Quasi Elastic Neutron Scattering model library

Céline Durniak[1,*], *Miguel Angel* González[2], *Anders* Markvardsen[3], *Sanghamitra* Mukhopadhyay[3], *Franz* Lang[3], and *Thomas Holm* Rod[1]

[1]European Spallation Source ERIC, Ole Maaløes Vej 3, DK-2200 Copenhagen, Denmark
[2]Institut Laue Langevin, 38042 Grenoble, France
[3]ISIS Neutron and Muon Source, STFC Rutherford Appleton Laboratory, Oxford, UK

Abstract. This paper reports on the development of a collection of dynamical models of one-dimensional peak profile functions used to fit dynamic structure factors $S(Q,\hbar\omega)$ of Quasi Elastic Neutron Scattering (QENS) data. The objective of this development is to create a maintainable and interoperable Python library with models reusable in other projects related to the analysis of data from Quasi Elastic Neutron Scattering experiments. The ambition is that the library also will serve as a platform where scientists can make their models available for others. We illustrate how the library can be used by newcomers to the field as well as by experts via different examples. These examples, provided as Jupyter notebooks, show how the QENS models can be integrated in the whole QENS data processing pipeline.

1 Motivation and significance

Quasi elastic Neutron Scattering (QENS) is a neutron scattering technique to probe displacements in temporal and spatial scales, typically from pico- to tens of nanoseconds and from Ångströms to nanometres, respectively. In this technique one measures the dynamic structure factor $S(Q,\hbar\omega)$ around the elastic line with $\hbar\omega = 0$. Here Q is the momentum exchange between the neutron and the investigated sample and $\hbar\omega$ is the energy transfer. The broadening of the elastic peak can be related to diffusive stochastic motions taking place in the sample, including translational diffusion (both unconstrained long-range translations and confined translation), molecular reorientations, localised motions (e.g., methyl rotations), etc. [1] Correspondingly, QENS finds applications in various scientific domains, such as biology (e.g., to explore the dynamics of proteins) [5, 6], soft-matter (e.g., to investigate different relaxation modes in polymers) [7, 8] or materials science (e.g., to probe H-diffusion in hydrogen storage materials) [9], among others. Each type of motion is characterised by a specific Q-dependence of the intensity and line-width of the quasi elastic signal for a given Q, which can be represented by an analytical model. Therefore, a large series of analytical models corresponding to the different types of motions mentioned above exist and can be used to fit the experimental data in order to extract the relevant physical parameters (e.g., self-diffusion constants, residence times, jump lengths, etc.) from the QENS spectra [1].

The QENS library makes a range of these mathematical models freely available to practitioners of QENS and developers of QENS analysis software in a format that makes them inter-operable and reusable. As an open-source library, the implementations can be scrutinised and validated by peers, and scientists are encouraged to add their own models to make them available for the QENS community at large.

The QENS library at its core provides Python models of mainly one-dimensional (1D) line profiles to fit experimental or simulated $S(Q,\hbar\omega \approx 0)$ data. This work was part of SINE2020 Work Package 10 on Data Treatment [10] to develop a comprehensive library of dynamical models in order to increase interoperability and reusability, for instance for rapid prototyping. The library provides different building blocks that users can combine, convolute and plug into different frameworks for visualizing or fitting.

Our approach is similar to that implemented by the `SasView` [11] developer community for small angle scattering with its SasView Marketplace [12].

The library is written in Python and depends on standard scientific Python modules (i.e., `scipy` [13] and `numpy` [14]). It is installed using the package installer for Python, `pip`.

2 Software description

The library is developed under an open-source license (BSD 3-Clause) and the source is stored in a GitHub repository at https://github.com/QENSlibrary/QENSmodels/. The models are written in Python facilitating integration with other packages of the large Python eco-system as well as easy integration in Python based workflows. Moreover, Python is a very popular scripting and programming

*e-mail: celine.durniak@ess.eu

[1]Refer to [1–4] for more details about the technique.

