Science and Technology Facilities Council

Technical Report
STFC-TR-2026-004

Analysing SpinW 2D powder fitting using FitBenchmarking

J Huntley, L Protopapa, R Waite

February 2026

Enquiries concerning this report should be addressed to:

RAL Library
STFC Rutherford Appleton Laboratory
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445577
email: library@stfc.ac.uk

Science and Technology Facilities Council reports are available online at:
https://epubs.stfc.ac.uk

Accessibility: a Microsoft Word version of this document (for use with assistive technology) may be available on request.

DOI: 10.5286/stfctr.2026004

ISSN 2753-5797

STFC Author Identifiers (ORCIDs)

Author ORCIDs are provided where available.

Jessica Huntley	0009-0008-7651-6149
Letizia Protopapa	0009-0008-6781-5876
Richard Waite	0009-0006-4521-7950

Analysing SpinW 2D Powder Fitting using FitBenchmarking

Jessica Huntley, Letizia Protopapa, Richard Waite
Science and Technology Facilities Council, Rutherford Appleton Laboratory,
Harwell Campus, Didcot, OX11 0QX

February 9, 2026

1 Introduction

FitBenchmarking [9] is an open-source Python package for comparing different data-fitting software. In this report, we utilise FitBenchmarking to analyse the performance of different implementations of fitting algorithms for 2D powder averaging problems in SpinW [17], in particular examining the $CrCl_2$(pwm) dataset. SpinW is a linear spin wave theory (LSWT) modelling code written in MATLAB, and its `sw_fitpowder` class provides the capability to fit powder-averaged spectra to inelastic neutron scattering (INS) data.

1.1 FitBenchmarking

FitBenchmarking compares how different fitting algorithms perform for the same dataset, model, and initial parameter guesses. It determines the best model parameters by solving a non-linear least squares problem, using minimizers drawn from a range of data-fitting software packages.

Different optimization algorithms are more or less suited to different types of problems, and FitBenchmarking is designed to support scientific software developers in identifying the most robust and efficient algorithms for the type of data analysis they support within their software. It also enables scientists to explore the best algorithm for fitting their data to a given model, and is beneficial to mathematicians who want to expose new methods to users, and see what kinds of data are problematic.

FitBenchmarking is designed to interface directly to the source of the data, and the `parser` class is easily extendable, allowing new datasets to be added. The output generated by FitBenchmarking includes results tables and performance profiles that summarise minimizer performance using metrics such as runtime, accuracy, and energy usage. More detailed fitting reports containing plots and fitted parameters for each problem-minimizer combination are also generated. For this work, the existing `horace_parser` was extended to accept 2D powder fitting problems, and additional 2D plots were added to the FitBenchmarking reports.

1.2 SpinW

SpinW can be used to simulate spinwave dispersions and INS cross-sections. INS measurements typically require relatively large samples, which are not always possible to grow as single crystals and hence powder samples must be used.

The processed INS data are binned in energy transfer, ω, and the magnitude of the momentum transfer, $|Q|$, of the neutron, producing 2D voxels each with an associated intensity and error. An ideal powder sample measurement is an average over all crystal orientations - as such the scattering in 3D reciprocal space cannot be directly measured in a powder. This is problematic for magnetic scattering, as neutrons interact only with the magnetisation (and fluctuations thereof) perpendicular to the momentum transfer of the neutron. This presents a challenge for model fitting, as the resulting data typically exhibit broad features for which different models can provide an adequate description. It is often necessary for users to place some reasonable bounds on the initial starting parameters either from

the magnetic structure itself, other measurements such as susceptibility, or calculations such as DFT derived exchanges or point-charge calculations for single-ion anisotropies.

The functionality to fit and visualize INS powder data was recently added in SpinW v4.0, along with improvements to the treatment of energy resolution and parallelization over q-points required for performant powder averaging. Fitting is performed by a derivative-free simplex minimizer or a Levenberg-Marquardt (LM) algorithm developed in-house to support parameter bounds. There is also a global optimizer based on a particle-swarm algorithm [13]. The cost-functions minimised are unweighted and error-weighted least-squares.

1.3 Challenges fitting powder INS data

Powder averaging of the simulated INS cross-section is calculated by numerical integration at points distributed over a spherical surface of radius $|Q|$ around the origin in reciprocal space for each $|Q|$ considered. Points can be generated randomly (Monte-Carlo) or using a lattice of sampling points based on a pair of successive Fibonacci numbers, as described in [3]. The powder average is quite computationally expensive and fitting powder INS data can be slow.

