Nested sampling for Fully Bayesian Unfolding (PENDING TITLE)

by

Jens Bratten Due

THESIS

for the degree of

MASTER OF SCIENCE

Faculty of Mathematics and Natural Sciences
University of Oslo

June 2021

Abstract

This here is the abstract

Acknowledgements

Contents

CHAPTER 1

Introduction

- Introduce the theme, FBU, OMpy/Folding iteration method, the spectra
- Talk about what I have done
- Describe the parts and chapters and what they will address

Part I

Theory

CHAPTER 2

Bayesian statistics

> Probability theory is nothing but common sense reduced to calculation.
>
> Pierre-Simon Laplace [1]

Sivia's book *Data Analysis - a Bayesian tutorial* [1] is a great read, and provides the theoretical foundation for the majority of topics discussed in this chapter.

1 Bayes' theorem

The realm of probability is commonly considered to be split into two camps of interpretation:

- The frequentist view, which describes probability as a number representing the frequency of which an outcome occurs, after performing an infinite amount of experiments. This view does not permit a guess about a probability before actually performing a given experiment and observing the result.
- The Bayesian view, which labels probability as a degree of belief, subjectivity included. Applying a Bayesian probability means to make a statement about what the outcome of an experiment will be, and how certain we are. We are able to use whatever *prior* knowledge and experience (or lack thereof) we possess to make this statement, as well as making any changes depending on the result. Frequentists can only base such a statement on the result of the experiment itself, infinitely repeated to be certain.

As one might guess from the titles, we will be assuming the second interpretation, the Bayesian view, and explore how it is used to describe the happenings of nature. First, we consider probability theory and its basic algebra which includes the sum rule

$$P(X|I) + P(\overline{X}|I) = 1 \tag{2.1}$$

and the product rule

$$P(X, Y|I) = P(X|Y, I) \times P(Y|I). \tag{2.2}$$

Here P stands for probability, the bar "|" means "given" and $\overline{X}$ means "not X". Lastly, we have the symbol I, meaning all relevant background information. The sum rule can then be stated as "the probability of X being true plus the probability of X not being true, both given all relevant background, equals 1".

Using the product rule, and the fact that $P(X, Y|I) = P(Y, X|I)$ we get the following.

$$P(X|Y, I) \times P(Y|I) = P(Y|X, I) \times P(X|I)$$

Rearranging this leads to *Bayes' theorem*

$$P(X|Y, I) = \frac{P(Y|X, I) \times P(X|I)}{P(Y|I)} \tag{2.3}$$

To get a clearer picture of the significance of Bayes' theorem, we can replace X and Y with *hypothesis* and *data*. $P(hypothesis|data, I)$ is then given the formal name *posterior probability*, $P(data|hypothesis, I)$ is called the *likelihood* and $P(hypothesis|I)$ is called the *prior probability*, representing our knowledge about the truth of the hypothesis before any data has been analysed. The term in the denominator, $P(data|I)$, often called the *evidence*, is in many cases not shown, due to it often being absorbed by a normalization constant. We can then replace the equality sign with a proportionality.

$$P(hypothesis|data, I) \propto P(data|hypothesis, I) \times P(hypothesis|I) \tag{2.4}$$

In summary, Bayes' theorem describes a learning process, showing how a probability should be augmented by the introduction of data.

Another useful result from using the sum and product rule is the *marginalization* equation

$$P(X|I) = \int_{-\infty}^{\infty} P(X, Y|I) dY \tag{2.5}$$

with a normalization condition

$$\int_{-\infty}^{\infty} P(Y|X,I)dY = 1. \tag{2.6}$$

The marginalization equation gives us the ability to integrate out so-called nuisance parameters, that is parameters of no interest to the question we are investigating, such as background signals and measurement byproducts. These rules of probability are generally applicable and provide a strong foundation for tackling data analysis problems. [1]

2 Parameter estimation

We will now look at the act of estimating a single parameter using Bayes' theorem, such as the mass of a planet, or the charge of the electron. We will firstly go through the example of deducting the fairness of a coin. This can be represented by the *bias-weighting* H. $H = 1/2$ will mean the coin is fair, while $H = 1$ and $H = 0$ means the coin is showing only heads or tails every flip. This value is continuous on the range $[0, 1]$, and $P(H|\{data\}, I)$ describes how much we believe H to be true. For a range of H-values, $P(H|\{data\}, I)$ is a *probability density function* (pdf). To find this, we use Bayes' theorem.

$$P(H|\{data\}, I) \propto P(\{data\}|H, I) \times P(H|I) \tag{2.7}$$

We can, if needed, find the normalization constant using equation (2.6). To express ultimate ignorance, we can assign a flat pdf for the prior.

$$P(H|I) = \begin{cases} 1 & 0 \leq H \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

meaning we assume every value of H to be equally probable. Assuming each flip is an independent event, the likelihood function takes the form of the binomial distribution.

$$P(\{data\}|H, I) \propto H^R(1 - H)^{N-R} \tag{2.9}$$

where R is the number of heads and N is the number of flips.

Plugging (8) and (9) into Bayes' theorem results in the posterior probability, the shape of which varies significantly for the first few data points. When the number of data increases however, the pdf becomes sharper and converges to the most likely value. The choice of prior becomes mostly irrelevant when we have a large of number of data, as the majority of propositions will converge to the same solution, but the speed of convergence may vary. A very confident, but wrong prior will often approach the correct solution more slowly than an ignorant one.

2.1 Reliabilities: best estimates, error-bars and credible intervals

One way to summarize the posterior pdf is with two quantities: the best estimate and its reliability. The best estimate is given by the maximum value of the pdf

$$\left.\frac{dP}{dX}\right|_{X_O} = 0 \tag{2.10}$$

where X_O denotes the best estimate. To make sure we have a maximum, we also need to check the second derivative

$$\left.\frac{d^2P}{dX^2}\right|_{X_O} < 0. \tag{2.11}$$

This is assuming X is continuous. If not, the best estimate will still be the value corresponding to the max of the pdf.

The reliability of the best estimate is found by considering the width of the pdf about X_O. We take the logarithm of the pdf as this varies more slowly with X, making it easier to work with.

$$P_{ln} = ln[P(X|\{data\}, I)]. \tag{2.12}$$

Doing a Taylor expansion about X_O and using the condition

$$\left.\frac{dP_{ln}}{dX}\right|_{X_O} = 0, \tag{2.13}$$

which is equivalent to (10), leads to

$$P(X|\{data\}, I) \approx A \exp\left[\frac{1}{2}\left.\frac{d^2P_{ln}}{dX^2}\right|_{X_O}(X - X_O)^2\right]. \tag{2.14}$$

Here, we only show the dominating quadratic term of the expansion, with A being a normalization constant. We have now approximated our pdf by the *normal distribution*, typically taking the form

$$P(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{2.15}$$

The parameter σ is called the *error-bar* and is defined as

$$\sigma = \left(-\left.\frac{d^2P_{ln}}{dX^2}\right|_{X_O}\right)^{-1/2}. \tag{2.16}$$

We then infer the quantity of interest by the following

$$X = X_O \pm \sigma. \tag{2.17}$$

By calculating the integral of the normal distribution in this range, we get a 67% chance that X lies within $X_O \pm \sigma$ and a 95% chance that it lies within $X_O \pm 2\sigma$.

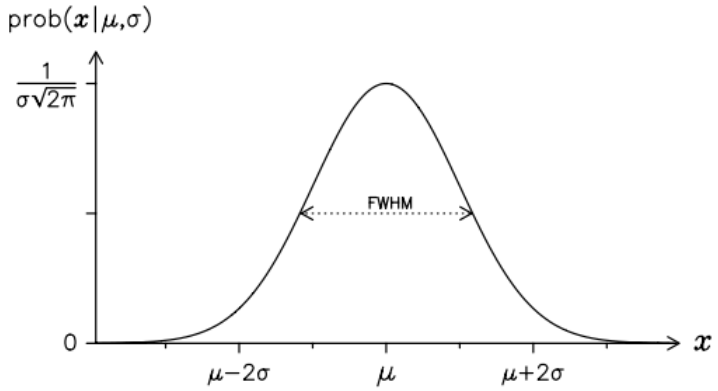


Figure 2.1: The normal distribution with a maximum at $x = \mu$ and a full width at half maximum (FWHM) of 2.35σ. (Sivia, 2006, p. 22) [1]

2.1.1 Asymmetric pdfs

The error-bar needs a symmetric pdf to be valid, something that is often not the case. This is solved by replacing the error-bar with a *credible interval* as a measure of reliability. It is defined as the shortest interval that encloses a given percentage of the total probability, conventionally set to 68% or 95%. In the case of 95%, we find X_1 and X_2 such that

$$P(X_1 \leq X \leq X_2|\{data\}, I) = \int_{X_1}^{X_2} P(X|\{data\}, I)dX \approx 0.95, \tag{2.18}$$

assuming the pdf is normalized.

In the case of an asymmetric pdf, we may consider using the *mean* or *expectation* as the best estimate. This quantity takes the skewness of the pdf into account, and is given by

$$\langle X \rangle = \int XP(X|\{data\}, I)dX. \tag{2.19}$$

If the pdf is not normalized, we also need to divide the right-hand side by $\int P(X|\{data\}, I)dX$.

If the pdf is *multimodal*, meaning it has multiple maxima, it becomes more difficult to calculate a best estimate and its reliability. If one maximum is much greater than the others, we can ignore those other contributions and focus on the largest. However, if multiple peaks are of similar size, we would be better off displaying the pdf itself.

Another common pdf is the Cauchy distribution, shown below.
This distribution has very wide wings and is in this case given by

$$P(x|\alpha, \beta, I) = \frac{\beta}{\pi[\beta^2 + (x - \alpha)^2]} \tag{2.20}$$

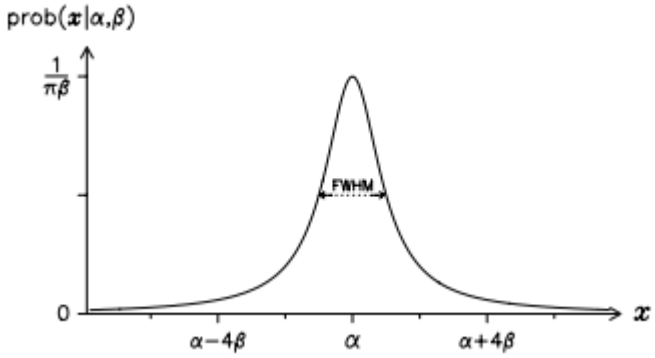


Figure 2.2: The Cauchy distribution, symmetric about $x = \alpha$ and has a FWHM of 2β. (Sivia, 2006, p. 31) [1]

3 Notation

The terms in eq. 2.3 are notationally very similar, forcing us to keep track of vertical bars and the order of parameters. Thus, for ease of reading, the following notation will be used for the Bayesian terms from now on:

- The posterior distribution: $P(H|D)$
- The likelihood: $L(H)$
- The prior distribution: $\pi(H)$

Bayes' theorem then takes the following form:

$$P(H|D) \propto L(H) \times \pi(H). \tag{2.21}$$

Part II

Methods

CHAPTER 3

Unfolding Methods

1 What is unfolding (maybe in Theory-chapter?)

- The inverse problem $f = \boldsymbol{R}u$
- The physics behind?
- The detector/experimental setup
- SiRi, OSCAr
- The raw data matrix, how it is produced?

1.1 The Response Matrix

The response matrix contains what we know about the circumstances of the experiment, a large part of which are the characteristics of the detector. The response matrix has elements given by:

$$R_{tr} = P(\text{reconstructed in bin r} \mid \text{true in bin t}) \tag{3.1}$$

[2]. This can be read as the probability of observing an event in energy bin r, given the true event happening in bin t. In a nutshell, the response matrix describes how a signal is smeared over the other bins in the spectrum.

- Multiplying with $\boldsymbol{R}^{-1}$ leads to fluctuations because we cannot assume the observed data equals the *expectation values* for the data. Statistical fluctuations in the data is assumed to come from a real structure in the true spectrum and will be magnified. http://www-library.desy.de/preparch/desy/proc/proc14-02/P52.pdf

2 The Folding Iteration Method

The following section describes the methods developed by Guttormsen et al. [3].

The folded spectrum f is on the form

$$f = \boldsymbol{R}u, \tag{3.2}$$

HAVE TO SWITCH AROUND TRUTH AND RECO BINS FOR THE ABOVE TO BE CORRECT, I BELIEVE where $\boldsymbol{R}$ is the response matrix and u is the expectation values for the true spectrum. The iterative method can then be described in 4 parts.

- First we use the observed spectrum $\boldsymbol{D}$ as an initial guess,
  $u_0 = \boldsymbol{D}$
- We then fold this with the response matrix,
  $f_0 = \boldsymbol{R}u_0$
- The difference between the folded and the raw spectrum is calculated and added to the initial guess, and we end up with the next trial spectrum,
  $u_1 = u_0 + (\boldsymbol{D} - f_0)$
- This is then repeated according to the following iteration scheme,
  $u_{i+1} = u_i + (\boldsymbol{D} - f_i)$

This method is performed until $f_i \approx \boldsymbol{D}$ within the experimental uncertainties [3]. It is important to note that for each new iteration, the oscillations between channels increase, as the solution approaches the inverted matrix solution $u = \boldsymbol{R}^{-1}\boldsymbol{D}$, which exhibits large oscillations. [2][3]

2.1 The Compton Subtraction Method

As the resulting spectrum from the folding iteration method often contains some degree of fluctuations, the Compton subtraction method is performed to obtain a significantly more stable spectrum.

The first step is to define a new spectrum $v(i)$ as the observed data excluding the Compton contribution:

$$v(i) = p_f(i)u(i) + w(i)\,, \tag{3.3}$$

where $u(i)$ is the spectrum obtained from the folding iteration method, which multiplied with p_f gives the full energy contribution. The remaining contributions are contained in $w(i) = u_s + u_d + u_a = p_s(i)u(i) + p_d(i)u(i) + \sum p_{511}(i)u(i)$, representing single escape, double escape and annihilation (note the missing Compton contribution "u_c"). To match the observed energy resolution, each contribution is then smoothed with a Gaussian function. Next, we subtract this from the raw spectrum to obtain the Compton background spectrum:

$$c(i) = r(i) - v(i)\,. \tag{3.4}$$

This spectrum may exhibit significant oscillations, and is thus further smoothed. This smoothing carries a low risk of loss of important information due to the nature of the spectrum not containing any sharp, narrow peaks. After this smoothing procedure on the individual contributions, we now "return" to the unfolded spectrum like so:

$$u(i) = \frac{r(i) - c(i) - w(i)}{p_f}\,. \tag{3.5}$$

Finally, to get closer to the true number of events, we correct for the total detector efficiency:

$$U(i) = \frac{u(i)}{\epsilon_{tot}(i)}\,. \tag{3.6}$$

This final spectrum shows higher stability compared to the result of the iteration method, while keeping similar statistical fluctuations to the raw spectrum. [3]

3 Fully Bayesian Unfolding

Choudalakis created the method of *Fully Bayesian unfolding* (FBU) by applying Bayesian thinking to the problem of unfolding. He states that the method provides the ability to observe all possible answers to a given unfolding problem via the posterior distribution, as opposed to other methods which result in point estimates of one of the possible answers through iteration. Below, we describe FBU and its components as developed by Choudalakis [4].

Bayes' theorem succinctly describes what we are asking for in the problem of unfolding, showing the relation between the expected truth spectrum $\boldsymbol{T}$, and the data we have obtained $\boldsymbol{D}$.

$$P(\boldsymbol{T}|\boldsymbol{D}) \propto L(\boldsymbol{T}) \cdot \pi(\boldsymbol{T}) \tag{3.7}$$

The expected truth spectrum $\boldsymbol{T}$ and the raw spectrum $\boldsymbol{D}$ are binned with N_t and N_r bins, respectively. In this thesis we operate with $N_t = N_r = N$ as we do not expect the number of energy bins to change during an experiment, but the mathematics do permit such a difference either way. Each bin in $\boldsymbol{T}$ is assigned a prior probability distribution $\pi(\boldsymbol{T})$, describing our belief of the number of events expected to be present. We assume the data follows a Poisson distribution, meaning

$$L(\boldsymbol{T}) = \prod_{r=1}^{N_r} \frac{f_r^{D_r}}{D_r!} e^{-f_r} \tag{3.8}$$

where

$$f_r = \sum_{t=1}^{N_t} T_t \cdot R_{tr}. \tag{3.9}$$

Here, R_{tr} is the element of the response matrix $R_{N_t \times N_r}$, corresponding to the probability that an event produced in the truth bin t is observed in the reconstructed bin r: $P(r|t)$. If we wish to include the background spectrum, all we have to do is add it to the sum:

$$f_r = B_r + \sum_{t=1}^{N_t} T_t \cdot R_{tr}. \tag{3.10}$$

The next step is to employ a sampling scheme of the parameter space, usually a MCMC algorithm, to calculate $L(\boldsymbol{T}) \cdot \pi(\boldsymbol{T})$ and arrive at a posterior distribution per bin in the expected truth spectrum.

3.1 Priors

As mentioned above, each bin in the spectrum is assigned a prior distribution. This means we are choosing the exact range and weighting of values we believe to be possible for that bin, independent of other bins. In fact, since the prior has to equal 0 outside its defined range, we say that it is impossible for there to be values beyond the boundaries. Since we are dealing with physical experiments, these boundaries need to be finite, and thus we are forced to restrict the realm of possibilities to whatever we deem reasonable. There is practically an infinite amount of choices one can make for assigning a prior, depending on what knowledge one has beforehand. We will now take a look at two possible prior distributions.

3.1.1 Uniform prior

If one wishes to make the least amount of assumptions about the truth, a *uniform* prior is suitable. The pdf of the uniform distribution is given as:

$$f(x;a,b) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

Ref? This flat distribution assigns equal probability to every outcome in the space of possibilities. The only assumption to be made here is determining the boundaries on this space. Complete ignorance would strictly be represented with a uniform prior without any boundaries. This would mean we believe all numbers on the interval $[-\infty, \infty]$ to be equally likely in a one-dimensional space. Such a space is of course not possible to explore completely, and otherwise extremely large limits will be computationally unfeasible. This is especially true considering the fact that many problems are complex and demand multidimensional parameter spaces. One thing to note is that unfolding in physics is often related to physical experiments pertaining to the counting of a number of events measured by a detector. In these cases, the existence of negative counts is unphysical, meaning a lower prior limit can safely be set to 0 (Discuss possibility for negative counts?).

Choosing the upper limit is not as straightforward. The ideal choice would be the largest possible limit that still allows for reasonable computational performance. Of course, if we have some knowledge about the size and location of the domain of the possible truth-values, there is no need to pick a limit located significantly beyond this domain. Computational resources are wasted if spent on exploring a region we strongly believe (know?) will not improve our estimate. A good check is to fold the upper prior limit with the response and make sure the raw data is contained within.

In this thesis, an upper prior limit of 10 times the raw data will be used for the uniform prior. In other words, we believe that the true value must be contained within an area with size and limits dependent on the observed value.

- Image of uniform distribution

3.1.2 Log-uniform prior

Another prior distribution we will use is the *log-uniform* distribution, also called the reciprocal distribution ref?. This distribution has the characteristic that its logarithm is uniformly distributed. What this means for our prior belief is that each order of magnitude

is given equal probability. In the case of a logarithm with base 10, we say that it is equally probable for our value of interest to lie between limits $a = 10^0$ and $b = 10^1$, as between $a = 10^6$ and $b = 10^7$, even though the second range is much larger. The pdf of the log-uniform distribution is defined as:

$$f(x; a, b) = \frac{1}{x \ln(b/a)} \quad \text{for } a \leq x \leq b \text{ and } a > 0. \tag{3.12}$$

Ref?

- Image of logarithmic distribution

Using the log-uniform distribution allows us to define a very large range of possible truth-values while keeping a high probability for values close to 0. The amount of counts per experiment is finite, and in cases with significant differences between peaks and valleys, we do not want to 'dampen' these by probabilistic distribution of counts into bins which should be containing none.

Say we know there exists one or multiple peaks in our truth-spectrum consisting of a very large amount of counts, i.e. $\sim 10^{10}$. If we also know that other bins in the spectrum should have close to 0 counts, how do we make sure both of these conditions are met? If we were to use a uniform prior between 0 and 10^{10}, we would firstly have an incredibly large space to explore, with 10^{10} possible values for each bin in the spectrum. Secondly, the probability of sampling a value close to 0 would be very small. Let's say that any value between 0 and 10000 is considered 'close' to 0, which itself seems very imprecise. According to the **uniform** prior, the probability of the true value being 'close' to 0 is thus:

$$P(0 \leq T_t \leq 10^4) = \int_0^{10^4} \frac{1}{10^{10} - 0} dT_t = \frac{10^4}{10^{10}} = 10^{-6} = 0.0001\%, \tag{3.13}$$

and the same result holds for values equally 'close' to the maximum of 10^{10}. This low probability means we might run out of computational resources long before our algorithm gets to explore those areas.

Now, if we instead use the **log-uniform** distribution as our prior, the probability of a value close to 0 is much higher, while still allowing for those tall peaks.

$$P(1 \leq T_t \leq 10^4) = \int_1^{10^4} \frac{1}{T_t \ln(10^{10}/1)} dT_t = 0.4 = 40\%. \tag{3.14}$$

Note here that we integrate from a count value of 1 instead of 0, as the distribution is undefined at $T_t = 0$. Since the logarithm of this function is uniformly distributed, looking

at the exponents will give a good indication of the probability value. In the simple case above, the exponents are 4 and 10, $4/10 = 0.4 = 40\%$. Similarly, we can estimate the probability of the area between 10^6 and 10^{10} to be $(10-6)/10 = 0.4 = 40\%$, the same result as in eq. (3.14), even though this range is much larger. The estimation turns out to be correct when we perform the proper calculation:

$$P(10^6 \leq T_t \leq 10^{10}) = \int_{10^6}^{10^{10}} \frac{1}{T_t \ln(10^{10}/1)} dT_t = 0.4 = 40\% \tag{3.15}$$

In summary, when we believe the truth spectrum to exhibit a very large difference between minima and maxima, the log-uniform distribution is a good candidate for the prior. This way, we increase the chance of reaching the proper relation between peaks and valleys, at the price of lower precision for higher values. The high probability for values close to 0 will also allow for less distribution of counts into bins where there should be almost none, due to the nature of probability. Should the true spectrum instead be composed of peaks with similar magnitudes and smaller differences, a uniform prior may be better suited.

For the cases examined in this thesis, similar limits to the uniform distribution has been chosen as the range has been sufficiently wide. Expanded limits have been found not to make a significant impact on the unfolded results, but different spectra may warrant such changes. The lower limit cannot be set to zero for which the distribution is not defined, thus we set it to 10^{-1} to be close enough. The upper prior limit is set to the following:

$$\text{upper} = \max(10 \cdot \text{raw}, 100 \cdot \text{lower} = 100 \cdot 10^{-1} = 10). \tag{3.16}$$

This makes sure the upper limit is never smaller than the lower in cases with low raw values. The implementation of the prior distributions is discussed in part III.

3.2 Multiplying response with (total) efficiencies

Cite Vala for this?

3.3 Likelihood

As mentioned above, the likelihood used in FBU is given by the Poisson distribution, given by eq. (3.8). It is important to note that the likelihood is a function rather than a pdf, meaning it does not necessarily integrate to 1.

When we assign the prior probability for our problem, we do so on a per-bin basis, meaning we end up with a set of N independent distributions, represented by histograms,

each describing the probability of possible truth-values for one bin. The same applies to the posterior probability, the only difference being the histograms having different shapes, due to the fact that we have been provided new knowledge from the data. This reshaping stems from the multiplication of the prior with the likelihood. One might then be tempted to construct a 1-dimensional Poisson distribution for a given bin, multiply with the prior and call this product the posterior. This is incorrect due to two reasons:

- Firstly, one must remember that the posterior is only proportional to the product likelihood$\times$prior, meaning it may need normalization to be a proper pdf and integrate to 1.
- Secondly, and most importantly, the likelihood cannot be assigned on a per-bin basis like the prior, as it does not consist of N independent distributions of which we can aggregate.

refer to plots of the wrong assumption of 1-dim likelihood? The likelihood is an N-dimensional function dependent on the total collection of data as well as the entire response matrix. For a given bin, a 1-dimensional Poisson distribution based on that data does not equal the contribution from the actual likelihood in that bin, unless our spectrum consists of only one bin. This also means we have no easy way of plotting the likelihood, should we wish to compare with the prior and posterior in a model test, unless we restrict the spectrum to contain a maximum of $N = 3$ bins and plot the complete multidimensional function. Most experiments are conducted with many more bins than this, however there is still some value to be had from performing such a visualization. Mainly, this will help us achieve an increased understanding of the process of Fully Bayesian Unfolding, its components, as well as the inner workings of the PyFBU-package. Due to its (lack of documentation? and) several layers of abstraction, both in itself and through PyMC3 and Theano, the PARTS are not immediately apparent. The symbolic variables and objects, while computationally efficient, do not allow for simple printing or plotting during intermediate steps. Understanding why the results appear as they do is therefore not straightforward, but we are able to use what we know about the prior and likelihood. The prior is defined by the user, but the likelihood is not. In fact, there is no simple multiplication of prior and likelihood performed in the source code of PyFBU at all. This is due to the way Bayes' theorem is being implemented. In the analythical formula, eq. (3.7), we see the posterior as a rescaled version of the prior, through multiplication with the likelihood. In the code, there is instead of this product, a definition of the space for which a sampling algorithm explores. By PyMC3 convention, this space is defined as the likelihood evalu-

ated on the prior, i.e. plugging the prior-values in for T_t in eq. (3.9). This is shown by creating an object of class 'Poisson' from the PyMC3 package, with the prior (folded with the response) and data as arguments. The class refers to one of the built-in distribution classes found in PyMC3. To make sure we are correct about this indeed representing the likelihood, we will want to compare the resulting posterior with a Poisson distribution we construct ourselves, multiplied with the prior. If the posterior has the same shape and location as this product, we have verified our knowledge of the likelihood and its parameters. As mentioned above, we will have to plot the entire multidimensional Poisson distribution to show a correct picture. Therefore, a 2-bin constructed spectrum will be used for this purpose. This implementation will be discussed in part III.

- Modified Poisson to take into account the total amount of counts

With a greater understanding of what happens under the hood, we can experiment with modification of the likelihood. We believe the Poisson distribution to be a good representation of the data, and will not switch it out completely. Such a switch is possible through PyFBU and PyMC3, but we will only perform a modification of the standard Poisson.

3.4 Sampling

- NUTS

There are several sampling methods possible for the problem of unfolding, a common example being Markov Chain Monte Carlo (MCMC) algorithms such as the Metropolis-Hastings algorithm. In the PyFBU-package, a variant of a Hamiltonian Monte Carlo (HMC) Markov Chain Monte Carlo algorithm is the default sampler. HMC aims to be much more efficient than regular MCMC algorithms by avoiding both sensitivity to correlated parameters and random walk tendencies [5]. A drawback to this is a significant sensitivity to step size as well as the number of steps, requiring manual tuning of these parameters. To circumvent this, Hoffman and Gelman created the No U-turn Sampler (NUTS), a variant of HMC which removes having to specify the number of steps. They also implemented an adaptive step size, meaning no manual tuning is necessary for running NUTS. Furthermore, they observed similar to better performance than other fine-tuned HMC algorithms [5]. The NUTS algorithm is implemented in the PyMC3 package [6] and is the default sampling algorithm in PyFBU.

- Initializing?

PyMC3 has several methods for initializing NUTS, the default being 'jitter+adapt_diag' CONTINUE Another sampling method which will be used in this thesis is called Automatic Differentiation Variational Inference (ADVI) [7].

