The Journal of Open Source Software

islatu: A Python package for the reduction of reflectometry data

Richard Brearton [1,2], **Andrew McCluskey** [3], **and Tim Snow** [1,4]

1 Diamond Light Source Ltd, Diamond House, Harwell Campus, Didcot, Oxfordshire, OX11 0DE, United Kingdom **2** Department of Physics, Clarendon Laboratory, University of Oxford, Oxford, Oxfordshire, OX1 3PU, United Kingdom **3** European Spallation Source ERIC, P.O. Box 176, SE-221 00, Lund, Sweden **4** School of Chemistry, University of Bristol, Bristol, BS8 1TS, UK

DOI: 10.21105/joss.04397

Software

- Review
- Repository
- Archive

Editor: Jeff Gostick

Reviewers:

- @andyfaff
- @daguiam

Submitted: 28 April 2022
Published: 29 September 2022

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

The interaction between light and matter provides a sensitive probe of the electronic structure of materials, on length scales determined by the difference between the incident and outgoing wavevector of the light. Reflectometry techniques involve scattering off the surface of a material, placing a detector at a point such that any light reaching the detector must scatter through a vector approximately parallel to the material's surface normal. Typically, for various experiment-specific reasons, the raw data recorded by a detector will not be proportional to the quantity of interest: the modulus squared of the scattering matrix element $\langle \vec{k}' | \hat{V} | \vec{k} \rangle$. This is particularly true when the length of the scattering vector $|\vec{Q}| = |\vec{k} - \vec{k}'|$ is small, as is the case in reflectivity experiments. Then, in addition to corrections that must be applied in any scattering experiment, the finite size of the sample will affect the intensity of the reflected beam, and it is often necessary to also correct for manual changes to the beam's attenuation. For the above reasons, all reflectivity experiments need at least some form of data reduction, with the exact requirements being experiment specific and often numerous. `islatu` provides a simple, performant and rigorously tested library and command-line interface for carrying out these correction steps, which aims to substantially simplify the process of converting instrument data to a reflectivity curve. This curve can then be analysed using one of the many widely available reflectivity fitting tools, such as (Björck & Andersson, 2007), (A. R. Nelson & Prescott, 2019) and (A. Nelson, 2006).

Statement of need

`islatu` (Brearton et al., 2022) is a Python package that simplifies the process of reducing raw reflectometry data to data that has scientific value. This package is designed to serve three purposes. Firstly, it provides an interface that can be used to easily script custom reflectometry reduction pipelines. As the fitting of reduced reflectivity data is an ill-posed problem, it is often challenging to fit reflectivity curves, even with significant a-priori knowledge of the structure of the material of interest. In some cases, this could be related to errors made at data reduction time. `islatu` gives users the ability to script data reduction at analysis time. This can be particularly important when combining data sets with very different statistical uncertainties (as would be the case when comparing neutron and x-ray reflectivity curves), as errors are computed at data reduction time.

The second purpose of `islatu` is to provide a simple command-line interface, that can be used in conjunction with a configuration file, to make reflectivity reduction as automatic as possible. For example, at large-scale facilities, to make the most of valuable beamtime, it is imperative that feedback on scans is given to users as quickly as possible after a scan has been performed.

The final purpose of `islatu` is to simplify the handling of uncertainties. In `islatu`, statistical errors are automatically calculated and efficiently propagated from the raw data to the reduced dataset using optimized `numpy` (Harris, 2020) routines. Despite their fundamental simplicity, the propagation of uncertainties is error prone. The assurance provided by unit tested error propagation gives scientists more time to focus on data analysis and less time spent worrying about re-implementing standard routines.

Conversion of raw instrument data to a meaningful reflectivity curve is not desirable, but an absolute requirement. Cutting corners or making mistakes at this stage in the data analysis process would result in physically incorrect values of roughnesses and thicknesses being derived from measurements. `islatu` needed to be written to address the demand for a lightweight, rigorously tested package that can carry out the above-described functions.

Overview

There are a multitude of instruments around the world capable of recording reflectivity data. `islatu` has been designed with this in mind, with a focus on directly supporting international standard file formats (including the NeXus (Könnecke, 2015) and ORSO file formats) for the initial release, making `islatu` compatible with most modern synchrotrons. Thanks to `islatu`'s modular design, it is a straightforward task to extend this functionality to other data sources; only one parsing function needs to be added per supported file type.

As there are many instruments worldwide being used to record reflectometry data, other packages exist that reduce reflectivity profiles. However, these packages tend to be specific to a technique or instrument and are often closed source. For example, the reduction of neutron reflectivity data is possible with `Mantid` (Arnold et al., 2014), but it is an enormous piece of software designed to work specifically with neutron and muon-based techniques. In the x-ray world, manufacturers of laboratory x-ray sources typically develop their own closed-source solutions, such as Bruker's DIFFRAC.XRR package and Rigaku's x-ray reflectivity software (Yasaka & others, 2010). Open source alternatives to `islatu` do exist, but tend to not share `islatu`'s focus of being highly scriptable (either on the command line or through python). `Reductus` (Maranville et al., 2018), for example, is an excellent web-based reflectometry reduction tool, but its focus is on reducing neutron data via a graphical user interface.

