Evaluating Multi-Dimensional Fitting Methods: Joint Maximum Likelihood vs Weighted *sWeights* Fits

Shahab Yousef-Nasiri (sy475)
Department of Physics, University of Cambridge
Word Count: 2989

Abstract

This study evaluates two fitting techniques for parameter estimation in a two-dimensional dataset; (1) a joint extended likelihood fit performed simultaneously across both variables X and Y, and (2) a sequential weighted fit using *sWeights*, which decouples the fitting process. The signal is modeled as a truncated Crystal Ball function in X and an exponential decay in Y, alongside a background component. Using `iminuit`, extended likelihood fits were performed with well-defined parameter bounds and initial guesses for optimization. A parametric bootstrap study was conducted to assess the bias and uncertainty of the estimated decay constant λ using both techniques. The joint fit demonstrated negligible bias and reduced uncertainties as the sample size increased, leveraging the full two-dimensional information. Conversely, the *sWeights* method, while computationally efficient, introduced a small positive bias due to its decoupling assumption. These findings highlight the trade-offs between precision, computational efficiency, and the treatment of multidimensional data in parameter estimation tasks.

Contents

1 Introduction

The estimation of model parameters from experimental data is a foundational task in statistical modeling. In this study, we evaluate two fitting techniques for multi-dimensional likelihood estimation: a joint maximum likelihood fit that utilizes the full two-dimensional probability density function (PDF), and a sequential weighted fit using *sWeights*, which decouples the fitting process into two stages.

The model under investigation comprises a signal component and a background component in two dimensions, $X \in [0, 5]$ and $Y \in [0, 10]$. The signal is modeled using a truncated Crystal Ball distribution in X and an exponential decay in Y. The background components are modeled by a uniform distribution in X and a truncated Gaussian distribution in Y.

1.1 Joint Probability Density Function

The joint PDF for the signal and background components is expressed as:

$$f(X, Y) = f_s \cdot g_s(X)h_s(Y) + (1 - f_s) \cdot g_b(X)h_b(Y), \tag{1}$$

where:

- f_s: Signal fraction, defining the proportion of the total probability density attributed to the signal.
- $g_s(X)$: The signal PDF in X, defined by a truncated Crystal Ball distribution over $[0, 5]$ (detailed in Section 1.2).
- $h_s(Y)$: The signal PDF in Y, defined by an exponential decay truncated over $[0, 10]$:

$$h_s(Y) = \frac{\lambda e^{-\lambda Y}}{1 - e^{-10\lambda}}, \quad Y \in [0, 10], \tag{2}$$

  where the denominator $1 - e^{-10\lambda}$ ensures proper normalization over the range.

- $g_b(X)$: The background PDF in X, modeled as a uniform distribution over $[0, 5]$:

$$g_b(X) = \frac{1}{5}, \quad X \in [0, 5]. \tag{3}$$

- $h_b(Y)$: The background PDF in Y, defined by a truncated Gaussian distribution with mean μ_b and standard deviation σ_b, over $[0, 10]$:

$$h_b(Y) = \frac{1}{\sigma_b\sqrt{2\pi}} e^{-\frac{(Y-\mu_b)^2}{2\sigma_b^2}} \cdot \frac{1}{\Phi\left(\frac{10-\mu_b}{\sigma_b}\right) - \Phi\left(\frac{0-\mu_b}{\sigma_b}\right)}, \tag{4}$$

  where $\Phi(x)$ is the cumulative distribution function (CDF) of the standard normal distribution.

This formulation ensures proper normalization of the joint PDF over the truncated regions for both the signal and background components.

1.2 Crystal Ball Probability Distribution

The Crystal Ball distribution consists of a Gaussian core and a power-law tail. It is defined as:

$$g_s(X; \mu, \sigma, \beta, m) = N \cdot \begin{cases} e^{-Z^2/2}, & \text{for } Z > -\beta, \\ \left(\frac{m}{\beta}\right)^m e^{-\beta^2/2} \left(\frac{m}{\beta} - \beta - Z\right)^{-m}, & \text{for } Z \leq -\beta, \end{cases} \tag{5}$$

where:

- $Z = \frac{X-\mu}{\sigma}$: The normalized variable.

- μ: Location parameter, controlling the center of the Gaussian core.
- σ: Scale parameter, determining the width of the Gaussian core.
- β: Transition point between the Gaussian core and the power-law tail.
- m: Slope of the power-law tail.

The distribution is valid for $\beta > 0$ and $m > 1$, ensuring a well-defined transition and finite normalization.

1.2.1 Normalization Constant

To ensure that the Crystal Ball PDF integrates to unity, the normalization constant N is derived from the following condition:

$$\int_{-\infty}^{\infty} p(X; \mu, \sigma, \beta, m) dX = 1.$$

Using the change of variable $Z = \frac{X-\mu}{\sigma}$, we have $dX = \sigma\, dZ$. Substituting this into the integral:

$$\int_{-\infty}^{\infty} p(X; \mu, \sigma, \beta, m) dX = N \int_{-\infty}^{\infty} p(Z; \beta, m) \sigma\, dZ = 1.$$

Rearranging for N, the normalization constant is given by:

$$N^{-1} = \sigma \int_{-\infty}^{\infty} p(Z; \beta, m) dZ. \tag{6}$$

Splitting the Integral: The PDF $p(Z; \beta, m)$ is piecewise-defined, allowing us to split the integral into two parts:

$$N^{-1} = \sigma \left[I_{\text{Gaussian}} + I_{\text{Power-law}} \right], \tag{7}$$

where:

- I_{Gaussian} accounts for the Gaussian core ($Z > -\beta$).
- $I_{\text{Power-law}}$ accounts for the power-law tail ($Z \leq -\beta$).

