falass Documentation

Release 1.0.4

Andrew R. McCluskey

Jul 06, 2018

Contents:

CHAPTER 1

falass

1.1 falass package

1.1.1 Subpackages

falass.test package

Submodules

falass.test.test_compare module

class falass.test.test_compare.**TestCompare**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_compare()

test_fit_noi()

test_fit_noq()

test_return_fitted()

test_scale_and_background()

falass.test.test_dataformat module

class falass.test.test_dataformat.**TestAtom3Position**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_atom3positions()

class falass.test.test_dataformat.**TestAtomPositions**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_atompositions()

class falass.test.test_dataformat.**TestQData**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_qdata()

class falass.test.test_dataformat.**TestSLDPro**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_sldpro()

class falass.test.test_dataformat.**TestScatLens**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_scatlens()

falass.test.test_job module

class falass.test.test_job.**TestJob**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_check_array_false()

test_check_array_true()

test_job()

test_set_lgts()

test_set_run()

test_set_times()

falass.test.test_readwrite module

class falass.test.test_readwrite.**TestFiles**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_check_duplicates_false()

test_check_duplicates_true()

test_check_update_increase()

test_check_update_nochange()

test_files_standard()

test_files_variation()

test_flip_zpos()

test_flip_zpos_neg()

test_get_qs()

test_iterate_time()

test_line_count()

test_read_dat2()

test_read_dat3()

test_read_dat4()

test_read_dat_not_defined()

test_read_lgt()

test_read_lgt_not_defined()

test_read_pdb()

test_read_pdb_flip()

test_set_datfile()

test_set_lgtfile()

test_set_pdbfile()

falass.test.test_reflect module

class falass.test.test_reflect.**TestReflect**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_average_reflect()

test_average_reflect_noq()

test_calc_ref_basic()

test_calc_reflect_noq()

test_knext_and_rj()

test_make_kn()

test_reflect()

falass.test.test_sld module

class falass.test.test_sld.**TestSLD**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_average_sld_profile()

test_get_scatlen()

test_get_scatlen_fail()

test_get_sld_profile()

test_set_av_sld_profile()

test_set_sld_profile()

test_sld()

Module contents

1.1.2 Submodules

1.1.3 falass.compare module

class falass.compare.**Compare**(*exp_data*, *sim_data*, *scale*, *background*)
Bases: object

Comparison and fitting.

For the comparison and fitting of calculated and experimental reflectometry.

Parameters

- **exp_data** (*array_like falass.dataformat.QData*) – The experimental reflectometry data read from the datfile.
- **sim_data** (*array_like falass.dataformat.QData*) – The calculated reflectometry data from the simulation.
- **scale** (*float*) – The amount by which the calculated reflectometry should be scaled.
- **background** (*float*) – The height of the uniform background to be added to the calculated reflectometry.

fit(*bounds=((1e-100, 0), (inf, inf))*)
Fit scale and background.

Perform the fitting of the scale and background for the calculated data to the experimental data. Currently only a logarithmically transformed fitted can be conducted.

plot_compare(*fitted=True*, *rq4=True*)
Plot a comparison.

Plotting the comparision between the calculated and experimental reflectometry.

Parameters

- **rq4** (*bool*) – Should the data be plotted in rq4 space.
- **fitted** (*bool*) – Should the fitted reflectometry data be used.

return_fitted()
Return fitted.

Return the fitted calculated reflectometry data for use.

falass.compare.**scale_and_background**(*sim_data*, *scale*, *background*)
Apply scale and background.

Apply a scale factor and uniform background to the calculated reflectometry data.

Parameters

- **sim_data** (*array_like float*) – The data to be scaled and have a background added.
- **scale** (*float*) – The amount by which the data should be scaled.
- **background** (*float*) – The size of the uniform background to be added.

Returns The scaled and background added reflectometry in log space.

Return type array_like float

1.1.4 falass.dataformat module

class falass.dataformat.**Atom3Positions**(*atom*, *x*, *y*, *z*)
Bases: object

3-Dimension positions.