It is included in PyMC3 as a possible choice for the **initialization** of NUTS. In some cases, the use of this initialization will help when FBU would otherwise crash. Maybe due to the 'jitter' part of 'auto'?

3.5 Posterior inference

Now that the unfolding has been performed, how do we interpret the resulting posterior distribution? While other methods may only return a point value, not necessarily accompanied by the uncertainties, FBU allows us to directly look at the final distribution per bin, and thus observe the result and its corresponding degree of belief. Of course, we are able to quantify these concepts in multiple ways. Here, we take a look at some of the methods of posterior inference.

3.5.1 Point estimates

Since we are dealing with 1-dimensional raw spectra visualized as plots with counts on the y-axis, and energy (bins) on the x-axis, it is desirable to represent the unfolded result the same way. The final output from FBU is a list containing N sets of posterior samples, allowing us to create N histograms representing the respective posterior distribution in each bin. One can then simply stack these histograms to form a band through the entire energy range, where higher probabilities can be shown with higher color intensities. However, it is customary to operate with point values for the unfolded spectrum when performing further analysis, like the results OMpy supplies. Point estimates will also allow us to directly compare performance with the folding iteration method, of which point values are the only output. Finally, error metrics such as the R^2-score are evaluated on point values, and allows for a quantitative measure of performance. Since we have N posterior distributions, we create N point estimates which we aggregate and represent as the unfolded spectrum. We can use this to directly compare the **folded** representation with the original raw spectrum, as well as the regular representation with the true spectrum, should we possess it.

It is important to remember that a point estimate does not summarize an entire distribution, and may in many cases paint a wrong picture. In these cases, the fact that we can

access and look at the complete posterior at any time may be the greatest advantage of using FBU.

We will consider two different point estimates, the posterior mean, median and mode:

- Posterior mean: The mean of the posterior distribution which minimizes the mean squared error (MSE) [8].
- Posterior median: The median of the posterior distribution which minimizes the expected absolute error [8].
- Posterior mode: drop this? The mode of the posterior distribution, also called the Maximum A Posteriori (MAP), which represents the most likely value for the parameter in question. This does not take into account any skewness of the posterior nor the existence of multiple modes of similar magnitudes.

3.5.2 Credible intervals

The credible interval is the Bayesian version of the frequentist confidence interval. It depends on the posterior and is defined as any interval that encompasses a certain percent of the posterior density. The difference between confidence and credible intervals is subtle, but not negligible. In the case of frequentist inference, the parameter in question, lets say θ, is treated as an unknown, but fixed value. The limits of the confidence interval are treated as random variables. Therefore, a confidence level of 95% means that for 100 repeated experiments, 95 of the confidence intervals will contain θ. Note that this does not mean there is a 0.95 probability of finding θ in every confidence interval. [9][10]

For bayesian inference however, the random-trait is switched, with the credible interval limits being fixed, and θ treated as the random variable. The credible interval takes our prior belief into account, while the confidence interval relies only upon the data. A 95% credible interval covers 95% of the posterior and can then be said to contain θ with a probability of 0.95. [9][10]

There are many types of credible intervals, the only requirement being that it covers a certain amount of area of the posterior. Some examples of ways of constructing credible intervals are:

- Using the posterior mean as the interval center.
- Making sure the probability of being outside the interval is equal on all sides (equal tailed).

- Making the interval as narrow as possible, the Highest Posterior Density interval (HPD). This will include the most likely values, as well as the mode of the posterior if it is unimodal.

We will be using the HPD interval which, together with the point estimates mentioned above, will give a solid estimate of the true energies of the γ-ray spectrum.

3.5.3 Variance

Each bin in the truth space are assigned a set of samples approaching the posterior distribution for that bin. The variance is a measur mean variance?

4 Error metrics

Now that we have a candidate unfolded spectrum consisting of expectation values for the counts in each bin, we need to assess the accuracy. If we should possess the true spectrum, we can directly compare our unfolded result with this. Usually however, we do not have the true spectrum, only the observed data. The solution is to refold our result with the response matrix, generating a candidate for the observed spectrum. Doing this assumes the response matrix perfectly represents the detector attributes, which will likely lead to some errors. Keeping this in mind, we use the following error metrics for comparing our model with the observed data.

4.1 Mean absolute error (MAE)

Given an estimated spectrum $\tilde{y}$ and an observed spectrum y, the mean absolute error is given by:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y}_i| \tag{3.17}$$

[11]. In our context, $\tilde{y}$ is the unfolded result refolded with the response matrix to estimate the observed data y. MAE is a simple measure of average error, allowing us to see how many counts we expect to be off the mark per bin. The lower this value, the better model, with 0 signifying a perfect match. Note that this does not take scale into account, as it does not measure relative difference. A model may output a value of $\tilde{y}_i = 90$ where the observed value is $y_i = 100$, while also outputting $\tilde{y}_i = 900$ for an observed value $y_i = 1000$. While the relative differences are the same, the MAE of the second case is 10

times larger than the first, falsely pointing to a worse accuracy. Therefore, we must take care to not blindly trust the MAE when comparing model accuracy on different scales. The MAE remains a good metric as long as we make sure to examine it within the context of scale.

4.2 R^2-score

Another metric of how well our estimated spectrum fits with the observed data is the R^2-*score*, also called the coefficient of determination. It is given by:

$$R^2(y, \tilde{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2}, \tag{3.18}$$

[12] where $\overline{y}$ is the mean of the observed spectrum. The R^2-score is a measure of the overall similarity between the estimate and the observed, a perfect match resulting in a score of 1. The R^2-score is scale invariant, meaning the mentioned limitation of MAE does not apply. This allows for direct comparison of model accuracy on different data sets, without the need to worry about scale differences. What constitutes as a 'good' R^2-score depends on the case and data, but in general the closer to 1, the better.

4.3 Residual plots

The MAE and R^2-score are summarizing metrics, boiling all errors down to a single number. They do not describe the actual error for each bin in the spectrum, nor if the total error is spread out over many bins, or focused around a few. To directly observe the error for each bin, we plot the residuals, which is simply given by the difference $y_i - \tilde{y}_i$. This way, we can determine the per-bin model accuracy and assess if there are any local dependencies affecting the total result.

4.4 Uncertainty plot

5 1-dimensional test spectrum

6 2-dimensional test spectrum

- Compare posterior with likelihood*prior

7 28 Si spectrum

- Compare result with Valas, using logscale prior, modified likelihood?
- Background?

Valsdóttir found that, when the background was known, including it in the unfolding significantly improved the results.

8 146 Nd spectrum

- 250 channels
- Background, nanoseconds
- 500 channels
- Background? nanoseconds?
- 453 keV

  Discrepancy at lower energies, unavoidable with this response, we have information to correct this. Maybe on the other states too? Results should match OMPy and thats what we want?
- 1-1.4 MeV
- 6-6.2 MeV

9 Model testing

- Cannot construct 1D likelihood Poisson distribution around data in bin X due to cross-bin dependencies.
- 2D example with 2D likelihood plots.
- Prior must be large enough such as the folded prior includes the observed counts.
- Can compare 1D prior and posterior.
- Can compare 2D likelihood and posterior if we only have 2 bins.

9.1 Synthetic data

For easier and more predictable testing, a synthetic data set is used for the raw spectrum. It consists of 3 excited states, a simplified representation of a physical case.

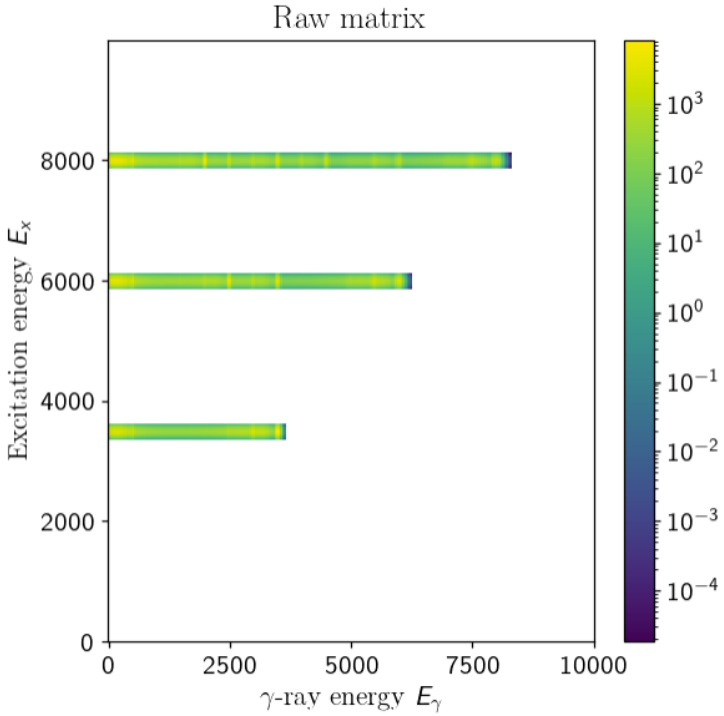


Figure 3.1: Synthetic raw $E_\gamma - E_x$ matrix.

9.2 Bayesian terms

An interesting aspect to look at is the terms in Bayes' theorem after unfolding has been performed. We should be able to reproduce the shape of the resulting posterior samples by multiplying the prior and likelihood. As the likelihood depends on f_r, rather than the truth-values t, direct comparison between this and the other terms is not easy at first glance. To arrive at comparative foundation, we perform the following modifications to the prior and the posterior to achieve f_r-dependence:

$$P(\boldsymbol{T}|\boldsymbol{D})_{f_r} = \boldsymbol{TR}, \tag{3.19}$$

$$P(\boldsymbol{T})_{f_r} = \boldsymbol{SR}, \tag{3.20}$$

where $\boldsymbol{S}$ is a matrix containing random samples between the upper and lower prior limits, with shape corresponding to the output $\boldsymbol{T}$. There is no easy way of extracting the likelihood from the PyFBU package, nor its corresponding f_r-values. However, as we know

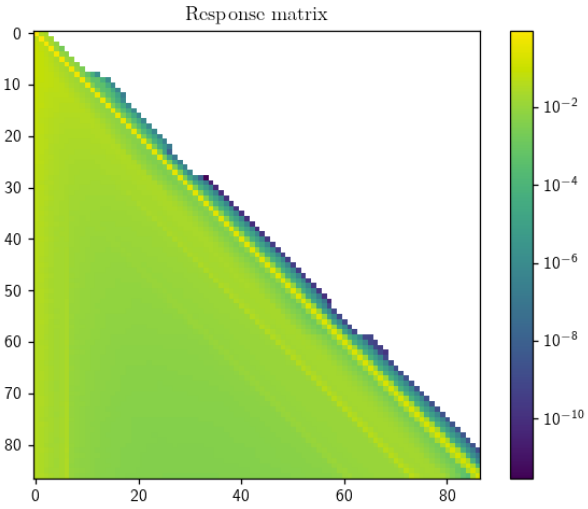
that it is a Poisson distribution, we are able to define a range of f_r-values determined by the prior and posterior, and use these as input to equation (3.8).

9.3 Response matrices

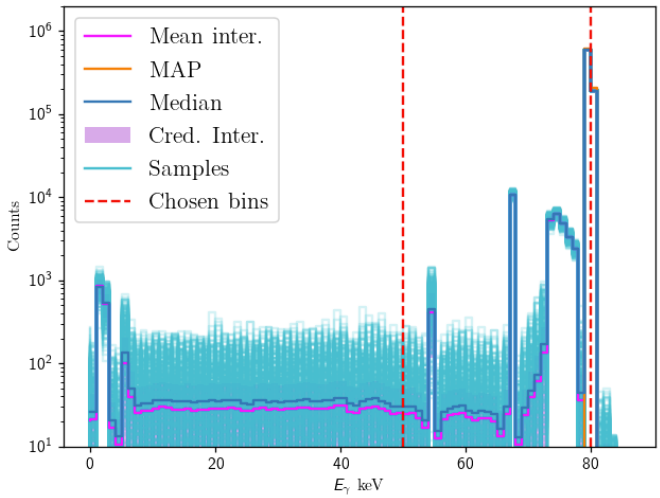
Here, different trial response matrices are tested for examining the impact on the final result, as well as the convergence of the implemented FBU method. As the response matrix is a vital part of the procedure, significant differences are expected when changes are made. In the end, the experimentally determined response should provide the best unfolded spectrum. It is however interesting to see the effect on the resulting Bayesian terms.

9.3.1 Normalized response from OCL

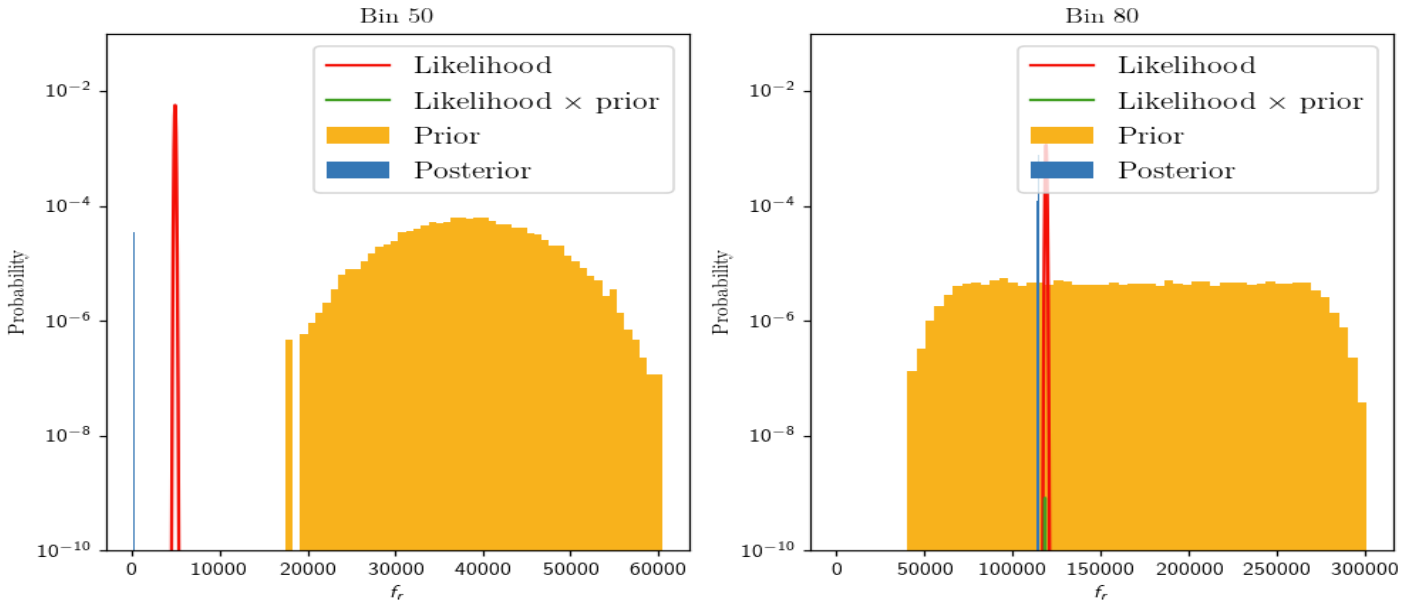
The following figures are results from running FBU using the response matrix from OCL, with normalization performed on each row.



(a) Response matrix



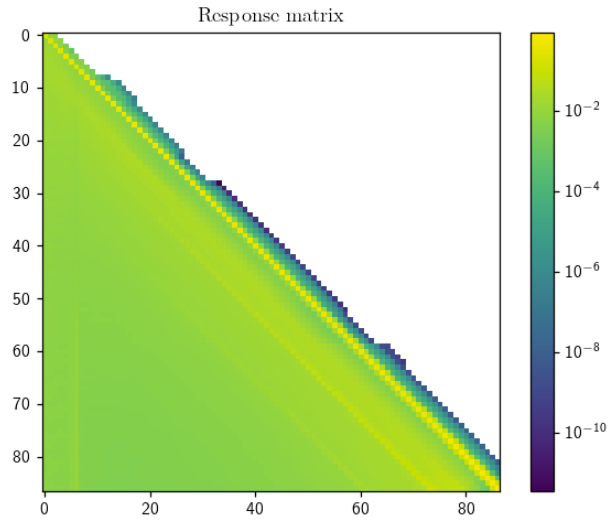
(b) Result of unfolding. Dashed lines show the bins chosen for the analysis below.



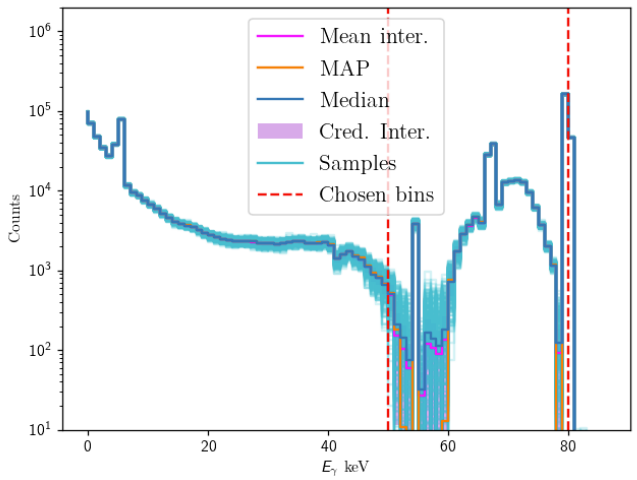
(c) The components of Bayes' theorem after unfolding, for the bins chosen above. Included is also a $L(\boldsymbol{D}|\boldsymbol{T}) \times P(\boldsymbol{T})$-function, which should have the same shape and position as the posterior, only differing by a normalisation constant.

9.3.2 Response from OCL with normalized columns

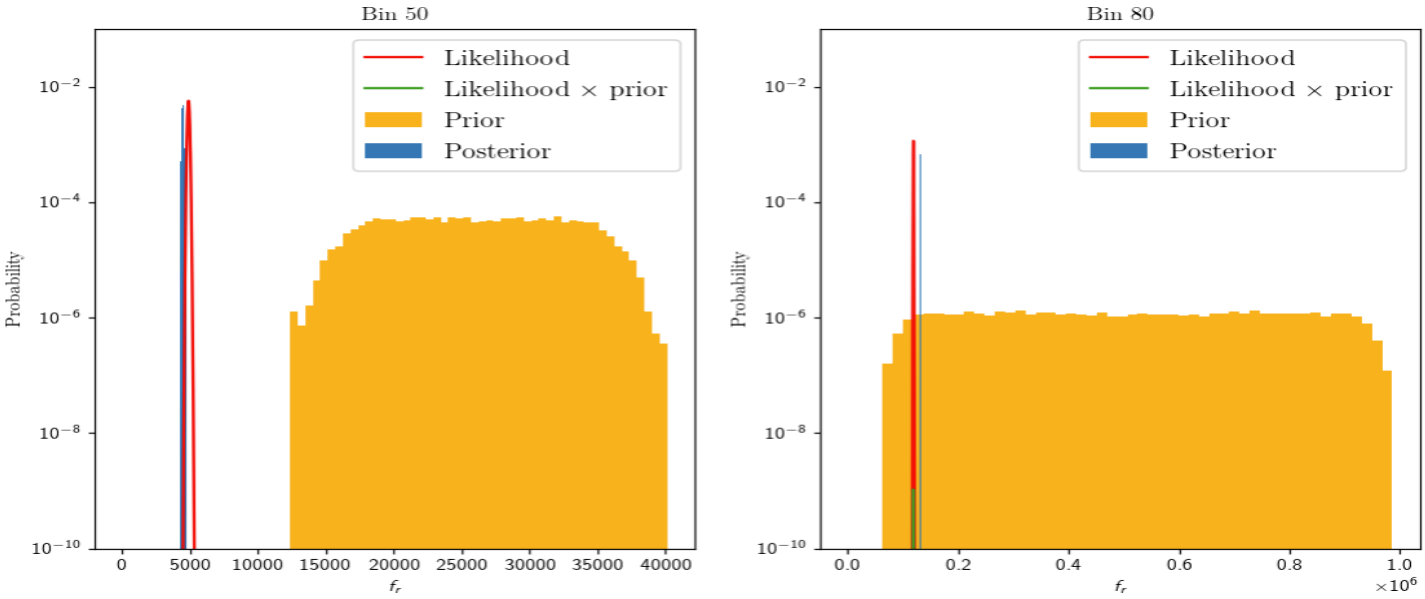
The following figures are results from running FBU using the response matrix from OCL, with normalization performed on each column instead.



(a) Response matrix



(b) Result of unfolding. Dashed lines show the bins chosen for the analysis below.

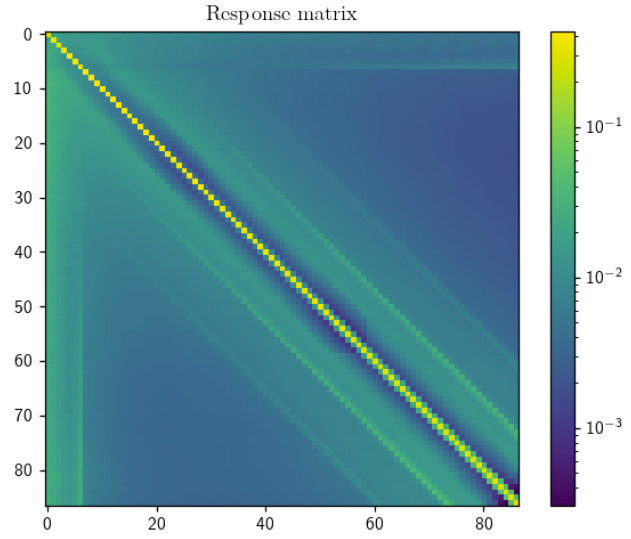


(c) The components of Bayes' theorem after unfolding, for the bins chosen above. Included is also a $L(\boldsymbol{D}|\boldsymbol{T}) \times P(\boldsymbol{T})$-function, which should have the same shape and position as the posterior, only differing by a normalisation constant.

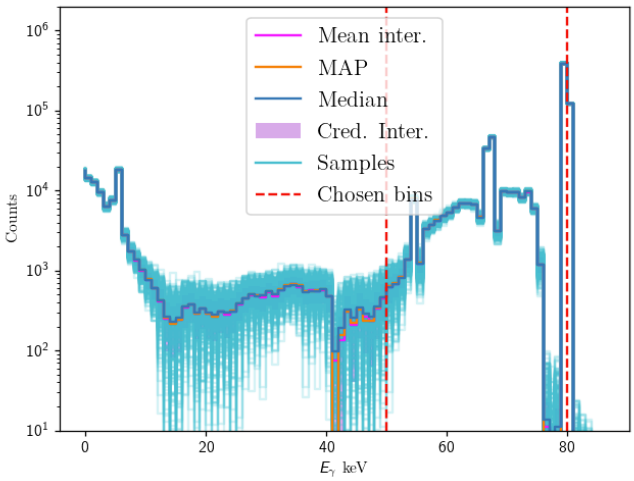
9.3.3 Symmetrized response from OCL

The following is produced using a symmetrized version of the OCL response matrix, that is

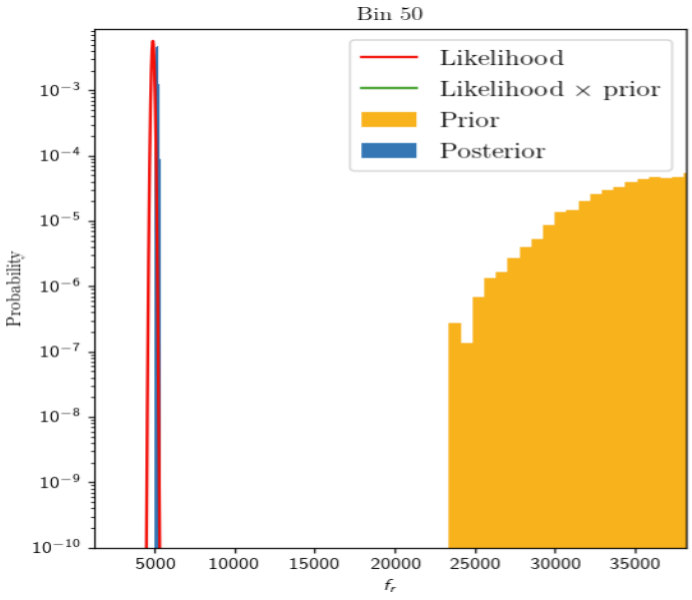
$$R = \frac{R_{OCL} + R_{OCL}^T}{2} \tag{3.21}$$

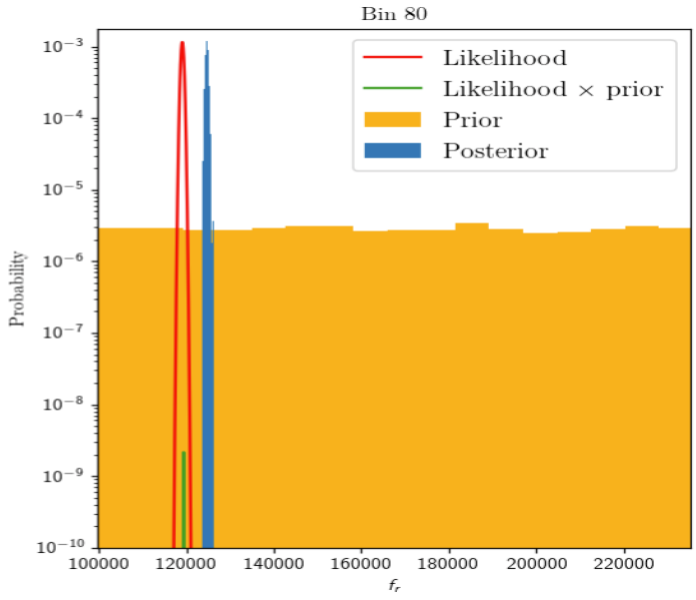


(a) Response matrix



(b) Result of unfolding. Dashed lines show the bins chosen for the analysis below.



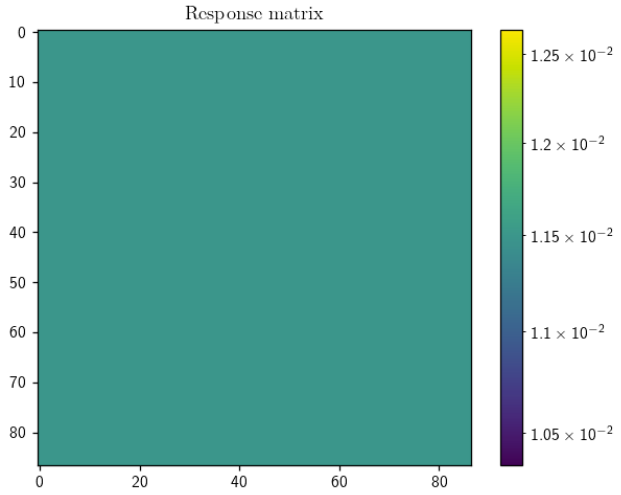


(c) The components of Bayes' theorem after unfolding, for the bins chosen above. Included is also a $L(\boldsymbol{D}|\boldsymbol{T}) \times P(\boldsymbol{T})$-function, which should have the same shape and position as the posterior, only differing by a normalisation constant.

