# Authors

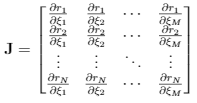
# Affiliations

# Abstract

# Introduction

# Methods

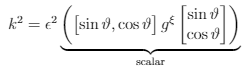
1. Our simulated datasets give us the model reflectivity (scaled [0,1] by definition)
2. We have a function which we use to add noise and a datafile which has a record of incident neutrons as a function of wavelength for OFFSPEC. We can use these to effectively shorten the experimental counting time, and to give us the number of incident neutrons for a simulated experiment.
3. 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.
4. The first step is to use the data and a traditional fitting algorithm to find the optimal model parameter values, as judged by some metric (chi squared or similar). This intrinsically assumes we have the right model (a point to work on later).
5. From here on, we no longer need the data, since the model + Poisson statistics describe the data well enough. Therefore from here all mention of data point refers to the model.
6. We calculate for every model data point (e.g. every Q value which the data was taken for every condition), the gradient of the reflectivity (*r*) with every model parameter (*ξ*). This forms our Jacobian:



1. Using this Jacobian, and the diagonal matrix of incident counts divided by the reflectivity (**M**) we can calculate the Fisher information matrix:



1. This matrix contains all of the information about parameter variances and covariances we just need to extract them.
2. For a single parameter’s error bar, ε, we know that the Fisher information is just the inverse of the variance, so we say ε = 1/gξ. For multiple parameters, this refers to the diagonal elements of the Fisher information matrix.
3. We can test this against other methods which are able to get the error bar for a parameter (and should!).
4. To calculate the parameter covariances is slightly trickier but still not so hard. We assume that 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 [0, 2π] and plotting the points (ε(θ)\*sinθ, ε(θ)\*cosθ).

1. This again should be cross checked against other methods.
2. The time dependence for any one error bar (and indeed the covariance) is determined by sqrt(measurement time), so by doing this once, we can know how the uncertainties will change, and thus give a time estimate for improvements in uncertainty.
3. We should test this using our noise function.

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

# Results and Discussion

# Future Work

# 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).