

Dataflow: a Web-based Data Reduction Framework for Neutron Scattering

William Ratcliff,^{a,*} Brian Maranville,^a Paul Kienzle,^a Andrew Tracer,^{a,b} Ophir Lifshitz,^a Elakian Kanakaraj,^a Brendan Rowan,^a Alex Yee^a and Joseph Redmon^c

^aNational Institute of Standards and Technology, Gaithersburg, Maryland 20899 USA, ^bPrinceton University, Princeton, NJ 08544 USA, and ^cUnknown. Correspondence e-mail: william.ratcliff@nist.gov

Neutron scattering is a technique which necessarily requires centralized facilities, with distributed access (a user facility). At such a facility, outside users collect data on one or more instruments, which is typically recorded in instrument-specific coordinates. Each user needs to do two things with this data: convert it into more universal (physical) coordinates, and correct for any instrumentation-specific artifacts in the data. This process is termed “data reduction”. A common paradigm for data reduction is for the instrument operator to create a new program and interface for each instrument. The source data files are reduced at user facility, where there is direct access to both the programs and the data. The reduced output files are then carried home by the users. This project aims to simplify this part of the user experience. A visual-programming interface is created in the user’s browser, allowing reduction of the source data by remote control. Reduction protocols are built up using facility-supplied functions (in icon form) wired together to create a data flow diagram. The user (or instrument operator) can change the reduction protocol, and repeat the procedure as needed, and the resultant reduced files can be downloaded by the users at will. Standard reduction protocols are made available by the facility and instrument operator. Changes to the protocol (and the addition of new instruments and new protocols) will not require changes to the user interface, and instrument-specific functions for reduction will be maintained by the instrument operators, rather than a full-blown unique reduction program for each instrument or type of instrument.

© 2011 International Union of Crystallography
Printed in Singapore – all rights reserved

1. Introduction

Every dataset that is produced at a neutron user facility must be reduced to physical coordinates, with artifacts removed and uncertainties calculated, before it can be analyzed or modeled. As a result, every type of instrument must be equipped with a software routine for performing this reduction. In addition, if the reduction programs are to be distributed to users so that they might work with the raw data at their home institutions after the experiment is complete, each program must be compiled for multiple target platforms (Microsoft Windows, Mac OS, Unix, Linux, etc.) At a large facility this requires the writing and maintenance of dozens of programs, with multiple target platforms for each program. The capability for self-update (through the internet) can be built into these programs, though typically the extra coding needed to achieve this is high. More commonly, software updates are accomplished through notification of all the users who might have downloaded or copied the programs, requesting that they download a newer version from a publicly-available repository.

We present an alternative system for providing data reduction services at a user facility: the Dataflow framework, consisting of a user interface which exists solely within the user’s web browser, and a centralized server which is maintained by the

facility staff. The core functionality of the system is the graphical representation of a reduction algorithm as a series of discrete operations on data in a user-editable workflow diagram, an example of which can be seen in Fig. 1.

The coding required to add a new set of reduction routines (a new instrument) to the framework is much reduced compared to writing an entirely new program (user interface and underlying machinery) from scratch. In addition to the workflow editing application, a common library for plotting data, importing files from a public (FTP) server, and creating and associating files with existing reduction workflows is provided as part of the base system.

Software updates are no longer dependent on the user, as changes to the server are implemented without any user interaction, and updates to the user interface code (in the browser) require only that the user refresh their view of the reduction site.

Also, by presenting the user interface in a web browser, the installation requirement for users is shifted to the browser providers. All that is required of the user is that they run a modern browser (with Javascript enabled.) Current browser support is indicated in Table 1.

2. Dataflow system

computer programs

In the client and server, every reduction session is associated with a specific, defined *instrument*. The editor for workflows allows a user to add and delete nodes in the diagram from a pool of available modules (specific to the chosen *instrument*), and attach the outputs of each node to the inputs of another node as necessary with wires. Each module, visually represented with a name and icon in the editor, corresponds to a subroutine in the backend, running on the server.

These modules have input and/or output terminals, which correspond to the inputs and/or outputs of the underlying subroutine, and wiring two terminals together in the editor corresponds intuitively to linking those inputs and outputs in the underlying reduction chain. In that sense the editor is a very simple programming environment, in which the number of available operations (modules) is extremely limited and well-defined, as well as the number of variables (input and outputs). Each module can also have associated configuration parameters, e.g. the desired resolution of rebinned data, or the magnitude of a scalar offset. These are accessible in the editor as well, by clicking on the module in question on the workflow. Parameter values are stored as part of the *configuration* of the workflow instance when the session is saved.

