() (in module log.log), 52
- timestampFormatted() (in module log.log), 52
- title (LogWidget attribute), 34
- title (TitleMixin attribute), 106
- TitleHandler (class in gui.bases.mixins.titlehandler), 27
- TitleMixin (class in bases.dataset.titlemixin), 106
- toDisplay (ParameterFloat attribute), 100
- toDisplay() (Temperature method), 67
- toDisplay() (Unit method), 68
- ToolBox (class in gui.mainwindow), 47
- toolTipFmt (SciEntryBox attribute), 50
- topLevelItems() (DataList method), 32
- toSi (ParameterFloat attribute), 100
- toSi() (Temperature method), 67
- toSi() (Unit method), 68
- total (Moments attribute), 63
- tr() (in module gui.utils.translate), 40
- TrapezoidSmearing (class in dataobj.sasconfig), 85
- tryDisconnect() (in module gui.utils.signal), 40
- ## U
- Ui\_MainWindow (class in gui.bases.mainwindow.ui\_mainwindow), 25
- uiWidgets (AlgorithmWidget attribute), 41
- uiWidgets (OptimizationWidget attribute), 48
- unblockSigValueChanged() (AlgorithmWidget method), 41
- Unit (class in utils.units), 67
- unit (DataVector attribute), 82
- unit (ParameterFloat attribute), 100
- update (ScatteringModel attribute), 109
- update() (AlgorithmBase method), 92
- update() (DataItem method), 29
- update() (ProgressDialog method), 40
- update() (TitleHandler method), 28
- updateAll() (AlgorithmWidget method), 42
- updateConfig() (DataObj method), 81, 89
- updateConfig() (SASData method), 88, 90
- updateData() (DataList method), 32
- updateFile() (AppVersion static method), 37
- updateFMasks() (DataConfig method), 79, 91
- updateFuMin() (DataConfig method), 79, 91
- updateHistograms() (RangeList method), 49
- updateItems() (DataList method), 32
- updateMenu() (ContextMenuWidget method), 27
- updateParam() (AlgorithmWidget method), 42
- updateParamBounds() (ScatteringModel method), 109
- updatePLimits() (GaussianSmearing method), 82
- updatePLimits() (SmearingConfig method), 84
- updatePLimits() (TrapezoidSmearing method), 85
- updatePUnit() (GaussianSmearing method), 83
- updatePUnit() (SmearingConfig method), 84
- updatePUnit() (TrapezoidSmearing method), 86
- updateQLimits() (GaussianSmearing method), 83
- updateQLimits() (SmearingConfig method), 85
- updateQLimits() (TrapezoidSmearing method), 86
- updateQUnit() (GaussianSmearing method), 83
- updateQUnit() (SmearingConfig method), 85
- updateQUnit() (TrapezoidSmearing method), 86
- updateRange() (Histogram method), 61
- updateRanges() (Histograms method), 62
- updateSmearingLimits() (GaussianSmearing method), 83
- updateSmearingLimits() (SmearingConfig method), 85
- updateSmearingLimits() (TrapezoidSmearing method), 86
- updateToolTip() (gui.scientrybox.SciEntryBox class method), 50
- updateUi() (AlgorithmWidget method), 42
- updateWidget() (AlgorithmWidget method), 42
- updateWidgets() (AdvancedSettings method), 51
- updateX0Limits() (DataConfig method), 79, 91
- updateX0Unit() (DataConfig method), 79, 91
- updateX0Unit() (SASConfig method), 84
- updateX1Limits() (DataConfig method), 79, 91
- updateX1Unit() (DataConfig method), 79, 91
- updateX1Unit() (SASConfig method), 84
- upper (Histogram attribute), 61
- upper (RandomExponential attribute), 94
- upper (RandomExponential2 attribute), 94
- upper (RandomExponential3 attribute), 95
- upperDisplay (Histogram attribute), 61
- url2href() (in module gui.bases.logwidget), 34
- utils (module), 68
- utils.binning (module), 54
- utils.classproperty (module), 55
- utils.devtools (module), 55
- utils.error (module), 55
- utils.findmodels (module), 56
- utils.hdf (module), 57
- utils.lastpath (module), 58
- utils.loadstore (module), 59
- utils.mixedmethod (module), 59
- utils.parameter (module), 59
- utils.pickleinstancemethods (module), 63
- utils.propertynames (module), 63
- utils.tests (module), 64
- utils.units (module), 65
- ## V
- valid (RawArrayMixin attribute), 106
- validate() (SciEntryValidator method), 51
- validIndices (DataVector attribute), 82