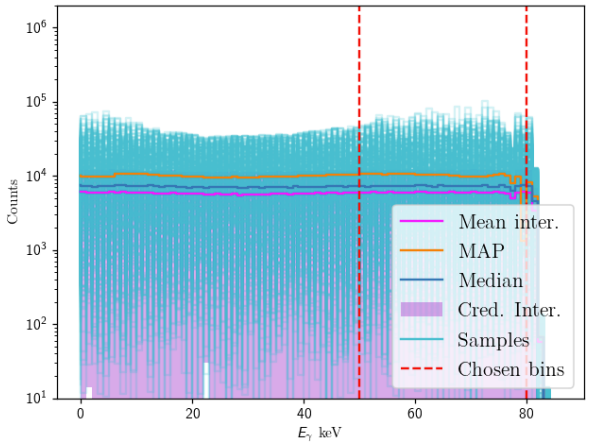
9.3.4 Response as a normalized matrix of ones

Now, the response is a normalized matrix of ones:

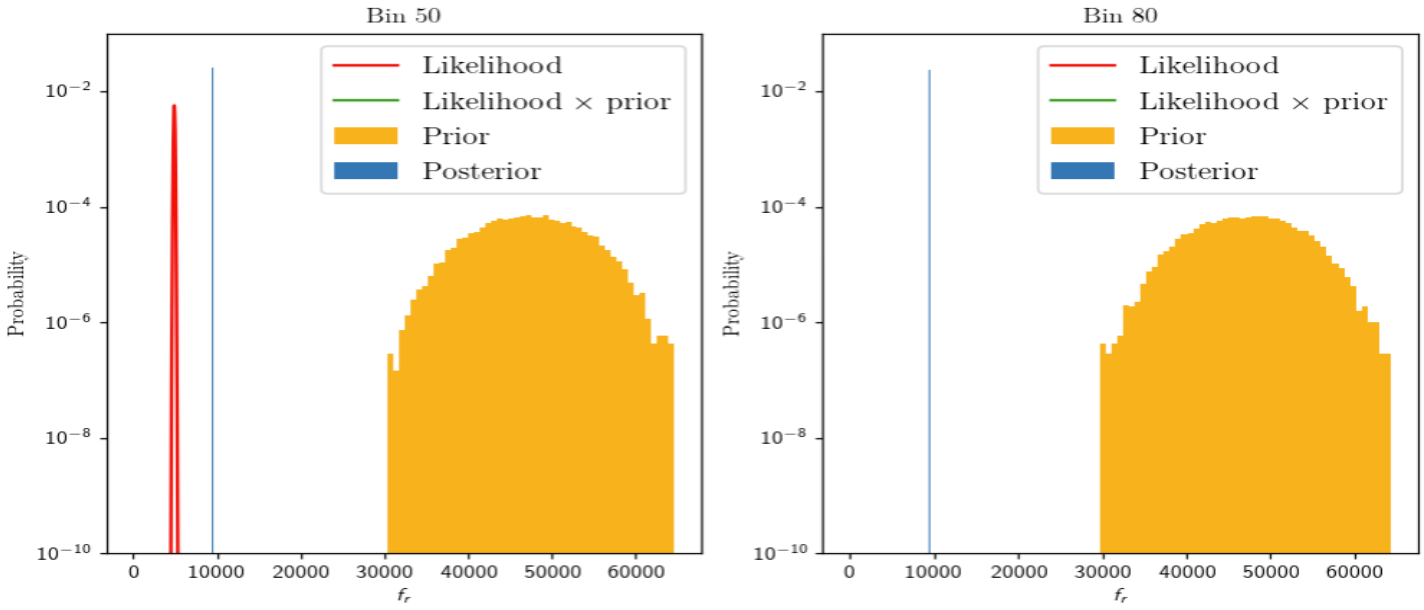
$$R^{N\times N} = \begin{bmatrix} 1/N & 1/N & \dots \\ 1/N & \ddots & \\ \vdots & & \ddots \end{bmatrix} \tag{3.22}$$



(a) Response matrix



(b) Result of unfolding. Dashed lines show the bins chosen for the analysis below.

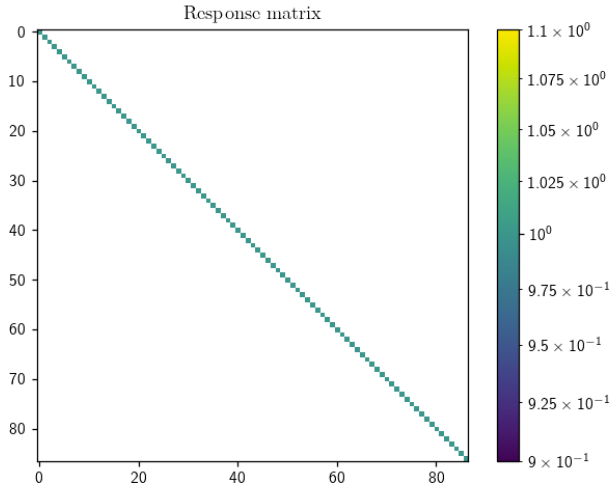


(c) The components of Bayes' theorem after unfolding, for the bins chosen above. Included is also a $L(\boldsymbol{D}|\boldsymbol{T}) \times P(\boldsymbol{T})$-function, which should have the same shape and position as the posterior, only differing by a normalisation constant.

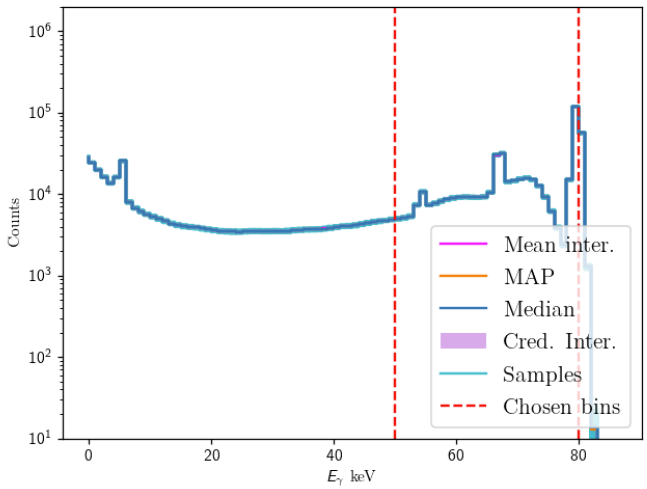
9.3.5 Response as the identity matrix

The results below are produced using an identity response matrix.

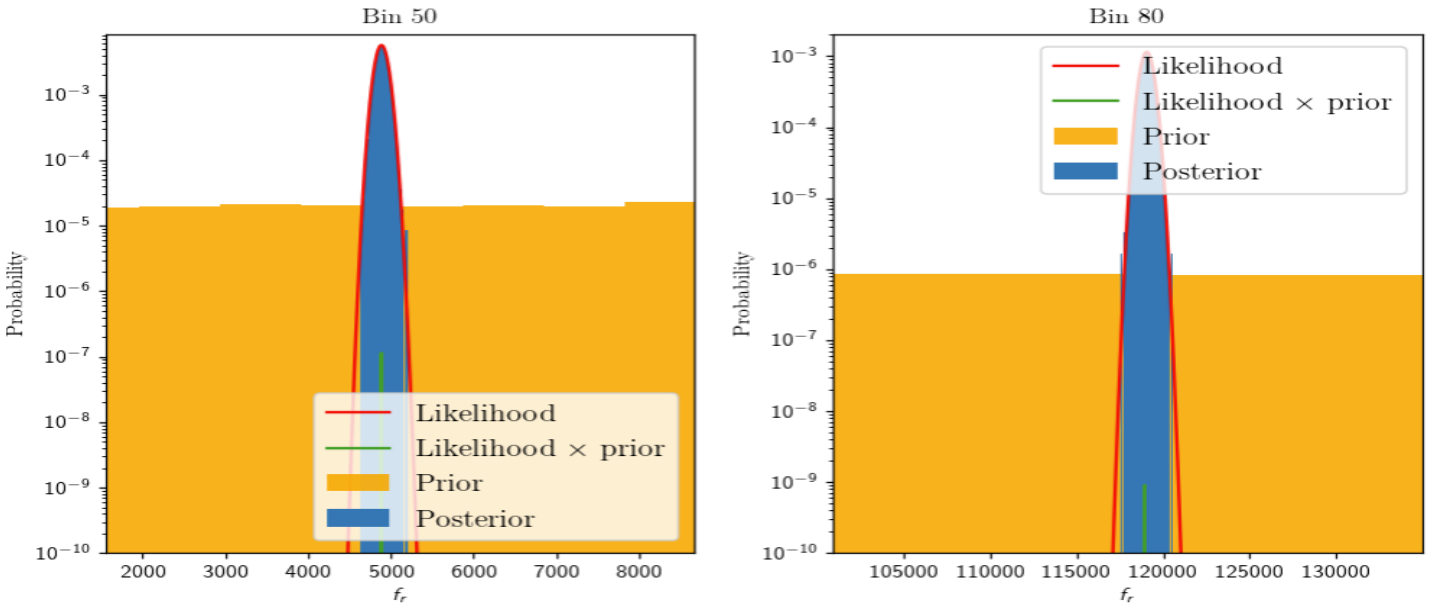
$$R = \mathbb{I} \tag{3.23}$$



(a) Response matrix



(b) Result of unfolding. Dashed lines show the bins chosen for the analysis below.



(c) The components of Bayes' theorem after unfolding, for the bins chosen above. Included is also a $L(\boldsymbol{D}|\boldsymbol{T}) \times P(\boldsymbol{T})$-function, which should have the same shape and position as the posterior, only differing by a normalisation constant.

Part III

Implementation

CHAPTER 4

PyFBU and PyMC3

For the implementation of FBU, we utilize the PyFBU package, made by Gerbaudo, Helsens and Rubbo [13]. It is directly based on the original FBU article by Choudalakis [4], and it is made to receive the observed data, response matrix and prior limits as inputs. Using these and other optional inputs such as the background spectrum or a specified number of sampling steps, it performs the modeling and sampling with the PyMC3 package [6]. PyMC3 is a statistical modeling library which has built-in probability distributions usable for both priors and likelihoods, e.g. Uniform, Normal, Poisson etc., as well as truncated versions of some distributions. PyMC3 leverages the Theano package for array operations and linear algebra with the use of symbolic variables [14].

A significant effort has been made attempting to fully understand the packages used in this thesis. Externally, they (PyFBU and PyMC3) are moderately simple to learn and the experience of using them is quite pleasant, should you be content with the limits they pose and the results you receive. Due to the multiple layers of class-references and abstraction, it is not immediately apparent how the code relates to the analytical procedure of FBU, and hence, why the results look as they do. Now, the authors may never have intended for the direct manipulation of their source code, and to expect them to facilitate the possibility would be unfair, seeing as the likely intended use is completely functioning. However, we are in a search of a greater understanding of the process and the results. If we should receive a result we do not expect, we want to know how it came to be, as well as the ability to fine-tune the individual elements in the name of improvement. This chapter focuses on how PyFBU and PyMC3 is modified to achieve an increased versatility of the unfolding process, by enabling more direct control over the individual Bayesian terms.

The following sections are based on the source code of the PyFBU and PyMC3 packages, as well as the documentation available for PyMC3 [13][6].

1 Usage and modification

We start by describing the general setup of PyFBU and how PyMC3 comes into play with the unfolding process. In practice, PyFBU can be said to be used in the following way:

- Create an object of the PyFBU class
- Supply the necessary variables, i.e. observed data, response matrix and the upper and lower prior limits
- Optionally supply parameters such as background data, systematic uncertainties, number of sampling chains and steps, etc.
- Run unfolding

Many things happen behind the scenes which the user does not see, of which the main parts will be discussed here, starting with the prior distribution.

1.1 Creating the prior

An important part of the process is how the prior is defined. The user only has to supply the limits for the prior, not the distribution itself. This is due to the fact that the uniform distribution is the default prior in PyFBU, and we have some ability to change that by supplying a string with the name of a different distribution. This string is then used to collect one of the built-in distribution classes in PyMC3. There is a good variety of these classes representing many popular distributions used in statistics. Unfortunately, there are only four of these that accept the lower and upper limits as parameters, namely the classes `Uniform` (default), `DiscreteUniform`, `Triangular` and `TruncatedNormal`. If we are to attempt to input the name of any other distribution, we will be met by errors due to the `lower` and `upper` parameters. PyMC3 does however include a class `Bound` that takes the limit parameters and constrains any of the built-in distributions. The resulting distribution is not normalized anymore, and we are still restricted to using the distributions included in PyMC3, unfortunately ruling out the log-uniform distribution (part II, subsection 3.1).

Another possibility is the `DensityDist` class, made for supporting custom distributions. This requires supplying a function returning the log-probability of the distribution you want to use, as well as a `random` method if the distribution is to be sampled from. These functions are not straightforward to implement, as complex distributions may not

easily be represented as analytical formulas, and attempts to use this class have not been successful by the author of this thesis.

It is possible to create an entirely new distribution class that mimics the functionality of the other classes, which of course requires some effort to correctly implement the underlying methods, Theano logic and inheritance to parent classes. Luckily, we can avoid this due to the final possibility for implementing custom distributions; the `Interpolated` class. This class belongs to the collection of continuous distributions in PyMC3 and allows for a higher degree of user influence. The parameter inputs are two arrays, one containing a lattice of `x_points` (counts) and one containing the corresponding `pdf_points` (probability densities). The distribution is then generated by linear interpolation of these probabilities. The `Interpolated` class can be found here (GitHub link). Now we are free to design whichever distribution shape we want, by directly controlling the probability height for each count in our assigned prior range. Furthermore, the prior limits are collected from the first and last element of the `x_points` array, meaning the previously mentioned `lower` and `upper` arguments are unnecessary. Lastly, the resulting distribution is automatically normalized by PyMC3, allowing for the direct use as a prior in PyFBU. The `Interpolated` class is very promising, and the integration of this class into PyFBU will now be described.

When the `run()` method is called, a PyMC3 model is created, wherein the main math and sampling is performed. The prior distribution is created here, using an external `wrapper` method which returns a PyMC3 object representing a stack of N tensors, a prior distribution for each bin. The important part is found inside this `wrapper`, where the type of distribution is determined by the `priorname` parameter. Originally, this creates a new distribution object from PyMC3 for each bin in the spectrum, assuming there exists one for the current `priorname`, and that it can take the `lower` and `upper` arguments. All these distributions are then stacked and passed on to the main PyFBU program. The suggested changes to this method is including an alternative creation of distribution objects if `priorname = Interpolated`, where `lower` and `upper` are not used. The `x_points` and `pdf_points` arguments are passed through to the `other_args` dictionary by assigning them to the `priorparams` variable accessible in PyFBU. The original code and suggested changes are shown in figure 4.1 and 4.2, respectively.

priors_original.py > ...

```python
import pymc3 as mc

priors = {
    }

def wrapper(priorname='',low=[],up=[],other_args={},optimized=False):


    if priorname in priors:
        priormethod = priors[priorname]
    elif hasattr(mc,priorname):
        priormethod = getattr(mc,priorname)
    else:
        print( 'WARNING: prior name not found! Falling back to DiscreteUniform...' )
        priormethod = mc.DiscreteUniform

    truthprior = []
    for bin,(l,u) in enumerate(zip(low,up)):
        name = 'truth%d'%bin
        default_args = dict(name=name,lower=l,upper=u)
        args = dict(list(default_args.items())+list(other_args.items()))
        prior = priormethod(**args)
        truthprior.append(prior)

    return mc.math.stack(truthprior) #https://github.com/pymc-devs/pymc3/issues/502
```

Figure 4.1: The original prior-creation function in PyFBU [13], which returns a stack of N prior distributions, one for each bin in the data. The file has been renamed from `priors.py` to `priors_original.py` to distinguish from the modified file in figure 4.2.

priors.py > ...

```python
import pymc3 as mc
priors = {
    }

def wrapper(priorname='',low=[],up=[],other_args={},optimized=False):
    # Suggested changes are in blocks enclosed by #---# borders
    #-------------------------------------------------------------------------#
    # Get non-keyword arguments from other_args, return empty list if not found
    non_kwargs = other_args.get('non_kwargs', [])
    #-------------------------------------------------------------------------#

    if priorname in priors:
        priormethod = priors[priorname]
    elif hasattr(mc,priorname):
        priormethod = getattr(mc,priorname)
    else:
        print( 'WARNING: prior name not found! Falling back to DiscreteUniform...' )
        priormethod = mc.DiscreteUniform

    truthprior = []
    #-------------------------------------------------------------------------#
    # If the Interpolated class is to be used, use arguments from non_kwargs
    if priorname == 'Interpolated':
        for bin,(l,u) in enumerate(zip(low,up)):
            name = 'truth%d'%bin
            prior = priormethod(name, non_kwargs[0][bin], non_kwargs[1][bin])
            truthprior.append(prior)
    else:
    #-------------------------------------------------------------------------#
        for bin,(l,u) in enumerate(zip(low,up)):
            name = 'truth%d'%bin
            default_args = dict(name=name,lower=l,upper=u)
            args = dict(list(default_args.items())+list(other_args.items()))
            prior = priormethod(**args)
            truthprior.append(prior)

    return mc.math.stack(truthprior) #https://github.com/pymc-devs/pymc3/issues/502
```

Figure 4.2: Modified version of the `priors.py` file in PyFBU [13], shown in figure 4.1. The changes are shown in blocks enclosed by comment borders, allowing for the use of the `Interpolated` class in PyMC3. This enables a much greater freedom in designing the shape of the prior distribution, done by determining prior range and corresponding pdf-values in the users code and passing to PyFBU.

The user is now able to externally define the exact shape of the prior distribution which makes it possible to use an endless variety of distributions like the log-uniform distribution discussed in part II, subsection 3.1.

1.2 The likelihood

Picking up the thread from the creation of the prior, the next step in PyFBU is to incorporate the likelihood function. By PyMC3 convention this is done by creating another distribution object, from the Poisson class in our case, and evaluating it on the **folded** prior object. The reason for this can be understood by considering the spaces where our Bayesian terms are defined. The prior $\pi(\boldsymbol{T})$ is an assumption of the truth, meaning it is defined in the truth-space, it represents what we believe the true count-value can be in each bin. However, the Poisson distribution we define lives in the folded space, along with the observed data, as it is dependent on the folded parameter f_r, see eq. 3.8. This means it describes the spread of possible observed values given the already supplied observed values. We can transform this to a function in the truth space by specifying a truth-range and making the f_r-parameter dependent on that range, see eq. 3.9. Evaluating the likelihood on this truth-dependent range makes it describe the spread of possible **truth**-values that can lead to the given observed data. This Poisson distribution object evaluated on the folded prior will then represent a space of likelihood-weighted prior-values, which when sampled will result in the posterior distribution. After this, the next step in PyFBU is running the NUTS sampling algorithm, which when finished, outputs the final posterior samples for each bin in the spectrum.

1.2.1 The modified likelihood

CHAPTER 5

Synthetic spectra

- 1dim test

Here, we perform unfolding on a raw spectrum consisting of 2 bins, to display all Bayesian terms and their relation. Even though experimental spectra usually consist of a much larger amount of bins, we have chosen 2 to be able to show the likelihood properly. As mentioned in part II, the likelihood is an N-dimensional function which is not easily decomposed into 1-dimensional contributions. Using 2 bins allows us to plot the likelihood in its entirety on the plane and compare with the posterior. We do this to firstly, display and compare the complete prior and posterior distributions, and the effect of the response matrix. Secondly, we wish to confirm what we believe to be correct of the built-in, symbolized likelihood, by comparing the posterior with an independent, externally constructed Poisson distribution. If we are correct, we expect our Poisson distribution to overlap and exhibit a similar shape as the posterior distribution from PyFBU.

We choose a simple true spectrum $\boldsymbol{T} = (120, 120)$ and construct a 2×2 response matrix with arbitrary values and normalized rows (preferably not the identity matrix, as no changes would happen when folding, i.e. a perfect detector):

$$R_{2\times 2} = \begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \end{bmatrix} \tag{5.1}$$

The response matrix is visualized in figure 5.1.

Chapter 5. Synthetic spectra

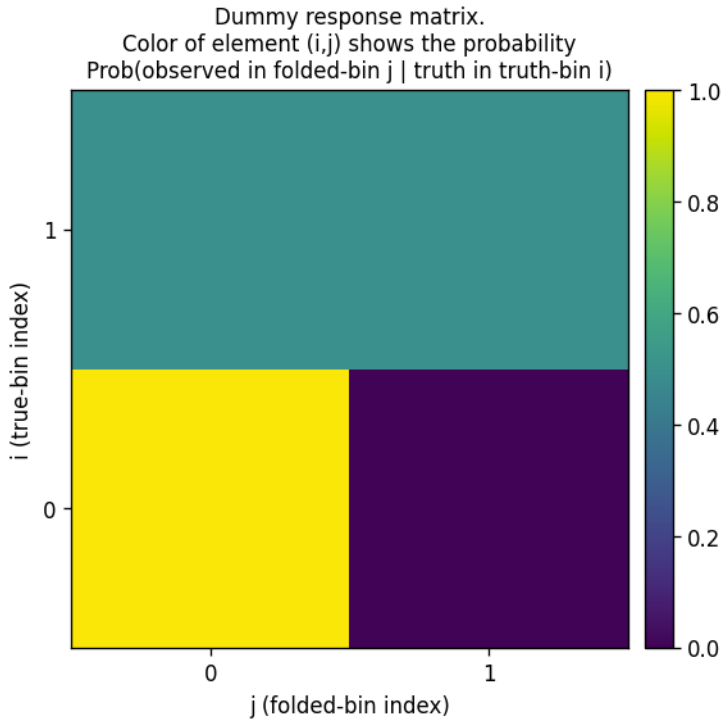


Figure 5.1: The constructed 2-bin response matrix. Note that when plotting the response, the origin for the indices is at the bottom left corner, as opposed to a mathematical matrix which has indices starting at the top left.

Next, we generate an 'observed' spectrum by folding the true spectrum with the response: Maybe explain why its TR instead of RT

$$\boldsymbol{D} = \boldsymbol{TR} = \begin{bmatrix} 120 & 120 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 180 \\ 60 \end{bmatrix} \tag{5.2}$$

Note that by generating the data this way, we take away the inherent randomness of the response matrix. Since it consists of probabilities of an event being reconstructed in bin r given that it originated in truth-bin t, the data would not remain constant for repeat experiments; flipping 10 fair coins does not result in 5 tails and 5 heads every time. A more realistic generation of observed data can be performed by randomly sampling a value in each bin according to the given probabilities. mention multinomial distribution? The mean value of these samples will be the same as the result we get from direct multiplication with the response. That is sufficient in this case, where the focus is on investigating the Bayesian terms and verifying our knowledge of the likelihood, for which the actual values of the observed data does not matter. Is this true? Should I sample the values instead of direct multiplication? For both bins, we assign a uniform prior in the range

$[0, 200]$ since we know the true values, and perform the unfolding. The resulting posteriors, together with priors and true values are shown in figure 5.2.

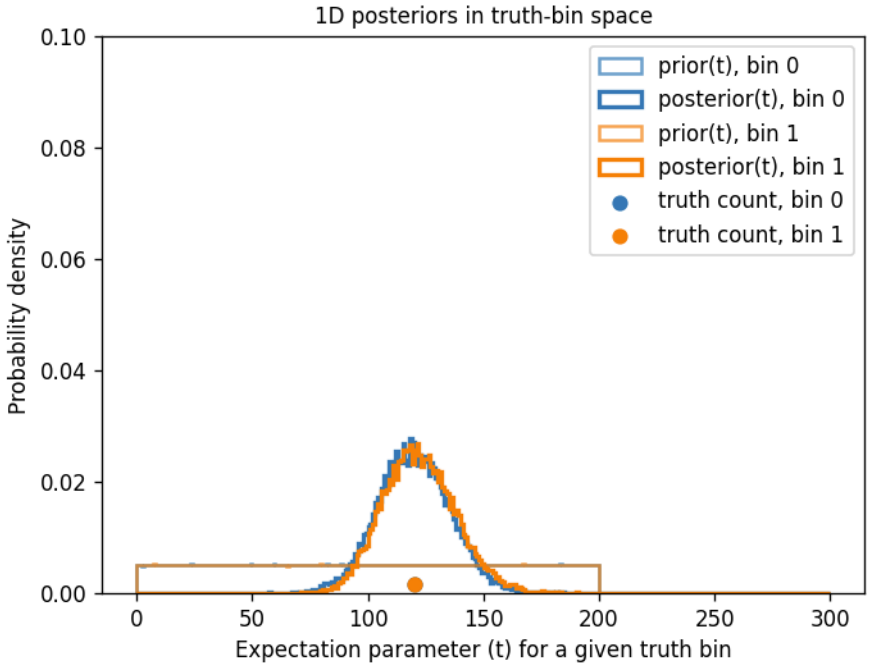


Figure 5.2: Prosterior distributions after unfolding a 2-bin constructed spectrum, along with corresponding priors and true values. Both posteriors point to the true value being located around 120, which is correct. The spread of the posteriors represents the uncertainty. We can also see that our uniform priors were suitable choices, as the true values are located within, and the posteriors are not truncated.

The plotted priors and posteriors are histograms consisting of samples from their respective distributions, and we may combine both bins to show the complete 2-dimensional histograms with 1 bin per axis. The complete prior is shown in figure 5.3.

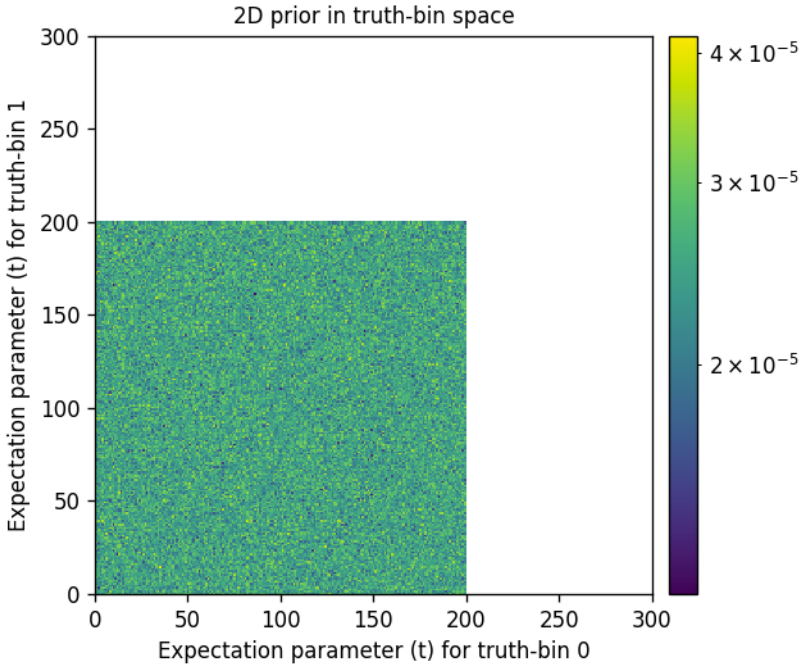


Figure 5.3: 2-dimensional complete prior distribution for both bins in the spectrum, consisting of random samples from the uniform distribution in the range $[0, 200]$.

Next, we plot our 2-dimensional Poisson distribution in the truth-bin space and see that it belongs to the same domain as the combined prior. This is shown in figure 5.4.

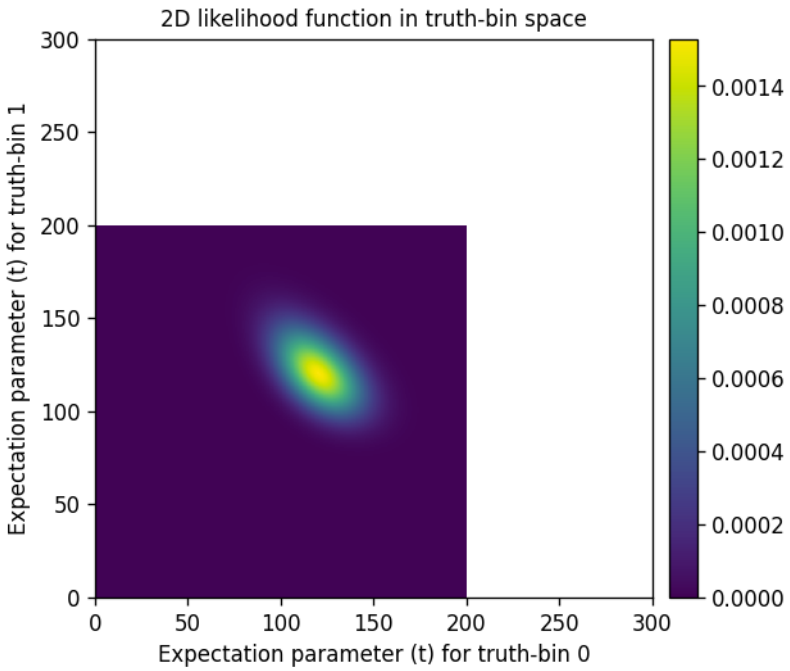


Figure 5.4: 2-dimensional Poisson distribution in the truth-bin space, dependent on the observed data and our defined ranges of possible true values. Note that this, as opposed to the prior and posterior, is a function defined over both dimensions, rather than a collection of samples.