The set of reduction subroutines available for an instrument typically consists of one or more *loaders* which import raw data from datafiles, and *filters* which operate on the data. It is necessary that the *loaders* and *filters* provide compatible objects within each instrument's namespace, but it is not required that the *filters* for one instrument operate on the data of another instrument. Thus every defined *instrument* is independent of every other, and modifications to the algorithms, data types, etc. in one will not affect any other.

2.1. Data management

Sessions in the workflow editor are initiated through another online tool, the user project manager. Users register with a unique id, and then they can log in to a personal work space. In this environment, the user can create new *projects*, which are collections of *experiments*, which are defined simply as the grouping of uploaded raw data files with a particular *instrument* and associated workflows.

The list of associated workflows is empty when a user begins the reduction process for a new experiment, at which point standard workflow *templates* can be loaded in the editor, and then saved as specific workflow instances associated with the *experiment*. Each reduction workflow may have more than one *loader*, and so in the editor there is also a tool for associating the files in a given *experiment* with individual *loaders*. This association information is stored along with any user-supplied module *configuration* when the workflow instance is saved to the *experiment* workspace.

2.2. Plotting

Data reduction is often an interactive process, in which corrections are adjusted and regions of interest are defined midway in a reduction workflow. Plotting of intermediate results in any workflow is possible by clicking on the wires themselves, trig-

gering an evaluation of the workflow as defined up to the point of the selected wire and plotting the result in a panel in editor window.

Currently, plotting is supported for two types of data: 2-dimensional arrays with defined dimensions (which are plotted as a colormap, similar to heat maps) as seen in Fig. 2, and multi-dimensional data where multiple sets of 1-d data (all of the same length) can be flexibly plotted, choosing any x- and y-axis from the sets (see Fig 3).

2.3. Caching

In many reduction processes, intermediate results can be costly to calculate. In order to have a responsive, flexible system, small changes to the dataflow that do not require recalculation of previous steps should not force this to happen.

3. Implemented instruments

The process for implementing a particular instrument, involves porting existing algorithms for data reduction to individual Python modules that act as filters, passing modified data to the next filter in the chain.

Currently, three classes of instrument-specific reduction are implemented at the host institution for this project (NIST center for Neutron Research)

3.1. Triple-Axis Spectrometer

Enter Alex Yee.

3.2. Small-Angle Neutron Scattering

By Elakian. For instance, in the Small-Angle Neutron Scattering (SANS) reduction, many of the algorithms used were adapted from existing Igor Pro code (Kline, 2006).

3.3. Offspecular

I imagine Brendan will contribute something here.

Appendix A Libraries used in Dataflow Application

A.1. Client

The user interface is entirely implemented in Javascript, with added libraries as described below.

A.1.1. Editor The main workflow (wiring) editor is implemented as a fork of the WireIt library (<http://neyric.github.com/wireit/>), with added routines for running data reductions and configuring modules with user input.

A.1.2. Plotting Plotting in the client is accomplished with the jqPlot library (<http://www.jqplot.com/>). Additional plugins written as part of this project include one for handling 2-dimensional plotting (colormaps), as well as one for handling

multi-axis data with selectors. The jqPlot library itself depends on the jQuery library (<http://jquery.com/>).

A.1.3. Layout Menus and layout (exclusive of the wiring editor) were done in extJS (<http://www.sencha.com/products/extjs/>)

A.2. Server

The backend server is written in python (<http://www.python.org>), which makes the connection between the client (web browser) and the data reduction modules, which are also written in Python.

There is no particular requirement that the reduction code be written in Python, just that the code be callable from Python. However, because of the large available body of numerical libraries in Python (<http://numpy.scipy.org/> (Oliphant, 2007)), it was not necessary to use any other languages in the backend.

A.2.1. Web server All dynamic content, including calls to run the reduction workflows and interactive forms in the user project manager environment, is served using the Django python library (dja, ????) (<http://www.djangoproject.com>). User information (project structure, sessions, experiments, uploaded data files, saved workflow instances...) is stored in a postgresql relational database on the server, (<http://www.postgresql.org/>) along with instrument definitions and default workflow templates for each instrument type.

A.2.2. Caching intermediate results As mentioned in the text, intermediate results are cached in memory on the server to minimize computation time and make the user interaction more responsive. This is accomplished using the Redis key-value store (<http://redis.io/>). This software allows old entries to automatically expire after a configurable time.

Acknowledgements

This work utilized facilities supported in part by the National Science Foundation under Agreement No. DMR-0944772. Andrew Tracer and Joseph Redmon were supported as part of the Summer Undergraduate Research Fellowship (SURF); Elakian Kanakaraj, Brendan Rowan and Alex Yee were supported by the Summer High-school Internship Program (SHIP), both of which are also part of the above grant.

