DOCUMENTATION FOR BRAGG EDGE ANALYSIS PYTHON CODE

JACOB MARESCA

13/07/2016

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This document serves to describe the motivation and usage of the Bragg Edge Analysis python application. It will aim to explain the scientific reasoning behind the actions available within the software and also to serve as a guide for using the application.

This application was designed to analyse neutron imaging data collected from the MCP detectors (ISIS) at the Rutherford Appleton Laboratory UK. The aim of the application is to be able to take the raw data produced by an experiment, and process it so that information such as the position of Bragg edges and strain maps can be produced.

2 BACKGROUND

2.1 Scientific Motivation

When a polychromatic neutron beam passes through a crystalline material, neutrons of different wavelengths are attenuated to varying degrees. As a result, the energy spectrum of the 'open beam' looks very different to the spectrum produced when a sample is being imaged. By observing the ration of these two intensities at each wavelength we can produce a transmission plot. Figure 1 is an example of such a plot produced by the application described here.

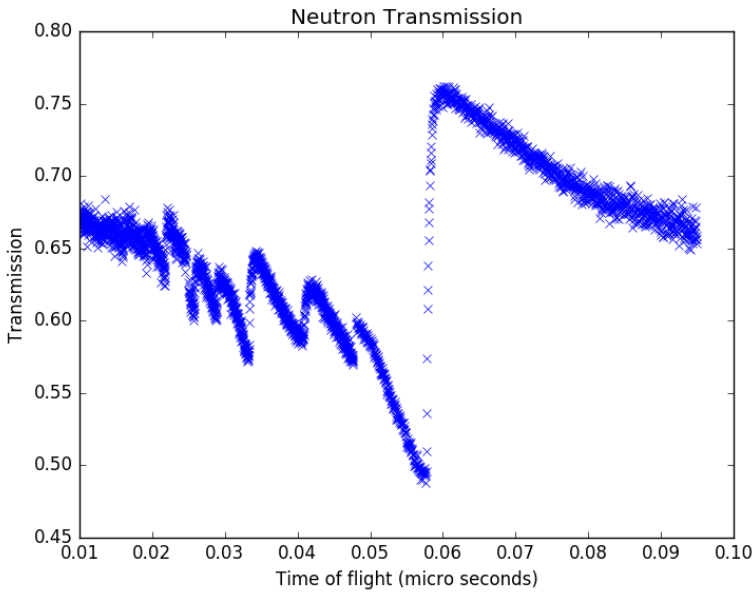

Figure 1: a plot showing the ratio of transmitted intensity of a neutron beam as a function of wavelength for a sample vs open beam.

Event counting detectors employed at ISIS, are capable of registering more than 10^4 simultaneous $events/cm^2/s$ with a timing resolution of more than 1ons within each independent $55x55micrometer^2$ pixel. Due to the fast parallel readouts of > 1kHz it is possible to detect input fluxes of as high as $10^7 events/cm^2/s$. However, only one event per frame can be registered by these first generation Timepix readouts. Due to this, in applications with very high incident fluxes, events arriving later in the shutter cycle have a lower probability of being detected as their is a higher probability of the pixel being already occupied. An event overlap correction algorithm has been developed that is applicable after the fact. i.e it can be applied to already corrected data with no requirement to change the experimental setup.

The probability $P(t_i)$ that a Timepix pixel is already occupied, and thus cannot detect any further events until the frame is readout is given by

$$P(t_i) = \frac{\sum_{j=0}^{j=i-1} N(t_j)}{S^m}, i > 1; P(t_0) = 0. \quad (1)$$

Therefore, it follows that the probability for detecting an event at time t_i is $(1 - P(t_i))$. The number of events $N'(t_i)$ that should be detected, if the Timpeix were able to detect multiple events per shutter is

$$N'(t_i) = \frac{N(t_i)}{(1 - P(t_i))}, i > 1; N'(t_0) = N(t_0). \quad (2)$$

Equations (1) and (2) have been adapted from "Optimization of Timepix count rate capabilities for the applications with a periodic input signal" - A.S. Tremsin et al, with the slight change being to the limits of the sum in equation (1).

