# Authors

# Affiliations

[1] ISIS Neutron and Muon source, Rutherford Appleton Laboratory, Harwell Campus, OX11 0QX

[2]

# Abstract

# Introduction

In this paper we propose a methodology for simulating an experiment from a given model and measurement angle. In practice, this methodology has low computational overhead and therefore has potential applications in generating large volumes of realistic noisy neutron reflectivity data such as in training neural networks.

# Methods

## Experiment Simulation

As input, a model is provided that defines the structure to probe with neutrons. This structure consists of the scattering length density (SLD), thickness and roughness for each layer. A number of these structures can be created for multiple contrast experiments. Using this structure and given resolution, scale and background (we used 2%, 1 and 1e-6 respectively), we can calculate model reflectivity using the Abeles matrix formalism.

The simulation utilises a two column ACSII CSV file which records incident neutrons as a function of wavelength for OFFSPEC; these values are not binned at this stage. We can multiply this incident neutron flux by a constant to effectively shorten the experimental counting time, and to give us the number of incident neutrons for a simulated experiment. Care is taken in adjusting the flux with given measurement angle ( in radians) to account for slits; flux increases as the square of the ratio of the measurement and slit angles (0.3° in this case).

Next, the momentum transfer, Q, is calculated for each wavelength point () in the file using given measurement angle () and formula

These Q values are binned in equally log-spaced bins, using flux as weighting, and with the number of bins being set to the desired number of points for the simulate dataset.

Following this, we iterate over each Q bin centre and for each bin, we calculate the model reflectivity scaled [0, 1] and add a background noise (we used a background rate of 5e-7). Next this reflectivity point is multiplied by the bin’s flux to get the neutron count for the bin. The error bar in count-space is the square root of this count. Using this count and error bar as the mean and standard deviation of a normal distribution respectively we obtain a noisy count. To obtain the noisy reflectivity and associated reflectivity error, we divide both the noisy count and its error bar by the bin’s flux.

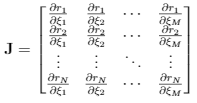
After applying these calculations to each Q bin, you are left with a range of Q values and for each value you have a noisy reflectivity, reflectivity error and neutron flux.

## Applying Fisher Information

We have used the fact that all of our errors are Poisson, and are treating every experimental data point independently as a way of analytically calculating the Fisher information, and therefore the variance in the model parameters. This approach intrinsically assumes that we have the correct model before fitting.

The Fisher information can be applied to both the simulated experimental data previously detailed and real data taken on an instrument. In both cases, a model must be provided with a given set of parameters to vary. Using refnx, we create an objective using the model and data provided. This objective can then be fit using a traditional fitting algorithm to find the optimal model parameter values, as judged by chi squared or similar. Such fitting algorithms include L-BFGS-B, differential evolution, MCMC and nested sampling. We used differential evolution as a global optimiser and then “polished” with the local optimiser L-BFGS-B. From here on, we no longer need the data, since the model and Poisson statistics describe the data well enough. Therefore from here all mention of data point refers to the model.

Next, we calculate the gradient of the model reflectivity (*r*) with every model parameter (*ξ*) for every model data point (e.g. every Q value which the data was taken for every condition). This is calculated using a two-point gradient with 0.5% step and forms our Jacobian:



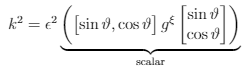
Using this Jacobian, and the diagonal matrix (**M**) of incident counts divided by the reflectivity (zeros for off-diagonals), we can calculate the Fisher information matrix (FIM):



This matrix contains all of the information about parameter variances and covariances but requiring extraction. For a single parameter’s error bar, ε, the Fisher information is just the multiplicative inverse of the variance, so we say

For multiple parameters, we take the square root of multiplicative inverse (reciprocal) of the diagonal elements of the Fisher information matrix.

If we want the covariance between two parameters then our equation is:



We can graphically solve (plot) this by inserting a number of theta values in and plotting the points

In summary, we will need to fit a dataset which is complexity dependant and then calculate the gradients of N (points) \* M (parameters), i.e. at least 2\*N\*M reflectivity points, but this is fast and fairly optimised; these calculations do not add any noticeable overhead to getting results in real time over the actual data fitting.

# Results and Discussion

## Comparing Traditional Fitting Methods

1. Provide a structure (such as one from the structures.py file) and simulate an experiment using the method above using a chosen angle, time and number of points.
2. Pass the resulting model(s) and dataset(s) to the Fitting class which will create a global objective that can be later fit.
3. Fit the objective using differential evolution / L-BFGS-B, MCMC or nested sampling. For MCMC, you can provide the number of samples for the initial burn-in and also the number of steps and thinning for the main sampling.
4. Each method will provide a fit of the data and an estimate of the parameter covariance matrix. MCMC and nested sampling will additionally provide a corner plot.

* Comparison of different fitting methods (L-BFGS-B/differential evolution, MCMC and nested sampling)
* Relationship of traditional fitting errors and Fisher errors with increasing time constant. The time dependence for any one error bar (and indeed the covariance) is determined by sqrt(measurement time)
* Corner plots and Fisher confidence ellipses.
* Comparison of multiplicative inverse of Fisher information (variance) with variance in parameter estimation using traditional fitting for many fits (1000+).
* Fisher information approach applied to two or more real datasets.

# Future Work

* Experiment design: using Fisher information gain to influence choice of measurement angle and/or contrast choice.
* Using the Fisher information to determine how parameter uncertainties will change with time and estimate how much longer data needs to be counted in order to reduce the uncertainty in given parameters to a desired level.

# Conclusions

# Acknowledgements

# References

## refnx

“Nelson, A.R.J. & Prescott, S.W. (2019). J. Appl. Cryst. 52, <https://doi.org/10.1107/S1600576718017296>.”

## dynesty

**Code and Methods:**

Speagle (2019): ui.adsabs.harvard.edu/abs/2019arXiv190402180S

**Nested Sampling:**

Skilling (2004): ui.adsabs.harvard.edu/abs/2004AIPC..735..395S

Skilling (2006): projecteuclid.org/euclid.ba/1340370944

**Bounding Method:**

Feroz, Hobson & Bridges (2009): ui.adsabs.harvard.edu/abs/2009MNRAS.398.1601F

## corner.py

https://dx.doi.org/10.21105/joss.00024

## matplotlib

J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

## NumPy

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2. (Publisher link).

## L-BFGS-B

To do