bfit: A Python Application For Beta-Detected NMR

Derek Fujimoto[1,2]

1 Stewart Blusson Quantum Matter Institute, University of British Columbia, Vancouver, BC V6T 1Z4, Canada **2** Department of Physics and Astronomy, University of British Columbia, Vancouver, BC V6T 1Z1, Canada

Summary

Beta-detected nuclear magnetic resonance (β-NMR) measures the beta-decay of probe radioactive nuclei to infer the electromagnetic character of the probe's local environment. Similar to muon spin rotation (μSR), this technique allows for unique insight of material properties not easily measured by conventional NMR. The `bfit` package provides a graphical user interface (GUI) and application programming interface (API) to facilitate the analysis of implanted-ion β-NMR measurements taken at TRIUMF.

Background

β-NMR leverages the parity-violating nuclear weak interaction to measure the spin precession of a ensemble of radioactive probe nuclei (MacFarlane, 2015). These nuclei can either be activated by neutrons or implanted as a foreign species in the form of a low-energy particle beam. Upon decay, the direction of the emitted electron is correlated with the nuclear spin orientation. As with many nuclear and particle physics experiments, the data collected is the counted number of electrons emitted in a given direction. These counts are then histogrammed and processed to yield a signal of interest.

The activation or implantation of the probe nuclei require high-intensity particle beams, restricting the technique to large nationally-supported facilities. Even today, there are only a handful of locations capable of conducting β-NMR measurements, such as TRIUMF, which is situated in Vancouver, Canada. This facility has been running β-NMR experiments for the past 20 years, and has developed the Muon Data (MUD) file format (Whidden et al., 1994) as a means of storing μSR and β-NMR data.

Statement of need

At TRIUMF, β-NMR receives approximately 5 weeks of radioactive beam time per year. As with other large-facility experiments employing particle beams, this data is extremely limited and expensive to generate. Having the tools for rapid on-line analysis is therefore crucial for efficient and informed measurement. Additionally, many of the experimenters using the β-NMR spectrometer are visiting scientists or students who have little experience with the technical aspects of the measurement.

As with many older science applications, the MUD API is written in C and FORTRAN. These statically-typed and compiled languages are known for their computational efficiency, but are accompanied by long development times, relative to modern languages. In many

communities, scientific computing has shifted to languages such as Python: a dynamically-typed and interpreted language. As a result, Python has amassed a massive library of data analysis tools (Virtanen et al., 2020). The short development time of Python programs is particularly important in the context of scientific analyses, which are typically run only a few times by select individuals. As a result, the development time of the analysis code comprises a large part of the program's effective run time. The aim of this work is to bring this rapid prototyping style of analysis to β-NMR. To further streamline on-line analyses, `bfit` provides an intuitive GUI capable of a moderately high degree of sophistication.

It should be acknowledged that, while a large body of analysis software exists to support μSR workers (such as WIMDA (Pratt, 2000), MANTID (Arnold et al., 2014), and Musrfit (Suter & Wojek, 2012)), β-NMR does not have a comparably extensive suite of maintained analysis programs. While there have been some recent improvements to this situation (Saadaoui, 2018), the analysis required for any non-trivial β-NMR experiment necessitates the development of new code to meet the individual requirements of each experiment. While such code may employ Musrfit, which is compatible with the MUD file format, this approach may be cumbersome for complex or rapid analyses, and presents a entry high entry barrier for new users. The Python API of `bfit` is well suited for addressing these issues.