With these pieces in place we can examine whether our Poisson distribution corresponds to the likelihood that is built into PyFBU. We do this by combining the posteriors in the same way we did with the priors (figure 5.3) and plot this together with the Poisson distribution, shown in figure 5.5.

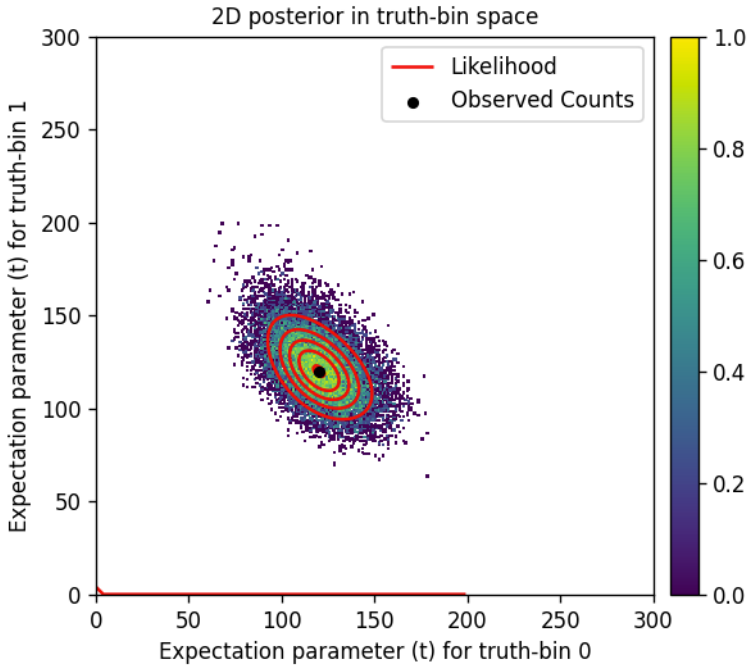


Figure 5.5: 2-dimensional complete posterior distribution for both bins, output from PyFBU, as well as contour lines from the 2-dimensional Poisson distribution shown in figure 5.4. Remember that the contours belong to a function we have defined based only on our available data and assumptions, meaning there is no connection to PyFBU. Yet, the contours exhibit a similar shape and location of the Poisson distribution as the posterior, indicating that our externally constructed function matches the internal likelihood of PyFBU. As mentioned in part II, subsection 3.3, we strictly have to compare the posterior with Poisson$\times$prior. However, since our prior is a uniform distribution, neither the shape nor the location of the posterior is affected by the prior, meaning we can directly compare with the likelihood candidate.

We see that our assumption about the likelihood was correct, and we have gained a stronger understanding of how the PyFBU-package is built up. This reduction of the black-box trait lets us have a greater confidence in future results, and how they actually are produced. Next, in figures 5.6 - 5.9, we examine the corresponding plots of the distributions in the folded-bin space, i.e. the space where the observed data is contained. This is where the likelihood function is originally defined, i.e. being dependent on the variable f_r in eq. 3.9 and eq. 3.8, and we will see the effect of the response matrix on the prior and posterior.

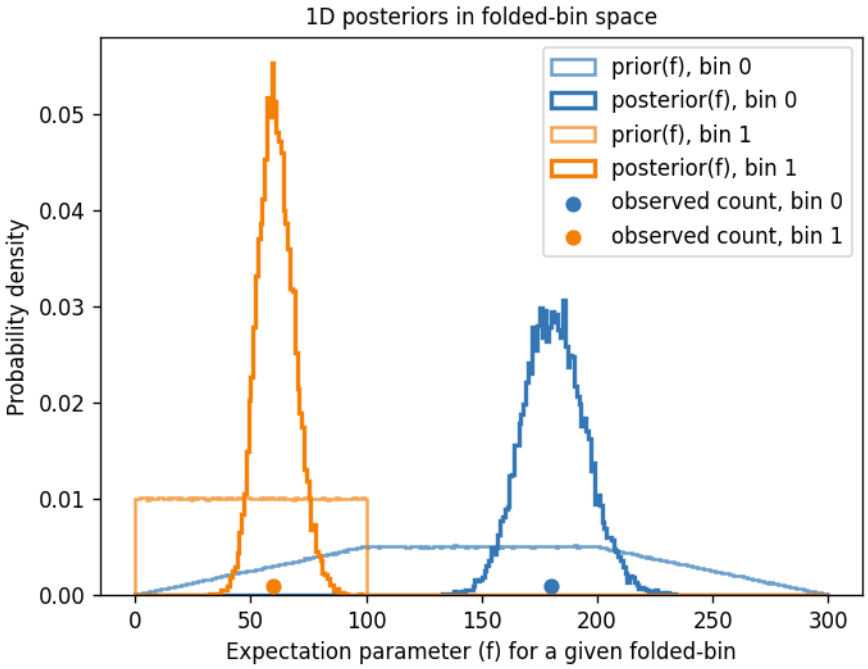


Figure 5.6: Posterior and prior distributions from figure 5.2 folded with the response matrix, along with the observed data, in folded-bin space. We see the smearing effect, how counts can be redistributed due to the detector response. The range for the prior for bin 0 is seen to have been resized from $[0, 200]$ to $[0, 300]$, and to $[0, 100]$ for bin 1. The corresponding heights have thus changed, preserving the total probability of 1 for both priors. The posteriors are centered around their respective observed counts, and show the spread for which possible observed values can lead to the true counts of 120 in this case.

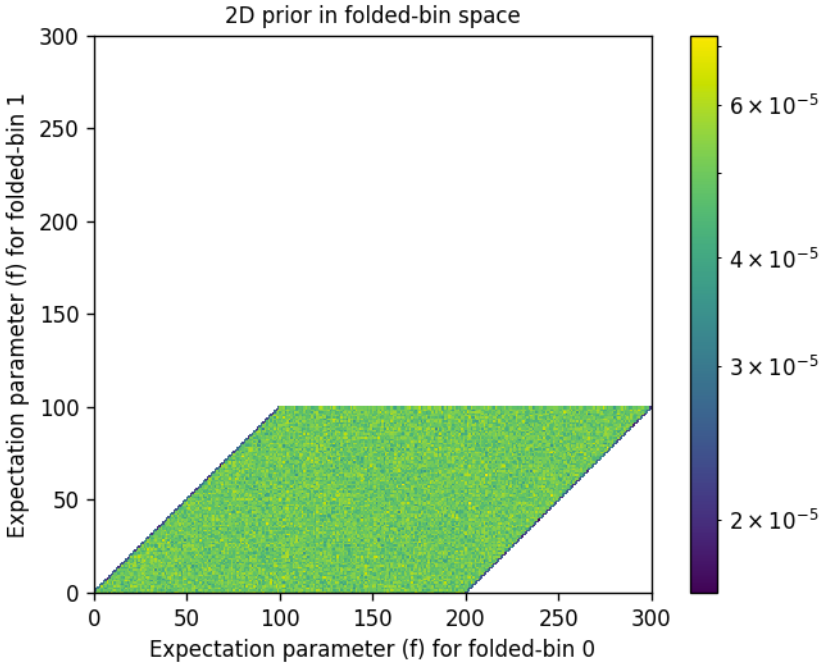


Figure 5.7: The complete 2-dimensional prior distribution from figure 5.3 in folded-bin space. We see that the prior has been skewed and occupies a smaller area than in the truth-bin space. We can also see that the prior heights, shown in color, are larger to compensate.

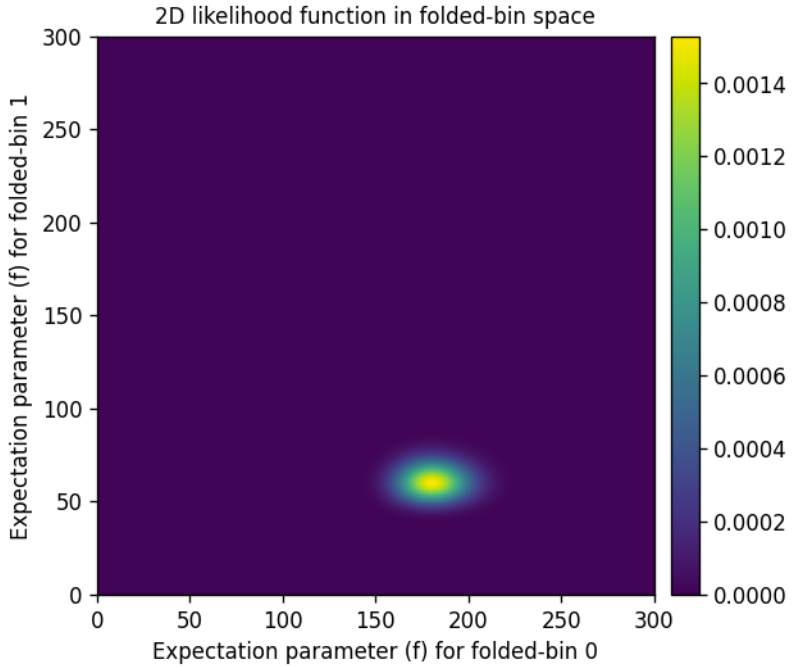


Figure 5.8: The 2-dimensional Poisson distribution from figure 5.4 in the folded-bin space, corresponding to the likelihood in PyFBU. We see a more concentrated peak here than in the truth space.

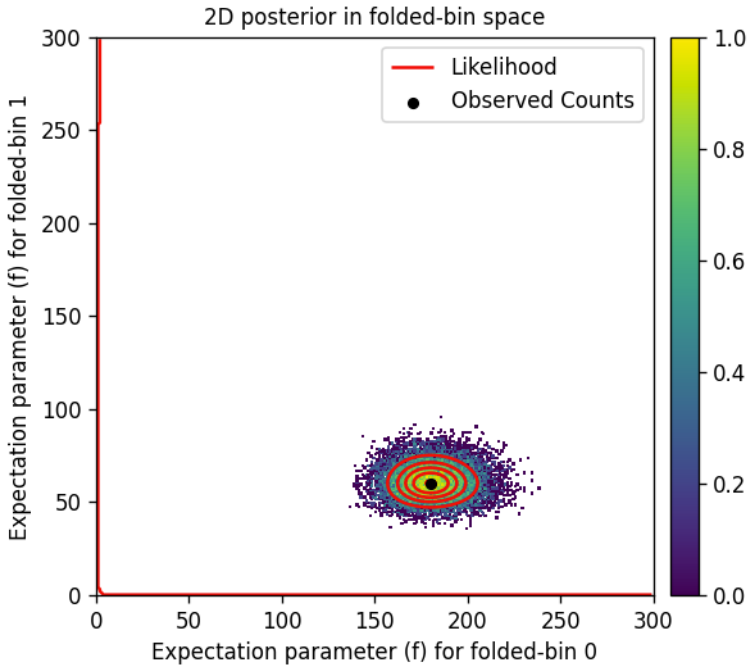


Figure 5.9: The 2-dimensional complete posterior distribution from figure 5.5, as well as contour lines from the likelihood function, in folded space. Here too, the likelihood and posterior shapes and locations match very well, confirming again our assumption about the likelihood. The narrower distribution, especially for bin 1, shows that there are less possible observed counts leading to a truth count of 120, than possible truth counts leading to the observed value from the detector.

Part IV

Results & Discussion

CHAPTER 6

Experimental spectra

1 The ^{28}Si spectrum

- Compare result with Valas, using logscale prior, modified likelihood?
- Background?

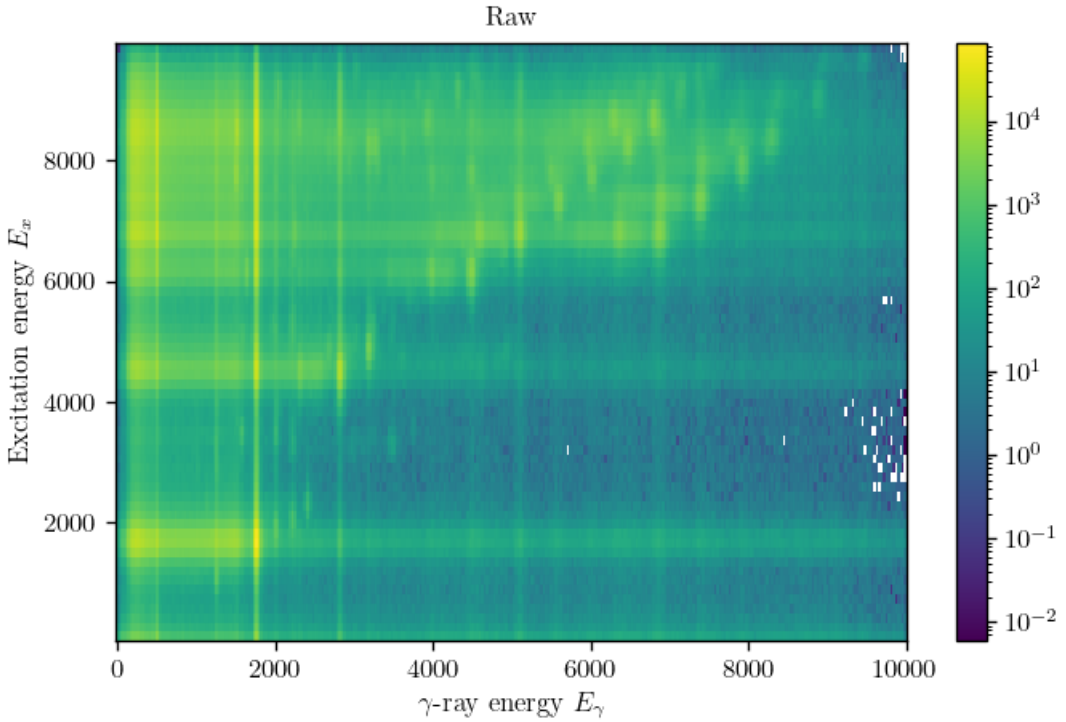


Figure 6.1: Complete raw spectrum for ^{28}Nd, showing the observed γ-ray spectra for each excitation energy E_x, received from Ann-Cecilie Larsen. This has been rebinned with a factor 3.

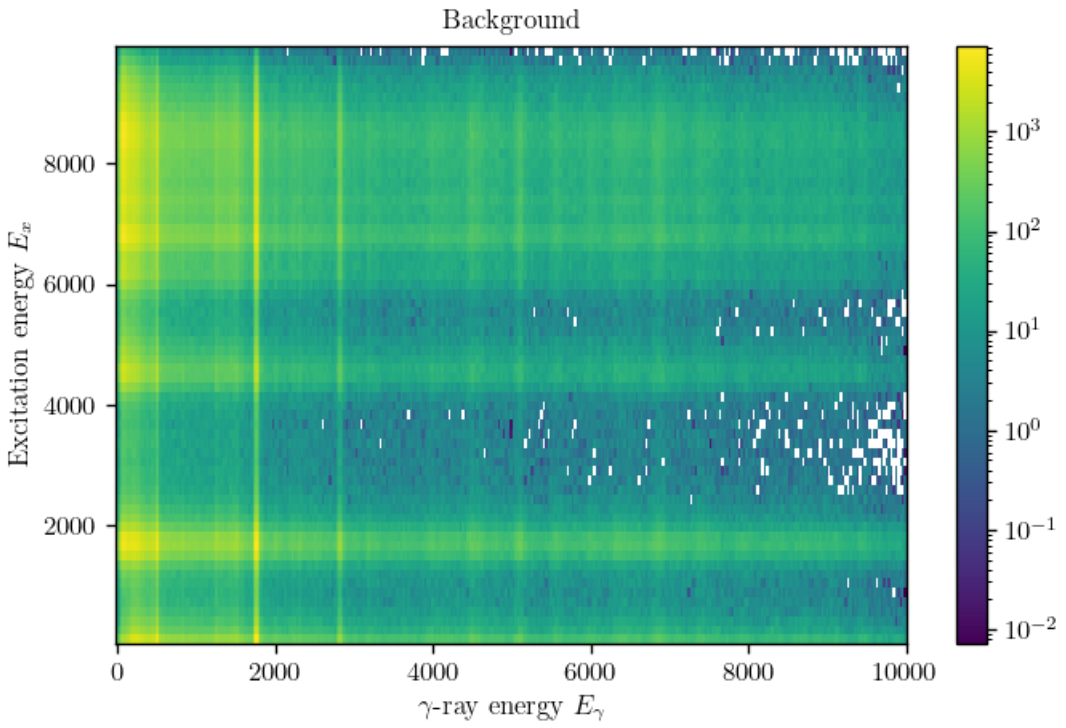


Figure 6.2: The background energies present in the raw data in figure 6.1, received from Ann-Cecilie Larsen. This has been rebinned with a factor 3.

- Response?

1.1 The first excited state

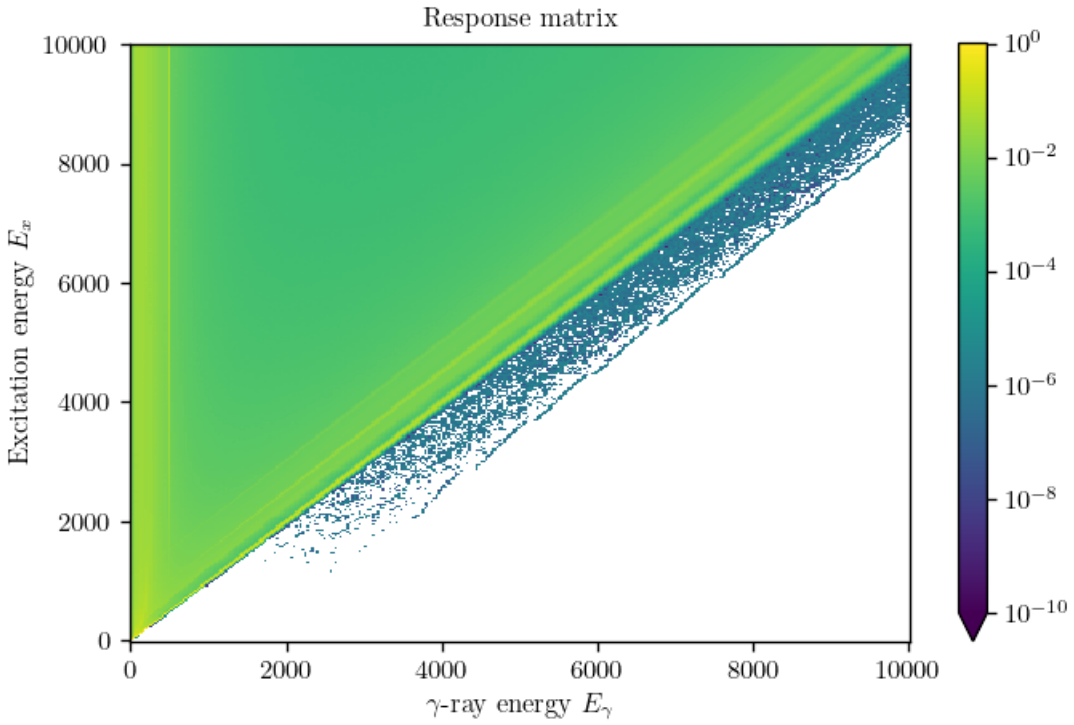


Figure 6.3: The new response matrix from 2020 from the OMpy library [15][16].

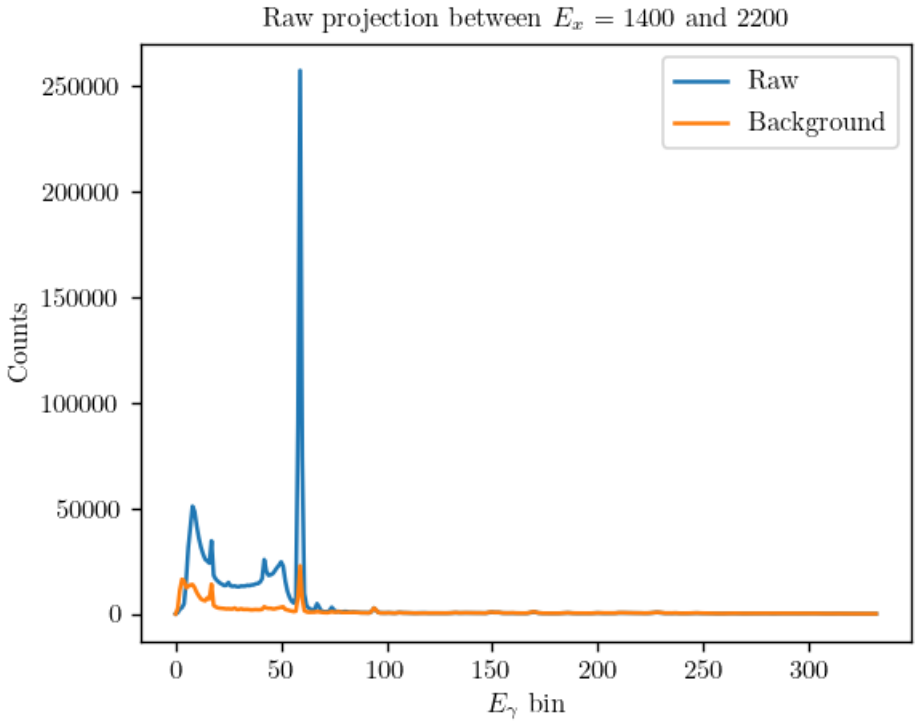
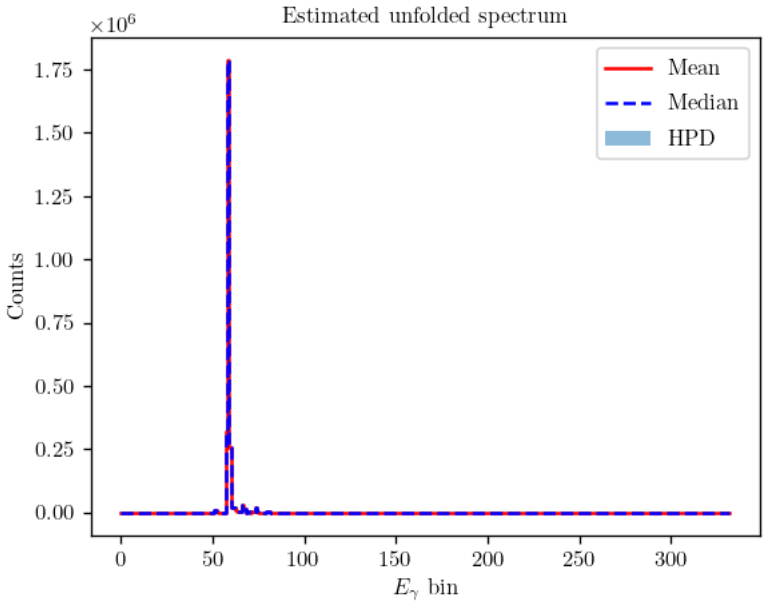
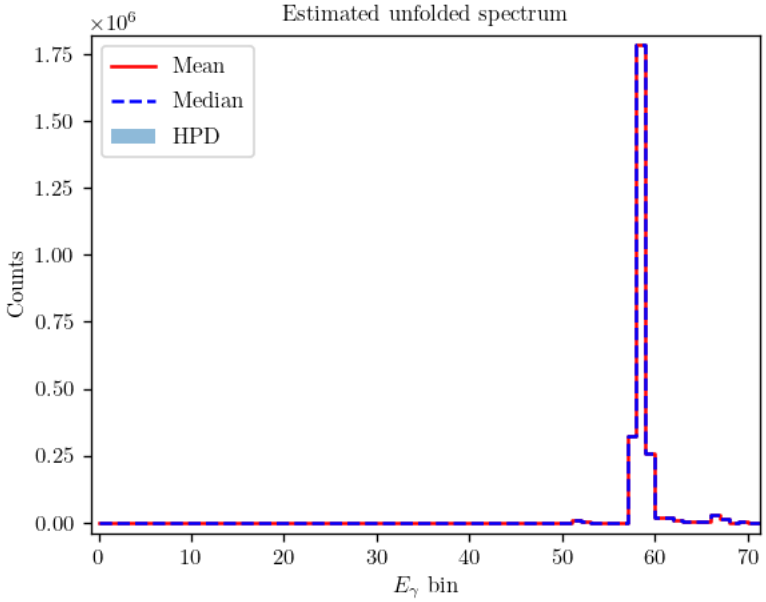


Figure 6.4: Projection of the raw spectrum and background for the first excited state, i.e. between $E_x = 1400$ and 2200, see figure 6.1, showing observed counts in each energy bin.