Gaussian Core Contribution: For $Z > -\beta$, the Gaussian probability density function (PDF) simplifies to:

$$p(Z; \beta, m) = e^{-Z^2/2}.$$

To compute the contribution over the region $Z \in [-\beta, \infty)$, we evaluate the integral:

$$I_{\text{Gaussian}} = \int_{-\beta}^{\infty} e^{-Z^2/2}\, dZ.$$

Using the property $\Phi(-\beta) = 1 - \Phi(\beta)$, we find:

$$I_{\text{Gaussian}} = \int_{-\infty}^{\infty} e^{-Z^2/2}\, dZ - \int_{-\infty}^{-\beta} e^{-Z^2/2}\, dZ = \sqrt{2\pi} - \left(\sqrt{2\pi} - \sqrt{2\pi}\Phi(\beta) \right) = \sqrt{2\pi}\Phi(\beta).$$

Thus, the Gaussian core contribution is:

$$I_{\text{Gaussian}} = \sqrt{2\pi}\Phi(\beta). \tag{8}$$

Power-Law Tail Contribution: For $Z \leq -\beta$, the PDF is defined as:

$$p(Z;\beta,m) = \left(\frac{m}{\beta}\right)^m e^{-\beta^2/2} \left(\frac{m}{\beta} - \beta - Z\right)^{-m}.$$

Using the substitution $u = \frac{m}{\beta} - \beta - Z$, such that $dZ = -du$, the limits transform as:

$$Z \to -\beta \implies u = \frac{m}{\beta}, \quad Z \to -\infty \implies u \to \infty.$$

The integral becomes:

$$I_{\text{Power-law}} = \int_{-\infty}^{-\beta} p(Z;\beta,m)dZ = \left(\frac{m}{\beta}\right)^m e^{-\beta^2/2} \int_{\infty}^{m/\beta} u^{-m}(-du).$$

Evaluating the integral:

$$\int_{m/\beta}^{\infty} u^{-m}du = \frac{u^{1-m}}{1-m}\bigg|_{m/\beta}^{\infty} = \frac{\left(\frac{m}{\beta}\right)^{1-m}}{m-1}.$$

Thus:

$$I_{\text{Power-law}} = \frac{m}{\beta(m-1)} e^{-\beta^2/2}. \tag{9}$$

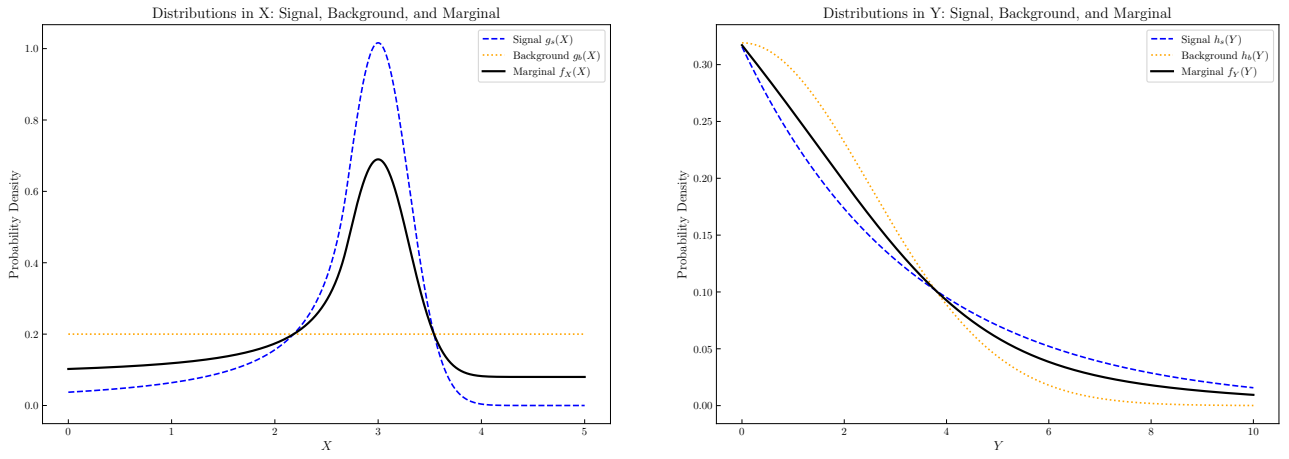
Combining Results: The normalization constant is:

$$N^{-1} = \sigma \left[\sqrt{2\pi}\Phi(\beta) + \frac{m}{\beta(m-1)} e^{-\beta^2/2}\right]. \tag{10}$$

This ensures the PDF is properly normalized over the entire domain.