The powder averaging also introduces noise in the cost-function surface which poses difficulties for derivative based minimizers such as LM. In particular the step-size used in the Jacobian estimation of the LM algorithm needs to be tuned by the user. A larger step-size may allow for fewer points in the powder average for quicker simulation, but if the step-size is too large, the minimizer may not perform well or estimate an accurate hessian at the minimum (from which parameter uncertainties are calculated). It can difficult and time-consuming to establish an adequate step-size, which is often problem dependent.

The derivative-free simplex minimizer does not have any tunable parameters and is anecdotally more robust to noise on the cost-function surface. However, it typically requires more iterations to reach the minimum, resulting in comparable (if not worse) execution times than the LM minimizer, particularly as the number of model parameters increases.

The least-squares based cost functions currently supported in SpinW behave well for data counted sufficiently long for the errors to be treated as Gaussian, however it is well known that in the Poisson regime these cost-functions are biased and a maximum-likelihood estimator (MLE) should be implemented [5, 7]. Such an MLE approach is not trivial to implement for data that have been scaled/normalised and had empty or high-temperature runs subtracted (e.g. [15, 2]). This is a particular problem for INS powder data as most of the voxels contain only background, which typically has fewer counts and lower stats. Anecdotally, the fitting of such data in SpinW seems to rely on a good initial estimate of the background for which the functional form is unknown (typically treated as polynomial in $|Q|$ and ω).

2 Problem Definition

In this report, analysis is run on INS spectra of $CrCl_2$(pym) (pwm = pyrimidine), collected using the LET spectrometer at ISIS. $CrCl_2$(pym) is a quasi-1D S=2 antiferromagnetic (AFM) metal-organic magnet, the structure and magnetic properties of which have been studied by Pitcarin et al [12]. In this previous study, measurements were taken at two temperatures, creating two datasets for the high and low temperature cases. Magnetic scattering was isolated from instrumental background by subtracting the high temperature dataset from the low temperature dataset. Both the high temperature dataset and the temperature subtracted dataset have been added as benchmarking problems within FitBenchmarking, and both will be considered within this report. We also added an extra dataset, which has been obtained from the high temperature one, by cropping the high energy values. This was done to facilitate fitting, as the full dataset included phonon scattering at high energy transfer that was not part of the model.

We are interested in fitting parameters of the magnetic Hamiltonian

$$\mathcal{H} = \sum_{\langle ij \rangle} -J_{ij} \boldsymbol{S_i} \cdot \boldsymbol{S_j} + \sum_i D(S_i^z)^2. \tag{1}$$

Here, $\boldsymbol{S_i}$ are spin vector operators, and the parameters to fit are J_{ij}, which are the three nearest neighbour

Heisenberg exchanges and D, which is a single-ion anisotropy corresponding the an easy-axis along the spin direction. For the temperature subtracted data, Pitcarin et al reported the parameter values shown in Table 1.

Parameter	Fitted Value (meV)
J_1	1.13
J_2	-0.10
J_3	-0.01
D	-0.11

Table 1: Parameter values reported in [12].

A similar unpublished study on the temperature subtracted dataset was carried out by R. Waite and M. D. Le, who aimed to replicate the results in Table 1 using SpinW. This work led to the results reported in Figures 1 and 2, which we will use as reference in this report to assess our results.

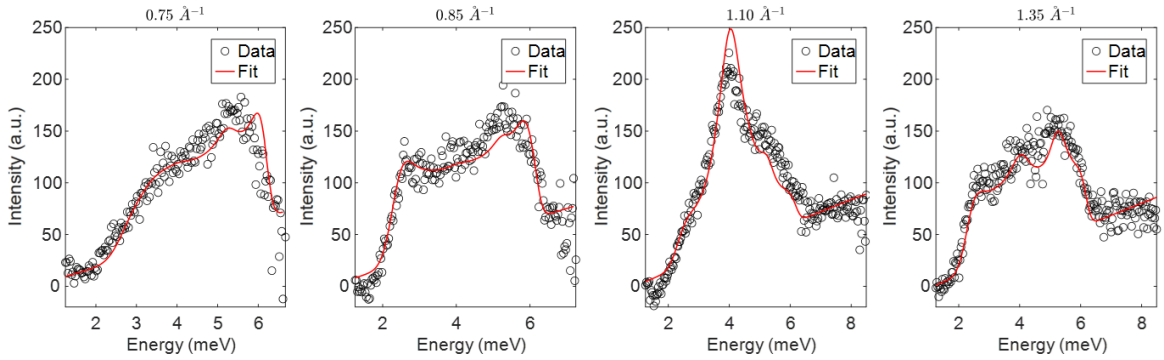


Figure 1: Reference 1D fits (temperature subtracted data).