A class to hold the 3D atom positions, constiting of a str atom type and each of the three dimensions a float.

class falass.dataformat.**AtomPositions**(*atom*, *position*)
Bases: object

z-Dimension positions.

A class to hold the atom positions in the z-dimension, consisting of the str atom type name and a float giving the position in the z-dimension.

class falass.dataformat.**QData**(*q*, *i*, *di*, *dq*)
Bases: object

Reflectometry data.

A class to hold the qdata information consisting of four floats associated with the q-vector, the intensity of the reflectometry, the uncertainty in the intensity and the uncertainty in the q-vector/resolution of the q-vector.

class falass.dataformat.**SLDPro**(*thick*, *real*, *imag*)
Bases: object

Layer information for SLD profile.

A class to hold the layer description of the sld profile consisting of three floats associated with the thickness, real scattering length density and imaginary scattering length density.

class falass.dataformat.**ScatLens**(*atom*, *real*, *imag*)
Bases: object

Scattering lengths.

A class to hold the scattering lengths of the different atom types consisting of a str atom type name, and two floats associated with the real and imaginary scattering lengths of that atom type.

1.1.5 falass.job module

class falass.job.**Job**(*files*, *layer_thickness*, *cut_off_size*)
Bases: object

The catch all.

This class is used for setting up the falass job – and is generally a catch all for inputs that do not fit into other parts.

Parameters

- **files** (*falass.readwrite.Files*) – A Files class item.
- **layer_thickness** (*float*) – The thickness of the layers that the simulation cell should be sliced into.
- **cut_off_size** (*float*) – The size of the simulation cell that should be ignored from the bottom – this is to allow for the use of a vacuum gap at the bottom of the cell.

set_lgts()
Assign scattering lengths.

Assigned the scattering lengths from the lgtfile to the different atom types. If no lgtfile is defined falass will help the user to build one by working through the atom types in the pdb file and requesting input of the real and imaginary scattering lengths. This will also occur if a atom type if found in the pdbfile but not in the given lgts file. falass will write the lgtfile to disk if atom types do not feature in the given lgtfile or one is written from scratch.

set_run(*files=None*, *layer_thickness=None*, *cut_off_size=None*)
Edit job inputs.

This allows parts of the class to be assigned after the initial assignment or changed

Parameters

- **files** (*falass.readwrite.Files*) – A Files class item.
- **layer_thickness** (*float*) – The thickness of the layers that the simulation cell should be sliced into.
- **cut_off_size** (*float*) – The size of the simulation cell that should be ignored from the bottom – this is to allow for the use of a vacuum gap at the bottom of the cell.

set_times(*times=None*)
Assign times to analyse.

The assignment of the simulation timesteps that should be analysed. if none are given all will be analysed.

Parameters **times** (*array_like float*) – The timesteps that should be analysed, in the unit of time that present in the pdbfile.

falass.job.**check_array**(*array*, *check*)
Checks if item is in array.

Checks if an item has already been added to an array.

Parameters

- **array** (*array-type*) – The array to check.
- **check** (*str*) – The item to try and find.

Returns true if the item is already present in the scatlen type array, false if not.

Return type bool

1.1.6 falass.readwrite module

class falass.readwrite.**Files**(*pdbfile=None*, *lgtfile=None*, *datfile=None*, *resolution=5.0*, *ierror=5.0*, *flip=False*, *xray=False*)
Bases: object

File parsing.

Definition of the files to be used in the analysis; such as the .pdb file, .lgt file (is one exists), .dat file (is one exists). Also defined at this time are the resolution (this is ignored if the .dat file has 4 columns) and the percentage error in the intensity (this is also ignored if the .dat file has >2 columns), and whether the simulation cell should be flipped in the xy-plane.

Parameters

- **pdbfile** (*str*) – Path and name of the .pdb file to be analysed.