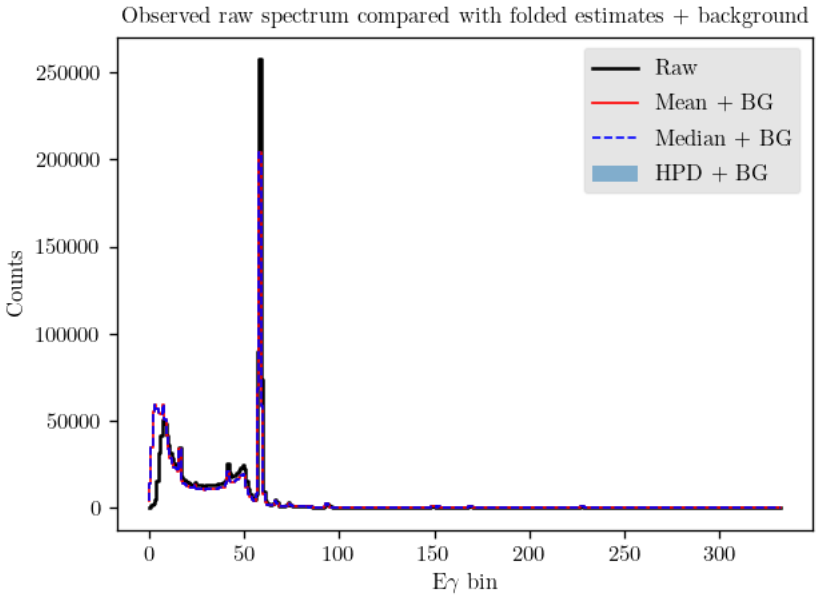


(a)

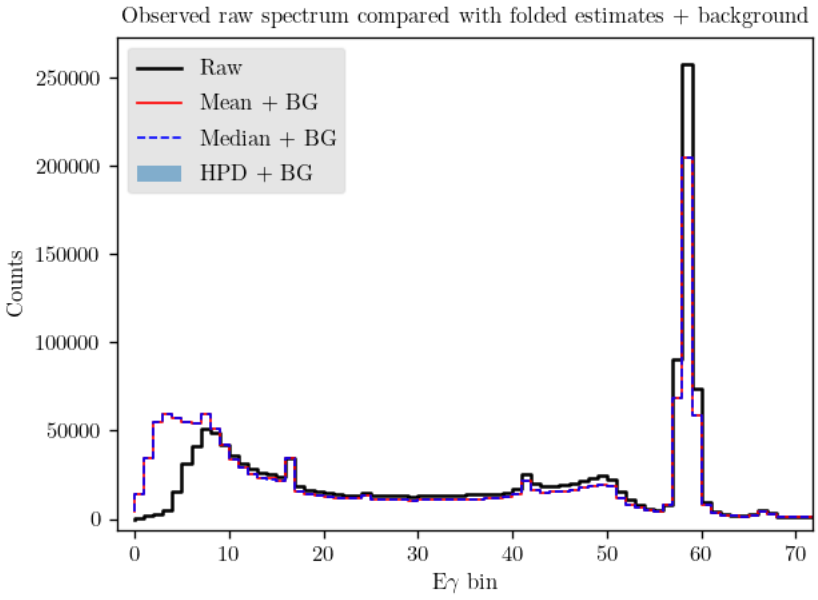


(b)

Figure 6.5: The aggregated estimates for the posterior distributions in each bin, representing an estimated unfolded spectrum, for all bins (a) and zoomed to the first 70 bins (b). We see that almost all counts have been redistributed to a single peak around bin 60 for our expectation. Further analysis is performed on this spectrum and shown in figures 6.6 and 6.7. The results from OMpy and subsequent comparisons are shown in figures 6.8, 6.10 and in table **??**. Lastly, we compare with Valsdóttir's results in figure **??**

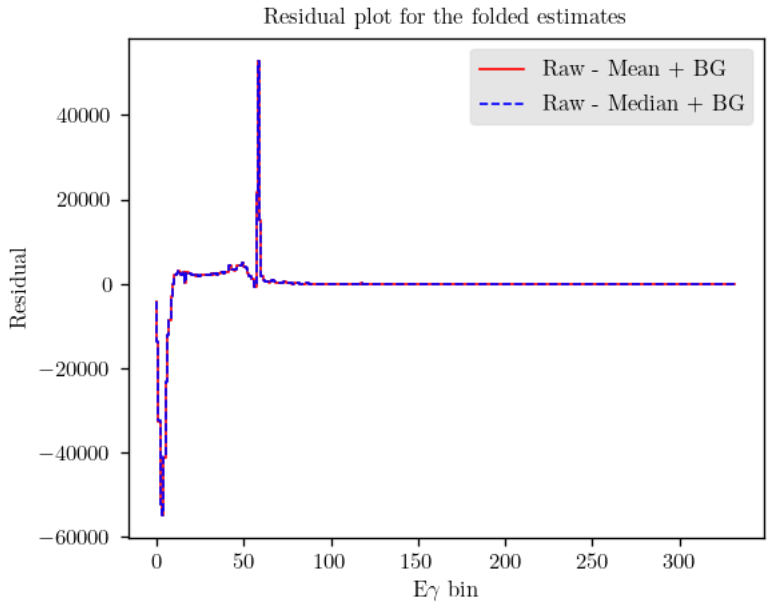


(a)

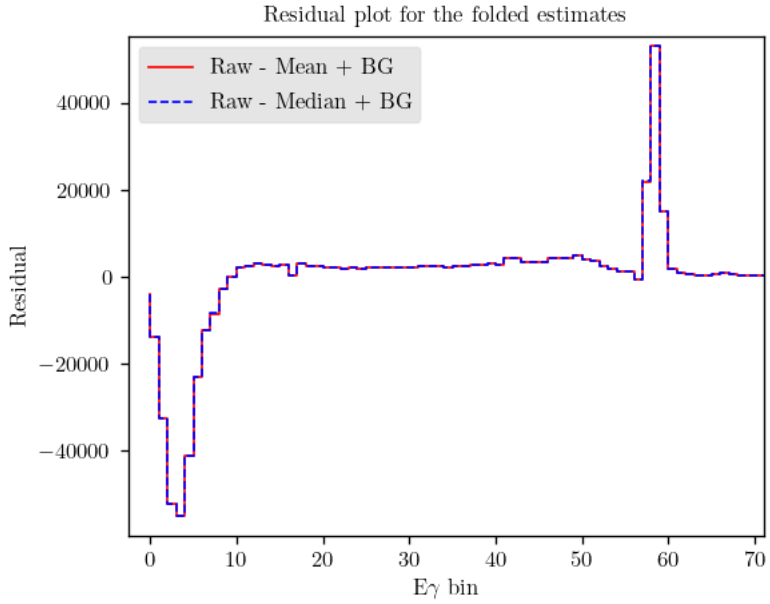


(b)

Figure 6.6: Refolding of the unfolded results compared with the raw observed data, in two ways. The first plot (a) shows the result of folding the truth-sample estimates (HPD, point estimates) shown in figure 6.5. The second plot (b) shows the result for the first 70 bins.

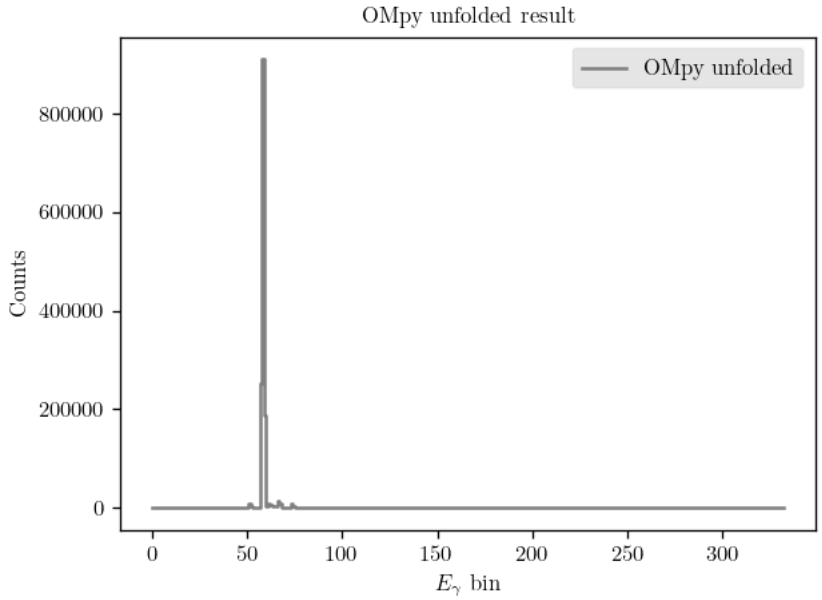


(a)

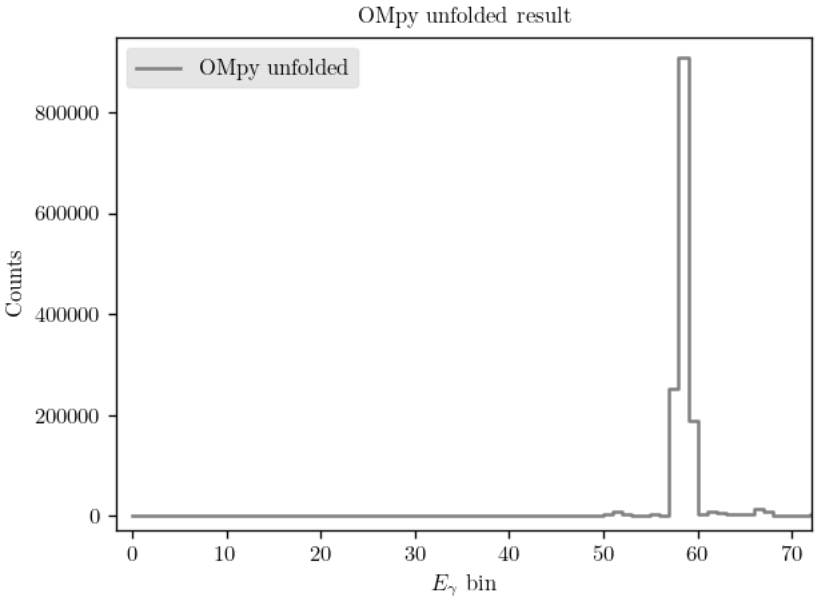


(b)

Figure 6.7: Residual plots showing the difference between the observed raw spectrum and the point estimates from FBU (figure 6.6a), first for all bins (a), then zoomed to the first 70 bins (b).

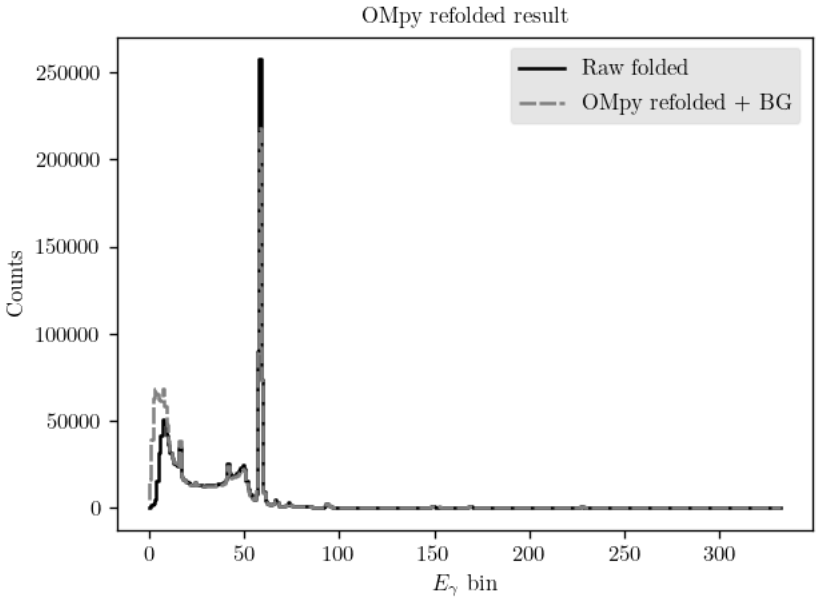


(a)

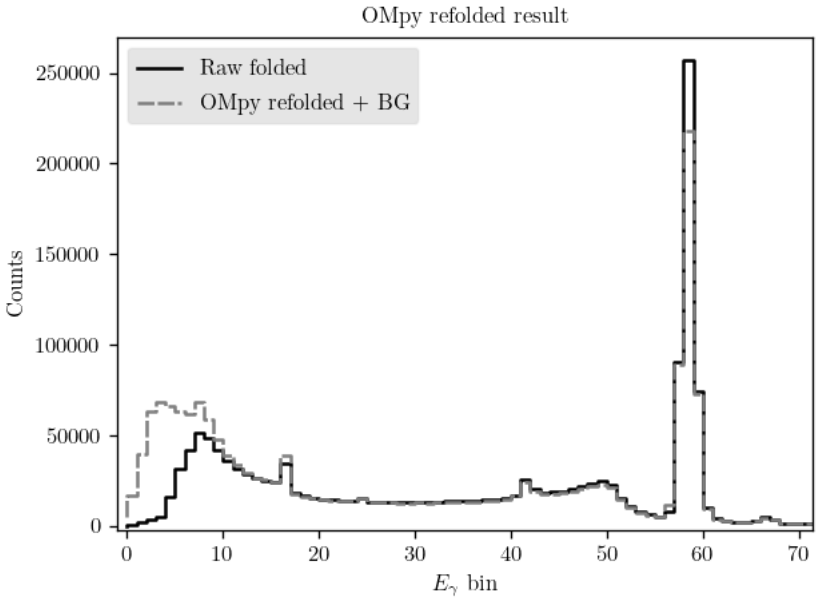


(b)

Figure 6.8: Result from unfolding using OMpy for all bins (a) and zoomed to the first 70 bins (b). The unfolded spectrum here contains a much lower peak (roughly 9×10^5) than the FBU result (roughly 1.75×10^6, see figure 6.5), while we see that the refolded spectra are of similar magnitudes, figure 6.10.

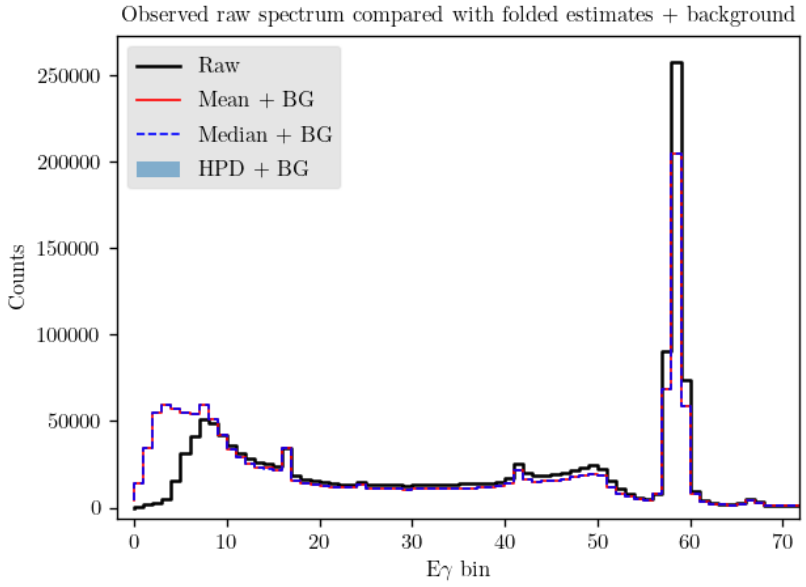


(a)

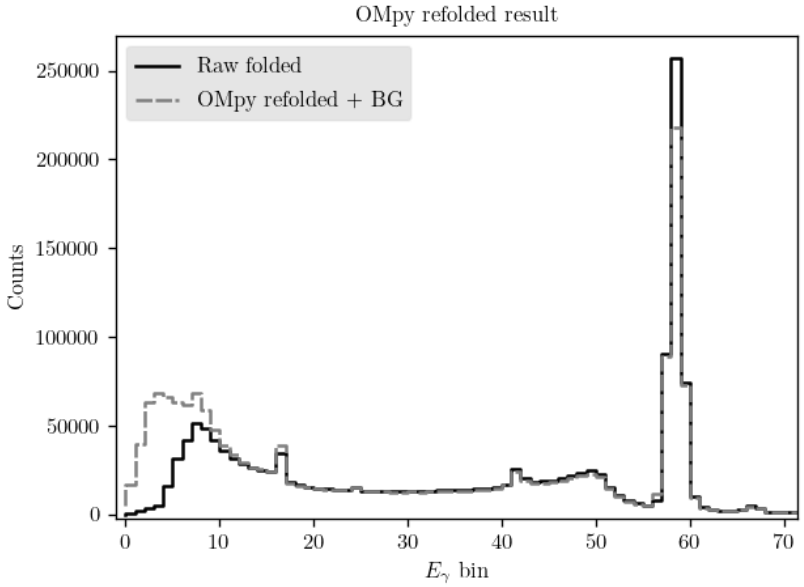


(b)

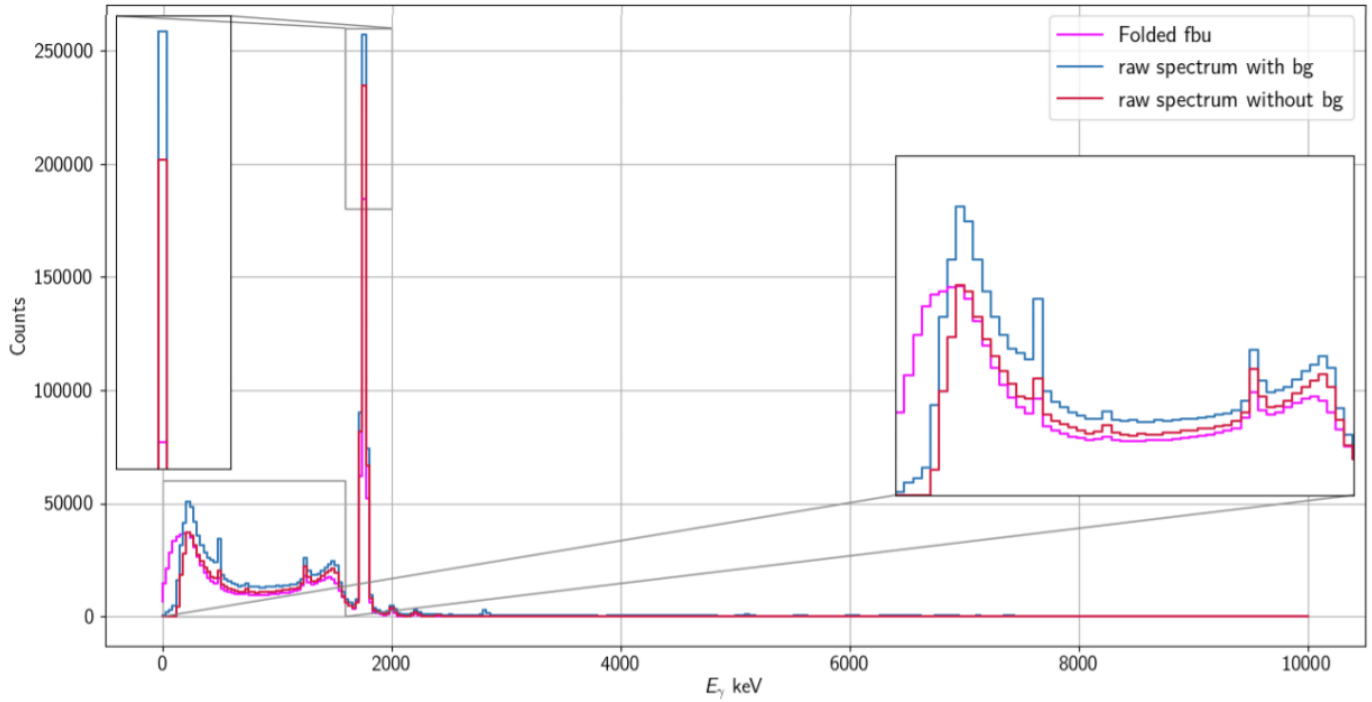
Figure 6.9: Refolding of the unfolded results compared with the raw observed data, in two ways. The first plot (a) shows the result of folding the truth-sample estimates (HPD, point estimates) shown in figure 6.8. The second plot (b) shows the result for the first 70 bins.



(a)



(b)



(c)

Figure 6.10: Zoomed versions of the refolded results (figures 6.6a and 6.9b) comparing FBU with OMpy for the first 70 bins, as well Valsdóttir's result for the same spectrum.[17]

Table 6.1: Error metrics for refolded results (rounded). We see all estimates score very well with low mean absolute errors, and R^2-scores all above 0.95. As seen in figure 6.19, the largest discrepancies are located at the lowest energy bins, however these are not large enough to have a big impact on the overall scores, since all bins are weighted equally in the metrics. There is an argument to be made about weighting these bins more, as this is where the most interesting structures are located. Doing so would require a manual definition of what is deemed the most interesting area on a case-by-case basis. Since we either way have the residual plots to directly show local errors, we are content to leave the MAE and R^2-score as summarizing error metrics for the entire spectrum. These scores would be even better if we were to include the mentioned additional information about the error in the response matrix for low energies, see figure 6.19 Again, is this true?. The results from both FBU and OMpy are very close, with a slightly higher R^2-score for the FBU estimates, signifying a very slightly overall better match with the observed data. Furthermore, we see a very small, almost negligible improvement in the FBU results when using the estimates calculated on the **folded** samples (figure 6.19b, here abbreviated to Fold. sampl.), over the folded **truth**-sample estimates (6.19a, Fold. est.).

	FBU Median		**FBU Mean**		**OMpy**
	Fold. est.	Fold. sampl.	Fold. est.	Fold. sampl.	
MAE	30.8	28.9	30.8	29.2	26.0
R^2-score	0.968	0.968	0.968	0.968	0.955

Table 6.2

	True samples	**Folded samples**
Mean variance	3535.7	90.0

2 The ^{146}Nd spectrum

- Raw? 2 dim and projected

- Response?

2.1 The first excited state

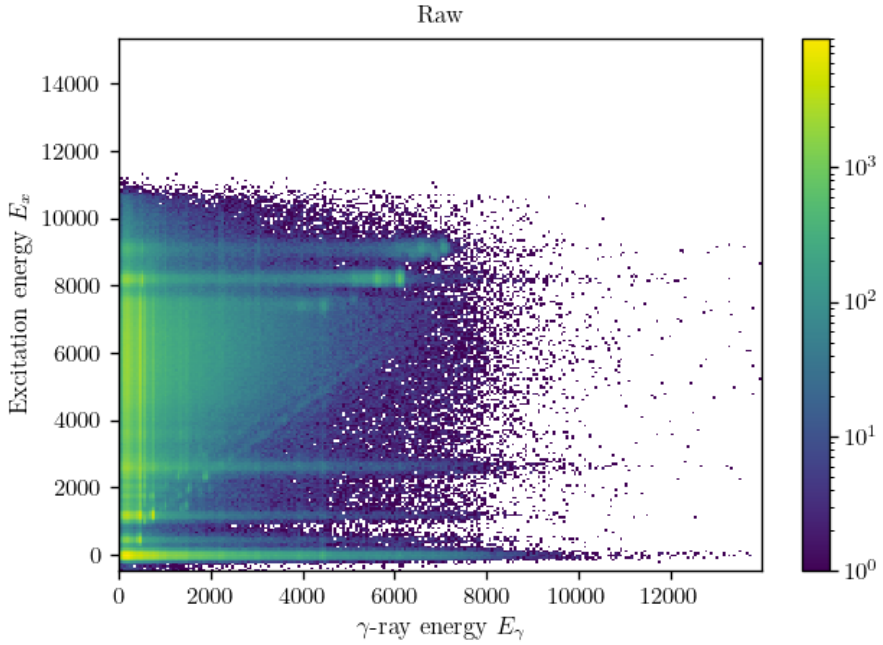


Figure 6.11: Complete raw spectrum for ^{146}Nd, showing the observed γ-ray spectra for each excitation energy E_x, received from Ann-Cecilie Larsen.

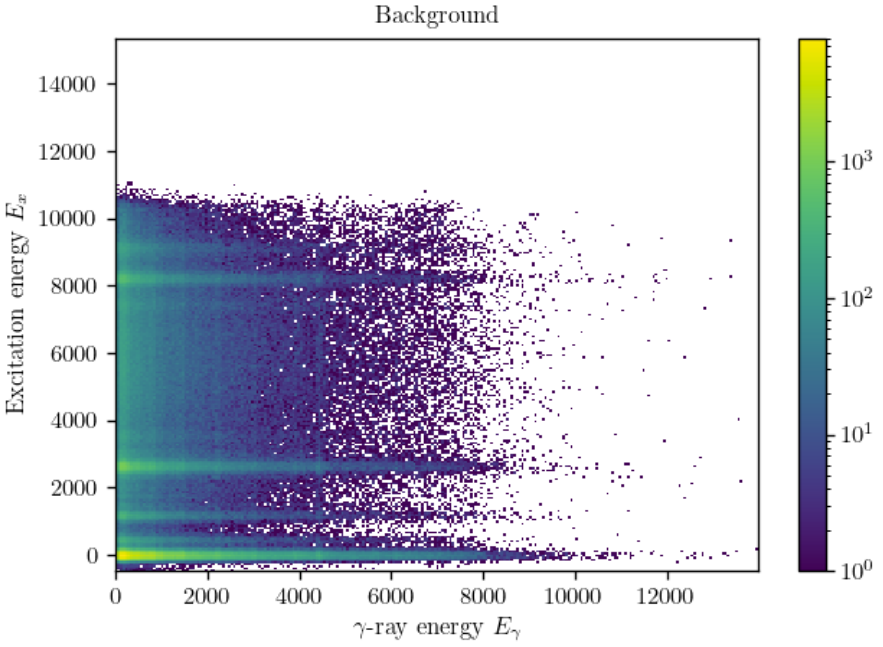


Figure 6.12: The background energies present in the raw data in figure 6.11, received from Ann-Cecilie Larsen.

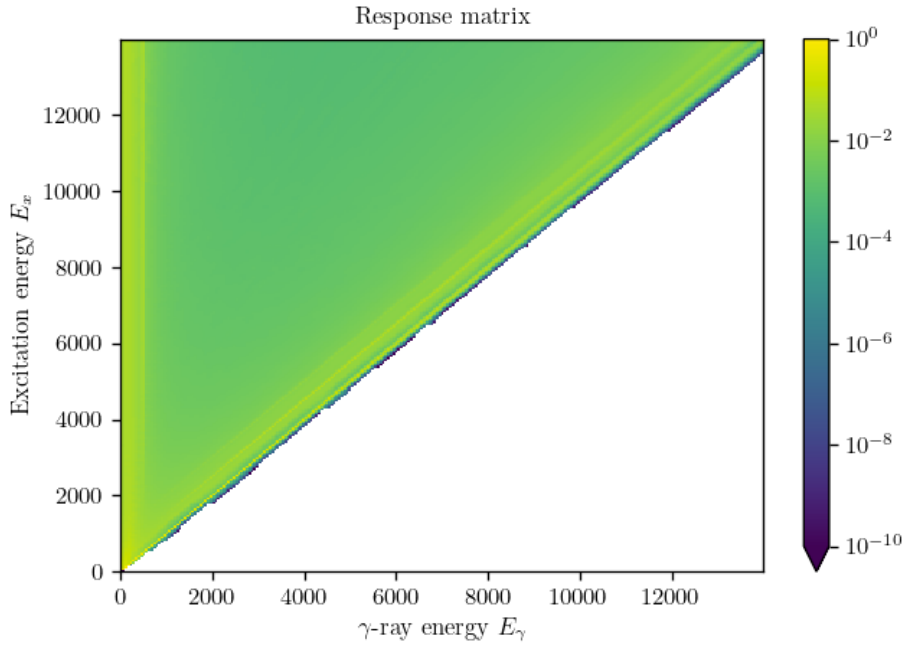


Figure 6.13: The complete response matrix from the OMpy library [15][16].