1.3 Distribution Plots

To better understand the behavior of the model components, we present the marginal distributions in X and Y, along with their respective signal and background components, in a single plot.



(a) Marginal distribution in X.

(b) Marginal distribution in Y.

Figure 1: Marginal distributions in X and Y, showing their respective signal and background components.

2 Methodology

2.1 Distributions

Correct implementation of the probability density functions (PDFs) for the signal and background components in both dimensions is critical for ensuring computational efficiency and accuracy. The following techniques were employed in defining and implementing the distributions:

Precomputing Truncation Normalization Constant For the signal component in X, the Crystal Ball distribution requires a truncation normalization constant to ensure the PDF integrates to unity over the truncated range. Instead of recalculating this constant during each evaluation, it is precomputed using numerical integration over the interval $X \in [0, 5]$. This approach significantly reduces runtime computational overhead, especially when the PDF is evaluated repeatedly during sampling and fitting.

Vectorization and JIT Compilation All distributions were implemented using vectorized operations to handle arrays of inputs efficiently. The use of Just-In-Time (JIT) compilation via `Numba` further optimizes the evaluation of PDFs by compiling Python functions to machine code at runtime [1]. This combination ensures that PDF evaluations are both fast and scalable, even for large datasets.

Error Function Approximation For the truncated Gaussian background in Y, the normalization constant involves the cumulative distribution function (CDF) of the Gaussian, which depends on the error function (erf). To ensure compatibility with JIT compilation and avoid incompatibilities with external libraries, an approximate error function (erf) was implemented using a polynomial expansion [2].

Normalization Testing To validate the correctness of the implementations, integrals of the PDFs were numerically computed over their respective domains. The results were checked to ensure they sum to unity, confirming that the normalization constants were correctly applied.

2.2 Sampling Techniques

The sampling process for generating data points is divided into two categories: signal and background.

2.2.1 Signal Sampling

Signal samples are drawn from $g_s(X)$ (Crystal Ball distribution) and $h_s(Y)$ (truncated exponential distribution). The sampling technique employed is **inverse transform sampling**, which involves [3]:

1. Compute the cumulative distribution function (CDF) of the target distribution.
2. Generate uniform random numbers $U \sim \text{Uniform}(0, 1)$.
3. Map U to the CDF to obtain samples X by solving $F(X) = U$. For simple distributions, this inversion can be done analytically; for complex ones, numerical methods are employed.

Sampling in Y: Truncated Exponential Distribution For the truncated exponential distribution $h_s(Y)$, the CDF is available in closed form:

$$F(Y) = \frac{1 - e^{-\lambda_s Y}}{1 - e^{-\lambda_s \cdot 10}}, \quad Y \in [0, 10].$$

The inverse of this CDF is given analytically as:

$$Y = -\frac{\ln(1 - U \cdot (1 - e^{-\lambda_s \cdot 10}))}{\lambda_s}, \quad U \sim \text{Uniform}(0, 1).$$

This direct inversion avoids numerical approximations and avoids the computational inefficiency of rejection sampling, guaranteeing exact samples from the truncated exponential distribution.

Sampling in X: Truncated Crystal Ball Distribution The Crystal Ball distribution $g_s(X)$, due to its more complex functional form, requires a numerical approach for sampling. The steps are as follows:

- **Numerical CDF Construction**: The CDF is computed by numerically integrating the Crystal Ball PDF over a finely spaced grid of X values within the truncation bounds $[0, 5]$. This is achieved using methods like the trapezoidal rule.
- **CDF Lookup via Interpolation**: To generate a sample X, the following steps are performed:
  1. Generate a uniform random number $U \sim \text{Uniform}(0, 1)$.
  2. Find two precomputed CDF values $F(X_i)$ and $F(X_{i+1})$ such that:
  $$F(X_i) \leq U < F(X_{i+1}).$$
  3. Approximate the corresponding X-value using linear interpolation between X_i and X_{i+1}:
  $$X = X_i + \frac{U - F(X_i)}{F(X_{i+1}) - F(X_i)} \cdot (X_{i+1} - X_i),$$
  where $F(X_i)$ and $F(X_{i+1})$ are the cumulative probabilities at grid points X_i and X_{i+1}, respectively.

2.2.2 Background Sampling

Background samples are drawn from $g_b(X)$ (uniform distribution) and $h_b(Y)$ (truncated Gaussian distribution).

Sampling in X: Uniform Distribution The uniform background distribution $g_b(X)$ is straightforward to sample:

$$X \sim \text{Uniform}(0, 5).$$

This involves directly generating random numbers within the specified range.

Sampling in Y: Truncated Gaussian Distribution The truncated Gaussian background $h_b(Y)$ is also sampled using the inverse transform sampling method:

The `scipy.stats.truncnorm` library function is used, which handles the truncation bounds $[0, 10]$.