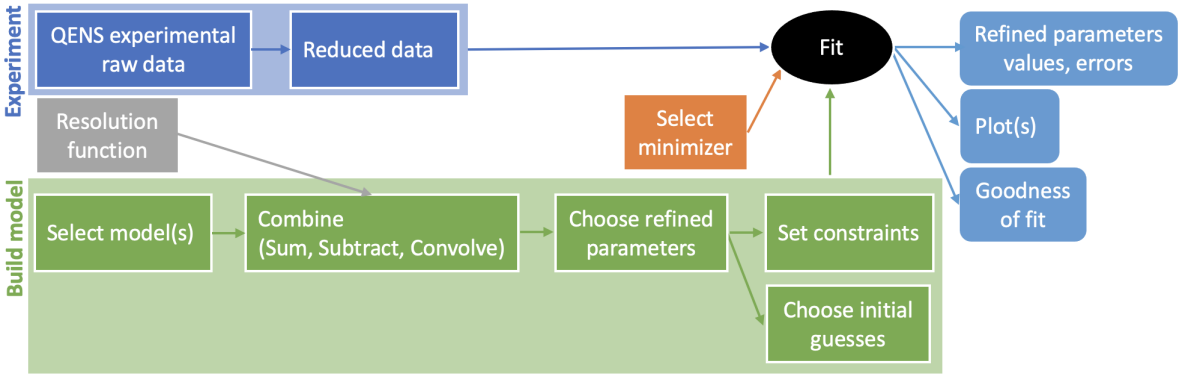

Figure 1. (Colour on-line) Workflow of QENS data processing from raw experimental data to output of fitting.

language in science and the Python interface therefore reduces the entry barrier for the many scientists already familiar with Python.

The library comes with examples of how it can be used in practise and in conjunction with existing Python modules for optimization (fitting), visualization, and for adding Graphical User Interface (GUI) elements.

Figure 1 shows a typical QENS data processing workflow. Data are collected during an experiment and then corrected (background subtraction, detector efficiency normalisation, absorption and multiple scattering corrections, etc.) in order to obtain the experimental $S(Q,\hbar\omega)$ for our sample [3]. The latter is then compared to the relevant mathematical models and typically some important parameters characterising the sample dynamics and the motion geometry are obtained by fitting the model parameters to the experimental signal. Data collected from multiple experiments can also be used in this procedure for simultaneous or sequential fittings. Different tools exist to deal with these stages with different levels of coverage of the whole workflow.

For example, MantidWorkbench [15] can deal with data reduction, fitting and plotting.

If we focus on fitting only, the available tools are more universal and not restricted to the QENS technique. There are, for example, `bumps` [16], `lmfit` [17] or `scipy` [13]. Therefore, in the present library we provide different models, (corresponding to cell 'Select models(s)' in Fig. 1) that can be combined and plugged in different frameworks. The following subsections detail the architecture of the library and its main functionalities.

2.1 Software Architecture

Figure 2 shows the structure of the repository. Each file in the QENSmodels folder contains the implementation of exactly one model in the form of a Python 3 function that returns $S(Q,\hbar\omega)$. For more complicated models, functions returning the Q-dependence of the half-width at half maximum (HWHM), the elastic incoherent structure factor (EISF) and the quasi-elastic incoherent structure factors (QISF) for the 1D peak are also provided. We refer to [8, 18] for definitions of these terms.

```
QENSmodels repository
├── QENSmodels/ ...
│   Python module files with QENS
│   models.
├── docs/
│   └── examples/ ...
│       Jupyter notebooks.
│       ├── data/ ...
│       │   Reference datafiles used in
│       │   Jupyter notebooks.
│       └── using_mantid/
├── tests/ ...   Python scripts of unit tests on
│                the QENS models.
├── tools ...    Jupyter notebooks and scripts
│                to run unit tests and provide
│                additional information on QENS
│                models.
├── CONTRIBUTING.rst
├── DISCLAIMER.rst
├── LICENSE.txt
├── MANIFEST.in
├── README.rst
└── pyproject.toml
```

Figure 2. Structure of QENS models repository.