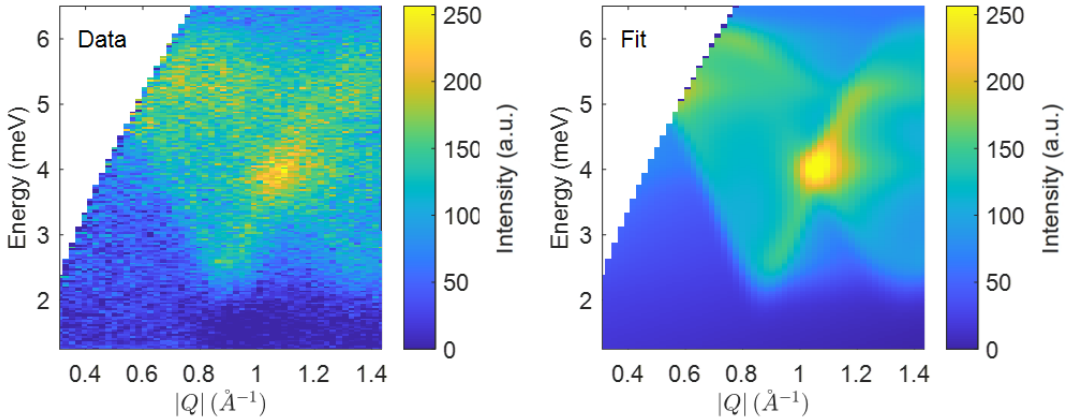


Figure 2: Temperature subtracted data and reference 2D fit.

3 Implementation in FitBenchmarking

3.1 Interfacing to SpinW

The existing `horace_parser` within FitBenchmarking was extended to allow for the parsing of SpinW 2D powder fitting problems. A user should define the problem of interest using a problem definition file, an example of which is shown in Listing 1. This file provides FitBenchmarking with the information required to load the data, construct the SpinW model, calculate the spinwave spectrum, and formulate a fitting problem to be run with the different minimizer options. Within this file, initial parameter values are set (shown on line 5), as well as acceptable parameter ranges (shown on line 8). The full set of initial parameter values and ranges used for the analysis in this report are shown in Section 3.3. For SpinW problems, in addition to the input data file, three MATLAB functions should be provided, defined in lines 5, 6, and 7 in the Listing, which load the data and simulate the spinwave spectrum using the fitpowder class in SpinW. In this report, FitBenchmarking is used to produce 2D plots of the fit, as well as 1D cuts at constant Q. The values at which to take the 1D cuts are provided in line 10 of the problem definition file, and in the case of this report, have been chosen to align with plots in Figure 1.

Listing 1: Example Problem Definition File

```
1  software = 'Horace'
2  name = 'CrCl2pym 2d'
3  description = 'CrCl2 pyrumidine data'
4  input_file = 'CrCl2Pym_1p7-25K_cropped.mat'
5  function = 'foreground=m_scripts/functions/CrCl2Pym.m, J1=0.85, J2=-0.075, J3=-0.005, D1=-0.1,
       slope_en=5, slope_modq=7, intercept=10, b1=2000'
6  wye_function = 'matlab_script=m_scripts/wye_functions/fb_wye_spinw_2d.m'
7  simulate_function = 'matlab_script=m_scripts/simulate_functions/fb_simulate_CrCl2Pym_2d.m'
8  parameter_ranges = {'J1': (0.0, 5.0), 'J2': (-5.0, 0.5), 'J3': (-5.0, 0.5), 'D1': (-2.0, 0.0)}
9  plot_type = '2d'
10 q_cens = '0.75, 0.85, 1.1, 1.35'
11 dq = '0.025'
```

3.2 Optimization Algorithms

The following algorithm types and implementations were analysed:

- The Levenberg-Marquardt (LM) algorithm, which is a trust-region approach that uses a combination of gradient descent and Gauss-Newton method [11]. Specific implementations considered in this work were `leastsq` from LMFIT [10] and `lm-bumps` from Bumps [8].
- Derivative-free methods, which do not compute the gradient of a function and so are often used to minimize problems with non-differentiable functions. Specific methods tested for this report were `dfols` from DFO [1, 4], `amoeba` from Bumps [8], `Powell` from SciPy [16], and `simplex` from MINUIT [6]. Additionally, the gradient-free optimizers [14] software package was considered, which offers local, global, population-based and sequential techniques. In particular, we include results from their `ParticleSwarmOptimizer` and `SimulatedAnnealingOptimizer`.

3.3 Initial Parameter Values and Ranges

Initial parameter values and ranges for model and background parameters used within FitBenchmarking are as shown in Tables 2 and 3. For global optimization algorithms, all parameters (including background parameters) require bounds. Background parameter bounds vary for each dataset, and are outlined in Table 4.