Since the neutron flux of the open beam is much higher than when a sample is being imaged, it is necessary to scale the trigger values. The open beam data is scaled by the ratio of trigger values for the sample data, and the trigger values for the open beam data.

With these steps completed the data is ready for visualisation and further analysis. The position of a Bragg edge can be determined accurately by fitting an analytical function based on the convolution of a Gaussian with back to back exponentials, with a least squares method. The function to be fitted depends on five parameters,

$$T(\lambda)|_{\lambda_0,\sigma,\tau,C_1,C_2} = C_1 + C_2[\text{erfc}(\frac{\lambda_0 - \lambda}{\sqrt{2}\sigma}) - \exp(\frac{\lambda_0 - \lambda}{\tau} + \frac{\sigma^2}{2\tau^2}) * \text{erfc}(\frac{\lambda_0 - \lambda}{\sqrt{2}\sigma} + \frac{\sigma}{\sqrt{2}\tau})] \quad (3)$$

2.2 Image Analysis

Due to the high variation in count rates between different regions of a single data set, and between data sets, it has been necessary to implement some basic image processing capabilities. One data set might be "dark" at the beginning, transitioning to "bright" and then "dark" again, and the meaning of "bright" and "dark" is relative to a particular data set.

The primary technique used is Histogram Equalisation. Consider an image f represented by a matrix of integer intensities ranging from $0 \rightarrow L - 1$, where L is the number of possible intensities. Let P denote the normalised histogram of f with a bin for each possible intensity, then the Histogram equalised image g is defined by

$$g_{ij} = \text{Floor}((L - 1) \sum_{n=0}^{f_{ij}} P_n) \quad (4)$$

3 HOW TO USE THE APPLICATION

Before running the application, be sure that you have python and all the requirements specified in the "requirements.txt" file installed. This application depends upon some modules that are not included in the standard library, namely astropy, numpy, matplotlib, and scipy. You can install all the requirements by typing

```
$pip install -r requirements.txt
```

into the command line, provided you are in the same directory as the requirements.txt file.

The application can be run from the command line, by typing

```
$python BraggEdgeAnalysis_V0.1.py
```

Once you have run the file, you will be greeted with a GUI that gives the user access to all of the currently available functionality. The current state of the application is that it is possible to apply the necessary corrections to the data, visualise the samples, select regions of interest, produce transmission plots and fit the analytical function specified earlier to a Bragg edge.

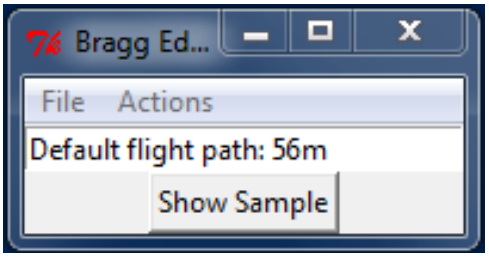

Figure 2: The main window upon opening the application.

3.1 Loading data

In the top left corner of the GUI there is a "file" menu. Clicking it will reveal three options. To load open beam data, to load sample data and to exit the program. If you click on either of the load data options, a file explorer window will pop up and you can select the folder that contains the data you wish to work with. The application will load overlap corrected/scaled data if it exists, otherwise it will load the original data.

When selecting folders containing the Data, it is important that you select the folder containing the original data. The application will find the overlap corrected data assuming that this is the starting point it was given. For example, figure 3 shows the selection of the "openbeam1" folder which contains the original data, but the program will automatically load the data inside "scaledOpenBeam".

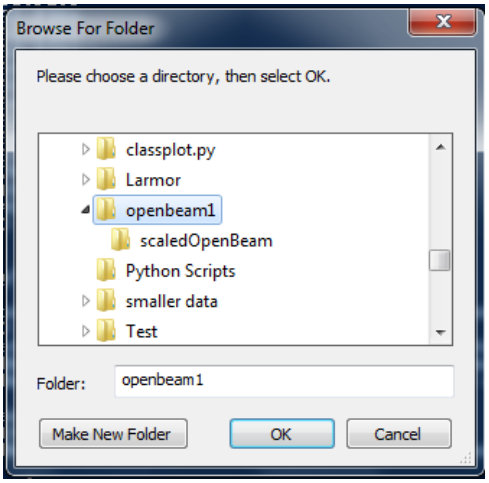

Figure 3: The file dialog opened by the program

Note: It is not necessary to reload data if the the overlap correction is applied. The program will update the open objects, and save a copy for later use.

