ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUMBCM004

REAL-TIME VISUALIZATION AND PEAK FITTING OF TIME-OF-FLIGHT NEUTRON DIFFRACTION AT VULCAN

B. A. Sobhani, Y. Chen, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Abstract

In situ neutron diffraction experiments on state-of-the-art diffractometers such as VULCAN at SNS, enable capturing key transients in the dynamic material evolution, for in-depth understanding of material science and engineering. Traditional data processing needs to transfer the time-of-flight neutron events to another computation cluster for post-experiment data reduction and analysis. This is not timely, and it thus cannot meet the demands of on-the-fly decision making and close-loop experiment control. This paper demonstrates an automatic and intuitive system at VULCAN that is developed in EPICS for a real-time visualization of live neutron diffraction patterns and Bragg peaks' fitting via generalized linear regression. The live plots and the results quantify the dynamic of the peak position, intensity, and width, which are indicators of materials straining, phase transformation, microstructure evolution and so on. It constructs a promising platform of future smart neutron experiment controls.

REGRESSION

Linear regression is a versatile technique for not only linear function fitting but also the parameter estimation of complex functions or non-linear curves. The "linear" requirement only needs to apply to the fitting coefficients, so a class of functions that is non-linear in the independent variable can be fit by linear regression, as long as it is linear with respect to the fitting coefficients.

However, if one wants to fit a gaussian function, for example, to data using fitting parameters of amplitude, mean, and standard deviation, linearity does not hold.

$$f(A, \mu, \sigma) = A * e^{-(x-\mu)^2/(2*\sigma^2)}$$

Fortunately, this is of little concern because the Levenberg-Marquardt algorithm allows one to fit functions that are nonlinear even in the fitting coefficients. Implementations of the Levenberg-Marquardt are widely available on the internet. In particular, scipy's optimize.curve_fit function calls this algorithm so it can be used to fit functions that are nonlinear in the fitting coefficients. As a matter of curiosity, scipy's optimize.curve_fit function is largely not implemented in python but is instead only a python interface to the compiled Fortran fitting library called MINPACK,[1] which was developed at Argonne National Laboratory in the 1980s, and LAPACK for calculating the covariance matrix.

PHYSICAL INTERPRETATIONS

The peak profile fitting exports some basic parameters such as peak position, area and width, which reflect the material status. In the following, the lattice spacing d as Bragg's peak position and the lattice strain are employed for demonstration. When neutrons with the wavelength λ are diffracted by the lattice plane of a crystal with spacing of d, the Bragg's Law as the following must be satisfied:

$$\lambda = 2d\ sin\theta$$

where 2θ is the scattering angle.

This can be related to time-of-flight data from SNS neutron detectors according to the following:

$$\frac{\mathrm{ht}}{\mathrm{mL}} = 2d\ sin\theta$$

Given the instrument optics, the measured time-of-flight (t) can be converted to the lattice d spacing [2]. This is presented as the Bragg's peak position. It is well known that the peak profile function at the TOF diffractometer can be as complex as a back-to-back exponential function convoluted with a pseudo-Voigt function. For the purpose on demonstrating the relative changes of the peaks, it is rational to employ a simpler Gaussian function in this paper. Therefore, the lattice spacing d of a particular crystal plane can be estimated as μ by fitting the Gaussian function over the corresponding Bragg peak in the neutron diffraction pattern.

The lattice spacing d is responsive to the external stimulus such as temperature and stress. The relative change from a reference state d_0 (such as room temperature and stress-free state) is calculated as lattice strain $\varepsilon = (d-d_0)/d_0$. Over a dynamic process, the phase-specific and the lattice-plane-specific lattice strains evolve, reflecting materials properties such as thermal expansion and stiffness, and indicating possible material changes via different mechanisms.

The change of the lattice spacing shifts the Bragg's peak. In this way, the μ fitting parameter can be used to quantify the strain.

PEAK FITTING INTERFACE

User can select the peak to be fit by adjusting draggable bounds on a CSS Phoebus interface. The bounds then get written to PVs, and the peak fitting algorithm is run only on data between those two bounds.