Parameter	Starting Guess High Temp. data `CrCl2pym 2d large`	Starting Guess Temp. subtracted data `CrCl2pym 2d subtracted`	Starting Guess Cropped data `CrCl2pym 2d cropped`
J_1	0.85	0.85	0.85
J_2	-0.075	-0.075	-0.075
J_3	-0.005	-0.005	-0.005
D	-0.1	-0.1	-0.1
slope_en	15	3.7	30
slope_modq	25	6.7	55
intercept	90	16.7	60
scale	2800	2332	2280

Table 2: Starting guesses for each parameter for each problem.

Parameter	Range
J_1	(0.0, 5.0)
J_2	(-5.0, 0.5)
J_3	(-5.0, 0.5)
D	(-2.0, 0.0)

Table 3: Parameter ranges considered for each of the model parameters.

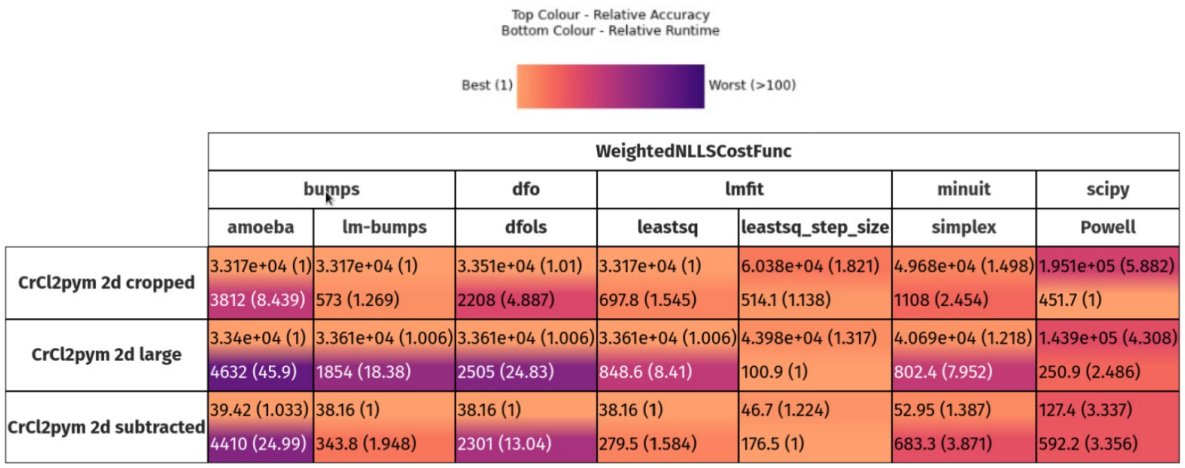
Parameter	Starting Guess High Temp. data `CrCl2pym 2d large`	Starting Guess Temp. subtracted data `CrCl2pym 2d subtracted`	Starting Guess Cropped data `CrCl2pym 2d cropped`
slope_en	(0.0, 30.0)	(0.0, 20.0)	(20.0, 50.0)
slope_modq	(10.0, 40.0)	(-20.0, 20.0)	(10.0, 70.0)
intercept	(50.0, 110.0)	(-20.0, 20.0)	(50.0, 80.0)
scale	(1000, 5000)	(1000, 5000)	(1000, 5000)

Table 4: Parameter ranges considered for each of the background parameters, set for global optimization algorithms only.

4 Benchmarking Results

In Table 3, we present results from LM and derivative-free methods, while results obtained with global minimizers are shown in Table 4. In Table 3, we also include `leastsq_step_size`, which is LMFIT's `leastsq` with a modified Jacobian step-size (rather than the default). Specifically, we used a step-size equal to 0.015, as this value lead to the best results.

Each row in these tables corresponds to a different problem and each column to a different minimizer. As depicted by the legend above the table, each cell shows accuracy and runtimes for a particular minimizer/problem pair. Results are colour-coded so that light orange corresponds to best results and purple to the worst ones.



	WeightedNLLSCostFunc						
	bumps		dfo	lmfit		minuit	scipy
	amoeba	lm-bumps	dfols	leastsq	leastsq_step_size	simplex	Powell
CrCl2pym 2d cropped	3.317e+04 (1) 3812 (8.439)	3.317e+04 (1) 573 (1.269)	3.351e+04 (1.01) 2208 (4.887)	3.317e+04 (1) 697.8 (1.545)	6.038e+04 (1.821) 514.1 (1.138)	4.968e+04 (1.498) 1108 (2.454)	1.951e+05 (5.882) 451.7 (1)
CrCl2pym 2d large	3.34e+04 (1) 4632 (45.9)	3.361e+04 (1.006) 1854 (18.38)	3.361e+04 (1.006) 2505 (24.83)	3.361e+04 (1.006) 848.6 (8.41)	4.398e+04 (1.317) 100.9 (1)	4.069e+04 (1.218) 802.4 (7.952)	1.439e+05 (4.308) 250.9 (2.486)
CrCl2pym 2d subtracted	39.42 (1.033) 4410 (24.99)	38.16 (1) 343.8 (1.948)	38.16 (1) 2301 (13.04)	38.16 (1) 279.5 (1.584)	46.7 (1.224) 176.5 (1)	52.95 (1.387) 683.3 (3.871)	127.4 (3.337) 592.2 (3.356)

Figure 3: Table showing accuracy and runtime for each minimizer (excluding global ones), per problem.