- **lgtfile** (*str, optional*) – Path and name of the .lgt file (which contains the scattering lengths of each of the atom types in the pdbfile). If a lgtfile is not defined falass will help in the creation of one. Currently the .lgt file style that is supported is a 3 column space separated txt file where the columns are atom_type, real_scattering_length, and imaginary_scattering_length respectively.
- **datfile** (*str, optional*) – Path and name of the .dat file (from which the analysis q vectors are drawn and also for subsequent comparison between theory and experiment). If a datfile is not defined falass will allow the user to define a range of q vectors to calculate the reflectometry over. Currently the .dat file style that is supported is a 2, 3, and 4 column space separated txt files where the columns are q, i, di, and dq respectively.
- **resolution** (*float, optional*) – Percentage of the q vector for the width of the resolution Gaussian function to be used in the data smearing, if a 4 column datfile is given this is ignored.
- **ierror** (*float, optional*) – Percentage error of the intensity to be assumed, if a >2 column datfile is used this is ignored.
- **flip** (*bool, optional*) – False if the system should be read as is, true is the simulation cell should be rotated through the xy-plane – note that falass treats the first side that the neutron or X-ray interacts with as that at z=0.
- **xray** (*bool, optional*) – True if the scattering length of the particles should be scaled by the classical radius of an electron.

get_qs (*start=0.005*, *end=0.5*, *number=50*)
Make custom q-vectors.

If no datfile exists this function should be used to generate a linear-spaced range of q-vector for the calculation of the reflectometry over.

Parameters

- **start** (*float, optional*) – The first q-vector for which the reflectometry should be calculated.
- **end** (*float, optional*) – The last q-vector for which the reflectometry should be calculated.
- **number** (*int, optional*) – The number of q-vectors.

plot_dat (*rq4=True*)
Plot the experimental data file that has been read in.

Parameters **rq4** (*bool, optional*) – Should the plot be created with a y-axis of Rq^4

read_dat ()
Parses .dat.

Parses the .dat file, supporting 2, 3, and 4 column files consisting of q, i, di, and dq with comments in lines where the first character is a '#'. If there is no .dat file the get_qs() function should be used to generate q vectors to allow for the calculation of the reflectometry profile.

read_lgt ()
Parses .lgt.

Parses the lgtfile. If no lgtfile is defined falass will help the user to build one by working through the atom types in the pdb file and requesting input of the real and imaginary scattering lengths. This will also occur if a atom type if found in the pdbfile but not in the given lgts file. falass will write the lgtfile to disk if atom types do not feature in the given lgtfile or one is written from scratch.

read_pdb ()
Parse .pdb.

Reads the .pdb file into memory. Currently the atoms must have the title 'ATOM', the timestep time needs to be the last text in the 'TITLE' line, and the cell dimensions are taken from the 'CRYST1' line, and assumed to be orthorhomic. Non-orthorhomic cells are not necessarily supported.

set_file(*pdbfile=None*, *lgtfile=None*, *datfile=None*)
Edits files.

Let the subsequent definition, or redefinition of the pdbfile, lgtfile or datfile.

Parameters

- **pdbfile** (*str*) – Path and name of the .pdb file to be analysed.
- **lgtfile** (*str, optional*) – Path and name of the .lgt file (which contains the scattering lengths of each of the atom types in the pdbfile). If a lgtfile is not defined falass will help in the creation of one. Currently the .lgt file style that is supported is a 3 column space separated txt file where the columns are atom_type, real_scattering_length, and imaginary_scattering_length respectively.
- **datfile** (*str, optional*) – Path and name of the .dat file (from which the /home/arm61/progs/refnxanalysis q vectors are drawn and also for subsequent comparison between theory and experiment). If a datfile is not defined falass will allow the user to define a range of q vectors to calculate the reflectometry over. Currently the .dat file style that is supported is a 2, 3, and 4 column space separated txt files where the columns are q, i, di, and dq respectively.

falass.readwrite.**check_duplicates**(*array*, *check*)
Stops duplicate atom types.

Checks if an atom type has already been added to an array.

