# 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.

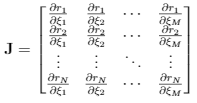
After applying these calculations to each Q bin, you are left with a range of Q values and for each value you have a noisy reflectivity, reflectivity error and neutron flux.

## 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:



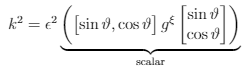
Using this Jacobian, and the diagonal matrix (**M**) of incident counts divided by the reflectivity (zeros for off-diagonals), we can calculate the Fisher information matrix (FIM):



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

## Experiment Simulation versus Real Data

To demonstrate the robustness of our experiment simulation, we compare two real datasets to their simulated counterparts. In both cases of simulation, the data used the same measurement angle, counting time and number of points as the real data was measured with and used the “ground truth” model. As can be seen from the following figures, the simulated and real datasets are very similar…

## Comparing Traditional Methods

Before demonstrating our new approach using the Fisher information, it is worth comparing traditional fitting methods. We used a selection of structures for this comparison each with a particular quality making it challenging to fit. For example, samples with many layers and thus many parameters to fit, samples with very thin layers or samples with very similar SLDs and therefore have strongly correlated parameters. Using these structures, we simulated experiments using a chosen angle, time and number of points. The resulting model(s) and dataset(s) were used with refnx to fit the simulated data using differential evolution / L-BFGS-B, Markov chain Monte Carlo (MCMC) sampling and nested sampling. For MCMC, the number of samples for the initial burn-in was set to 400, the number of steps was 30 and thinning 100.

Each method provides a fit of the data and an estimate of the parameter covariance matrix. In each case, we found the three methods produced very similar estimates for the covariance matrix. MCMC and nested sampling additionally provide a corner plot which is “an illustrative representation of different projections of samples in high dimensional spaces”. These corner plots closely relate to the Fisher information confidence ellipses….

The multiplicative inverse of Fisher information is the parameter variance. We investigated how this variance compares with the variance in parameter estimation obtained using a traditional fitting algorithm for many fits. We ran L-BFGS-B 1000 times...

## Time Dependence

We explored the relationship of traditional fitting errors and Fisher errors with increasing time. We found that the time dependence for any one error bar (and indeed the covariance) was determined by the square root of the measurement time…

To further demonstrate this time dependence, we applied the Fisher information approach to two real time-sliced datasets and found the relationship to hold…

# Future Work

The applied Fisher information approach presented in this paper has many potential applications in neutron reflectometry and other neutron/x-ray techniques. Experiment design is one such application where the Fisher information gain could be used to influence real-time decisions of measurement angle and/or contrast choice; an equivalent Bayesian model selection approach would be infeasible for real-time application due to computational overhead.

Another avenue for future work would be using the Fisher information to determine how parameter uncertainties 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. This would allow experiments to be stopped when they reach their uncertainty threshold, saving beam time; or to tell the experimenters to carry on counting, saving experimental datasets.

# 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