		WeightedNLLSCostFunc	
		gradient-free	
		ParticleSwarmOptimizer	SimulatedAnnealingOptimizer
CrCl2pym 2d cropped	8 params, 7829 points	2.102e+05 (1) 1008 (1)	2.169e+05 (1.032) 2420 (2.4)
CrCl2pym 2d large	8 params, 9497 points	7.342e+04 (1) 1419 (1)	1.048e+05 (1.428) 2482 (1.749)
CrCl2pym 2d subtracted	8 params, 9011 points	106.4 (1.25) 1757 (1)	85.18 (1) 2558 (1.455)

Figure 4: Table showing accuracy and runtime for gradient-free minimizers, per problem.

In terms of accuracy, the best performing algorithms are `lm-bumps`, `dfols`, and `leastsq`, which result in similar accuracy on all problems. However, while the two LM algorithms have comparable runtime, `dfols` is considerably slower. The difference in performance between `leastsq_step_size` and `leastsq` emphasizes the impact of the Jacobian step-size on the results. We further explore this in Section 4.1. Derivative-free minimizers like `simplex` and `amoeba` are slightly less accurate than the above mentioned algorithms, and require either comparable or longer runtimes. `ParticleSwarmOptimizer`, `SimulatedAnnealingOptimizer` and `Powell` have poor accuracy, with long runtimes across all problems.

Figure 5 shows the best 2D fit for each of the problems listed above. Comparing these pictures with Figure 2, it is evident that the fits for the cropped and subtracted data more closely resemble the reference fit. This is because the large dataset includes some phonon scattering at high energy transfer that is not part of the model. In the other two datasets the phonon scattering intensities have been removed by either cropping the data at high energies or by subtracting the Bose-corrected high temperature data, thus leaving only magnetic scattering which can be described using linear spin wave theory.

Figure 6 illustrates the fit to 1D cuts of the data for the best performing minimizer (in terms of accuracy), for both the large dataset (where no cropping has been applied) and the temperature subtracted one. Cuts were taken at the same locations as in the reference fit, i.e., Figure 1.

Table 5 shows the fitted parameters for each minimizer (for the temperature subtracted dataset) together with the expected parameter values. For this dataset, `lm-bumps`, `dfols` and `leastsq` are the minimizers that obtain fitted parameters closest to the expected ones. Global optimizers are those resulting in the worst fitted parameter values. Figure 7 further highlights this, showing that `leastsq` and `dfols` achieve comparable accuracy, but `dfols` have a considerably longer runtime. The `SimulatedAnnealingOptimizer` meanwhile, has comparable runtime to `dfols`, but produces a less accurate fit. Although `leastsq` is much faster than `dfols`, there are additional parameters to tune like the step-size of the Jacobian that can affect the fit.

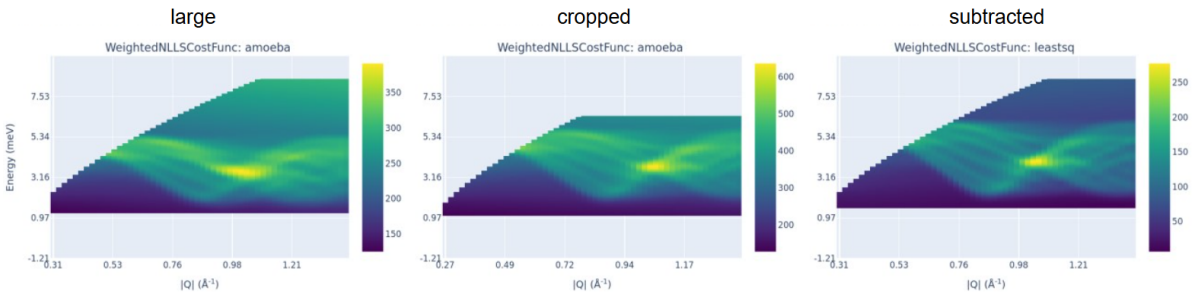


Figure 5: 2D plot of the best fit for each problem.