3.2 Overlap corrections

Once you have selected the two required folders, you can press the overlap correction button in the "actions" menu of the GUI. This will begin correcting the chosen data and create new folders in the same directory as the uncorrected data. Pressing this option without having previously loaded the necessary data will result in an error message popping up. Pressing this button when the overlap corrected data is already present will result in another error message. This option will produce corrected datasets

for both the sample and open beam data, one after the other. Trigger scaling will simultaneously be applied to the open beam data set.

3.3 Sample Visualisation

The "Show Data" button in the main window of the GUI will create a canvas embedded in the main window displaying the sample image data. It will also initialise a slider object, a text entry window, and a "Histogram Equalisation" button.

The slider object is used for scrolling through the image stack. Dragging the slider object along it's bar allows for scrolling quickly through the stack, whilst clicking in the bar either side of the object allows for moving one frame at a time.

The text entry window is used for defining a maximum range of the pixel values, which matplotlib uses for normalising the colour map scale. Trial and error for finding a sensible scale seems to work best, and the plot will be updated upon drawing the next image. A default value of 255 is supplied, and it won't always be necessary to adjust it's value, but it is a useful tool for improving the image shown on the screen.

The Histogram Equalisation button will apply the technique of histogram equalisation to the current image and update the canvas appropriately, and is generally a good tool for increasing contrast. Used in conjunction with the above tool, most dark images should have the potential to become much clearer.

To select a region of interest, click and drag the mouse anywhere in the canvas. Doing so will draw a translucent rectangle on the canvas that persists after you release the mouse and if you scroll through the images etc. To select a new region, simply click and drag again.

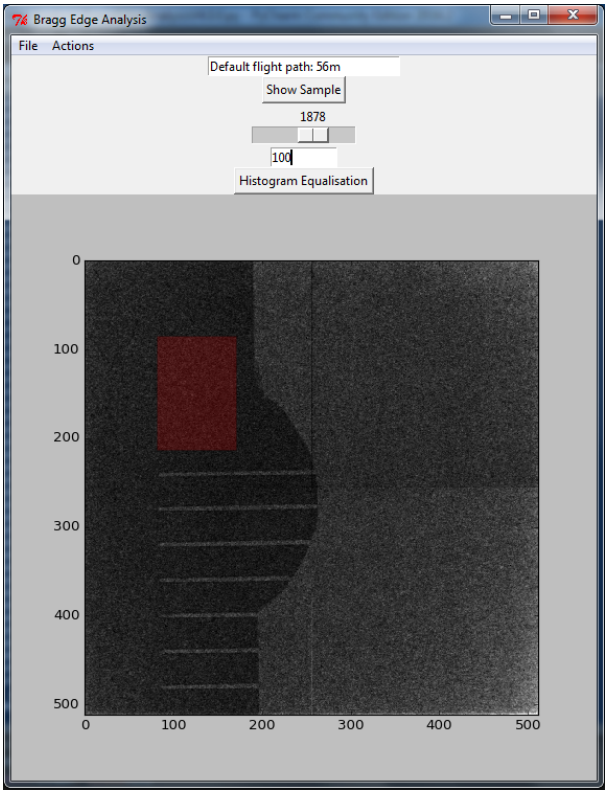

Figure 4: Selecting a Bragg edge from a transmission plot

3.4 Transmission plot

After selecting a region of interest from the sample, you can produce a transmission plot by navigating to the "Actions" menu and selecting "Transmission Plot". Here you have two options, to produce a plot

based on time of flight, or wavelength. If using the wavelength option, make sure to input the flight path of the instrument you are using into the entry box in the main window of the GUI. A default value of 56m is supplied, and when you change this value you can simply input "29.4" for example, removing any other text ("29.4m" is also an acceptable input. Any leading or trailing text will be ignored).