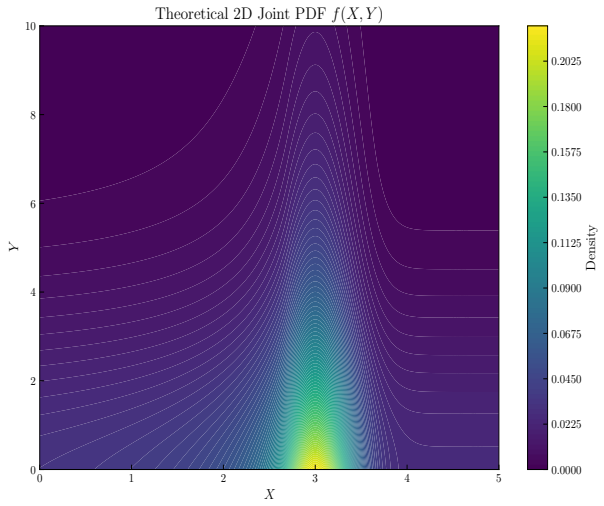
2.2.3 Combined Sampling

The combined sampling process integrates signal and background components:

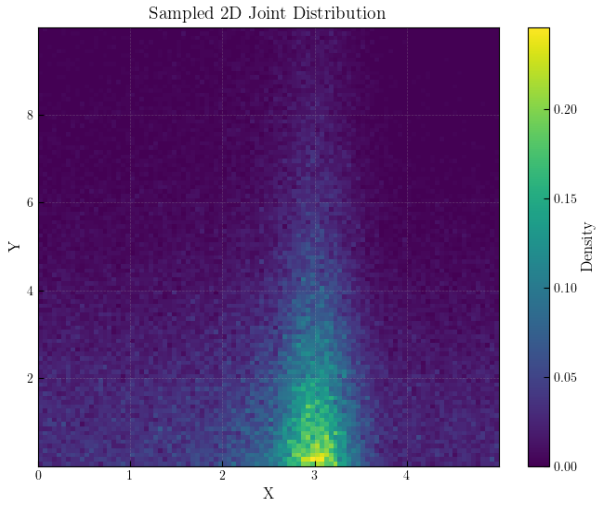
1. Each sample is probabilistically assigned to either signal or background based on the signal fraction f_s.
2. Samples are then drawn from the corresponding distributions: $g_s(X), h_s(Y)$ for the signal or $g_b(X), h_b(Y)$ for the background.

2.2.4 Joint and Marginal Sampling Comparisons

To validate the sampling techniques, the actual joint PDF and sampled joint data are visualized side by side, followed by individual marginal distributions for X and Y comparing actual and sampled distribution.

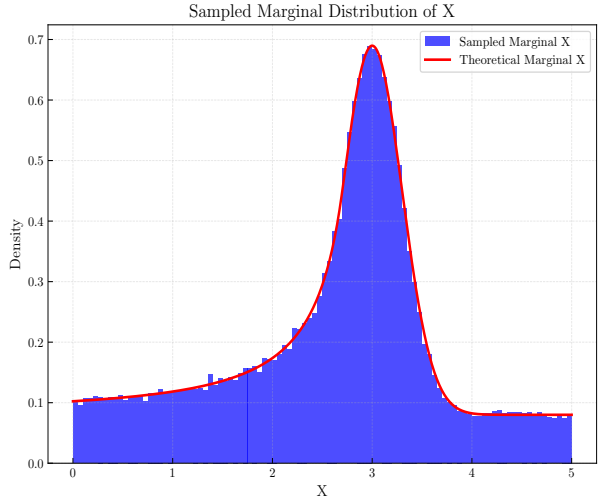


(a) Theoretical joint PDF of X and Y.

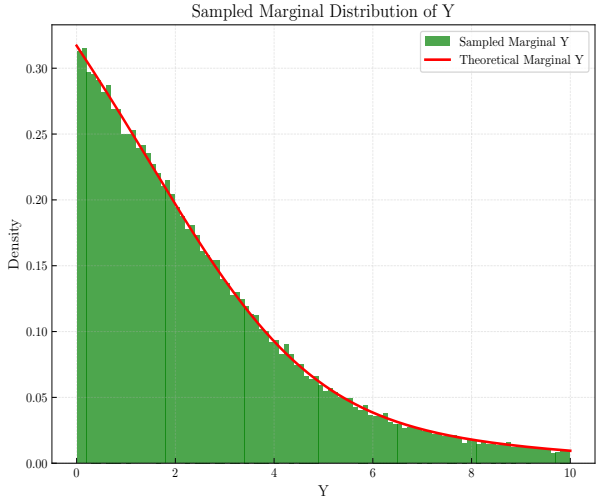


(b) 2D binned joint distribution of sampled data.

Figure 2: Comparison between the theoretical joint PDF and the binned joint distribution of sampled data.



(a) Actual and sampled marginal distributions for X.



(b) Actual and sampled marginal distributions for Y.

Figure 3: Comparison of actual and sampled marginal distributions for X and Y.

2.3 Fitting Techniques

The parameter estimation focuses on extracting the signal parameter λ, which defines the exponential decay of the signal in Y. Two distinct methodologies are employed: a joint extended likelihood fit and a weighted fit using *sWeights*. These approaches are detailed below.