`islatu` was designed for use with two-dimensional detectors, but support for point detectors is complete and all reduction steps can be carried out with identical syntax. To give an overview of `islatu`'s Python API, the first step in any data reduction with `islatu` is to instantiate a `Profile` object. A full reflectivity profile can generally be made up of more than one $|\vec{Q}|$ scan, where $\vec{Q}$ is the probe particle's scattering vector. To instantiate a `Profile` object, all that needs to be provided is a list of source files and a function that can be used to parse them. Once the profile has been instantiated, reduction takes place by calling the `Profile` object's methods. For example, for an instance of `Profile` named `my_profile` representing data acquired when a beam with a full width at half maximum of 100 μm was incident on a sample of length 10 mm, our reflectometry profile can be footprint-corrected by calling `my_profile.footprint_correction(beam_width=100e-6, sample_size=10e-3)`. The footprint correction is exact for Gaussian beam profiles, and, along with all other reduction methods available to `Profile` objects, propagates errors optimally (such that the number of mathematical operations required to propagate the errors is minimized). The other reduction methods can be used in entirely analogous ways, taking arguments where necessary, and using metadata scraped from the raw data files wherever possible.

An example of `islatu` being used to carry out corrections on a profile can be found in Figure 1. In Figure 1(a), the raw instrument data can be seen (the only analysis that was carried out prior to plotting was that $|\vec{Q}|$ was calculated for each image in the profile). The result of using `islatu` to apply corrections to the profile shown in Figure 1(a) can be seen in Figure 1(b) - this could now be analysed using a piece of fitting software.

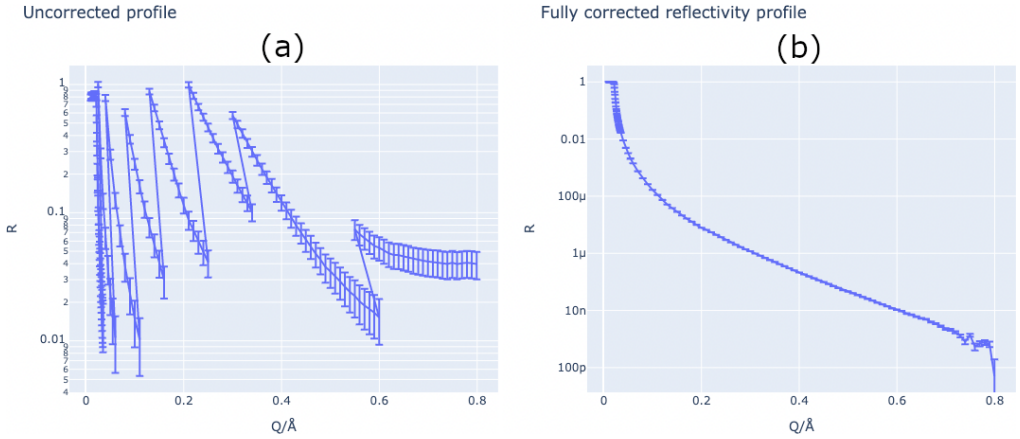

Figure 1: Visualization of the results of the reflectivity reduction process. (a) The reflectivity profile associated with the raw data. (b) The same data as in (a), after all relevant corrections have been applied.

As well as the above-described Python API, `islatu` also features a command-line interface. This application is used at the I07 beamline at Diamond (Nicklin et al., 2016) to process reflectivity data immediately after acquisition. The command-line interface runs a typical `islatu` processing script, where the arguments taken by the various data reduction methods in the script are extracted from a combination of a .yml configuration file and command-line arguments. The program outputs a human-readable metadata-rich .dat file, which at present aims to comply with the ORSO .ort file format definition, and will be converted to comply exactly with the .ort file format at such a time as the ORSO file format specification is finalized.

Acknowledgements

We acknowledge the support given to this project by the Ada Lovelace centre who funded this work under the grant number 103318.

References

Arnold, O., Bilheux, J.-C., Borreguero, J., Buts, A., Campbell, S. I., Chapon, L., Doucet, M., Draper, N., Leal, R. F., Gigg, M., & others. (2014). Mantid—data analysis and visualization package for neutron scattering and μ SR experiments. *Nuclear Instruments and Methods in Physics Research Section a: Accelerators, Spectrometers, Detectors and Associated Equipment*, *764*, 156–166. https://doi.org/10.1016/j.nima.2014.07.029

Björck, M., & Andersson, G. (2007). GenX: An extensible x-ray reflectivity refinement program utilizing differential evolution. *Journal of Applied Crystallography*, *40*(6), 1174–1178. https://doi.org/10.1107/S0021889807045086

Brearton, R., McCluskey, A., & Snow, T. (2022). *Islatu*. https://doi.org/10.5281/zenodo.7105217

Harris, C. R. et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Könnecke, M. et al. (2015). The NeXus data format. *Journal of Applied Crystallography*, *48*(1), 301–305. https://doi.org/10.1107/S1600576714027575

Maranville, B., Ratcliff, W., & Kienzle, P. (2018). Reductus: A stateless python data reduction service with a browser front end. *Journal of Applied Crystallography*, *51*(5), 1500–1506.

https://doi.org/10.1107/S1600576718011974

Nelson, A. (2006). Co-refinement of multiple-contrast neutron/x-ray reflectivity data using MOTOFIT. *Journal of Applied Crystallography*, *39*(2), 273–276. https://doi.org/10.1107/S0021889806005073

Nelson, A. R., & Prescott, S. W. (2019). Refnx: Neutron and x-ray reflectometry analysis in python. *Journal of Applied Crystallography*, *52*(1), 193–200. https://doi.org/10.1107/S1600576718017296

Nicklin, C., Arnold, T., Rawle, J., & Warne, A. (2016). Diamond beamline I07: A beamline for surface and interface diffraction. *Journal of Synchrotron Radiation*, *23*(5), 1245–1253. https://doi.org/10.1107/S1600577516009875

Yasaka, M., & others. (2010). X-ray thin-film measurement techniques. *The Rigaku Journal*, *26*(2), 1–9.