Usability in a scripting environment, such as Jupyter, is facilitated by providing rich documentation through so-called documentation strings using the Sphinx [19] standard including so-called doctest [20]. By adhering to the Sphinx standard, documentation pages can be automatically generated and updated from the code itself. Moreover, to help the interested user and to lower the entry barrier, examples using Jupyter notebooks and Python scripts are provided as well as instructions for scientists wishing to contribute their own models to the library. Online documentation is also available on `readthedocs` at https://qensmodels.readthedocs.io/.

Code quality and maintainability is facilitated by using git for version control and by implementing unit tests,

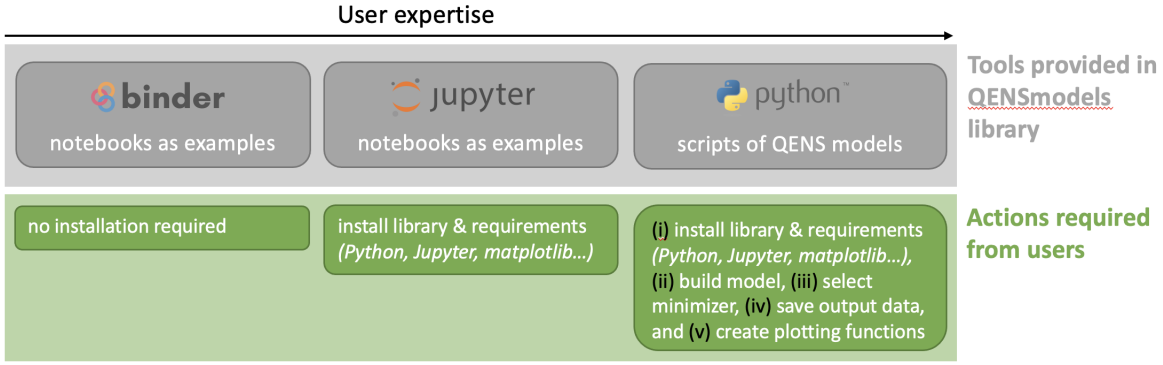

Figure 3. (Colour on-line) Diagram showing the tools provided in the QENSmodels library to help users with different levels of expertise in QENS and in programming.

including the aforementioned doctests,[2], which are used for continuous integration using GiHub Actions. This approach to software development is in alignment with the guidelines outlined in [21].

2.2 Software Functionalities

Figure 3 shows the options available for using the library. In order to give an idea of what can be done with the library without any installation, the Jupyter notebooks [22] are available through Binder [23]. Users only have to click on the link to be able to run the notebooks and check if the library can answer their needs.

On the other hand, for users only interested in the models, for example, to add them to their fitting pipelines, the installation of the library requires only a command line. In addition, they will have to perform all steps as listed in the third column of Figure 3 in order to use the model for their cases.

As mentioned in the previous section, examples are provided as Jupyter notebooks. They use a few QENS models and a few fitting engines, like `lmfit`, `scipy`, or `bumps`. The notebooks and modules required to make them run can be installed by following the guidelines on the github pages [24]. Note that the minimizers used in these notebooks were only to illustrate the specific syntaxes of the fitting engines and not because they were the most optimal engines for the investigated cases. Details about comparison of minimizers can be found at [25].

Physical units used by the models' parameters and variables are specified at the beginning of the Jupyter notebooks. A notebook, `Convert_Units.ipynb` is also provided to convert between different physical units in case the reference user's data are expressed in different units than the standards chosen for the QENS library.

[2] A unit test checks that a single component operates in the right way. It helps to isolate where something in the code is broken. Doctest provides a testing framework using code examples for documenting and testing Python code.

3 Illustrative Examples

3.1 Toy examples in the library

As tutorials, Jupyter notebooks [22] are provided showing how to (i) build a model using different functions from the QENS library, (ii) choose a fitting engine, (iii) define the settings and run the fit, (iv) extract visual and quantitative information from the outputs.