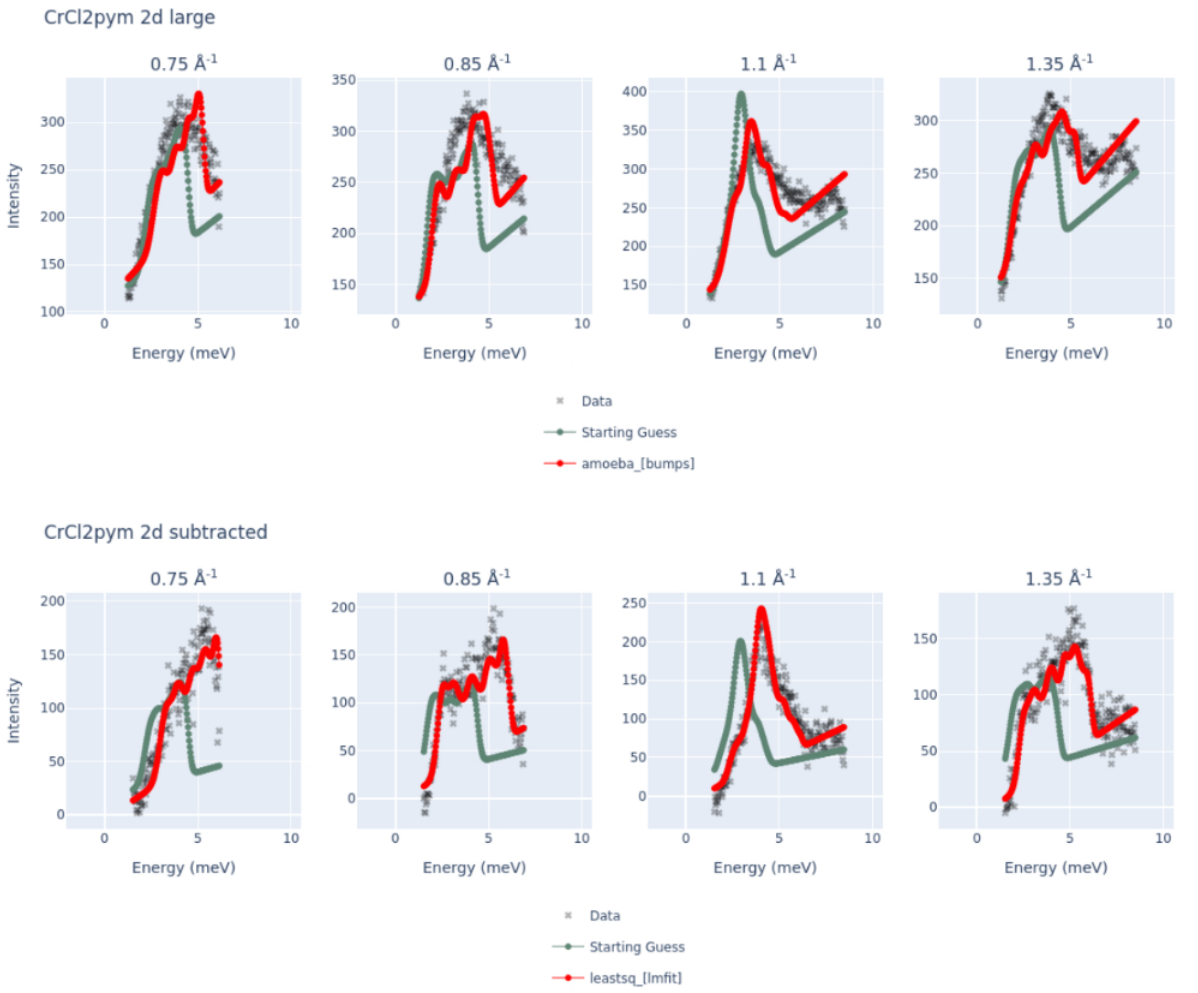


Figure 6: 1D cuts - best fit for each problem.

Parameter	amoeba	lm-bumps	dfols	leastsq	simplex	Powell	Particle Swarm	Simulated Annealing	expected
J_1	1.18	1.17	1.17	1.17	1.28	5.0	0.9	1.0	1.13
J_2	-0.00	-0.12	-0.12	-0.12	-0.07	0.39	-0.3	-0.10	-0.10
J_3	-0.12	0.00	0.00	0.00	0.00	-0.02	-0.1	-0.10	-0.01
D	-0.10	-0.11	-0.11	-0.11	-0.08	-0.9	-0.15	-0.08	-0.11

Table 5: Fitted parameter values for all minimizers (on the subtracted data) and expected value.