[value](#) (ParameterBase attribute), 99  
[value\(\)](#) (SciEntryBox method), 51  
[valueFormat](#) (AsciiFile attribute), 71, 76  
[valueRange](#) (ParameterNumerical attribute), 101  
[valueRange](#) (ParameterString attribute), 102  
[ValueRangeError](#), 102  
[variance](#) (Moments attribute), 63  
[varName](#) (Moments attribute), 63  
[VectorResult](#) (class in `utils.parameter`), 63  
[VerboseError](#), 56  
[Volume](#) (class in `utils.units`), 68  
[volume\(\)](#) (CylindersIsotropic method), 16, 17  
[volume\(\)](#) (CylindersRadiallyIsotropic method), 17  
[volume\(\)](#) (CylindersRadiallyIsotropicTilted method), 18  
[volume\(\)](#) (EllipsoidalCoreShell method), 19  
[volume\(\)](#) (EllipsoidsIsotropic method), 19  
[volume\(\)](#) (GaussianChain method), 20  
[volume\(\)](#) (Kholodenko method), 21  
[volume\(\)](#) (LMADenseSphere method), 22  
[volume\(\)](#) (ScatteringModel method), 109  
[volume\(\)](#) (Sphere method), 23  
[volume\(\)](#) (SphericalCoreShell method), 23  
[volumeFraction\(\)](#) (ModelData method), 107  
[vset](#) (ModelData attribute), 107

## W

[w](#) (in module `gui.bases.settingswidget_test`), 36  
[wasClickedAndChanged\(\)](#) (DataItem method), 29  
[weight\(\)](#) (SASModel method), 107  
[weight\(\)](#) (ScatteringModel method), 109  
[weights](#) (SmearingConfig attribute), 85  
[widget](#) (WidgetHandler attribute), 53  
[WidgetHandler](#) (class in `log.widgethandler`), 53  
[write\(\)](#) (DataFile method), 72, 75  
[write\(\)](#) (StdErrSink method), 53  
[write\(\)](#) (StdOutSink method), 53  
[writeAttribute\(\)](#) (HDFWriter method), 57  
[writeAttributes\(\)](#) (HDFWriter method), 57  
[writeConfig\(\)](#) (cInfo method), 13  
[writeData\(\)](#) (`datafile.DataFile` class method), 75  
[writeData\(\)](#) (`datafile.datafile.DataFile` class method), 72  
[writeDataset\(\)](#) (HDFWriter method), 58  
[writeFile\(\)](#) (`datafile.AsciiFile` class method), 76  
[writeFile\(\)](#) (`datafile.asciifile.AsciiFile` class method), 71  
[writeFile\(\)](#) (`datafile.DataFile` class method), 75  
[writeFile\(\)](#) (`datafile.datafile.DataFile` class method), 72  
[writeHeaderLine\(\)](#) (`datafile.AsciiFile` class method), 76  
[writeHeaderLine\(\)](#) (`datafile.asciifile.AsciiFile` class method), 71  
[writeMember\(\)](#) (HDFWriter method), 58  
[writeMembers\(\)](#) (HDFWriter method), 58  
[wset](#) (ModelData attribute), 107

## X

[x0](#) (DataObj attribute), 81, 89  
[x1](#) (DataObj attribute), 81, 89  
[x2](#) (DataObj attribute), 81, 89  
[xLowerEdge](#) (Histogram attribute), 61  
[xMean](#) (Histogram attribute), 61  
[xrange](#) (Histogram attribute), 61  
[xscale](#) (Histogram attribute), 61  
[xscaling\(\)](#) (Histogram static method), 61  
[xWidth](#) (Histogram attribute), 61

## Y

[yweight](#) (Histogram attribute), 62  
[yweighting\(\)](#) (Histogram static method), 62