2.3.1 Joint Extended Likelihood Fit

The joint extended likelihood fit accounts for both the observed number of samples and the probability density of each sample. The primary objective is to estimate λ while simultaneously fitting all other parameters in the joint PDF. The derivation of the likelihood function is as follows [4]:

Extended Likelihood Function The likelihood function for a dataset of N_{obs} samples is:

$$\mathcal{L} = P(N_{\text{obs}}|N_{\text{exp}}) \cdot \prod_{i=1}^{N_{\text{obs}}} f(X_i, Y_i; \theta),$$

where:

- $P(N_{\text{obs}}|N_{\text{exp}})$ is the Poisson probability of observing N_{obs} samples given an expected number N_{exp},
- $f(X_i, Y_i; \theta)$ is the joint PDF evaluated at the i-th sample point, and
- θ represents the set of model parameters, including λ.

Poisson Contribution The Poisson term is expressed as:

$$P(N_{\text{obs}}|N_{\text{exp}}) = \frac{N_{\text{exp}}^{N_{\text{obs}}} e^{-N_{\text{exp}}}}{N_{\text{obs}}!}. \tag{11}$$

Extended Log-Likelihood Taking the natural logarithm of the likelihood function yields:

$$\ln \mathcal{L} = \ln P(N_{\text{obs}}|N_{\text{exp}}) + \sum_{i=1}^{N_{\text{obs}}} \ln f(X_i, Y_i; \theta), \tag{12}$$

$$= -N_{\text{exp}} + N_{\text{obs}} \ln N_{\text{exp}} - \ln(N_{\text{obs}}!) + \sum_{i=1}^{N_{\text{obs}}} \ln f(X_i, Y_i; \theta). \tag{13}$$

The term $\ln(N_{\text{obs}}!)$ is constant for a given dataset and does not affect optimization, so it is ignored.

Extended Negative Log-Likelihood The extended negative log-likelihood, minimized to extract λ and other parameters, is therefore:

$$-\ln \mathcal{L} = -N_{\text{obs}} \ln N_{\text{exp}} + N_{\text{exp}} - \sum_{i=1}^{N_{\text{obs}}} \ln f(X_i, Y_i; \theta). \tag{14}$$

Implementation Details Key aspects of the implementation include:

- **Precomputed Truncation Normalization:** The truncation normalization constant for $g_s(X)$ is precomputed during each iteration to reduce computational overhead.
- **Numerical Stability:** A small offset (10^{-10}) is added to avoid log(0) errors.
- **Optimization:** The `iminuit` library is used for optimization [5]. It employs the MIGRAD algorithm, which minimizes the negative log-likelihood by iteratively updating parameters along the direction of steepest descent. MIGRAD dynamically adjusts step sizes and leverages second-order derivative approximations for efficient convergence. Two uncertainty estimation methods are provided:
- **HESSE:** Approximates the covariance matrix by inverting the Hessian matrix at the minimum, providing symmetric uncertainties under the assumption of a quadratic likelihood surface.
- **MINOS:** Computes asymmetric uncertainties by scanning the likelihood function along parameter directions, accounting for non-quadratic behavior near the minimum.

HESSE is chosen for its computational efficiency and sufficient accuracy given the well-behaved nature of the likelihood surface in this analysis.

Bounds and Initial Guesses for the Fitting Procedure To ensure convergence and accurate parameter estimation during the fitting procedure, we selected **bounds** and **initial guesses** for each parameter based on the behavior of the marginal distributions in X and Y, as shown in Figure 3.

- μ **(mean of the signal distribution):** The peak of the signal component $g_s(X)$ in Figure 3a suggests a location close to $\mu = 3.0$. We set the bounds $[2.8, 3.2]$ to focus the fit around this region and chose an initial guess of $\mu = 3.2$.

- σ **(width of the signal distribution):** The signal distribution is relatively narrow, as observed in Figure 3a. Hence, the bounds for σ are set to $[0.05, 0.5]$, with an initial guess of $\sigma = 0.2$.
- β **and** m **(shape parameters for the Crystal Ball distribution):** To allow flexibility in modeling the tail of $g_s(X)$, the bounds are chosen as $\beta \in [0.5, 1.5]$ and $m \in [1.0, 2.0]$. The initial guesses of $\beta = 1.2$ and $m = 1.2$ are close to standard values observed in signal modeling.
- λ_s **(decay rate for the signal in** Y**):** The exponential decay observed in Figure 3b motivated bounds of $\lambda_s \in [0.1, 1.0]$, with an initial guess of $\lambda_s = 0.5$.
- μ_b **and** σ_b **(background components):** The background distribution $g_b(X)$ in Figure 3a and $h_b(Y)$ in Figure 3b are flatter and broader. To capture this, we set $\mu_b \in [-1, 1]$ and $\sigma_b \in [2.0, 4.0]$, with initial guesses of $\mu_b = 0.3$ and $\sigma_b = 2.7$.
- f_{signal} **(signal fraction):** An approximate signal-background balance in the distributions suggested $f_{\text{signal}} \in [0.1, 0.9]$, with an initial guess of $f_{\text{signal}} = 0.5$.
- $N_{expected}$ **(expected number of events):** To account for Poisson fluctuations, the nominal sample size N is replaced by N_{Poisson}, a Poisson random variable with mean N. The bounds are then set as:
$$N_{\text{expected}} \in [0.75 \cdot N_{\text{Poisson}}, 1.25 \cdot N_{\text{Poisson}}],$$
with an initial guess of $N_{\text{expected}} = N_{\text{Poisson}}$.