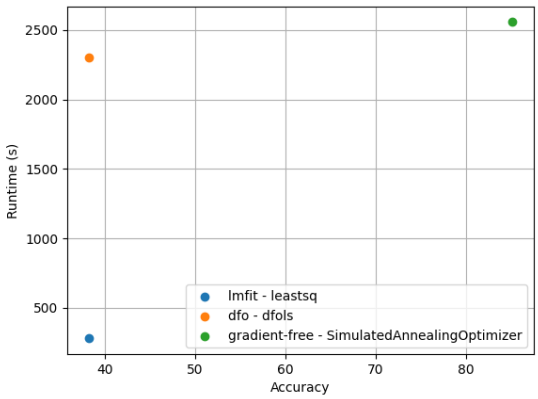


Figure 7: Scatter plot showing accuracy vs runtime for LMFIT `leastsq`, DFO `dfols`, and gradient-free-optimizer `SimulatedAnnealingOptimizer`, for the temperature subtracted dataset.

4.1 Jacobian Step-Size

For LMFIT's `leastsq`, changing the Jacobian step-size when running on `CrCl2pym 2d large` allowed us to obtain parameter values closer to the expected ones. FitBenchmarking typically uses the default Jacobian step-size for each particular minimizer, but this result indicates that tuning the step-size might sometimes be beneficial for improving the fit.

In Table 6, we report the fitted parameters obtained on the large, high temperature data, with both the default step-size (`leastsq` column) and the custom step-size (`leastsq_step_size` column). After testing with a few values, the custom step-size was set to 0.015, as this value lead to the best results. From a comparison of both these columns with the "expected" column, one can see that the parameters obtained with the custom step-size are closer to the expected ones. The difference in fitted parameter values could be caused by the existence of an additional local minimum, but more investigation would be required to confirm this. Figure 8 shows the 2D fit in the two cases, i.e., default step-size on the left and custom step-size on the right. Once again from a comparison of this 2D fit to the reference (Figure 2), it is evident that the fit obtained with the custom step-size resembles the reference 2D fit much more closely.

This emphasizes that the performance of LM methods can be heavily dependent on the Jacobian step-size, and for this reason derivative-free algorithms like `dfols` might be more robust for certain problems, if an optimal Jacobian step-size is unknown.

Parameter	leastsq	leastsq_step_size	expected
J_1	0.97	1.05	1.13
J_2	-0.10	-0.10	-0.10
J_3	0.06	0.02	-0.01
D	-0.14	-0.07	-0.11

Table 6: Fitted parameter values for LMFIT `leastsq`, obtained on the large (high temperature) dataset, both with default and custom Jacobian step-size.

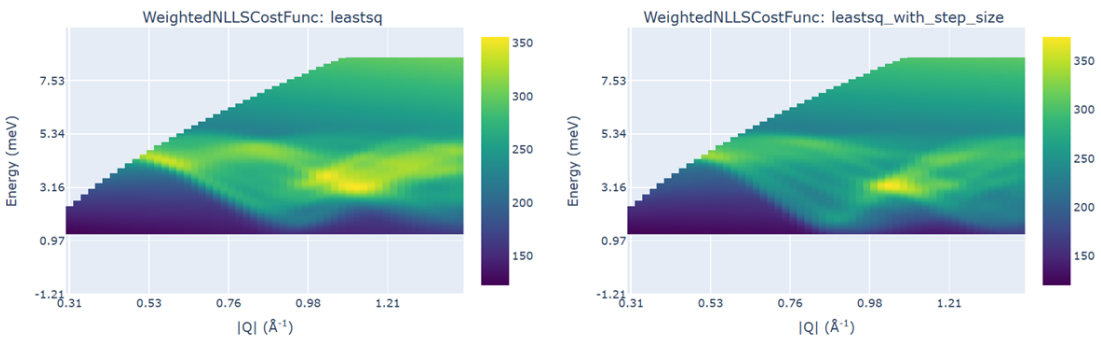


Figure 8: 2D fit obtained with LMFIT's `leastsq`, with default step-size (left) and with step-size 0.015 (right).

5 Conclusions

In conclusion, the results obtained have highlighted that all algorithm types perform better on the temperature subtracted and cropped datasets, which respectively isolate the magnetic scattering that can be modelled by SpinW, and remove phonon scattering intensity not included in the model. The benchmarking results also indicate that there is a trade off to be had between runtime of the optimization algorithm, and robustness. Whilst Levenberg-Marquardt algorithms and derivative-free methods (in particular `dfols` from DFO and `ameoba` from BUMPS) achieve almost identical accuracy and parameter values across all three datasets, the runtime of the derivative-free methods is significantly slower. However, the performance of LM algorithms does vary with Jacobian step-size, presenting a further parameter to be tuned, and so a derivative-free method may be favourable in some cases. Finally, the results have confirmed that for these problems the global optimization options that were tested perform poorly and should be avoided.

