Neutron imaging analysis using jupyter Python notebook

Jean-Christophe Bilheux, Hassina Bilheux, Jiao Lin, Ian Lumsden and Yuxuan Zhang
Oak Ridge National Laboratory, Neutron Scattering Division, Neutron Sciences Directorate, Oak Ridge, TN, United States of America
E-mail: bilheuxjm@ornl.gov

Keywords: imaging, python, analysis, jupyter, notebook

Abstract
Independently of the image modality (x-rays, neutrons, etc), image data analysis requires normalization, a preprocessing step. While the normalization can sometimes easily be generalized, the analysis is, in most cases, specific to an experiment and a sample. Although many tools (MATLAB, ImageJ, VG Studio …) offer a large collection of pre-programmed image analysis tools, they usually require a learning step that can be lengthy depending on the skills of the end user. We have implemented Jupyter Python notebooks to allow easy and straightforward data analysis, along with live interaction with the data. Jupyter notebooks require little programming knowledge and the steep learning curve is bypassed. Most importantly, each notebook can be tailored to a specific experiment and sample with minimized effort. Here, we present the pros and cons of the main methods to analyse data and show the reason why we have found that Jupyter Python notebooks are well suited for imaging data processing, visualization and analysis.

1. Imaging data analysis requirements

A prerequisite to image data analysis is normalization, and this step is necessary for most imaging modalities such as x-ray or neutron radiography. While the normalization process can sometimes easily be generalized, the analysis is, in most of the cases, sample specific. Depending on the scientific question to be answered, one might create, for example, a transmission profile through a 3-dimensional (3D) sample volume, or perform data segmentation prior to measuring porosity, size distribution, density changes, etc. Samples usually have complex geometries (that may require geometrical corrections) and elemental compositions, which may introduce artifacts that have to be corrected prior to the analysis. Visualization of the data on-the-fly is key in determining how to proceed with data analysis, after artifact correction and/or segmentation.

One must also consider the different levels of data science skills of the research and development communities the data analysis package is developed for. Some enjoy working with a scripting language such as MATLAB [1], use command lines [2], or others might prefer a graphical user interface (GUI) such as iMars [3] or MuhRec [4]. Moreover, training is often required prior to utilizing GUIs which adds time to an already lengthy data normalization and analysis process. The highest challenge is to develop a versatile tool complex enough that it is capable of responding to 90% scientific community requirements and provides an intuitive workflow at the same time.

We will first present the current analysis tools used at our facilities (MATLAB and Python [5]) and then our novel approach to unify the strength of each previously used method using Python Jupyter notebooks. The novel application of the Jupyter Python notebooks applied to neutron radiography provides an intuitive data analysis, thus lessening the learning curve process.

1.1. MATLAB and python
MATLAB and Python have several similarities. MATLAB is a commercial language that is mostly based on handling arrays. The open source language Python, thanks to the very powerful NumPy library [6] handles arrays in a similar way. These languages are well suited to work with radiographs, which are essentially two-dimensional matrices. Tools developed using MATLAB or Python can also be easily extended to a three-
dimensional data set, which is composed of 2-dimensional slices reconstructed from the original radiographs of the sample. An abundant list of image processing algorithms is available in each of these languages. Methods to read and display various file formats are also available. However, many users find that these languages have a steep and demanding learning curve. User facilities, such as the Spallation Neutron Source (SNS) and High Flux Isotope Reactor (HFIR) at the Oak Ridge National Laboratory (ORNL), are in high demand. Often, a research team visits the facility and perform experiments a few times a year due to the oversubscription of these facilities (oversubscription by a factor 4 at the CG-1D imaging beam line at HFIR, ORNL). In order to expedite the science result, it becomes very important to create intuitive analysis tools that are customized to an experiment or series of experiments.

A team effort is often required to successfully perform most of the experiments. That is also true for data analysis. This means sharing code (.m files for MATLAB and .py files for Python) between team members and counting on the understanding of the programming philosophy of the programming expert. Frequent and useful comments are a must, while also good coding practice is essential, but these requirements are not always met.

