ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA186

PARALLEL EXECUTION OF SEQUENTIAL DATA ANALYSIS

J. F. J. Murari*, K. Klementiev,
Max IV Laboratory, Lund, Sweden

Abstract

The Parallel Execution of Sequential Data Analysis (ParSeq) software has been developed to work on large data sets of thousands spectra of a thousand points each. The main goal of this tool is to perform spectroscopy analysis without delays on the large amount of data that will be generated on Balder beamline at Max IV [1]. ParSeq was developed using Python and PyQt and can be operated via scripts or graphical user interface (GUI). The pipeline is consisted of nodes and transforms. Each node generally has a common group of components: data manager (also serves as legend), data combiner, metadata viewer, transform dialog, help panel and a plot window (from silx library [2]) as main element. The transforms connect nodes, applying the respective parameters in the active data. It is also possible to create cross-data linear combinations (e.g. averaging, RMS or PCA) and propagate them downstream. Calculations will be done with parallel execution on GPU. The GUI is very flexible and user-friendly, containing splitters, dock widgets, colormaps and undo/redo options. The features mentioned are missing in other analysis platforms, which justifies the creation of ParSeq.

INTRODUCTION

ParSeq has been projected to perform data analysis on a large amount of spectroscopy data through a downstream pipeline. This paper aims to present the tool and the status of the project, as well as describe the features implemented and what software technologies were used.

MAIN WINDOW

ParSeq has a main window composed of transformation nodes. Each node has a separated tab and defines which stage of the pipeline the user is working on. The nodes have a common group of components that will be detailed in next section. Figure 1 shows an overview of the main window.

ParSeq can be operated both via GUI or via scripts, where the user can define programmatically what is the data processing pipeline wanted and what are the parameters for each transform and for each spectrum. Processing pipelines are Python modules imported by GUI or scripts and are predefined for any analysis technique and therefore are (a) ready to use and (b) extensible if required. The currently implemented pipeline is only one: for X-ray Absorption Spectroscopy.

All graphical elements were implemented using Python and PyQt4 and the main plot window is from the silx library. The initial layout was designed with Qt4 Designer program, as shown in Fig. 2.

* juliano.murari@maxiv.lu.se

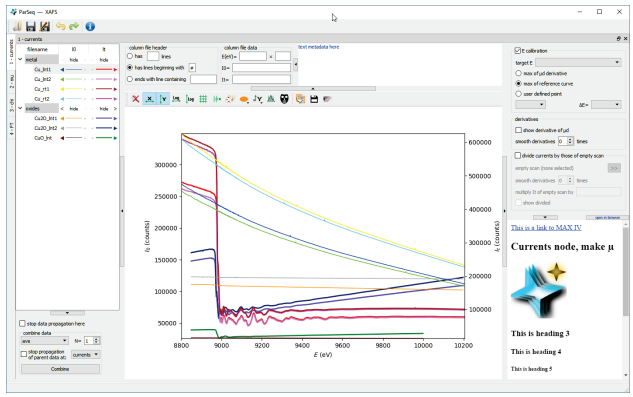

Figure 1: ParSeq Main Window.

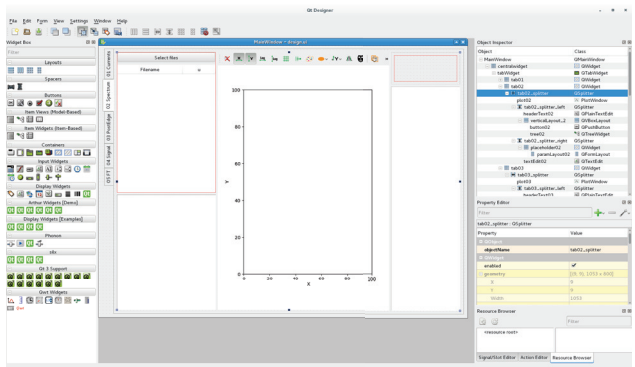

Figure 2: Qt4 Designer layout of ParSeq.

Dockable Widgets

The node widgets are dockable and can be placed wherever the user prefers, giving the flexibility to the user to decide where to position the windows or to have them grouped in tabs. ParSeq saves the current perspective of the widgets in order to be able to restore it next time the program is opened again.

Undo Redo Actions

ParSeq has implemented undo and redo options related to the transform operations. These options allow the user to revert one transform that was occasionally done with wrong parameters, or even that the result was not the expected. If one action is reverted (undo) it will then be available to be reapplied (redo).

Splitters

Corner widgets of ParSeq are divided by splitter bars (qtSplitter widget) that can be resized by the user. Furthermore, the splitters have a button to collapse/expand the corner wid-

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA186

get. This way, the user can hide the sub-windows that are not in use at the moment.

NODE STRUCTURE

The graphical elements that compose a node are described bellow:

Data Manager

Data manager is a QtTree with a list of names representing the data related to each spectrum. The list can be divided in different groups, where is possible to drag and drop the names between the groups.

A right click opens a context menu with options to interact with the data. The selected names determine the active data, that will be considered to apply transforms or options from the context menu. Bellow are presented the options available in the context menu:

- **Add**: add a new curve from a file. It will be inserted on the current group.
- **Remove**: remove the selected curves.
- **Create group**: create a new group on the QtTree.
- **Set a colormap**: apply a colormap to the selected curves, with a linear or logarithmic gradient (see Fig. 3). Colormaps available are: temperature, viridis, plasma, cool, copper, autumn, spring, summer, winter, brg, gnuplot and jet (from silx).