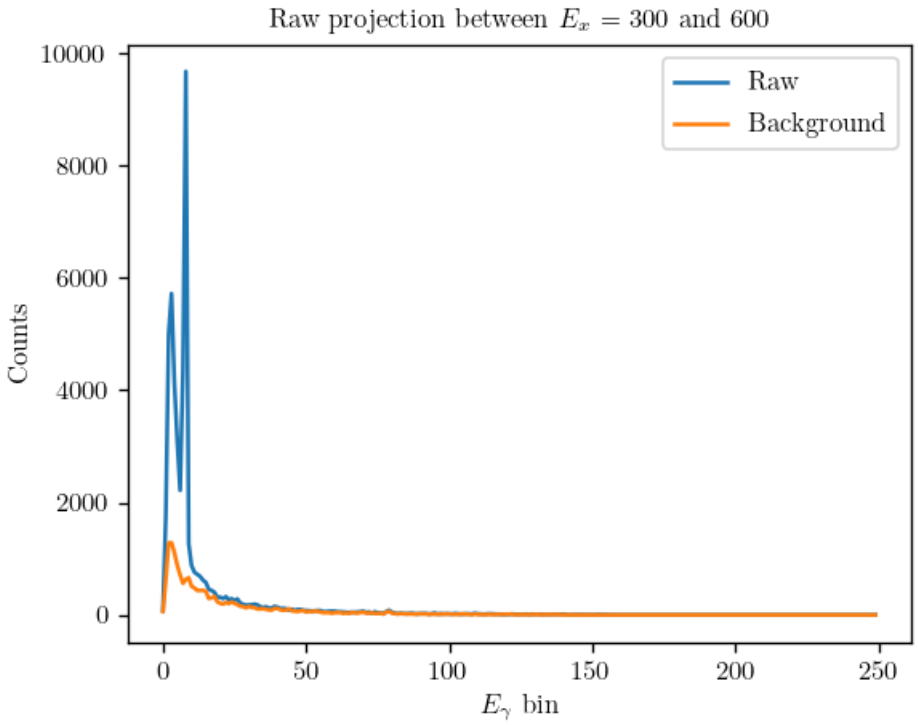
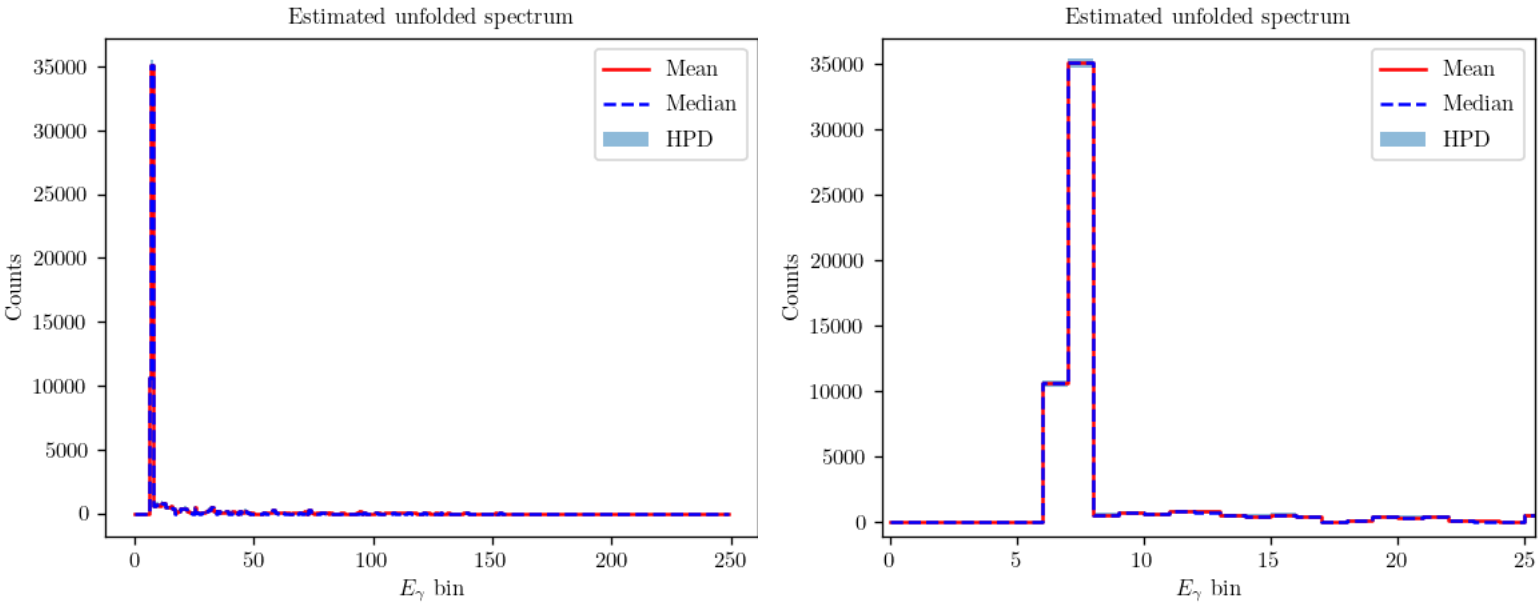
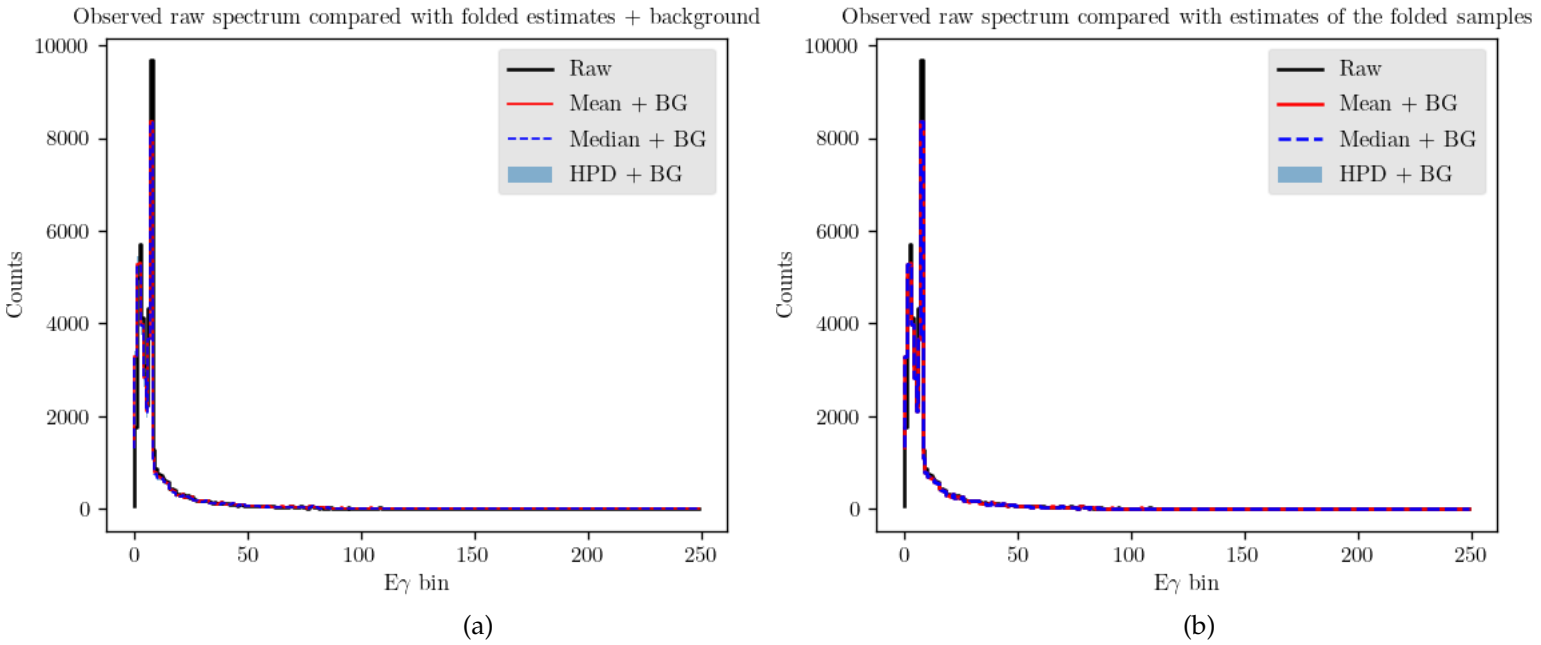


Figure 6.14: Projection of the raw spectrum and background for the first excited state, i.e. between $E_x = 300$ and 600, see figure 6.11, showing observed counts in each energy bin.



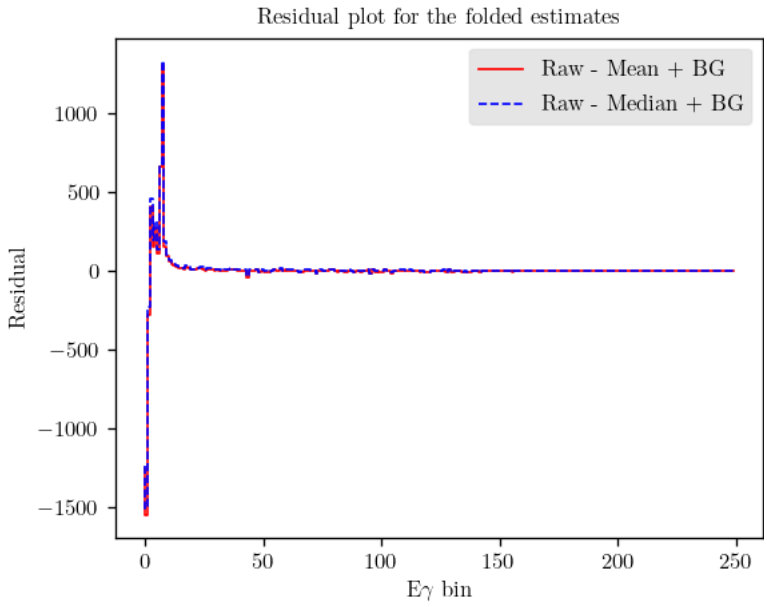
(a) (b)

Figure 6.15: The aggregated estimates for the posterior distributions in each bin, representing an estimated unfolded spectrum, for all bins (a) and zoomed to the first 25 bins (b). We observe a single peak by the low energy bins, showing that the double peak in the observed spectrum, figure 6.14, has almost been completely combined in our truth estimate. Further analysis is performed on this spectrum and shown in figures 6.16 and 6.17. The results from OMpy and subsequent comparisons are shown in figures 6.18, 6.19 and in table 6.6.

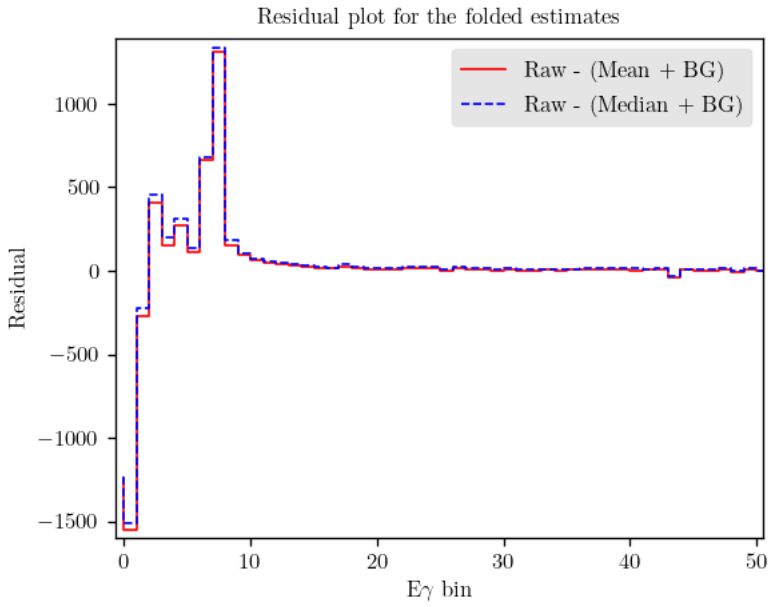


(a) (b)

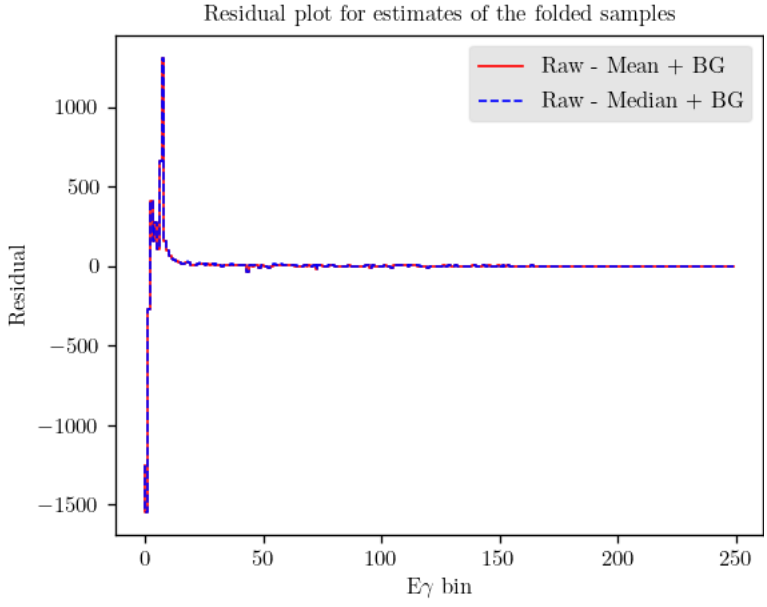
Figure 6.16: Refolding of the unfolded results compared with the raw observed data, in two ways. The first plot (a) shows the result of folding the truth-sample estimates (HPD, point estimates) shown in figure 6.15. The second plot (b) has been made by directly folding the truth-samples, and afterwards calculating a new HPD interval and point estimates based on the folded samples.



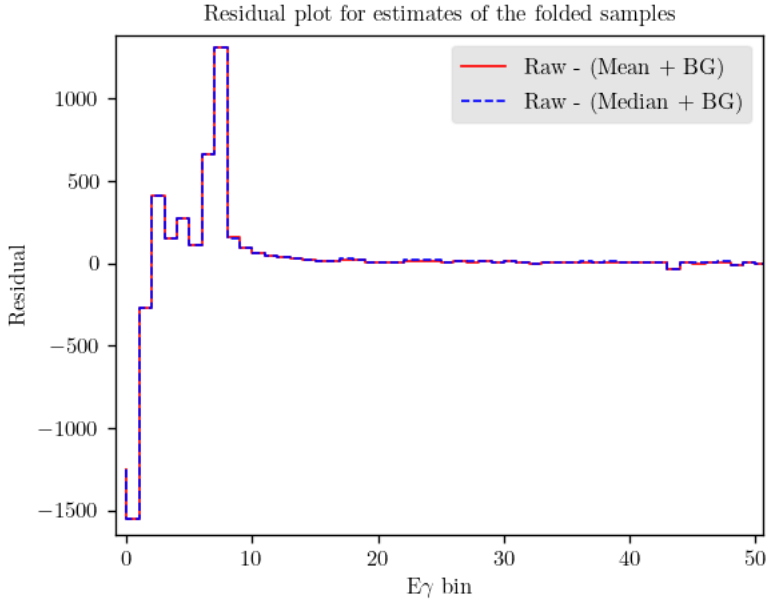
(a)



(b)

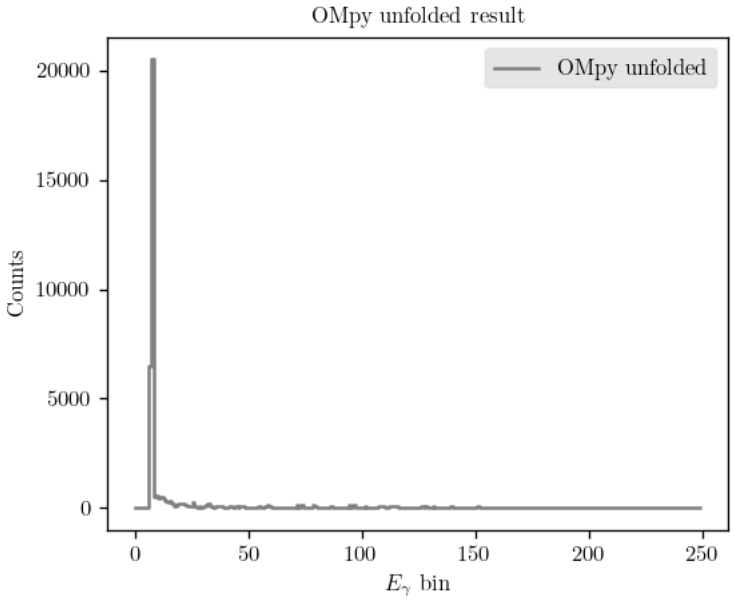


(c)

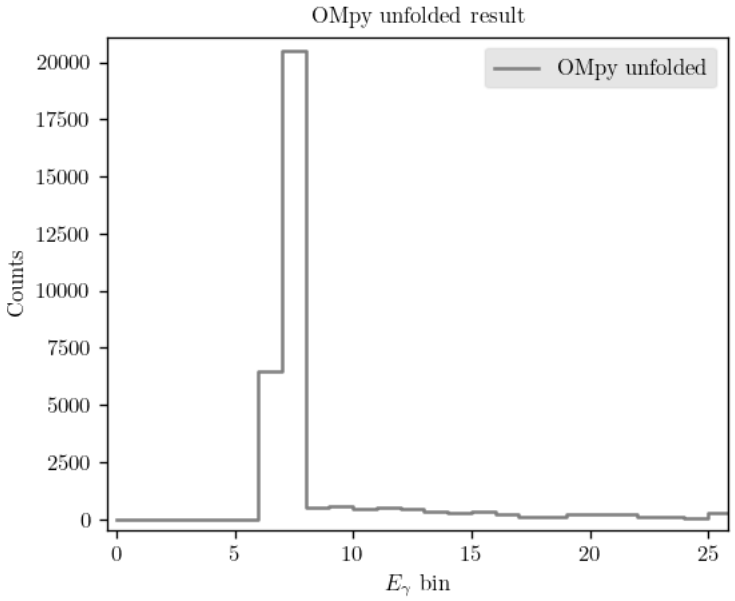


(d)

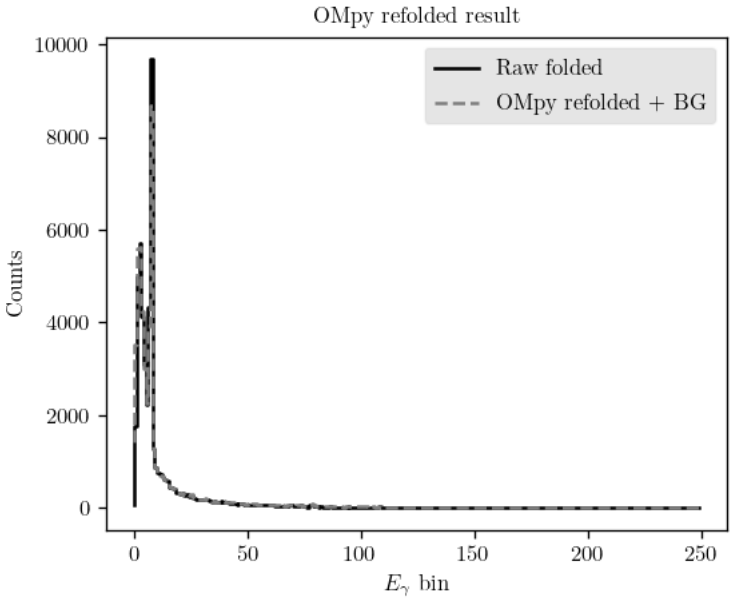
Figure 6.17: Residual plots showing the difference between the observed raw spectrum and the point estimates from FBU, first for all bins, then zoomed to the first 50 bins. Plots (a) and (b) pertain to the folded truth-sample estimates from figure 6.16a. Plots (c) and (d) are connected to the folded-sample estimates from figure 6.16b. These are very similar, but some differences appear in figure 6.19 and table 6.6.



(a)

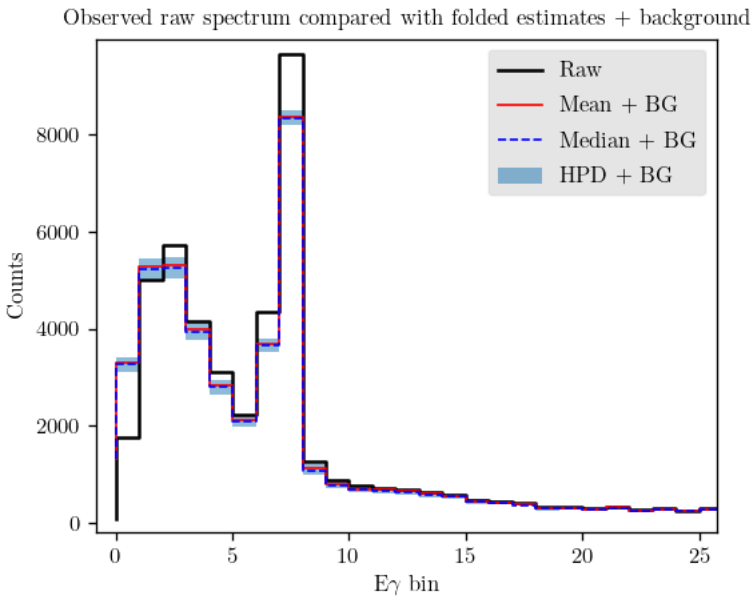


(b)

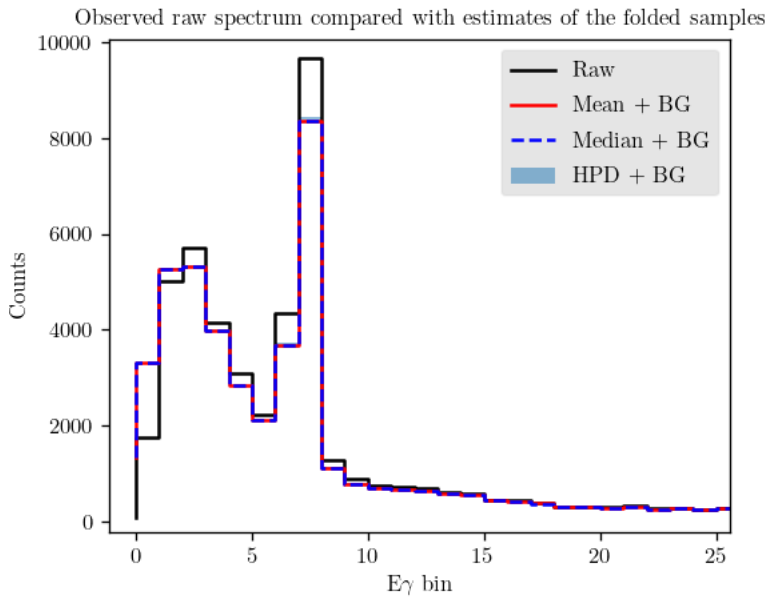


(c)

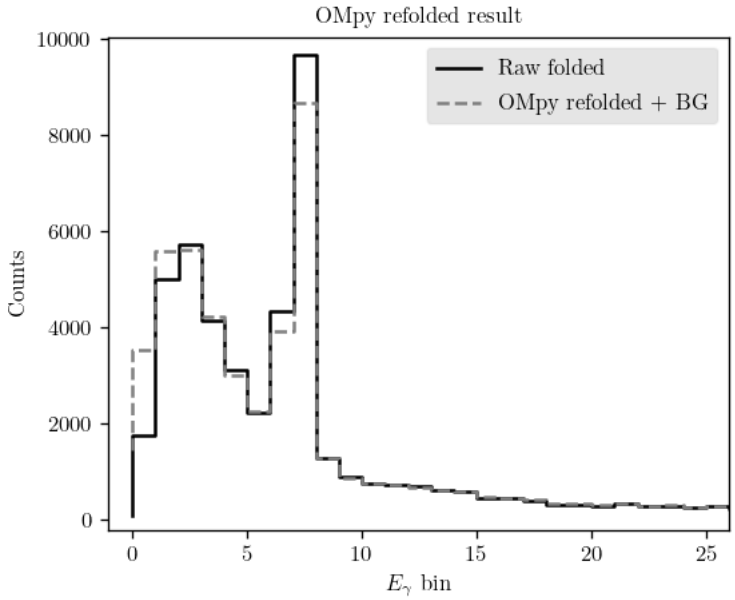
Figure 6.18: Result from unfolding using OMpy for all bins (a) and zoomed to the first 25 bins (b), as well as the subsequent refolded result compared with the observed spectrum (c). These results look similar to those from FBU, and a closer look will be shown in figure 6.19.



(a)



(b)



(c)

Figure 6.19: Zoomed versions of the refolded results (figures 6.16a 6.16b and 6.18c) comparing FBU with OMpy for the first 25 bins. First, we see one effect of folding the truth-samples estimates (a), the HPD interval has become larger than the interval calculated on the folded samples (b). Overall, all three plots are visually quite similar, with small variations per bin. We see a very similar discrepancy between refolded results and observed data for all three plots at the lower energy bins. This is actually due to a known error in the response matrix used Is this true?. With the correct information compensating for this error, we would be able to adjust the results and observe an even closer match. The error metrics are shown in table 6.6.

Table 6.3: Error metrics for refolded results (rounded). We see all estimates score very well with low mean absolute errors, and R^2-scores all above 0.95. As seen in figure 6.19, the largest discrepancies are located at the lowest energy bins, however these are not large enough to have a big impact on the overall scores, since all bins are weighted equally in the metrics. There is an argument to be made about weighting these bins more, as this is where the most interesting structures are located. Doing so would require a manual definition of what is deemed the most interesting area on a case-by-case basis. Since we either way have the residual plots to directly show local errors, we are content to leave the MAE and R^2-score as summarizing error metrics for the entire spectrum. These scores would be even better if we were to include the mentioned additional information about the error in the response matrix for low energies, see figure 6.19 Again, is this true?. The results from both FBU and OMpy are very close, with a slightly higher R^2-score for the FBU estimates, signifying a very slightly overall better match with the observed data. Furthermore, we see a very small, almost negligible improvement in the FBU results when using the estimates calculated on the **folded** samples (figure 6.19b, here abbreviated to Fold. sampl.), over the folded **truth**-sample estimates (6.19a, Fold. est.).