Future investigations should focus on fully understanding the effect of varying the Jacobian step-size for LM algorithms. It would also be beneficial to calculate the errors associated with parameter fits to ensure that results are physically feasible, and allow further analysis into which algorithms are most performant.

References

[1] Coralia Cartis et al. "Improving the Flexibility and Robustness of Model-based Derivative-free Optimization Solvers". In: *ACM Trans. Math. Softw.* 45.3 (Aug. 2019). ISSN: 0098-3500. DOI: 10.1145/3338517. URL: https://doi.org/10.1145/3338517.

[2] F. Guglielmetti, R. Fischer, and V. Dose. "Background–source separation in astronomical images with Bayesian probability theory – I. The method". In: *Monthly Notices of the Royal Astronomical Society* 396.1 (June 2009), pp. 165–190. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2009.14739.x. eprint: https://academic.oup.com/mnras/article-pdf/396/1/165/4069772/mnras0396-0165.pdf. URL: https://doi.org/10.1111/j.1365-2966.2009.14739.x.

[3] J. H. Hannay and J. F. Nye. "Polarization singularities in the diffraction of light". In: *Journal of Physics A: Mathematical and General* 37.48 (2004), pp. 11591–11602. DOI: 10.1088/0305-4470/37/48/004. URL: https://doi.org/10.1088/0305-4470/37/48/004.

[4] Matthew Hough and Lindon Roberts. "Model-Based Derivative-Free Methods for Convex-Constrained Optimization". In: *SIAM Journal on Optimization* 32.4 (2022). _eprint: https://doi.org/10.1137/21M1460971, pp. 2552–2579. DOI: 10.1137/21M1460971. URL: https://doi.org/10.1137/21M1460971.

[5] Philip J Humphrey, Wenhao Liu, and David A Buote. "$\chi 2$ and Poissonian data: biases even in the high-count regime and how to avoid them". In: *The Astrophysical Journal* 693.1 (2009), p. 822.

[6] F. James and M. Roos. "Minuit - a system for function minimization and analysis of the parameter errors and correlations". In: *Computer Physics Communications* 10.6 (1975), pp. 343–367. ISSN: 0010-4655. DOI: https://doi.org/10.1016/0010-4655(75)90039-9. URL: https://www.sciencedirect.com/science/article/pii/0010465575900399.

[7] Leccardi, A. and Molendi, S. "In search of an unbiased temperature estimator for statistically poor X-ray spectra". In: *AA* 472.1 (2007), pp. 21–27. DOI: 10.1051/0004-6361:20077290. URL: https://doi.org/10.1051/0004-6361:20077290.

[8] Brian Benjamin Maranville et al. *bumps/bumps: v1.0.0rc3*. Version v1.0.0rc3. June 2025. DOI: 10.5281/zenodo.15730428. URL: https://doi.org/10.5281/zenodo.15730428.

[9] Anders Markvardsen et al. "'FitBenchmarking': an open source 'Python' package comparing data fitting software". In: *Journal of Open Source Software* 6.62 (2021), p. 3127. DOI: 10.21105/joss.03127. URL: https://doi.org/10.21105/joss.03127.

[10] Matthew Newville et al. *LMFIT: Non-Linear Least-Squares Minimization and Curve-Fitting for Python*. Version 1.3.4. July 2025. DOI: 10.5281/zenodo.16175987. URL: https://doi.org/10.5281/zenodo.16175987.

[11] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN: 978-0-387-40065-5. URL: https://books.google.co.uk/books?id=VbHYoSyelFcC.

[12] Jem Pitcairn et al. "Low-Dimensional Metal–Organic Magnets as a Route toward the S = 2 Haldane Phase". In: *Journal of the American Chemical Society* 145.3 (2023), pp. 1783–1792. DOI: 10.1021/jacs.2c10916. eprint: https://doi.org/10.1021/jacs.2c10916. URL: https://doi.org/10.1021/jacs.2c10916.

[13] Tareq M. Shami et al. "Particle Swarm Optimization: A Comprehensive Survey". In: *IEEE Access* 10 (2022), pp. 10031–10061. DOI: 10.1109/ACCESS.2022.3142859.

[14] Simon Blanke. *Gradient-Free-Optimizers: Simple and reliable optimization with local, global, population-based and sequential techniques in numerical search spaces.* https://github.com/SimonBlanke. since 2020.

[15] W.J. Thompson. "Don't subtract the background [signal count analysis method]". In: *Computing in Science Engineering* 1.5 (1999), pp. 84–88. DOI: 10.1109/5992.790592.

[16] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[17] Simon Ward. *SpinW v4.0.0*. GitHub release at https://github.com/SpinW/spinw/releases/tag/v4.0.0. Retrieved December 15, 2025. 2025.