Since the syntax to build the fitting model, apply constrains and set the initial guesses depends on the chosen fitting engine, different minimizers have been used in the notebooks: `scipy` [13], `bumps` [16], `lmfit` [17]. For an easier identification, the notebooks have been named following this convention: "*[fitting engine]_[name of QENSmodels]*_fit.ipynb", e.g., "`scipy_lorentzian_fit.ipynb`".

Reference data used in the examples are either generated in the dedicated Jupyter notebook or stored in one of the files in the `data` sub-folder of the repository. We provide these reference data in binary (hdf5) and ascii formats. The resolution function is either a Gaussian profile or reduced data from a Vanadium run. And the convolution with the sample data is done using methods from `numpy` [14] or `lmfit` [17].

We are now going to detail one of these notebooks, `bumps_Brownian_Diff_fit.ipynb`. Figure 4 details its table of contents. After a short introduction describing the objective of the notebook, we specify the physical units of the refined parameters and import the required Python libraries as well as the reference data for the sample and resolution function. Figure 5 shows the evolution of these functions for different Q values. The convolution between the sample and the resolution functions is done using a method from `numpy` [14].

Brownian Translational diffusion *
Resolution with bumps

- Introduction
  - Physical units
- Import libraries
- Setting of fitting
  - Load reference data
  - Display units of input data
  - Create fitting model
  - Display initial configuration
  - Choice of minimizers for bumps
  - Setting for running bumps
- Running the fit
- Showing the results
  - Display final configuration

Figure 4. Table of contents of one of the Jupyter notebooks available in the library: `bumps_Brownian_Diff_fit.ipynb`

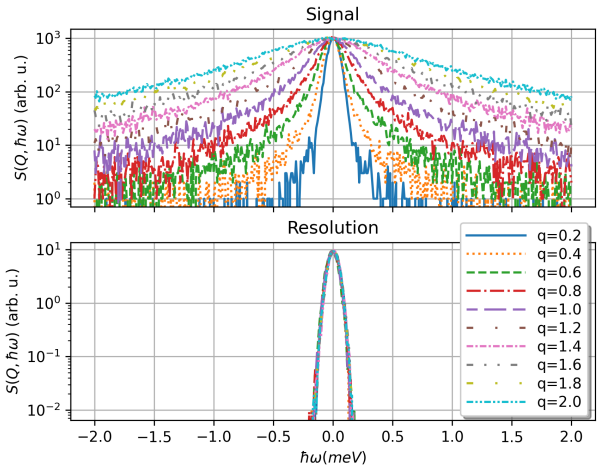

Figure 5. (Colour on-line) Plot of the reference data used in `bumps_Brownian_Diff_fit.ipynb`.

The fitting model is expressed as:

$$S(Q,\omega) = R(Q,\hbar\omega) \circledast \left[\frac{\text{scale}}{\pi} \frac{DQ^2}{(\hbar\omega - \text{center})^2 + (DQ^2)^2} \right],$$

where scale and center are the scale factor and center of the Lorentzian function describing the diffusion, of half-width at half-maximum equal to DQ^2, where D is the self-diffusion coefficient and Q the momentum transfer. Listing 1 shows how the fitting model was created following the syntax required by the fitting engine, `bumps`. In order to help less Python savvy users, several GUI widgets are provided in the notebooks, for example, to choose the minimizer, the number of iterations or for interactive plots as shown in Figure 6. After running the fit, the values and errors of the refined parameters and the fit quality are printed and a plot of the fitted model in comparison with the reference data is shown for the value of Q selected by the user via a widget.