PEAK QUALITY

Scipy's optimize.curve_fit function returns a covariance matrix as its second return value.[3] The diagonals of this matrix are the variance of the fitting parameters – these are measures of certainty of the fit, not related to the variance of the gaussian peak itself.

Since these variances give us a way to quantify the quality of the peak fit, this provides an efficient way of automating the process of determining when enough data has been collected in a measurement, so that the next measurement can begin.

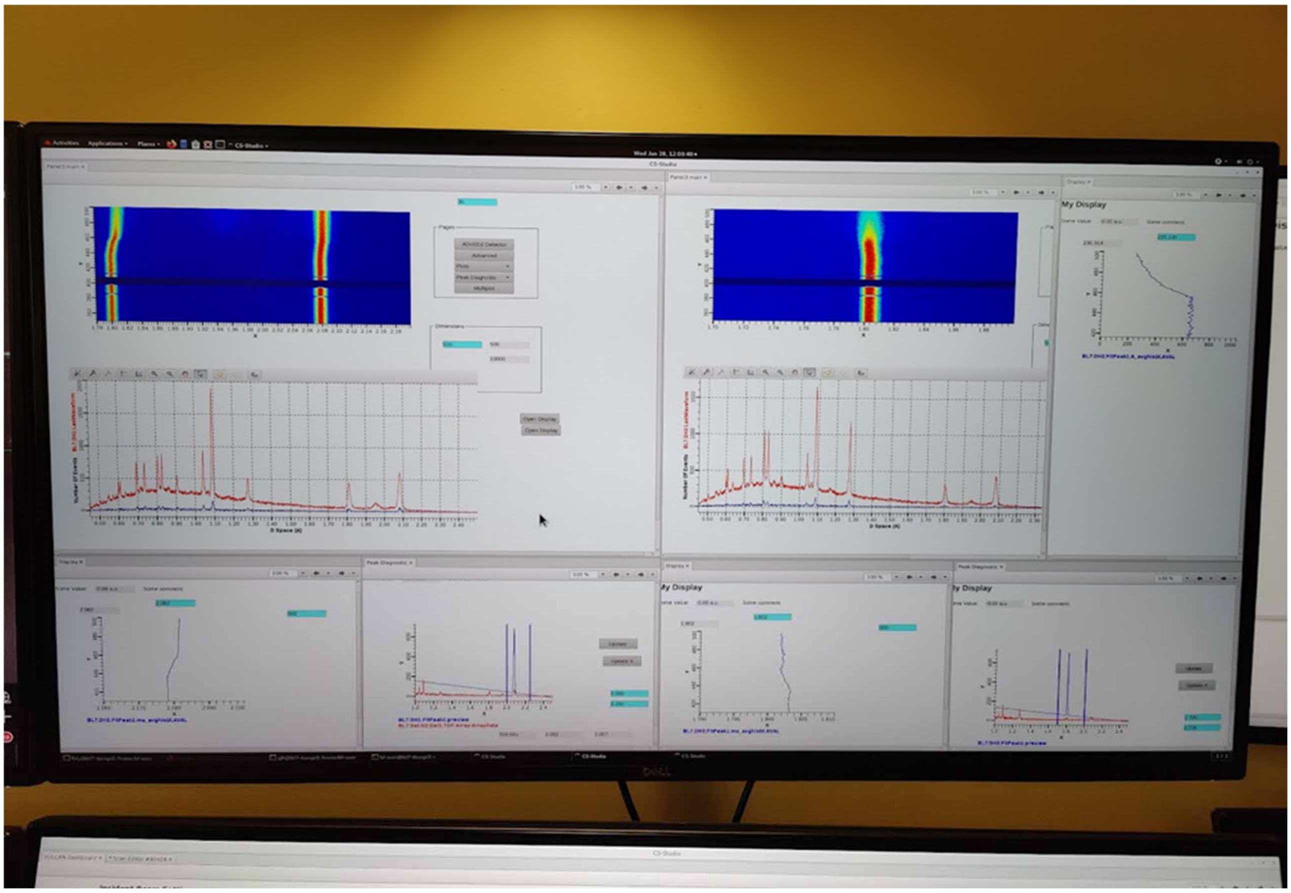

Figure 1: Peak fitting and waterfall plot in production use at VULCAN beamline at SNS.