Parameters

- **array** (*array-type ScatLens*) – The array to check.
- **check** (*str*) – The atom type to try and find.

Returns True if the atom type is already present in the scatlen type array, false if not.

Return type bool

falass.readwrite.**check_update**(*percentage*, *percentage_new*)
Check if an update string should be printed.

Assess if an update string should be printed to the screen.

Parameters **string** (*int*) – Percentage of way through reading.

Returns New percentage of way through reading.

Return type int

falass.readwrite.**flip_zpos**(*cell*, *zpos*)
Flip the z-position.

Flips the z-position through the xy-plane.

Parameters

- **cell** (*float*) – z-cell dimension.
- **zpos** (*float*) – z-position for an atom.

Returns z-position after the flipping

Return type float

falass.readwrite.**iterate_time**(*number_of_timesteps*, *line*)
Increases the number of timesteps

Iterates the number of timesteps and finds the new timestep value.

Parameters

- **number_of_timesteps** (*int*) – Number of timesteps found so far.
- **line** (*str*) – 'TITLE' line from the pdb file which has timestep information.

Returns

- *int* – Number of timesteps iterated by 1.
- *float* – Time of newest timestep.

falass.readwrite.**line_count**(*filename*)
File length.

Quickly counts the number of lines in a file

Parameters **filename** (*str*) – Name of the file that the number of lines is desired for.

Returns Number of lines in the file

Return type int

falass.readwrite.**print_update**(*percentage*)
Print percentage read-in.

Prints a percentage of how much as been read in at a given time.

Parameters **string** (*int*) – Percentage read-in complete.

1.1.7 falass.reflect module

class falass.reflect.**Reflect**(*sld_profile*, *exp_data*)
Bases: object

Reflectometry calculation.

A class for the calculation of reflectometry from the sld profile defined in the falass.sld.SLD class.

Parameters

- **sld_profile** (*array_like falass.dataformat.SLDPro*) – An array describing the scattering length density of the simulation cell under study.
- **exp_data** (*array_like falass.dataformat.QData*) – An array giving the experimental data from the datfile.

average_ref()
Average reflectometry profiles.

The averaging of the reflectometry profiles as calculated by the calc_ref() function.

calc_ref()
Calculate reflectometry.

The calculation of the reflectometry profiles based on the sld profiles calculated from each of the timesteps under study.

plot_ref(*rq4=True*)
Plot reflectometry profile.

The plotting of the calculated reflectometry data without comparison to experimental data.

Parameters **rq4** (*bool*) – Should the data be transformed to rq4 space.

falass.reflect.**convolution**(*exp_data*, *sld_profile*)
Convolution/smearing

The convolution of the reflectometry data by a gaussian of constant width (a percentage of the q-vector)

Parameters

- **exp_data** (*falass.dataformat.QData*) – The experimental data from the datfile.
- **sld_profile** (*falass.dataformat.SLDPro*) – The SLD profile calculated from the simulation trajectory.
- **lt** (*float*) – The thickness of the layers.

Returns The smeared reflectometry profile.

Return type array_like

falass.reflect.**knext_and_rj**(*kn*, *idx*, *k*)
Calculate the k in the next layer and the nature of rj.

This will determine the wavevector value in the next layer and therefore the value of rj which describes the propensity for the wavevector to reflect or refract at a given interface.

Parameters

- **kn** (*array_like*) – The wavevector array for a given layer.
- **idx** (*int*) – The particular layer.
- **k** (*array_like*) – The wavevector for the semi-infinite top layer.

Returns

- *array_like* – The wavector array in the next layer.
- *array_like* – The rj array for the interface.

falass.reflect.**layer_loop**(*kn*, *k*, *idx*, *layers*, *mrtot*)
Calculation that is conducted for each layer.

The is the calculation carried out for each layer in the Abeles optical matrix method calculation.