Listing 1. Code snippet showing how to create a fitting model using 'bumps' and the QENS library

```
import numpy as np
from QENSmodels import \
    sqwBrownianTranslationalDiffusion
import bumps.names as bmp
# Fitting model
def model_convol(x, q, scale=1, center=0, D=1,
resolution=None):
    model = sqwBrownianTranslationalDiffusion(x, q, scale,
center, D)
    return np.convolve(model,
                       resolution/resolution.sum(),
                       mode='same')
# Fit
model_all_qs = []

for i in range(len(q)):
    # Bumps fitting model
    model_q = bmp.Curve(
        model_convol,
        hw,
        sqw[:, i],
        err[:, i],
        name=f'q_{q[i]:.2f}',
        q=q[i],
        scale=1000,
        center=0.0,
        D=0.1,
        resolution=res[:,i])
    model_q.scale.range(0, 1e5)
    model_q.center.range(-0.1, 0.1)
    model_q.D.range(1e-12, 1)
    # Q-independent parameters
    if i == 0:
        D_q = model_q.D
    else:
        model_q.D = D_q
    model_all_qs.append(model_q)
problem = bmp.FitProblem(model_all_qs)
```

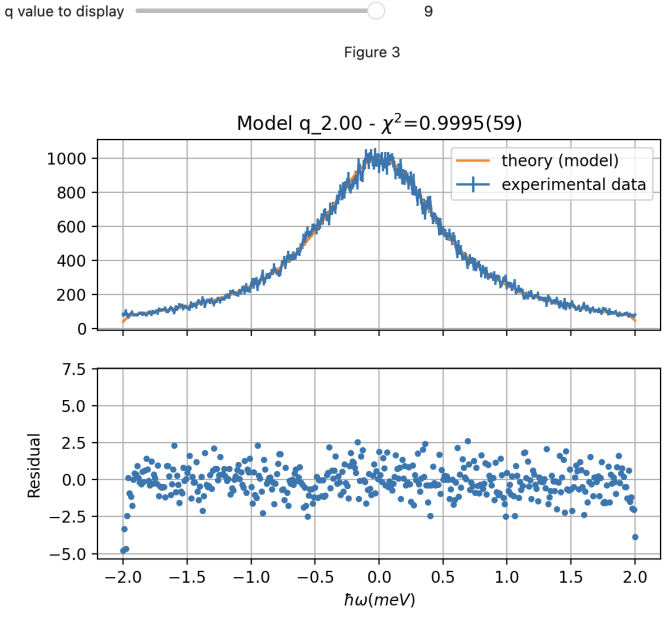

Figure 6. (Colour on-line) Widget to select the index in the list of available Q-values to plot the fitted model in comparison with the reference data as well as the residual. Here the configuration corresponding to Q = 2 Å is shown.

In addition to these notebooks, we also provide instructions to make the library importable in MantidWorkbench [15] as well as a Python script using one of the