Having selected either of these options, a transmission plot will be generated for your region of interest in a new window using matplotlib. A toolbar in the window natively supports zooming, panning, saving the plot as a .png file, resetting the axes and zooming to a specified region. Zooming to a specified region can be useful for selecting a Bragg edge. To use this feature, select it from the tool bar, use the mouse to draw a rectangle and then release the mouse. Once you are happy with the region you are zoomed to, press the button again to disengage the function. Pressing the "Home" button will reverse any changes you have made.

If you click and drag on the plot without having another function engaged, you will select a region of interest in much the same way as on the sample canvas. This is used for selecting the data points constituting a Bragg edge. To select a different region, simply click and drag again.

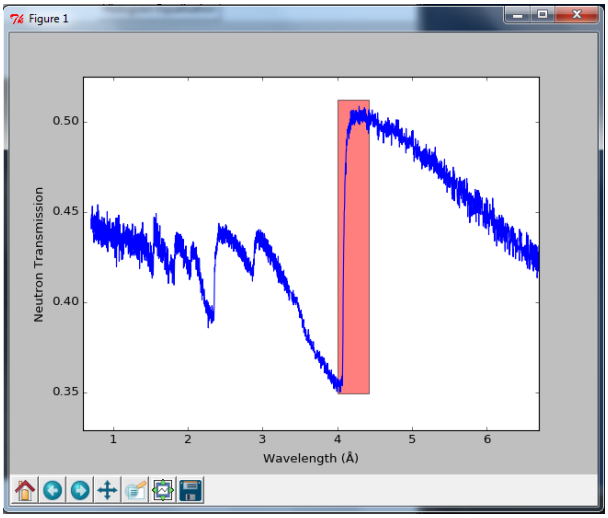

Figure 5: a plot showing the ratio of transmitted intensity of a neutron beam as a function of wavelength for a sample vs open beam.

3.5 Edge Fitting

Upon selecting a Bragg Edge from the transmission plot you are ready to fit the analytical function by selecting the "Fit Bragg Edge" option from the Actions menu. This will bring up a new window in the GUI with an embedded matplotlib window and several entry fields. The window will display the Bragg edge you selected earlier. The entry fields are used to provide initial guesses for the parameters used by the fitting function. If the values supplied were not good enough for the least squares algorithm to converge on a solution, an error message will pop up and the a plot of the function based on the supplied parameters is displayed. This is helpful to see how close your guess is to the data, and is a useful visual tool for further refining the parameters. A successful fit will result in each of the parameter fields being updated with the values that satisfied the least squares algorithm.

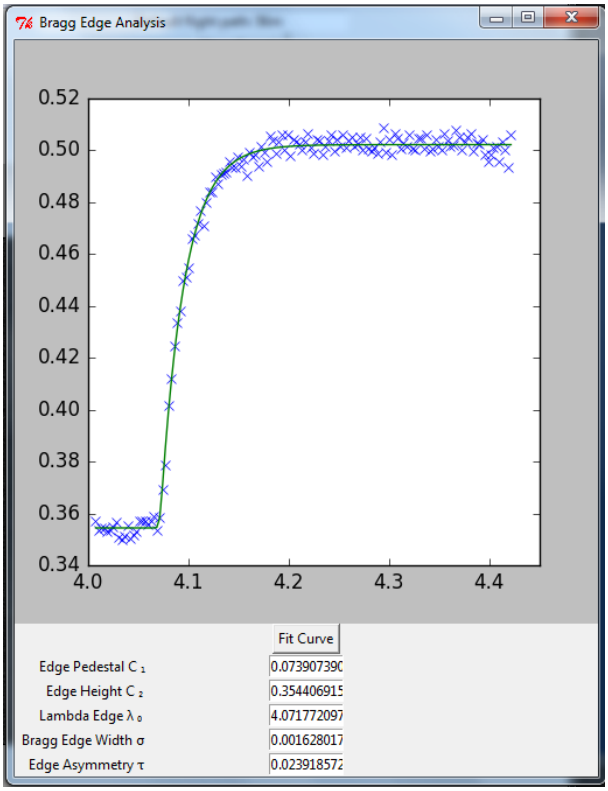

Figure 6: A selected Bragg edge with analytical function fitted and parameters used visible