Parameters

- **kn** (*array_like*) – The wavevector at the start of the current layer.
- **k** (*array_like*) – The wavevector at the start of the semi-infinite top layer.
- **idx** (*int*) – The layer number.
- **layers** (*array_like*) – An n by 4 array consisting of information about the layer; thickness, real SLD, imag SLD, and roughness (not used in falass), where n is the number of layers.
- **mrtot** (*array_like*) – A 2 by 2 array of float comprising the resultant matrix for the layered structure.

Returns

- *float* – The wavevector at the start of the next layer.
- *array_like* – The updated resultant matrix.

falass.reflect.**make_kn**(*npnts*, *nlayers*, *layers*, *qvals*)
Generate the starting kn and sld arrays.

This will generate the appropriate array for the Abele optical matrix method calculation.

Parameters

- **npnts** (*int*) – number of q-vectors to be calculated over.
- **nlayers** (*int*) – number of layers in system minus two.
- **layers** (*array_like*) – An n by 4 array consisting of information about the layer; thickness, real SLD, imag SLD, and roughness (not used in falass), where n is the number of layers.
- **qvals** (*array_like*) – q-vectors for calculation.

Returns The kn array for the Abeles optical matrix calculation.

Return type array_like

falass.reflect.**reflectivity**(*exp_data*, *sld_profile*)
Abeles optical matrix formalism.

The calculation of the reflectometry using the Abeles optical matrix method.

Parameters

- **exp_data** (*falass.dataformat.QData*) – The experimental data from the datfile.
- **sld_profile** (*falass.dataformat.SLDPro*) – The SLD profile calculated from the simulation trajectory.
- **lt** (*float*) – The thickness of the layers.

Returns The reflectometry profile.

Return type array_like

1.1.8 falass.sld module

class falass.sld.**SLD**(*assigned_job*)
Bases: object

SLD profile calculation.

This class enables the calculation of the SLD profile for each of the timesteps as defined in the falass.job.Job. Further it will then allow the calculation and plotting of the average SLD profile.

Parameters **assigned_job** (*falass.job.Job*) – The is the Job class for the particular falass run taking place. See the job.Job class for more information.

average_sld_profile()
Average SLD profiles.

Allows for the calculation of the average SLD profile across all of the timesteps that were studied.

get_sld_profile()
Calculate SLD profile.

This will calculate the SLD profile for each of the timesteps defined in the falass.job.Job. This is achieved by summing the scattering lengths for each of the atoms found in a given layer (of defined thickness). This total scattering length is converted to a density by division by the volume of the layer.

plot_sld_profile(*real=True*, *imag=False*)
Plot SLD.

Plots the average sld profile using matplotlib.

Parameters

- **real** (*bool*) – Should the real SLD profile be plotted (if both real and imaginary are true the real will be plotted).
- **imag** (*bool*) – Should the imaginary SLD profile be plotted (if both real and imaginary are true the real will be plotted).

set_av_sld_profile(*av_sld*, *av_sld_err*)

set_sld_profile(*sld*)

falass.sld.**get_scatlen**(*atom*, *scat_lens*)
Find scattering length.

This gets the scattering length for a given atom type

Parameters

- **atom** (*str*) – The name of the atom type that the scattering length is needed for.
- **scat_lens** (*array_like falass.dataformat.ScatLens*) – The array of the scattering lengths that is defined in the falass.readwrite.Files class.

Returns The real and imaginary scattering lengths for the given atom type.

Return type tuple_like

1.1.9 Module contents

CHAPTER 2

falass

falass is a pure python library for the calculation of neutron and X-ray relectometry data from molecular simulation.

This was originally developed during a co-funded PhD studentship split between the University of Bath and Diamond Light Source.

An example Jupyter notebook showing the usage of falass is available from the GitHub source: https://github.com/arm61/falass

If you have any question or idea for development please get in touch: arm61@bath.ac.uk

2.1 Todo

- Test documentation
- Greater functionalisation
- Greater test coverage
- Subsequent analysis of the MD trajectory

CHAPTER 3

Indices and tables

- genindex
- modindex

Python Module Index

f

Index

T