Usage and features

The `bfit` GUI has three primary functions which are contained in the *Inspect*, *Fetch*, and *Fit* tabs. The purpose of the *Inspect* tab (shown below) is to quickly view the file headers and plot the data in order to detect and solve problems as they may arise during measurement. The *Fetch* tab has been designed to prepare the data for analysis, loading runs in batch and allowing the user to draw and compare each run. The *Fit* tab provides the tools needed to fit a model to the data, and to view and analyze the result. These tools include global fitting (i.e., sharing fit parameters between data sets), constrained fitting (i.e., constraining a parameter to follow a specific model dependent on the experimental conditions, such as temperature), non-trivial fitting functions specific to pulsed-beam operation (leveraging double exponential integration (Cook, 2014)), multiple minimization routines, and more.
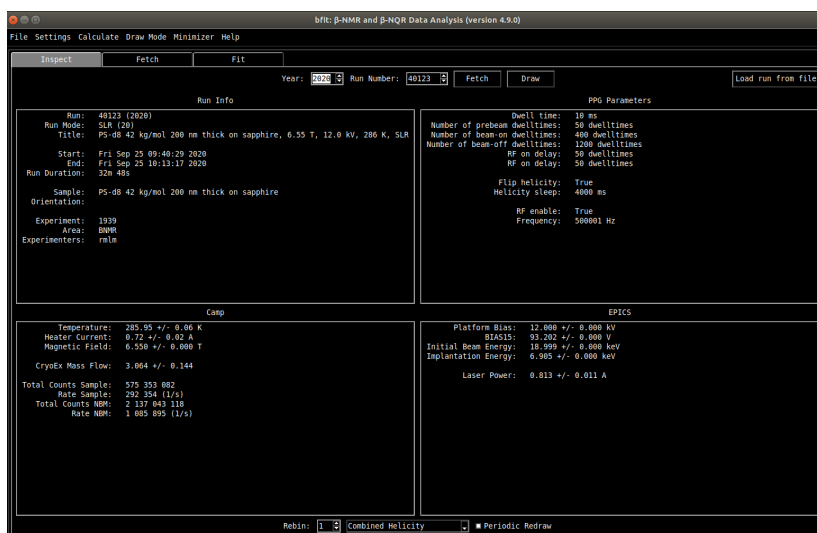

Figure 1: The inspection tab of the `bfit` GUI.

While the GUI greatly facilitates rapid on-line analysis, the `bfit` API provides the flexibility needed for publishable analyses. The analysis tools and functions utilized in the GUI are

readily accessible via the API, and documented in the wiki. Many of these tools are very general, easily interfacing with other Python packages, and can accommodate a great deal of complexity and sophistication.

Acknowledgements

References

Arnold, O., Bilheux, J. C., Borreguero, J. M., Buts, A., Campbell, S. I., Chapon, L., Doucet, M., Draper, N., Ferraz Leal, R., Gigg, M. A., Lynch, V. E., Markvardsen, A., Mikkelson, D. J., Mikkelson, R. L., Miller, R., Palmen, K., Parker, P., Passos, G., Perring, T. G., … Zikovsky, J. (2014). Mantid–Data analysis and visualization package for neutron scattering and μ SR experiments. *Nucl. Instrum. Methods Phys. Res. Sect. A*, *764*, 156–166. https://doi.org/10.1016/j.nima.2014.07.029

Cook, J. D. (2014). https://www.codeproject.com/Articles/31550/Fast-Numerical-Integration

MacFarlane, W. A. (2015). Implanted-ion βNMR: A new probe for nanoscience. *Solid State Nucl. Magn. Reson.*, *68-69*, 1–12. https://doi.org/10.1016/j.ssnmr.2015.02.004

Pratt, F. L. (2000). WIMDA: a muon data analysis program for the Windows PC. *Phys. B Condens. Matter*, *289-290*, 710–714. https://doi.org/10.1016/S0921-4526(00)00328-8

Saadaoui, H. (2018). BnmrOffice: A Free Software for β-nmr Data Analysis. *JPS Conf. Proc.* https://doi.org/10.7566/JPSCP.21.011049

Suter, A., & Wojek, B. M. (2012). Musrfit: A Free Platform-Independent Framework for μSR Data Analysis. *Phys. Procedia*, *30*, 69–73. https://doi.org/10.1016/j.phpro.2012.04.042

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Mulbregt, P. van. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Whidden, T., Arseneau, D., & Daviel, S. (1994). *MUon Data access*. http://musr.ca/mud/