The authors would like to thank Andrew Jackson, Steve Kline and Julie Borchers of the NIST Center for Neutron Research for their invaluable assistance on this work.

References

- (????). Django high-level python web framework.
URL: <http://www.djangoproject.com>
- Kline, S. R. (2006). *Journal of Applied Crystallography*, **39**(6), 895–900.
URL: <http://dx.doi.org/10.1107/S0021889806035059>
- Oliphant, T. E. (2007). *Computing in Science Engineering*, **9**(3), 10–20.
URL: <http://link.aip.org/link/?CSX/9/10/1>

Table 1

Supported browsers		
Browser	Version	Link
Firefox	5.0	http://www.firefox.com
Chrome	13.0	http://www.google.com/chrome
Internet Explorer	None	Not supported

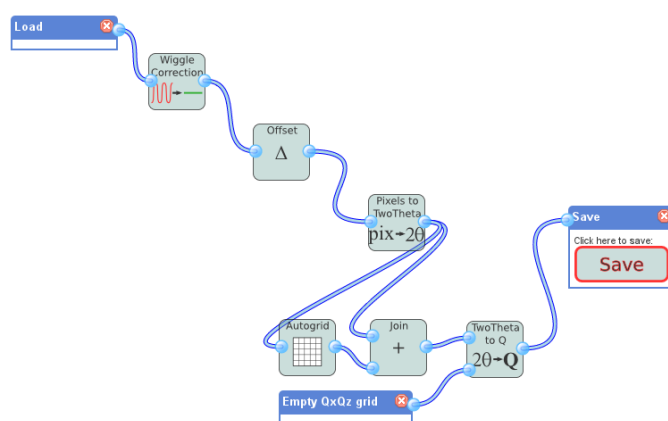


Figure 1

Sample data reduction workflow (off-specular neutron reflectometry)

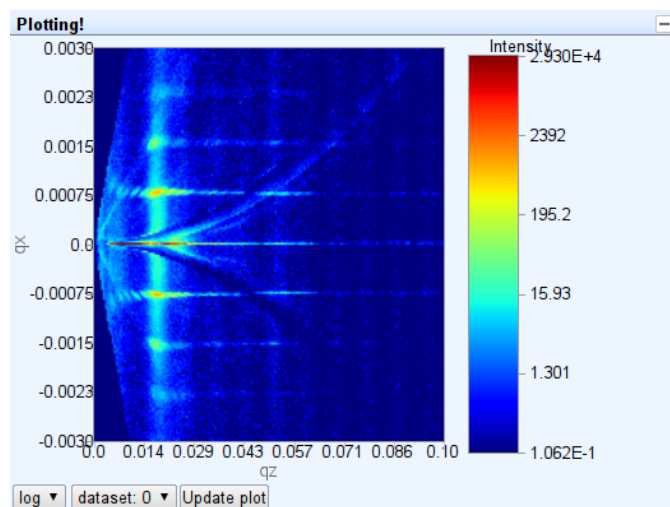
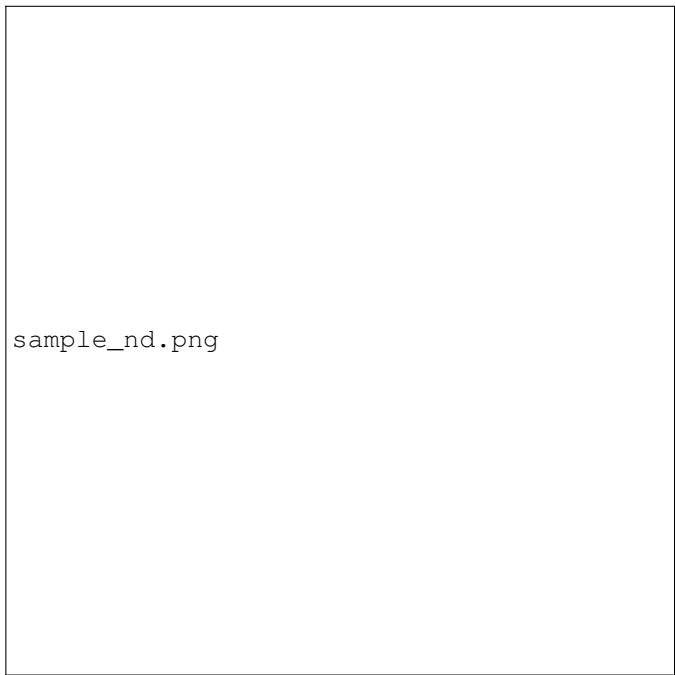


Figure 2

Two-dimensional color map of data. Log or linear mapping is selectable in the client



sample_nd.png

Figure 3

N-dimension plotter, where x- and y-axes are chosen by the user with drop-down selection inputs seen below the graph