2.3.2 Weighted Fit Using *sWeights*

The *sWeights* methodology [6] decouples the fitting process into two stages to isolate the signal component in Y and estimate the decay constant λ. The steps are as follows:

Step 1: Fit in X The goal of this step is to estimate the signal fraction f_{signal} and calculate the signal/background weights (*sWeights*) for each sample X_i. The marginal distribution in X, $f(X)$, is modeled as a mixture of signal and background components:

$$f(X_i) = f_{\text{signal}}\, g_s(X_i) + (1 - f_{\text{signal}})\, g_b(X_i),$$

where $g_s(X)$ is the signal PDF and $g_b(X)$ is the background PDF.

The weights are defined as:

$$w_{\text{signal},i} = \frac{f_{\text{signal}}\, g_s(X_i)}{f(X_i)}, \tag{15}$$

$$w_{\text{background},i} = \frac{(1 - f_{\text{signal}})\, g_b(X_i)}{f(X_i)}. \tag{16}$$

The parameters of $f(X)$, including $\mu, \sigma, \beta, m, f_{\text{signal}}, N_{\text{expected}}$, are estimated using the `iminuit` library by minimizing the extended negative log-likelihood:

$$-\ln \mathcal{L}_X = N_{\text{expected}} - N_{\text{Poisson}} \ln(N_{\text{expected}}) - \sum_{i=1}^{N_{\text{Poisson}}} \ln\left[f_{\text{signal}} g_s(X_i) + (1 - f_{\text{signal}}) g_b(X_i)\right]. \tag{17}$$

This fit allows us to determine the signal fraction f_{signal} and extract of the signal weights $w_{\text{signal},i}$, which are used to isolate the signal contribution in Y.

Step 2: Weighted Fit in Y The signal weights $w_{\text{signal},i}$ obtained from Step 1 are used to isolate the signal component and perform a weighted fit on the marginal distribution in Y to estimate λ. The weighted likelihood function in Y is:

$$\mathcal{L}_Y = \prod_{i=1}^{N_{\text{Poisson}}} h_s(Y_i)^{w_{\text{signal},i}}, \tag{18}$$

where $h_s(Y)$ is the signal PDF in Y.

Taking the negative logarithm gives the weighted log-likelihood:

$$-\ln \mathcal{L}_Y = -\sum_{i=1}^{N_{\text{Poisson}}} w_{\text{signal},i} \ln h_s(Y_i). \tag{19}$$

The parameter λ is extracted by minimizing this weighted negative log-likelihood using the `iminuit` library.

3 Computational Efficiency Benchmarking

To evaluate the computational performance of our implementation, we benchmarked three main tasks:

1. **Random Sampling from a Normal Distribution:** Using `np.random.normal` as a baseline.
2. **Sample Generation:** Generating samples from the combined signal and background distributions using the defined methods.
3. **Fitting:** Performing the extended likelihood fit using the generated samples.

The benchmarking was conducted for a dataset of $100,000$ samples, averaged over 100 repetitions. The results include average execution times, standard deviation of execution times, and relative times compared to `np.random.normal`.

Table 1: Benchmarking Results for Sampling and Fitting Techniques

Task	**Average Time (s)**	**Std Dev (s)**	**Relative Time**
`np.random.normal` (Baseline)	0.001797	0.000930	1.000
Joint PDF Sampling	0.009922	0.001255	5.521
Maximum Likelihood Fit	1.755148	0.023208	976.622

The results indicate that:

- **Sampling** is approximately 5.5 times slower than baseline normal sampling. This is due to the added complexity of sampling from non-standard distributions (e.g., truncated Crystal Ball and exponential distributions), which require numerical techniques such as CDF construction, interpolation, and inversion instead of direct sampling.
- **Maximum likelihood fitting** is significantly more computationally intensive, with a relative time of approximately 977 times the baseline. This is because the likelihood fit relies on iterative optimization using numerical methods. In the absence of analytic gradients, MIGRAD estimates gradient vectors numerically using finite differences by perturbing each parameter slightly and recalculating the negative log-likelihood across the entire dataset. Computing this negative log likelihood each time to calculate each partial derivative making up the gradient vector, per optimization iteration, until convergences is extremely costly. Additionally, MIGRAD dynamically approximates second-order derivatives (Hessian elements) to optimize step size and convergence, further increasing computational cost.

4 Bootstrapping Analysis

Bootstrapping is utilized to evaluate the performance of the two fitting methodologies—joint extended likelihood fit and *sWeights* fit—in estimating the decay constant λ. This involves generating multiple trials over varying sample sizes, performing fits for each trial using each method, and analyzing the resulting uncertainties and biases, along with the errors associated with these metrics.