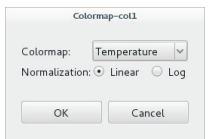

Figure 3: Colormap Dialog Window.

- **Set line properties**: apply a set of line properties to the selected curves (see Fig. 4). The properties are:
  - Color: full range of colors from matplotlib *ColorDialog*.
  - Symbol: circle, point, pixel, cross, x-cross, diamond, square or none (from silx API).
  - Width: value for line width.
  - Style: no line, solid, dashed, dash-dot and dotted (from silx API).

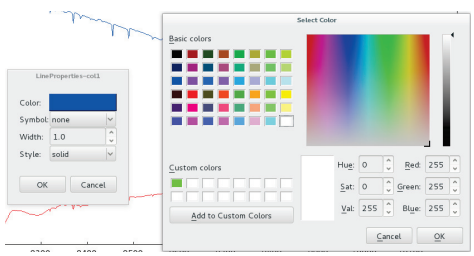

Figure 4: Line Properties Dialog Window.

- **Average**: combine the selected curves, averaging them in a new curve.

The widget also serves as a legend. For each node there are one or more columns representing the curves on the plot. Each legend entry has a control element for attributing it to left or right Y axis or for switching it off. There is also a button to control all the curves of the group.

Regarding the functionality as legend, each button has the color and the line from the curve. Through the context menu the user can change these characteristics. Furthermore, when more than one curve are selected, the user can apply a colormap to change their colors following the colormap gradient, as shown in Fig. 5.

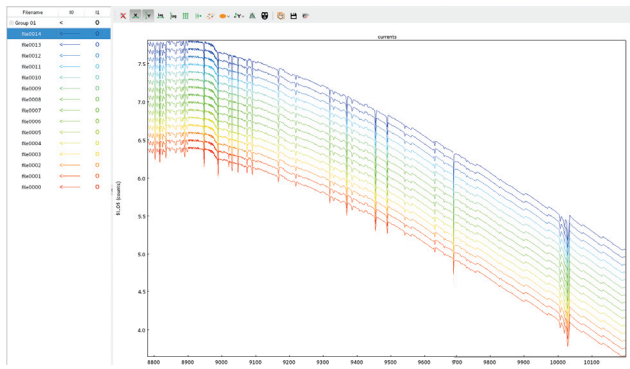

Figure 5: Example of colormap with temperature gradient.

Data Combiner

The data combiner creates a linear combination (average or RMS) or several linear combinations (for Principal Component Analysis) and optionally can stop the propagation of the contributing data at some downstream node in favor of the combined data. This feature is useful, for example, for averaging multi-element detector data when the quality of the individual data can be visualized and assessed not immediately but a few transforms downstream of the present node.

Metadata Viewer

Metadata viewer widget has the function to show what are the metadata related to the selected curves. If more than one curve are selected it shows only the metadata common between them.

Transform Dialog

The transform dialog widget is where the parameters options are available. At this point each node has a specific set of parameters. Each spectrum has, in general, its individual values for these parameters. The result of the transform appears in the next node. The last node of the pipeline does not have this widget.

Help Panel

Help panel widget is a space to present help text with instructions and information about the current node and the associated transform. It shows a rich text from *html* file.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA186

Plot Window

Plot window widget from silx is the central and most important element of the GUI. It is the widget that indeed shows spectra, usually as XY graph. The advantage to use the silx plot window is to have a lot of functionalities already implemented, as for example, zooming, regions of interest (ROI), grid, log scale.

Library silx provides an OpenGL backend which is faster than matplotlib backend, mainly working with 1D curves. It is an important aspect of performance, since we plan to work with thousands of curves.

GPU PROCESSING

Planned for a later development stage, the most expensive transforms will be computed in parallel on GPUs or multicore CPUs. The programming will be done in OpenCL [3]. The connection to ParSeq will be done in Python by means of PyOpenCL. The early implementation of the transforms will be done solely with numpy.

FUTURE WORK

ParSeq is a project under development and we are continously working on it. For the future we plan, besides the GPU processing, to improve the handling of data files, and to increase the number of formats supported (HDF, for example). We are also working on the implementation and integration of the transforms that will be available.

In the future ParSeq can be also very useful to run integrated with the beamline control system for on-the-fly quality checking.

CONCLUSION

The paper presented ParSeq and its features, as well as the direction and the current status of the project. In conclusion, ParSeq will be a very important and user-friendly software tool for data analysis and visualization at Balder beamline.

ACKNOWLEDGEMENT

The authors would like to acknowledge everyone involved in the project, specially the Controls and IT group (KITS) and the Balder team.

REFERENCES

[1] K Klementiev, *et al.* The Balder Beamline at the MAX IV Laboratory. *Journal of Physics: Conference Series*, 712(1):012023, 2016.

[2] Solé, V. A., *et al.*, "silx-kit: Scientific Library for eXperimentalists", Release 0.5.0 - 2017/05/12. http://www.silx.org/

[3] John E. Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *IEEE Des. Test*, 12(3):66–73, May 2010.