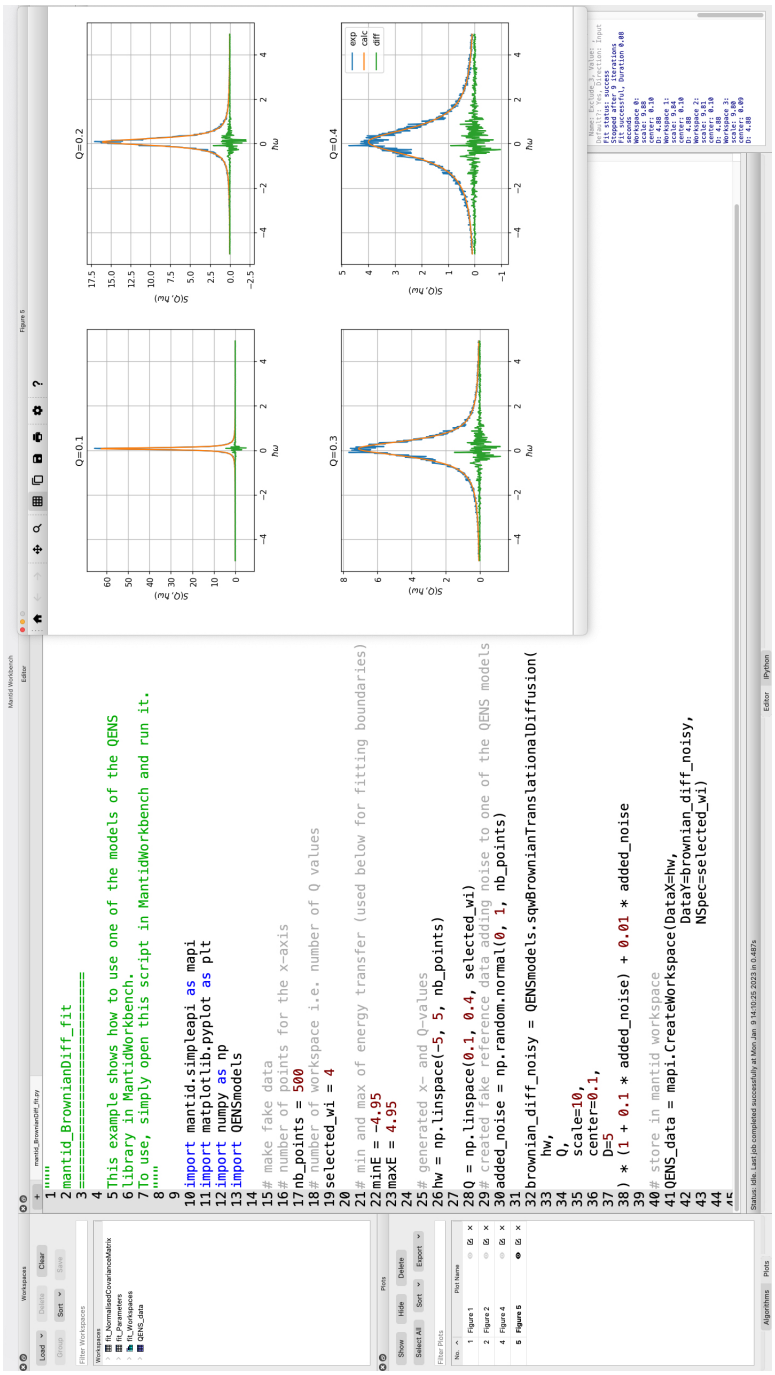

Figure 7. (Colour on-line) Graphical User Interface of MantidWorkbench [26] showing the use of the Python script `mantid_BrownianDiff_fit.py` provided in the library. The Python script is displayed in the editor. After its execution, workspaces, i.e., Mantid data structures, are available for further operations in the Graphical User Interface (left-hand side column) and a separate window displays the fitting result for the 4 Q-values considered in this example.

models to fit data and plot and display the output in MantidWorkbench (see Figure 7). Reference [27] details the QENS features available in this framework.

3.2 Benchmarking with experimental data

One of the notebooks, `lmfit_TOFTOF_delta_lorentz.ipynb` uses reduced data presented in [28].

In this publication, the authors used QENS, among other techniques, to investigate the microscopic details of the proton motions in two proton conducting, acceptor-doped, perovskites. The samples were measured at the time-of-flight spectrometer TOFTOF at MLZ at different temperatures using an incident wavelength of 2.5 Å. Nine "Q-cuts" were retained in the analysis. The reduced data files for one of the investigated samples $BaZr_{0.8}In_{0.2}O_3H_{0.2}$ at several temperatures have been provided by D. Noferini. The reduced experimental data were fitted using the fol-

lowing model:

$$S_{\text{meas}}(Q,\hbar\omega) = \{S(Q,\hbar\omega)[1 + n(\hbar\omega)]\hbar\omega\beta\} \circledast R(Q,\hbar\omega),$$

where $S(Q,\hbar\omega) = a_D(Q)\delta(\hbar\omega) + a_L(Q)L(Q,\hbar\omega) + \text{bkg}(Q)$, $n(\hbar\omega) = [\exp(\frac{\hbar\omega}{k_B T}) - 1]^{-1}$ is the Bose factor, and $\beta = (k_B T)^{-1}$ is the reciprocal of the thermodynamic temperature. $\delta(\hbar\omega)$ and $L(Q,\hbar\omega)$ are a Dirac and a Lorentzian peaks, respectively. $R(Q,\hbar\omega)$ is the resolution function (spectrum of a vanadium standard). bkg(Q) is the constant background, only Q-dependent and $\circledast$ is the convolution symbol.