4.1 Methodology

The bootstrapping procedure follows these steps:

1. **Generate Bootstrap Samples:** For each trial, synthetic datasets are generated using the sampling techniques described earlier. Sample sizes range from 500 to 10000 to evaluate scaling behavior.
2. **Perform Fits:** Each synthetic dataset is fitted using both the joint extended likelihood fit and the *sWeights* fit to extract the decay constant λ.
3. **Repeat:** The process is repeated for $N_{\text{trials}} = 250$ bootstrap trials per sample size to ensure statistical robustness.
4. **Calculate Metrics:**
   - **Bias:** The bias in the fitted decay constant λ is defined as:

$$\text{Bias } (\delta_\lambda) = \langle \hat{\lambda} \rangle - \lambda_{\text{true}}, \tag{20}$$

   where $\langle \hat{\lambda} \rangle = \frac{1}{N} \sum_{i=1}^{N} \hat{\lambda}_i$ is the mean of the fitted decay constants across all trials, and λ_{true} is the true value of the decay constant.

   - **Uncertainty:** The uncertainty σ_λ in the fitted decay constant is given by the standard deviation of the fitted values:

$$\sigma_\lambda = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left(\hat{\lambda}_i - \langle \hat{\lambda} \rangle \right)^2}, \tag{21}$$

   where N is the number of bootstrap trials, and $\hat{\lambda}_i$ is the fitted decay constant from the i-th trial.

   - **Bias Error:** The error on the bias Δ_{δ_λ} reflects the statistical uncertainty in estimating the mean of the fitted decay constants $\langle \hat{\lambda} \rangle$. This is derived as:

$$\Delta_{\delta_\lambda} = \frac{\sigma_\lambda}{\sqrt{N}}, \tag{22}$$

   where σ_λ is the standard deviation of the fitted values, and $\sqrt{N}$ arises because the mean is estimated from N independent trials.

   - **Uncertainty Error:** The error on the uncertainty Δ_{σ_λ} quantifies the uncertainty in the estimation of the standard deviation σ_λ. Using propagation of errors and the asymptotic variance of the sample standard deviation (derived from the chi-squared distribution with $N-1$ degrees of freedom), it is given by:

$$\Delta_{\sigma_\lambda} = \frac{\sigma_\lambda}{\sqrt{2(N-1)}}. \tag{23}$$

The results of the bootstrapping analysis are presented below. Figure 4 illustrates the bias in the fitted decay constant λ, while Figure 5 shows the uncertainties as a function of sample size.

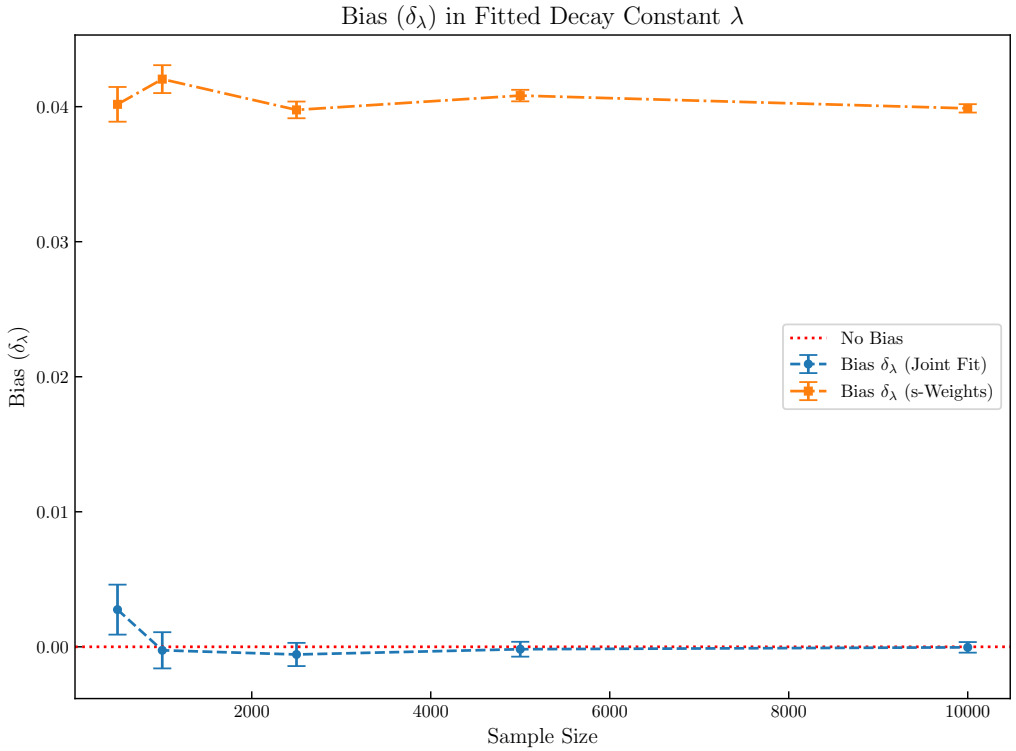


Figure 4: Bias δ_λ in the fitted decay constant λ as a function of sample size. The joint extended likelihood fit exhibits negligable bias ($\delta_\lambda \approx 0$), whereas *sWeights* method shows a constant bias of approximately $\delta_\lambda \approx 0.04$ across the sample sizes