	FBU Median		**FBU Mean**		**OMpy**
	Fold. est.	Fold. sampl.	Fold. est.	Fold. sampl.	
MAE	30.8	28.9	30.8	29.2	26.0
R^2-score	0.968	0.968	0.968	0.968	0.955

Table 6.4

	True samples	**Folded samples**
Mean variance	3535.7	90.0

2.2 High excitation energies (6.0-6.2 MeV)

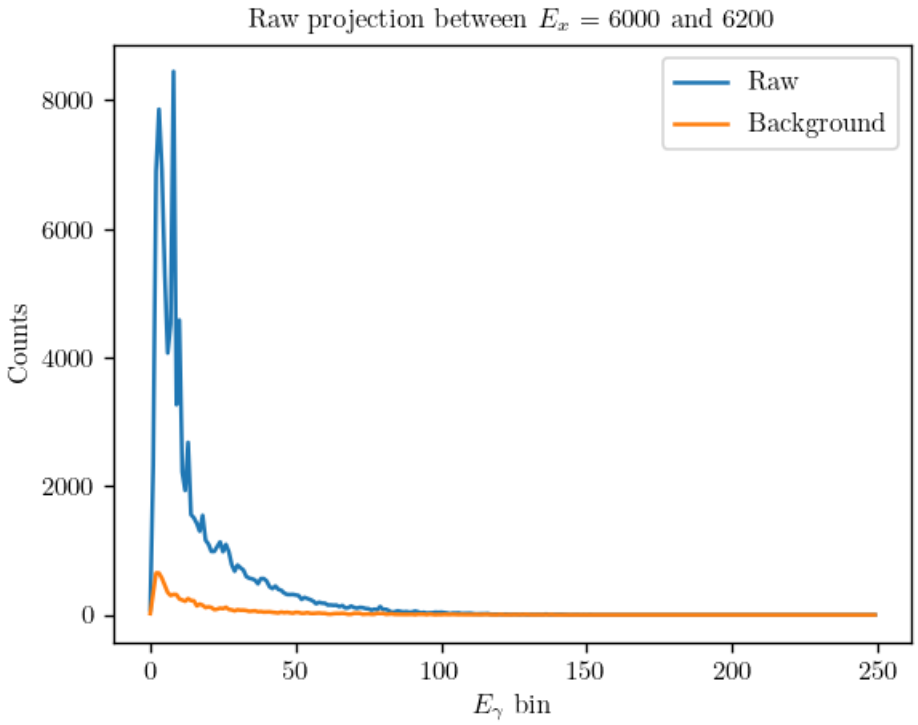
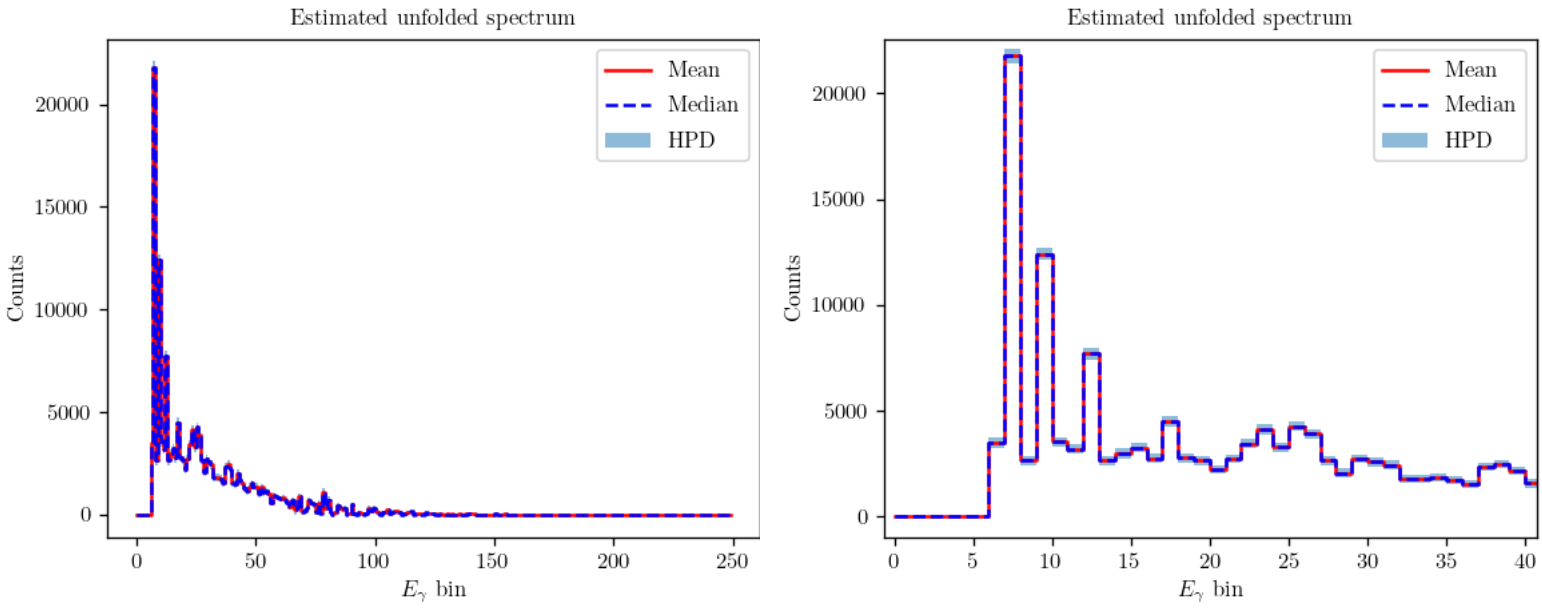
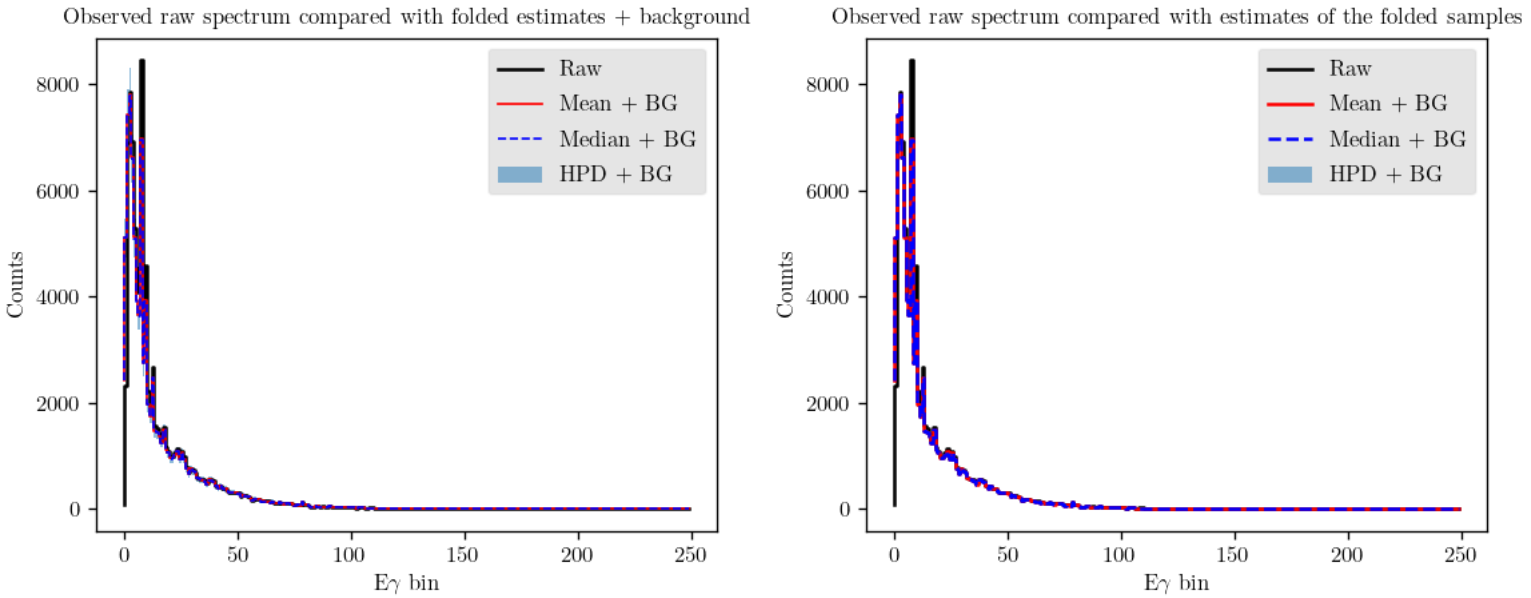


Figure 6.20: Projection of the raw spectrum and background for the high-energy area, i.e. between $E_x = 6000$ and 6200, see figure 6.11, showing observed counts in each energy bin.



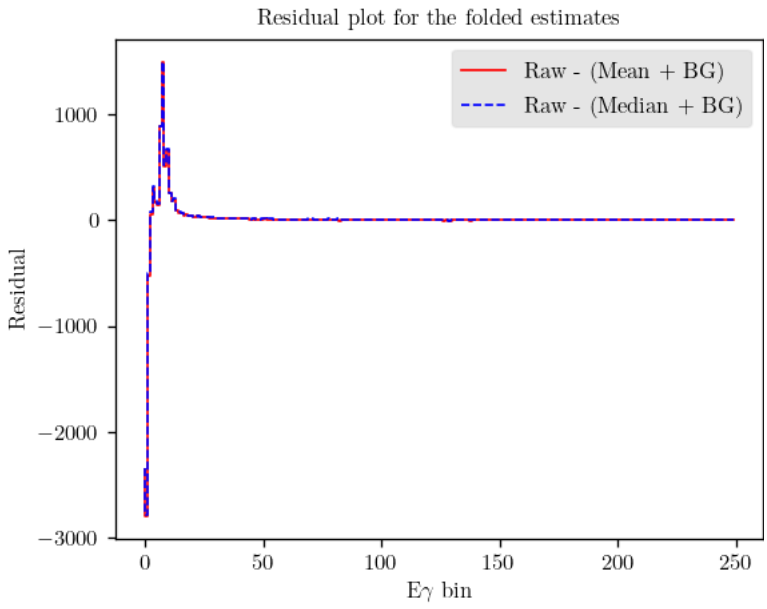
(a) (b)

Figure 6.21: The estimated unfolded spectrum from the raw spectrum in figure 6.20, for all bins (a) and zoomed to the first 40 bins (b). Here we see a more complex composition of peaks than for the first excited state. Further analysis is performed on this spectrum and shown in figures 6.22 and 6.23. The results from OMpy and subsequent comparisons are shown in figures 6.24, 6.25 and in table 6.5.

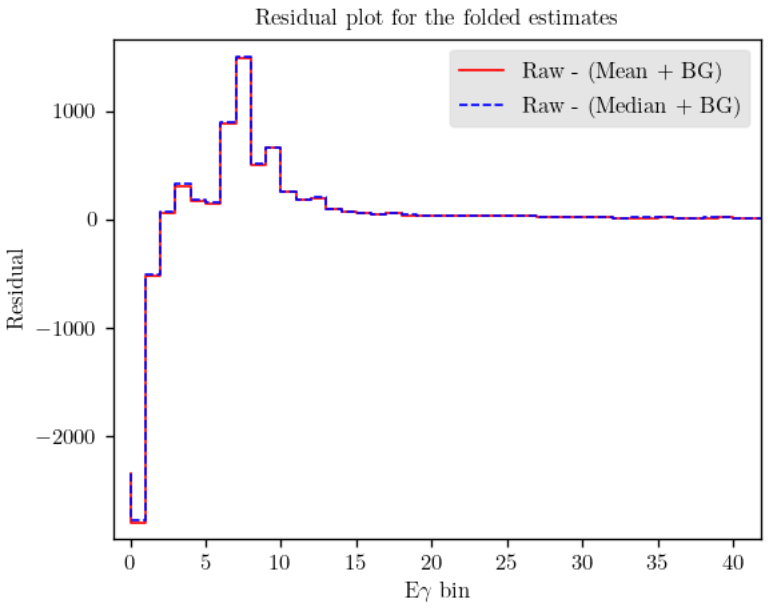


(a) (b)

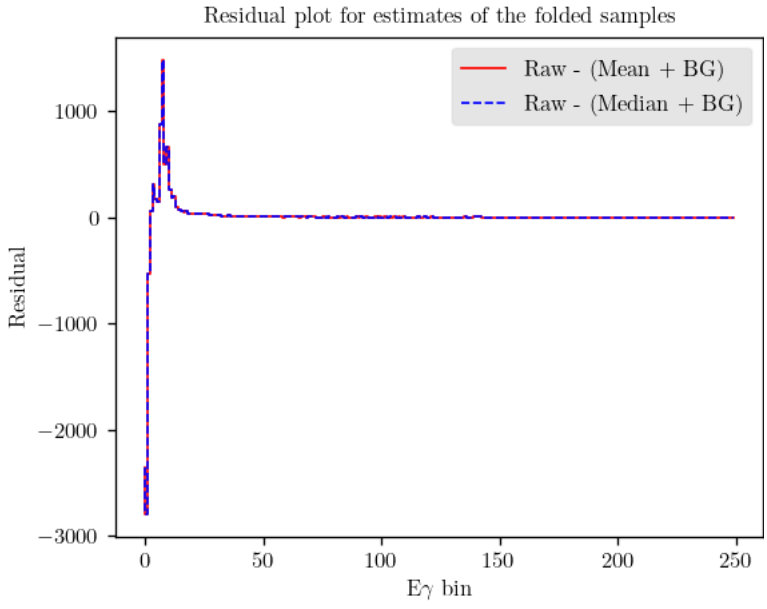
Figure 6.22: Refolding of the unfolded result compared with the raw observed data, in two ways. The first plot (a) shows the result of folding the truth-sample estimates (HPD, point estimates) shown in figure 6.21. The second plot (b) has been made by directly folding the truth-samples, and afterwards calculating a new HPD interval and point estimates based on the folded samples.



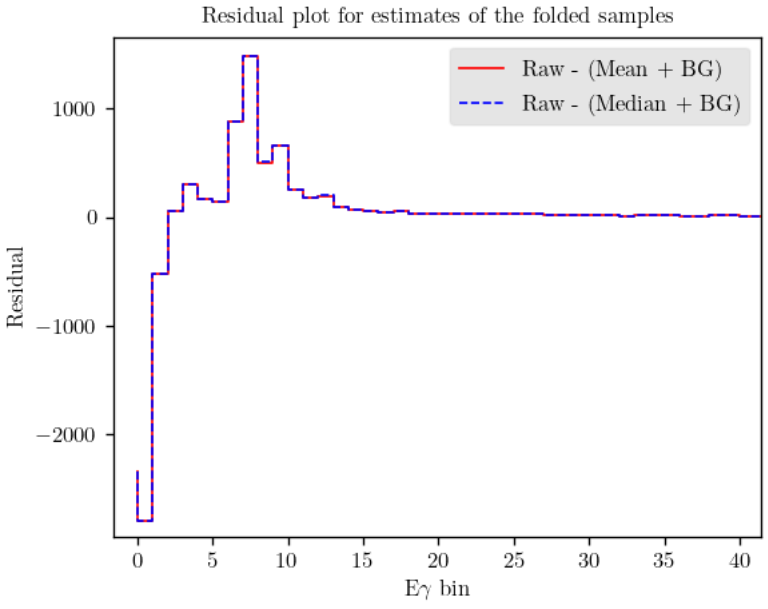
(a)



(b)

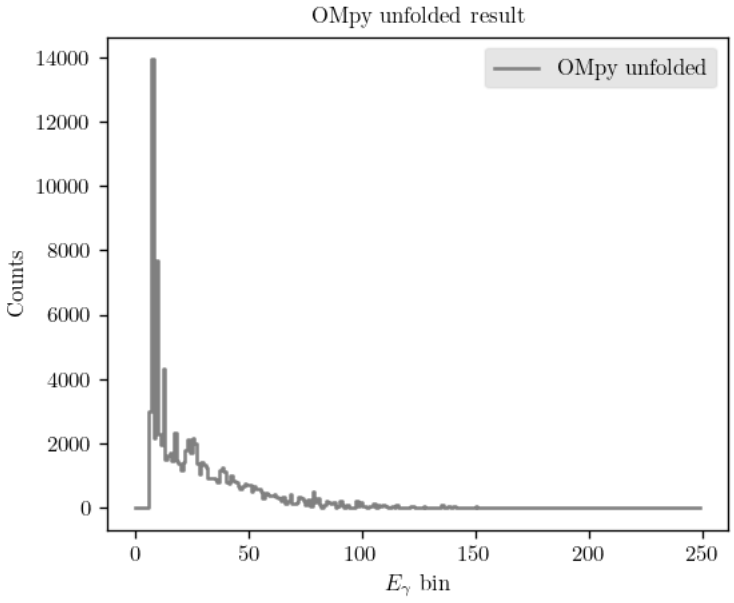


(c)

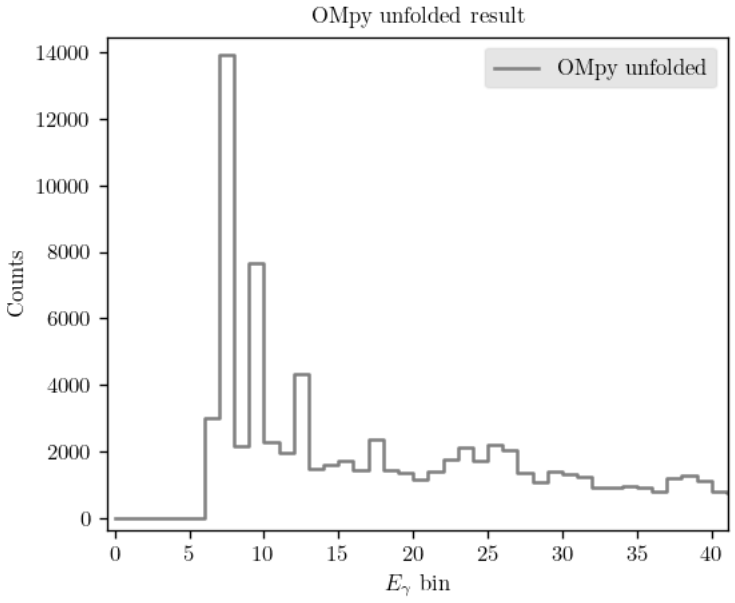


(d)

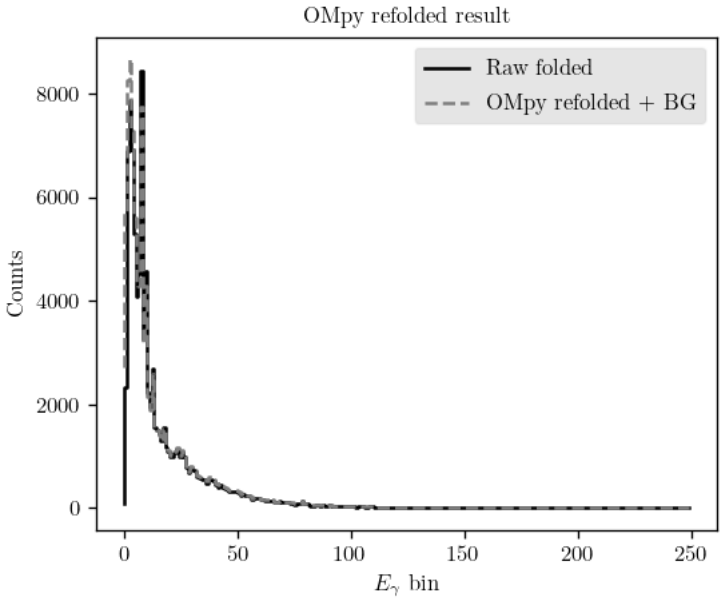
Figure 6.23: Residual plots showing the difference between the observed raw spectrum and the point estimates from FBU, first for all bins, then zoomed to the first 40 bins. Plots (a) and (b) pertain to the folded truth-sample estimates from figure 6.22a. Plots (c) and (d) are connected to the folded-sample estimates from figure 6.22b. Again we see no obvious visual differences between these, although some differences does appear in figure 6.25 and table 6.5.



(a)

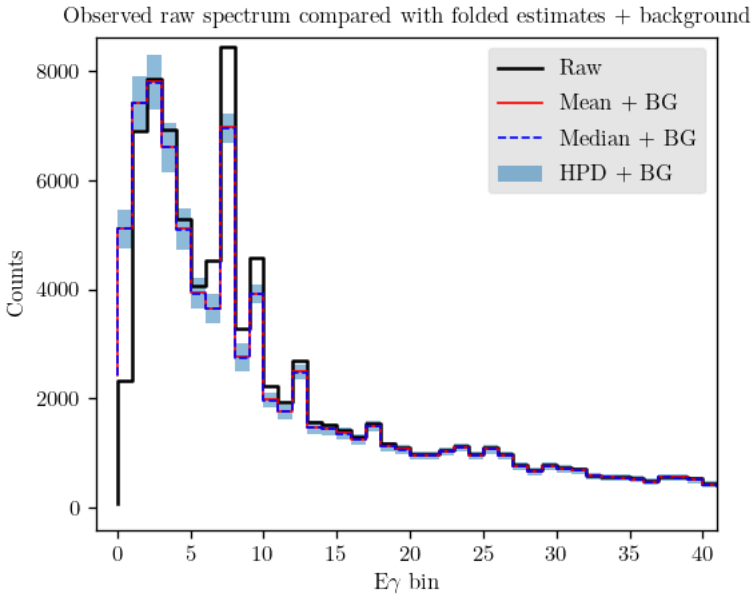


(b)

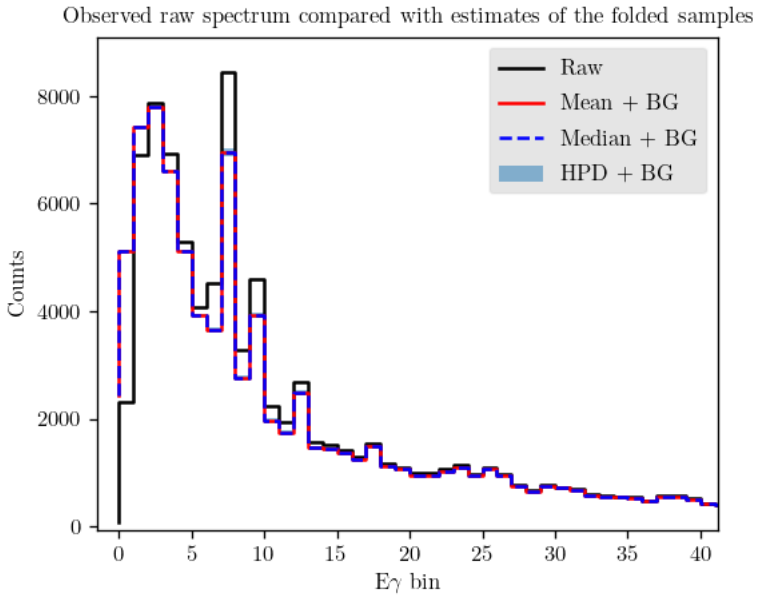


(c)

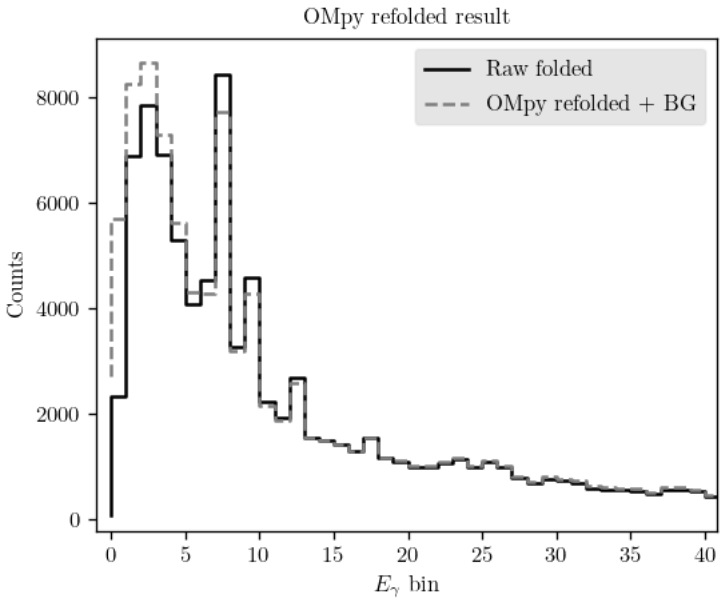
Figure 6.24: Result from unfolding using OMpy for all bins (a) and zoomed to the first 40 bins (b), as well as the subsequent refolded result compared with the observed spectrum (c). These results look similar to those from FBU, and a closer look will be shown in figure 6.25.



(a)



(b)



(c)

Figure 6.25: Zoomed versions of the refolded results (figures 6.22a 6.22b and 6.24c) comparing FBU with OMpy for the first 40 bins. We see again that the folded truth-sample HPD interval (a), has become larger than the interval calculated on the folded samples (b). We again observe similar results from FBU and OMpy, and the error metrics are shown in table 6.5.

Table 6.5: Error metrics for refolded results (rounded). Compared with the results for the simpler spectrum of the first excited state, table 6.6, we see a slight drop in accuracy. This is expected due to the increased complexity of the spectrum, however the accuracy remains very good with no R^2-scores below 0.9. As described in table 6.6, the largest discrepancies are located at a few low-energy bins, exhibiting the most interesting structures, while all other bins show a very close match between raw and refolded output. Since MAE and R^2 are overall metrics, such local errors may have a smaller impact on the score, but can still be observed directly in figure 6.23. Again, we see slightly better scores for FBU than OMpy, and the results from either using the folded truth-sample estimates (figure 6.25a, Fold. est.) or the estimates on the folded samples (6.25b, Fold. sampl.) are virtually identical.

	FBU Median		**FBU HPD Mean**		**OMpy**
	Fold. est.	Fold. sampl.	Fold. est.	Fold. sampl.	
MAE	48.1	47.3	47.9	47.5	49.3
R^2-score	0.948	0.948	0.948	0.948	0.936

Table 6.6: The mean variance for both the truth-samples output from FBU, and the same samples folded with the response.

	Truth-samples	**Folded samples**
Mean variance	12382.4	216.6

Part V

Conclusion and future work

3 Conclusion

Bibliography

1. D. S. Sivia, J. S. *Data analysis: A Bayesian Tutorial* 2nd ed. ISBN: 0–19–856831–2 (Oxford University Press, Great Clarendon Street, Oxford OX2 6DP, 2006).
2. Cowan, G. *A survey of unfolding methods for particle physics* in *Prepared for Conference on Advanced Statistical Techniques in Particle Physics, Durham, England* (2002), 18–22.
3. Guttormsen, M., Tveter, T., Bergholt, L., Ingebretsen, F. & Rekstad, J. The unfolding of continuum γ-ray spectra. *Nuclear instruments & methods in physics research. Section A, Accelerators, spectrometers, detectors and associated equipment* **374,** 371–376. ISSN: 0168-9002 (1996).
4. Choudalakis, G. *Fully Bayesian Unfolding* 2012. arXiv: `1201.4612 [physics.data-an]`.
5. Hoffman, M. D. & Gelman, A. *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo* 2011. arXiv: `1111.4246 [stat.CO]`.
6. Salvatier, J., Wiecki, T. V. & Fonnesbeck, C. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science* **2.** Publisher: PeerJ Inc., e55. ISSN: 2376-5992. `https://peerj.com/articles/cs-55` (2021) (Apr. 6, 2016).
7. Kucukelbir, A., Ranganath, R., Gelman, A. & Blei, D. M. *Automatic Variational Inference in Stan* 2015. arXiv: `1506.03431 [stat.ML]`.
8. Jaynes, E. T. *Probability Theory: The Logic of Science* (ed Bretthorst, G. L.) (Cambridge University Press, 2003).
9. Lee, P. M. *Bayesian statistics : an introduction* eng. Chichester, West Sussex ; 2012.

10. Edwards, W., Lindman, H. & Savage, L. J. Bayesian statistical inference for psychological research. *Psychological Review* **70.** Publisher: US: American Psychological Association, 193. ISSN: 1939-1471. http://psycnet.apa.org/fulltext/1964-00040-001.pdf (1964).

11. Cort J. Willmott & Kenji Matsuura. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research* **30,** 79–82. https://www.int-res.com/abstracts/cr/v30/n1/p79-82 (2005).

12. *The Coefficient of Determination* [Online; accessed 2021-06-25]. Jan. 11, 2021. https://stats.libretexts.org/@go/page/553.

13. Gerbaudo, D., Helsens, C. & Rubbo, F. *PyFBU* https://github.com/pyFBU/fbu.

14. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* **abs/1605.02688.** http://arxiv.org/abs/1605.02688 (May 2016).

15. Zeiser, F. & Tveten, G. M. *oslocyclotronlab/OCL_GEANT4: Geant4 model of OSCAR* version v1.0.3. Aug. 2018. https://doi.org/10.5281/zenodo.1339347.

16. Zeiser, F. *et al. The energy response of the Oslo Scintillator Array OSCAR* 2020. arXiv: 2008.06240 [physics.ins-det].

17. Valsdóttir, V. M. *Exploring Fully Bayesian Unfolding for γ-ray Spectra* MA thesis (University of Oslo, Oslo, 2020).

18. Valsdóttir, V. M. *Exploring Fully Bayesian Unfolding for γ-ray Spectra - GitHub repository* https://github.com/valamaria89/Exploring-Fully-Bayesian-Unfolding-for-gamma-ray-Spectra (2020).

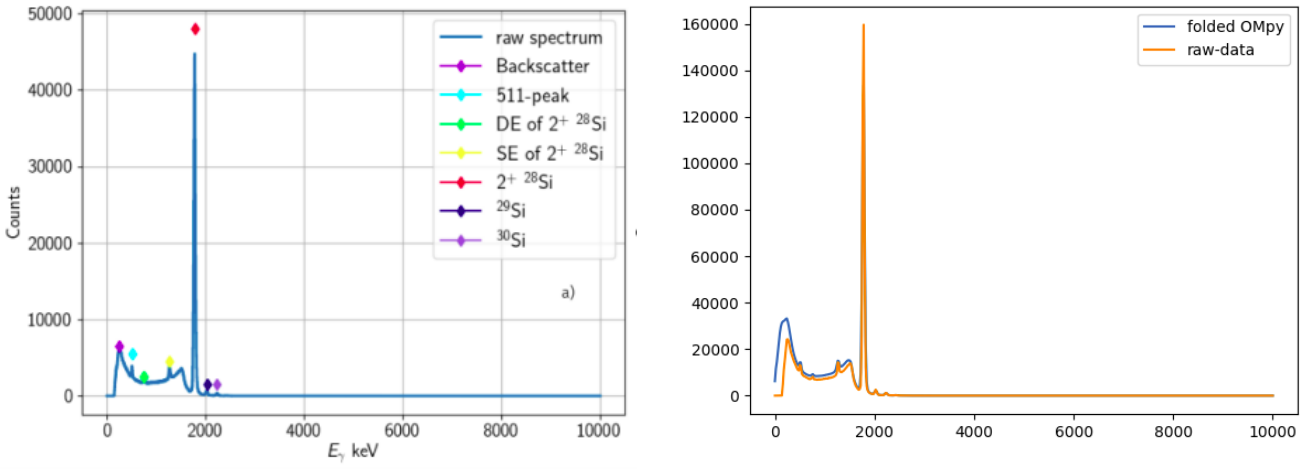
APPENDIX **A**

Reproduction of results

Under follows a reproduction of the results achieved by Valsdóttir [17]. This procedure is done to verify the results using the same methods.