The Jupyter notebook describes how to load the reduced data, build the model using the QENS library, select the minimizer and the range of Q-values to consider, run batch fitting over successive Q-values at different temperatures and extract, store and display results from the fit. Figure 8 shows the comparison between an experimental profile (bullet points with errorbars) and the fitted model (solid line) for $Q = 2.3$ Å at a temperature of $T = 540K$ together with the initial model (dashed curve). The good agreement between the final fit and the experimental data is clearly evident. Interested readers are encouraged to consult ref. [28] for further details.

This example could be further extended by providing loading of reduced data from different instruments or different data formats to, for example, easily compare the results collected during experiments at different institutes.

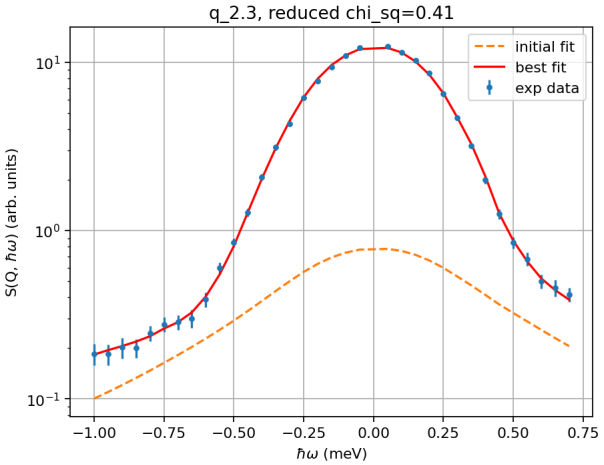

Figure 8. (Colour on-line) Experimental $S(Q,\hbar\omega)$ (bullet points with errorbars) from [28] and model with initial guesses (dashed orange line) and fitting outputs (solid red line).

4 Impact

The QENS library helps researchers and scientists by providing a list of 1D functions to fit QENS data making the analysis more FAIR, i.e., Findable Accessible Interoperable Reusable [29] by providing a public reference to fitting models allowing other researchers to reproduce the analysis.

The dynamics observed in QENS experiments are mainly related to the complexities of the samples studied. But collected signals can also contain the signature of the instrument and of the sample environment, which are getting more sophisticated. The combination of both factors may render the extraction and modelling of the experimental $S(Q,\hbar\omega)$ challenging. But this process can be made easier thanks to the flexibility of customising fitting models with the QENS library and different instrument resolution functions.

The library can be imported in different frameworks as illustrated in Section 3 with MantidWorkbench. This example shows how to use Python scripts. But the library could also be used in Graphical User Interfaces, like the QENS data analysis interface of Mantid described in [27].

Given that the library is installed using `pip`, it can be included in an automated data processing pipeline (reduction and analysis) or, for example, in a Machine Learning workflow, to build the optimal fitting model with ease, which increases the reach and usability of our development.

5 Conclusions

We have developed a Python package called QENSmodels, that provides dynamic models of 1D line profiles to fit QENS experimental or simulated data. In order to illustrate how to hook these models to the whole data processing pipeline to newcomers and experts, we provide Python scripts and Jupyter notebooks using different data and fitting engines. In the future, we plan to integrate more models and other examples using experimental data and other fitting engines. Scientists are also encouraged to contribute implementations of their work with additional models or examples.

The authors wish to thank Daria Noferini and her co-authors for providing experimental data used for testing some of the models.

This project received funding From the European Union's Horizon 2020 research and Innovation programme under grant agreement No 654000.