WATERFALL PLOT

While peak-fitting is useful, it is not a substitute for visualizing the raw data. One reason is because when the peak fitting code has bugs, it can be difficult to visual detect. A waterfall plot is more robust in this way, in that it is for the most part a plot of raw data. A waterfall plot is a 3D plot with two spatial dimensions and one color dimension. The x-axis is d-spacing, y-axis is time, and color is the number of counts (see Fig.~1). This is a very straightforward and simple way to visualize peak trends, without having to fit the data. By plotting the fitting data alongside the waterfall plot, peak fitting can be validated visually, and peak-fitting imperfections can be spotted.

SOFTWARE

The waterfall plot is simply an areaDetector plugin that takes the dspace plot (also an areaDetector plugin, via ADnED) as its input. When the input is processed, a new row is added and the others are shifted down. There are limit PVs to control the boundaries of the plot according to specifications of beamline scientists (at VULCAN it is useful to look at the top few rows, let the view expand until a user-specified maximum number of rows is reached, and then shift down the plot for new rows added after the maximum is reached). The rate at which rows are added can be controlled by the MinCallbackTime PV of areaDetector.

The peak-fitting is done using scipy. In order to make the data from the IOC accessible by scipy, PyDevice[4] is used. PyDevice is a module that allows for python code to be executed from a regular EPICS, so that one gets both the stability of a “real EPICS” IOC as well as the flexibility being able to run python code to process data in real-time. Additionally, this also allows for the developer to have access to all the standard EPICS records types with their associated fields (e.g. HIGH, HIHI) to allow for more seamless integration into a conventional EPICS system.

CURRENT STATUS AND FUTURE PLANS

Waterfall and peak fitting functionalities have been found to be stable, and have been used in production with neutron beam at the VULCAN beamline at SNS. It was found that recent versions of python (e.g. 3.11) and scipy must be used, because of a bug that affects multithreading in old versions of scipy.

The versatility of python and its dependencies make for easy expansion of this IOC into certain other territories. For example, a beamline at HFIR in this same kind of peak fitting to determine peak quality, except for 2-dimentional data. With numpy it is very easy to convert 1D data into 2D and vice versa, so this alone will make the task of extending this peak-fitting to 2D much easier.

Currently, both the peak-fitting and waterfall plot functionalities are packaged in the same IOC. There are a few small advantages to this, such as the deployment is simplified and the sharing of data can be convenient in some cases. However, the longer term plan is to split the peak-fitting and waterfall functionalities into separate IOCs. This can make configuration easier in cases where only one of the functionalities is needed, and reduce complexity of debugging.

The waterfall plot is an areaDetector plugin, written in simple C++. This is the probably the most optimal way for this plugin to be implemented.

The peak-fitting operates on data after it is exported to waveform record by areaDetector. This works, but further investigation will be needed in the performance of this method. It seems inefficient to export the data to a waveform record, and then again move the data from the waveform record to inside the python code. Speculatively, it seems like it may be more efficient to have an areaDetector plugin with the d-space data directly call the python interpreter on the data, or call some compiled version of it (e.g. with Cython).

ACKNOWLEDGEMENTS

Acknowledgements to VULCAN beamline scientist Yan Chen for guiding the development of this IOC, and to the VULCAN team in general.

This research used resources at the Spallation Neutron Source, a DOE Office of Science User Facility operated by the Oak Ridge National Laboratory.

REFERENCES

[1] https://github.com/scipy/scipy/blob/bf776169c753fff655200dc15ae26db95a083b02/scipy/optimize/minpack/lmdif.f

[2] http://pd.chem.ucl.ac.uk/pdnn/inst3/tof.htm

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

[4] https://github.com/klemenv/PyDevice