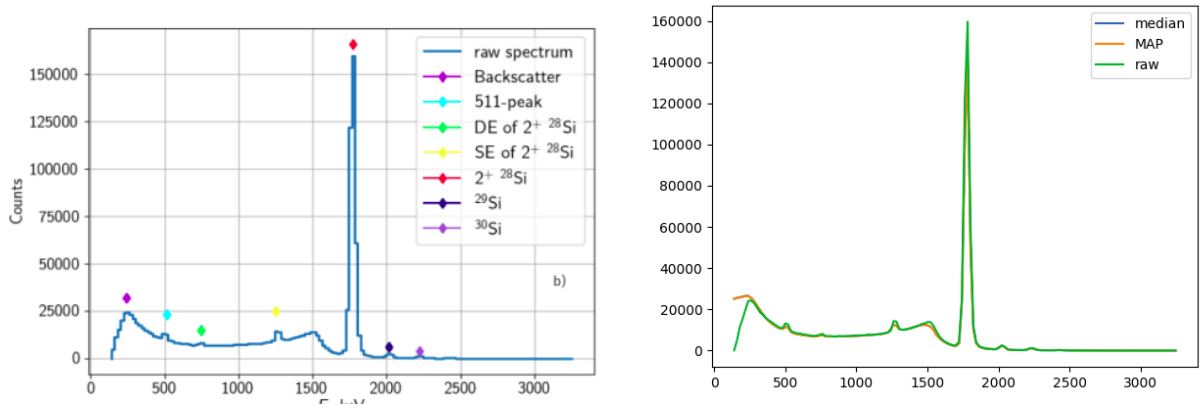
The figures presented as reproductions are outputs of the publicly available Jupyter notebooks on Valsdóttirs GitHub repository [18]. Some discrepancies are seen, pointing to the possibility that the results in Valsdóttirs thesis may stem from newer, locally stored versions of the files that have not been made accessible on the repository. Therefore, the below sections contain only the results for which corresponding output was found to be produced in the mentioned Jupyter notebooks.

0.1 Fully Bayesian Unfolding Spectrum From the First Excited state of ^{28}Si



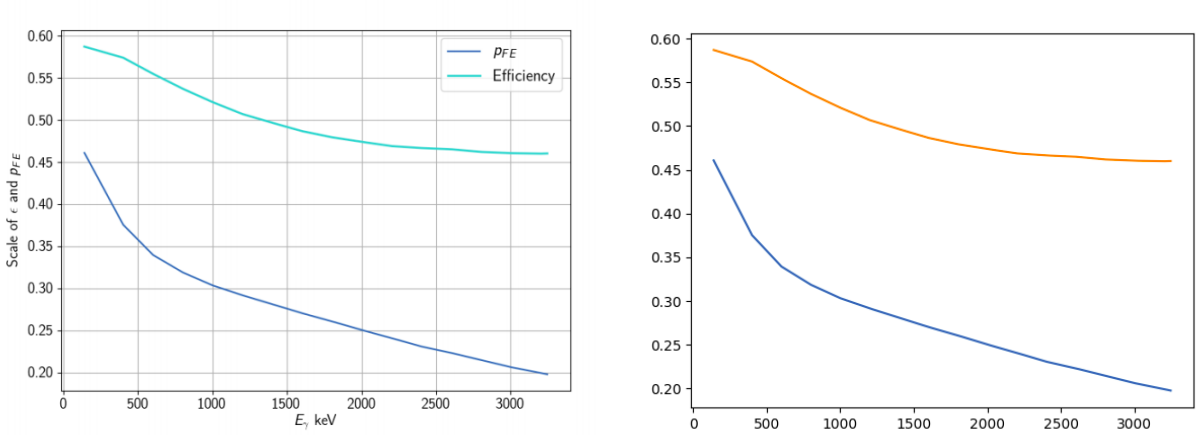
(a) Valsdóttir (b) Reproduction

Figure A.1



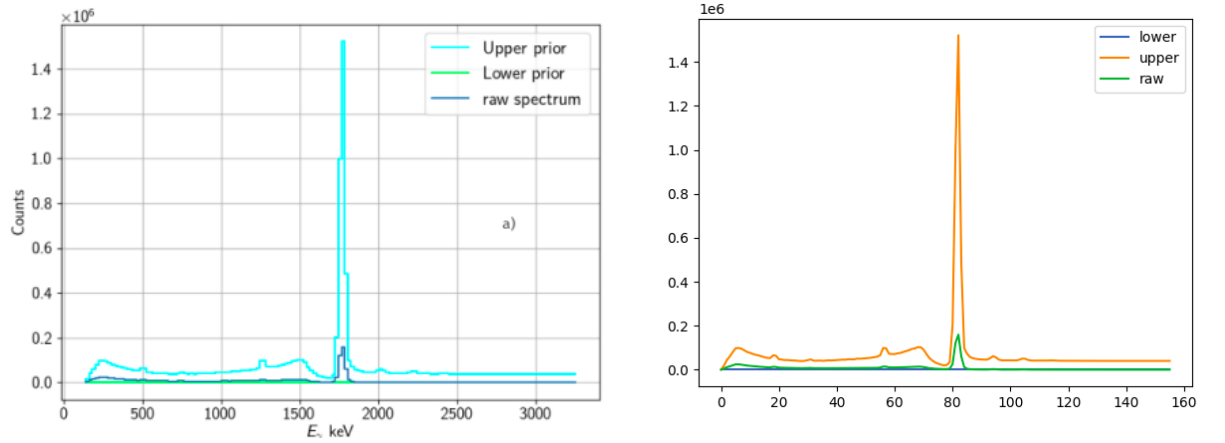
(a) Valsdóttir (b) Reproduction

Figure A.2



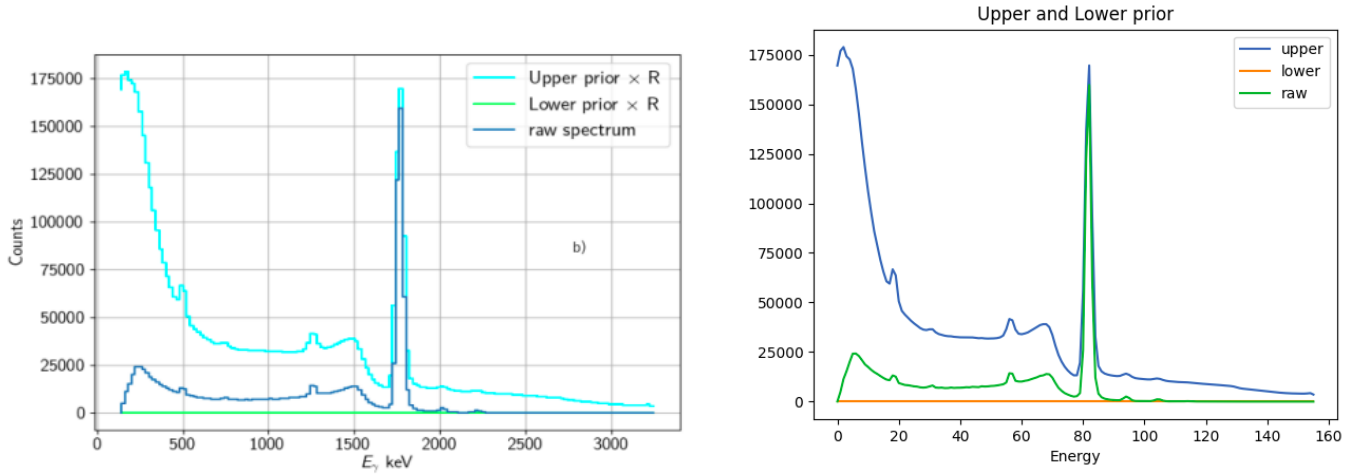
(a) Valsdóttir (b) Reproduction

Figure A.3



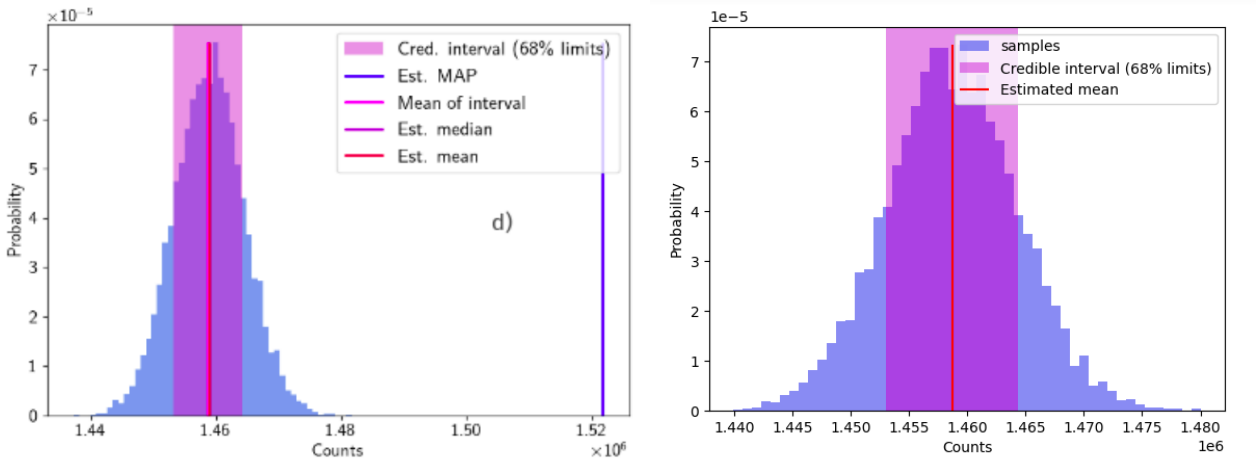
(a) Valsdóttir (b) Reproduction

Figure A.4



(a) Valsdóttir (b) Reproduction

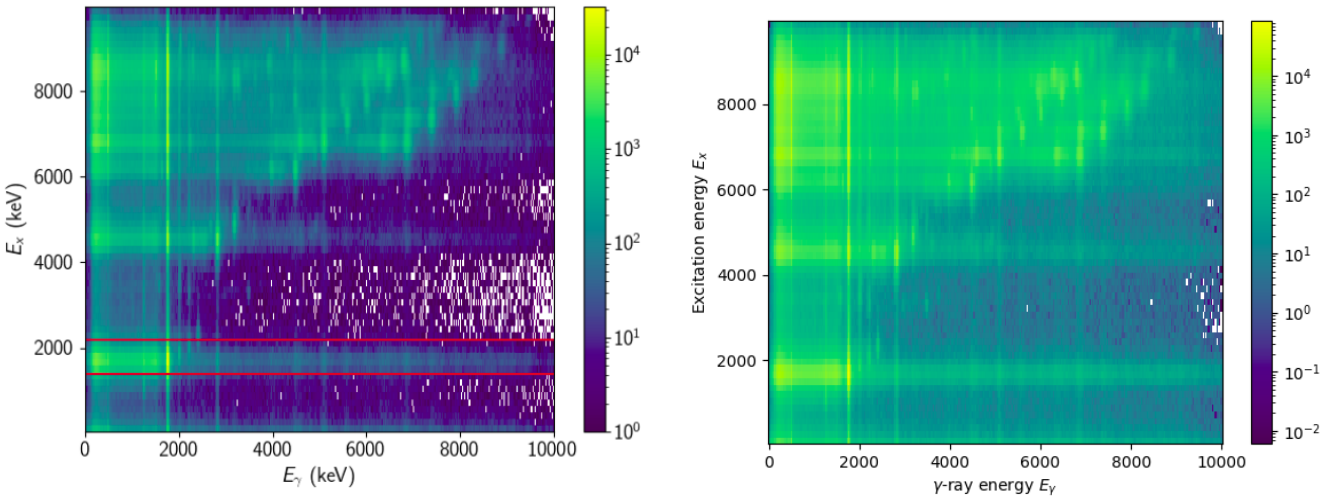
Figure A.5



(a) Valsdóttir (b) Reproduction

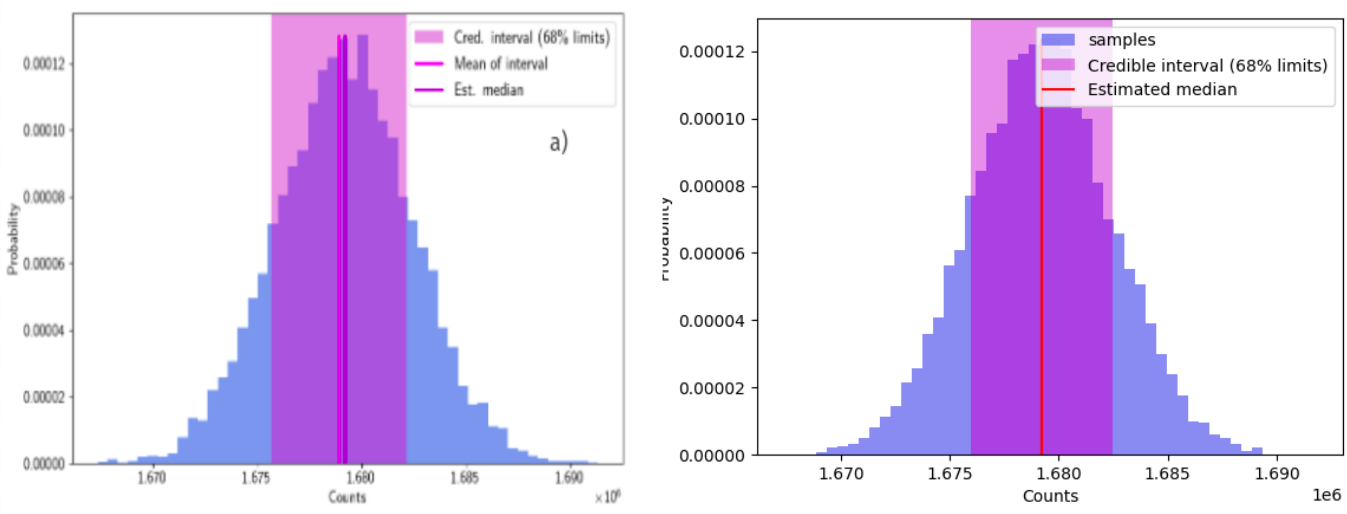
Figure A.6

0.2 Fully Bayesian Unfolding on first excited state of ^{28}Si including background



(a) Valsdóttir (b) Reproduction

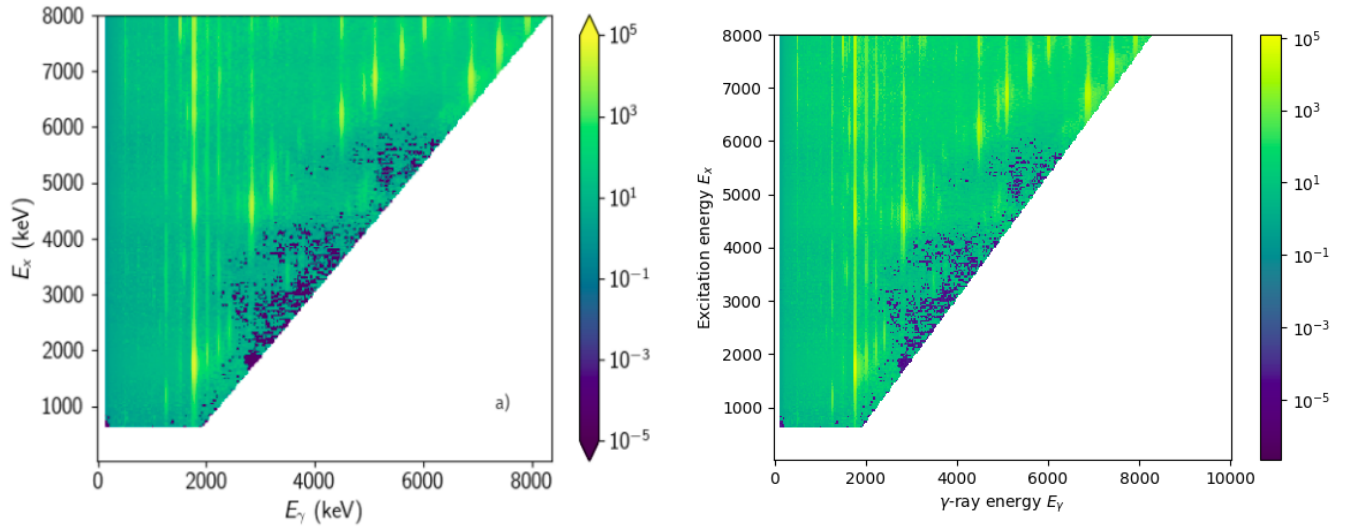
Figure A.7



(a) Valsdóttir (b) Reproduction

Figure A.8

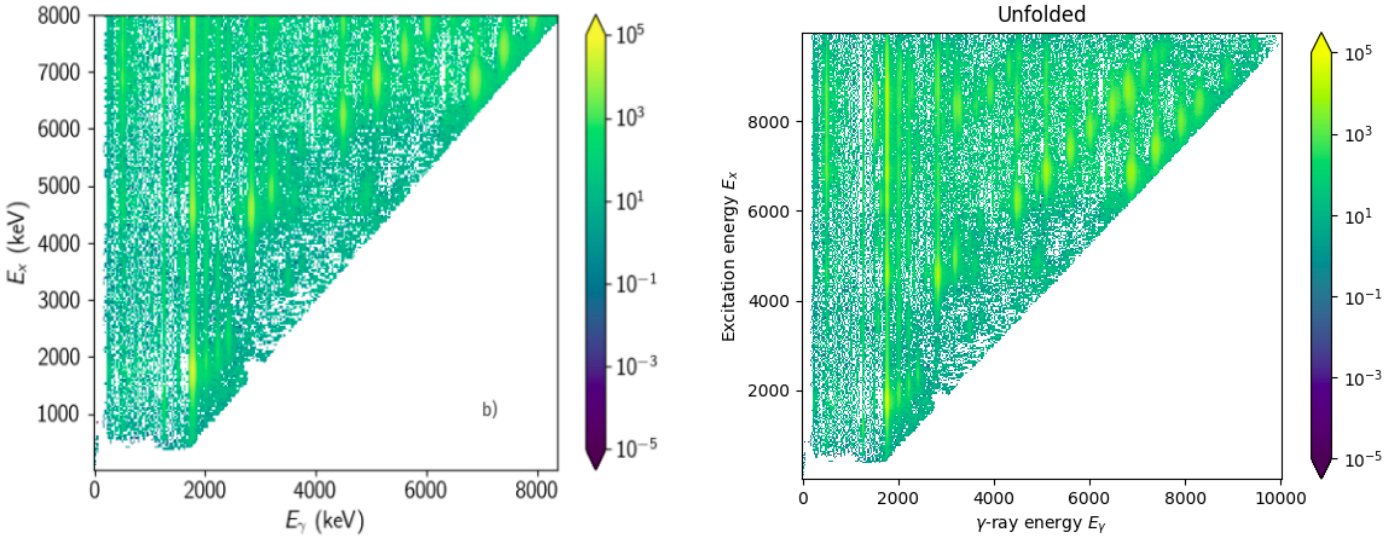
0.3 Fully Bayesian Unfolding for all Excited States



(a) Valsdóttir (b) Reproduction

Figure A.9

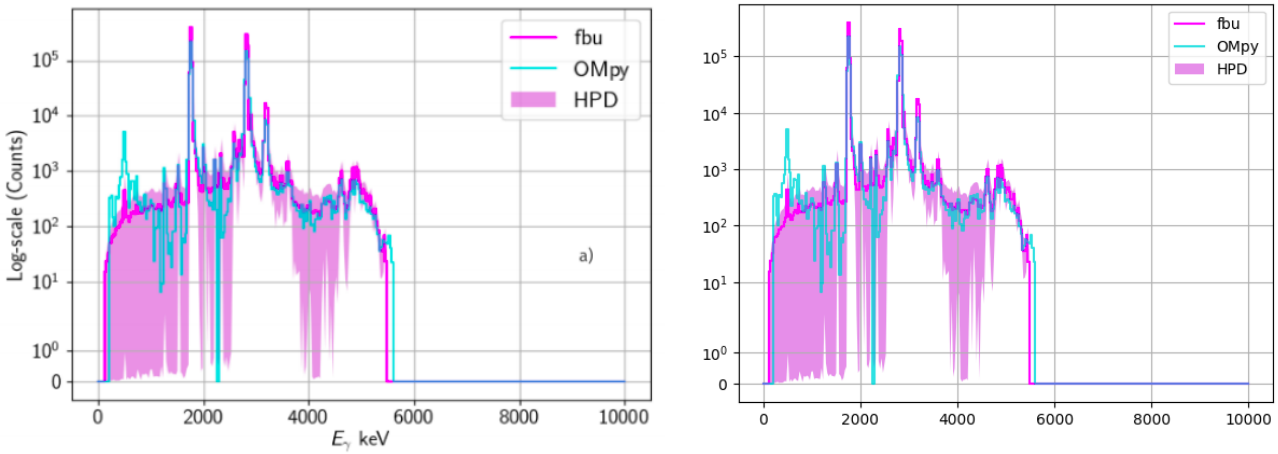
Appendix A. Reproduction of results



(a) Valsdóttir

(b) Reproduction

Figure A.10

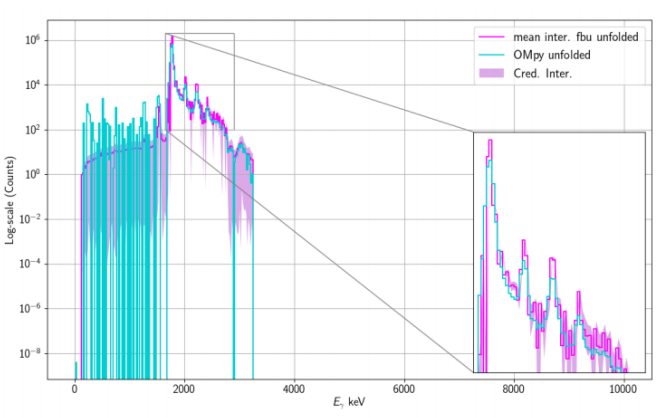


(a) Valsdóttir

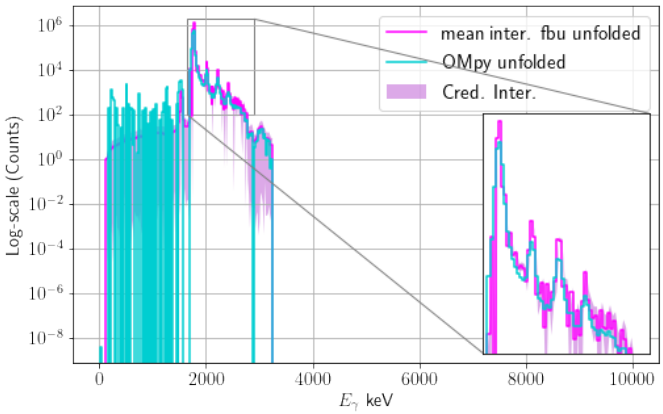
(b) Reproduction

Figure A.11

Appendix A. Reproduction of results

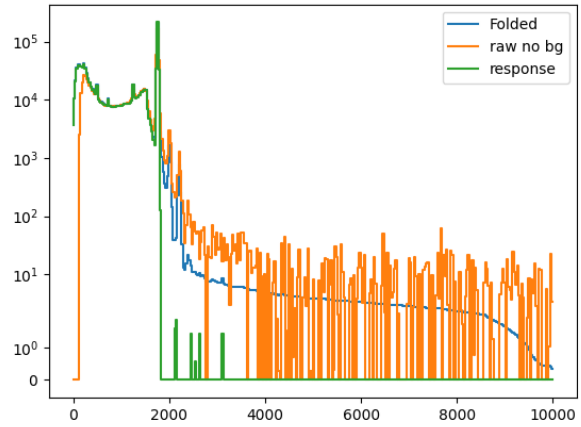


(a) Valsdóttir

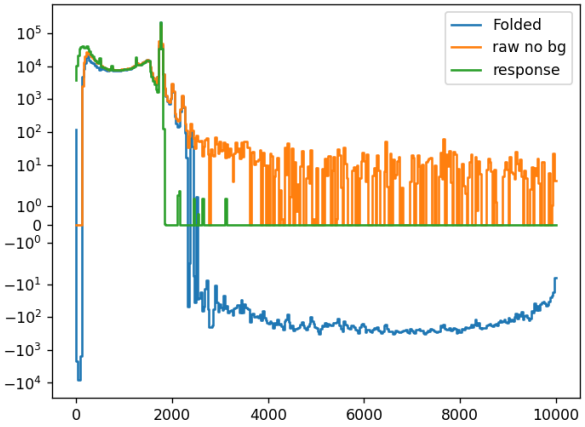


(b) Reproduction

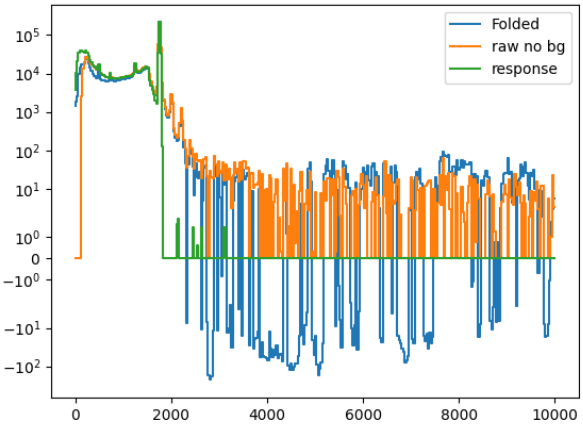
Figure A.12



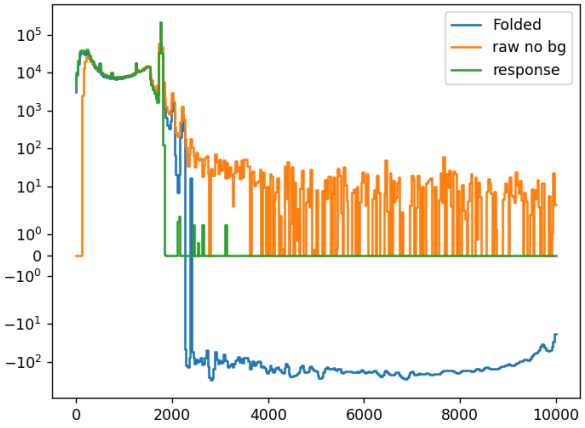
(a) Prior lower bound = raw/10



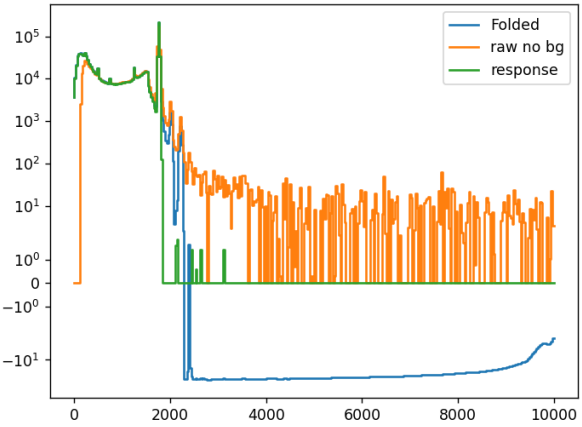
(b) Prior lower bound = raw/50



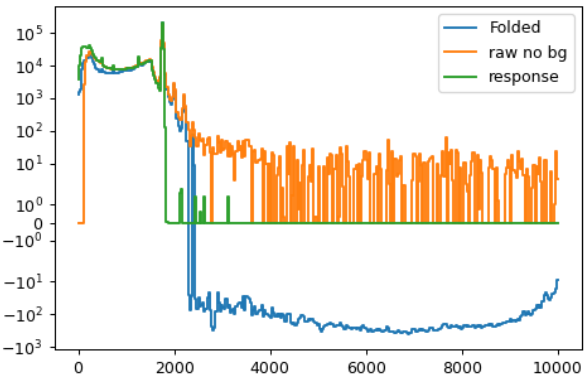
(c) Prior lower bound = raw/100



(d) Prior lower bound = raw/1000



(e) Prior lower bound = raw−100



(f) Prior lower bound = raw−20000

Figure A.13