At our imaging beamline, we exclusively use the Python programming language. Some of our users are more comfortable with a GUI, others are able to use a notebook with the flexibility for user to add their own script if they desire to do so.

1.2. Standalone graphical user interface
An alternative option to automate/simplify manual and cumbersome data analysis can be developed with a graphical user interface (GUI). MATLAB and Python provide tools to quickly implement GUIs. More programming is required but this can be beneficial for large data sets and repetitive analysis. Any algorithms requested by the users, such as fitting an intensity profile, can be implemented in python and accessed via the graphical user interface. But the more powerful they are, the more complex they become and the harder it is for the end-user to use the algorithms. Due to the many options and routes one can take via the GUI, a tutorial with detailed documentation must be provided, as the interface by itself does not provide direction to the user on how to navigate through it. Figure 1 shows the normalization tab of the MATLAB-based iMars [2] application. One must first load the raw radiographs, open beams, the dark fields (steps 1 to 5, as indicated in figure 1) then correct for background fluctuation by selecting one or more regions of interest, then run the normalization, which is automated. However, another path can be used if the normalization needs to be run as a batch job (steps 1 to 8, as illustrated in figure 1). The greatest strength of a GUI is its wide range of tools provided and multiple possible bifurcations, but that creates complexity which contributes to its weakness.

2. Jupyter python notebooks
The Jupyter project [7] is an electronic interactive computational notebook that can be used with various programming languages. A Python-based notebook allows the programmer to describe the step-by-step process used during data analysis, with embedded executable code, rich text, mathematics, radiographs, plots, etc. The open source Jupyter Python notebook can run using a standard web browser. For example, at the ORNL neutron imaging beamline, custom-made notebooks are selected to perform actions such as normalization, histogram-based segmentation, attenuation map as a function of sample height, etc. Our user community has been very pleased with the implementation of the Jupyter notebooks. Our effort focus on a platform that can be used by a researcher with little to no programming python experience. The most advanced users can, of course, introduce their own scripts in our notebooks too. The simplicity of using the notebooks and the combination of interactive widgets, explanation texts, output results within the notebook are the main advantages of such a tool over a python script for example. We use the notebooks as an interface and deliberately hides a lot of the complexity of the python libraries to the users.

These actions are represented by a link in the browser, i.e. in the notebook. By default, the code runs Python, however, other languages (e.g. MATLAB, Java, or JavaScript) can be used. The Jupyter notebooks also allow the implementation of multiple programming language in different cells. These cells can still work with the same data set. The notebook interface is very simple [8] with, from top to bottom, a menu, toolbar and the cells that will contain the code, comments (markdown), images, and output of cells. Moreover, a notebook can be easily modified by users with more programming experience by introducing algorithms into new cells of the notebook. The execution workflow of a notebook is straightforward as it goes from top to bottom one cell at a time. Each cell is comprised of lines of code that can be modified in the browser directly. This is an efficient and guided method that eliminates the need for a complex tutorial to follow the proper processing steps. Widgets, such as buttons, sliders, text field, can be inserted as well to add interactive options.
2.1. Working live with the data
All basics widgets, such as dropdown lists, buttons, check boxes, sliders and progress bars, are provided as part of the Jupyter Python framework. They can easily be added to a notebook to provide interactive functions. For example, the notebook can start by asking the user to select files that have been acquired via widgets inside the cell. Python will load the images. Then, a specific file is selected and displayed so the intensity profile plot can be interactively selected and changed using a horizontal and vertical lines on the selected file, or any other file that have been loaded in memory via the notebook. This allows live interaction with the data which is advantageous when one is interested in getting preliminary data analysis during an experiment. This feature is enabled by the access to the data via the data server, which is where our data and notebooks are located. Thus, the notebooks can readily communicate with the data as soon as they show up on disk. Figure 2 illustrates the capability to select a sector in order to calculate a radial intensity profile.

