Contribution ID: **51** Type: **not specified**

ReMU – Data sharing in a forward-folding framework

The canonical way to make experimental neutrino-cross-section data available for comparison with different theories and other experiments is to unfold it.
The aim of this is to remove the detector effects and efficiencies from the data to release distributions of variables of interest in "truth space".
Depending on the available data, the detector properties, and the variables of interest, it is not always possible to do this though.
For example, the detector response might depend on variables that are not measurable by the detector.
If the distribution of events in those variables is not constrained by external measurements,
it is impossible to predict the actual detector response without depending on a theoretical model.
The unfolded results would thus become dependent on said model, introducing a theoretical bias in what is supposed to be an objective measurement.

In these (and other) cases it would be preferable to calculate the expected measured distributions for each model separately,
and compare those predictions to the real data in the smeared (i.e. reconstructed) space.
This should always be possible as the different models will have to predict some sort of distribution in all variables that could affect the detector,
even if those distributions are poorly motivated.
The act of bringing the model predictions in truth space to expected distributions of reconstructed (i.e. measured) quantities is called "forward-folding".

This is done on a regular basis within the experimental collaborations using detailed simulations of the detectors.
Events are generated according to a specific model and these events are then propagated through the detector one-by-one.
It requires a large amount of computing power and expert knowledge of the detectors.
Also, the simulation software is usually integrated closely in the software stack of the experiments.
All this makes it impractical or impossible to do a full detector simulation outside the collaborations.
To make the data usable by external interested parties, e.g. model builders or other experiments,
it would be necessary to provide a simpler way of forward-folding.

ReMU (Response Matrix Utilities) tries to solve this problem by modelling the detector with a response matrix that describes both the detector efficiency and smearing.
Expectation values in reconstructed space are generated by simply multiplying this matrix with the truth space distributions provided by the different models.
It offers all necessary machinery to build the response matrix from simulations and test arbitrary models against the data.
Both Frequentist and Bayesian methods for model testing and parameter estimation are available.
Systematic detector uncertainties, statistical uncertainties, and model nuisance parameters can all be handled with a few lines of code.

The main design goal of ReMU is ease of use.
Users are not required to be a Python, statistics, or detector experts to use a provided response matrix and compare their models to data.
It is implemented as a pure Python package and requires only standard scientific Python packages.

It supports Python 2.6, 2.7, and 3.x.
Official releases are distributed via the Python Package Index, so installing it is as easy as “pip install remu”.

Primary author: KOCH, Lukas (Science and Technology Facilities Council)

Presenter: KOCH, Lukas (Science and Technology Facilities Council)

Session Classification: Welcome Reception and poster session