References

[1] M. Bée, *Quasielastic Neutron Scattering, Principles and Applications in Solid State Chemistry, Biology and Materials Science* (Taylor & Francis, 1988), ISBN 9780852743713

[2] R. Hempelmann, *Quasielastic Neutron Scattering and Solid State Diffusion*, Oxford Neutron Scattering in C (Clarendon Press, 2000), ISBN 9780198517436

[3] M. Telling, *A Practical Guide to Quasi-elastic Neutron Scattering* (Royal Society of Chemistry, 2020), ISBN 9781788019262

[4] J.P. Embs, F. Juranyi, R. Hempelmann, Zeitschrift für Physikalische Chemie **224**, 5 (2010)

[5] R.E. Lechner, S. Longeville, *Quasielastic Neutron Scattering in Biology, Part II: Applications* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 355–397, ISBN 978-3-540-29111-4, https://doi.org/10.1007/3-540-29111-3_16

[6] D. Vural, X. Hu, B. Lindner, N. Jain, Y. Miao, X. Cheng, Z. Liu, L. Hong, J.C. Smith, Biochimica et Biophysica Acta (BBA) - General Subjects **1861**, 3638 (2017), science for Life – Recent Advances in Biochemical and Biophysical Methods

[7] V.G. Sakai, A. Arbe, Current Opinion in Colloid & Interface Science **14**, 381 (2009)

[8] Q. Berrod, K. Lagrené, J. Ollivier, J.M. Zanotti, EPJ Web of Conferences **188**, 05001 (2018)

[9] M. Karlsson, Phys. Chem. Chem. Phys. **17**, 26 (2015)

[10] *SINE2020 homepage*, https://www.sine2020.eu

[11] *sasview webpage*, www.sasview.org

[12] *sasview marketplace webpage*, http://marketplace.sasview.org/

[13] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright et al., Nature Methods **17**, 261 (2020)

[14] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith et al., Nature **585**, 357 (2020)

[15] O. Arnold, J. Bilheux, J. Borreguero, A. Buts, S. Campbell, L. Chapon, M. Doucet, N. Draper, R.F. Leal, M. Gigg et al., Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **764**, 156 (2014)

[16] P. Kienzle, J. Krycka, N. Patel, I. Sahin, *Bumps (version 0.8.0) [computer software]*

[17] M. Newville, T. Stensitzki, D.B. Allen, A. Ingargiola, *Lmfit: Non-linear least-square minimization and curve-fitting for python* (2014), https://doi.org/10.5281/zenodo.11813

[18] T. Matsuo, J. Peters, Life **12**, 1259 (2022)

[19] G. Brandl, URL http://sphinx-doc.org/sphinx.pdf (2021)

[20] *doctest documentation*, https://docs.python.org/3/library/doctest.html

[21] J. Wuttke, S. Cottrell, M. Gonzalez, A. Kaestner, A. Markvardsen, T. Rod, P. Rozyczko, G. Vardanyan, Journal of Neutron Research **24**, 1 (2022)

[22] *Jupyter webpage*, https://jupyter.org/

[23] Project Jupyter, Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris Holdgraf, Kyle Kelley, Gladys Nalvarte, Andrew Osheroff et al., *Binder 2.0 - Reproducible, interactive, sharable environments for science at scale*, in *Proceedings of the 17th Python in Science Conference*, edited by Fatih Akici, David Lippa, Dillon Niederhut, M. Pacer (2018), pp. 113 – 120

[24] *Using the jupyter notebooks in a virtual environment*, https://github.com/QENSlibrary/QENSmodels/tree/main/docs/examples#using-the-jupyter-notebooks-in-a-virtual-environme

[25] *Mantid documentation: Comparing minimizers*, https://docs.mantidproject.org/v3.8.0/concepts/FittingMinimizers.html

[26] *Mantid 6.5.0: Manipulation and analysis toolkit for instrument data.; mantid project.* (2022)

[27] S. Mukhopadhyay, B. Hewer, S. Howells, A. Markvardsen, Physica B: Condensed Matter **563**, 41 (2019)

[28] D. Noferini, M.M. Koza, S.M.H. Rahman, Z. Evenson, G.J. Nilsen, S. Eriksson, A.R. Wildes, M. Karlsson, Phys. Chem. Chem. Phys. **20**, 13697 (2018)

[29] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.W. Boiten, L.B. da Silva Santos, P.E. Bourne et al., Scientific Data **3**, 160018 (2016)