2.2. Graphical interface via notebooks
Graphical interfaces developed using PyQt [9] for example, can be launched from the notebook itself. This allows to easily chain, via the notebook, several user interfaces (ui). Data can then be shared between the ui and the notebook and processed again inside the notebook itself for advanced users. Once again, the notebook is the heart of the system and users are guided by the workflow of the notebook. Figure 3 shows such a ui that was started from a Jupyter notebook in which users can overlap their images with metadata values or a plot of the evolution of the metadata. Processed images can be exported in any format (images, text, yaml, …) directly from the notebooks or the GUI, at the discretion of the user’s request. Data files will be located in the user home folder and can be transferred via Secured File Transfer Protocol (SFTP), at a speed that depends on the user’s home institution network capability.

2.3. Modifying and sharing notebooks
Once a plethora of notebooks have been created, they can be combined and modified in such a way that the algorithm workflow can be optimized for a specific data analysis requirement. Insertion of a new embedded
executable, removal and change of lines of code can be performed on the same notebook. External scripts can be called from the notebook as necessary. Notebooks can thus be tailored to a specific experimental plan in order to facilitate scientific data analysis. Because the notebooks are saved as a JavaScript Object Notation (json) file in the

**Figure 2.** A section of a Python notebook dedicated to calculate sectorized radial profiles.

**Figure 3.** Graphical interface used to add information on radiographs such as plots, or metadata information.
back, they can then easily be shared with collaborators via email, Dropbox accounts or repository sites such as GitHub [10]. Using GitHub offers the advantage of automatically rendering (commands, images and plots) a notebook via the browser. Moreover, websites such as nbviewer [11] facilitate the sharing of notebooks by posting notebooks on the site to be viewed, downloaded and run. This sharing process ensures the notebooks are peer-reviewed and improved upon by contributions from the scientific community. This ultimately contributes to the betterment of the notebooks.

One of the drawbacks in using the notebooks is the need for frequent up and down scrolling of the window. Users must also be disciplined in the way they run the cells as running them out of order may throw errors, or worse, give wrong outputs.

3. Conclusion

Although many tools offer a large collection of pre-programmed image analysis tools, they are difficult to learn in a short amount of time. We have implemented the use of Jupyter Python notebooks to allow easy, straightforward, live, and interactive data analysis. These notebooks require little programming knowledge and can be easily adapted to a specific research problem and image data analysis. Using Jupyter hub, Python notebooks can be displayed on the web browser, but run on a server machine, avoiding the need to install all the dependencies (i.e. libraries) necessary to execute the notebook. Our plan is to create a collection of common algorithms that one can start from to develop a more advanced and data-specific analysis in order to facilitate research and development at our ORNL user facilities.

Acknowledgments

Use of the CG-1D beam line at Oak Ridge National Laboratory’s High Flux Isotope Reactor was sponsored by the Scientific User Facilities Division, Office of Basic Energy Sciences, U. S. Department of Energy. This manuscript has been authored by UT-Battelle, LLC, under Contract No. DE-AC05 00OR22725 with the U.S. Department of Energy. The authors would like to thank all of the CG-1D scientific research community who helped, by their interactions and feedback, start the implementation of the Python notebooks and improve them.

ORCID iDs

Jean-Christophe Bilheux @ https://orcid.org/0000-0003-2172-6487
Hassina Bilheux @ https://orcid.org/0000-0001-8574-2449
Jiao Lin @ https://orcid.org/0000-0001-9233-0100
Yuxuan Zhang @ https://orcid.org/0000-0002-0083-1408

References

[1] MATLAB (Natick, MA, United States: The MathWorks Inc.) http://mathworks.com/
[2] Minniti Triestino 2019 Bragg Edge Analysis for Transmission Imaging Experiments software tool: BEATRIX Journal of Applied Crystallography 52 903–909
[3] Bilheux J and Bilheux H 2015 iMars (iMaging Analysis Research Software) Phys. Proc. 69 343–8
[4] Kaestner Anders P. 2011 MuhRec—A new tomography reconstructor NIMA 651 156–160
[5] Python.org
[6] numpy.org
[7] jupyter.org
[8] Introduction to the notebook user interface. https://neutronimaging.pages.ornl.gov/tutorial/how_to_run_notebooks/#notebook_layout
[9] riverbankcomputing.com/software/pyqt/intro
[10] github.com
[11] Nbviewer, a simple way to share Jupyter notebooks www.nbviewer.ipython.org