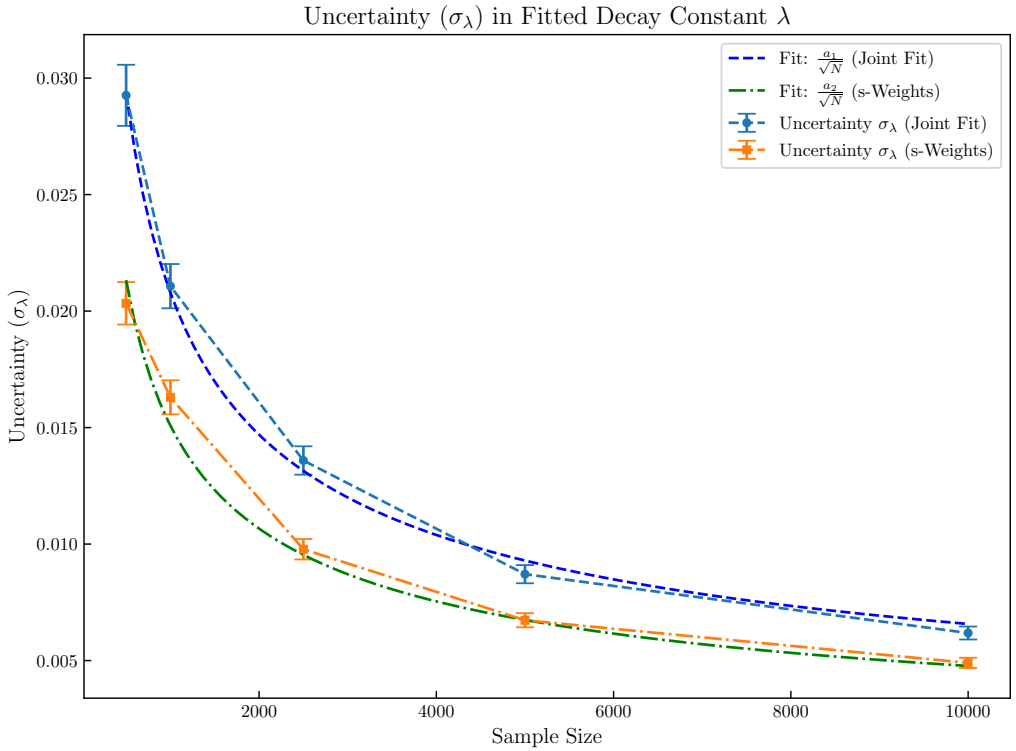


Figure 5: Uncertainty σ_λ in the fitted decay constant λ as a function of sample size. Both methods show decreasing uncertainties with increasing sample size, but the joint extended likelihood fit has consistently higher uncertainties compared to the *sWeights* method. The uncertainties are well described by a $1/\sqrt{N}$ scaling, as shown by the fitted curves.

4.2 Discussion

Bias: The joint extended likelihood fit exhibits negligible bias in the fitted decay constant λ, while the *sWeights* method shows a small but persistent bias of approximately $\delta_\lambda \sim 0.04$ across all sample sizes. This behavior arises due to fundamental differences in how the two methods utilize the full joint distribution $f(X, Y)$ and handle correlations between the variables X and Y:

- **Joint Fit:** The joint extended likelihood fit models the signal and background components simultaneously within a single optimization framework, using the full joint PDF $f(X, Y)$. By explicitly accounting for correlations between X and Y, this method avoids structural assumptions and achieves an unbiased estimation of λ. As the sample size increases, statistical fluctuations diminish due to averaging, further reducing the bias.
- ***sWeights* Fit:** The signal weights $w_{\text{signal},i}$ are calculated from a marginal fit in X, which estimates the signal fraction f from the marginal PDFs $g_s(X)$ and $g_b(X)$. In practice, finite sample sizes, imperfect bounds and initial guesses can lead to incorrect and local instead of global convergences during the fit introducing statistical uncertainties and inaccuracies in these estimates. These uncertainties propagate into the signal weights, such that they are imperfect representations of the true signal contribution. When these imperfect weights are applied in the weighted fit in Y, the reconstructed signal distribution does not perfectly isolate $h_s(Y)$, introducing a systematic bias in the parameter estimation.

Uncertainty: The uncertainties σ_λ in the fitted decay constant decrease as a function of the sample size N, as expected. The uncertainties scale as:

$$\sigma_\lambda \propto \frac{1}{\sqrt{N}},$$

where N is the number of samples in the dataset. This relationship exists because larger sample sizes provide more statistical information, allowing more precise estimation of the parameter when fitting, which reduces the standard error of the estimated parameters obtained from the trials. Specifically, the standard error of the fitted parameter decreases inversely with the square root of the sample size, as is typical in statistical inference.

In figure 5, the joint fit consistently shows slightly larger uncertainties compared to the *sWeights* fit. This difference likely arises because the joint fit involves simultaneous optimization over multiple parameters, introducing additional variability, whereas the *sWeights* method performs a simpler weighted fit.

4.3 Comparison of the Joint Fit and *sWeights* Methods

The joint extended likelihood fit and the *sWeights* method offer different trade-offs between accuracy and efficiency:

- **Joint Fit:** Provides unbiased parameter estimates by simultaneously fitting the signal and background components, accounting for correlations between X and Y. However, it is computationally intensive due to the simultaneous optimization of multiple parameters.
- ***sWeights* Fit:** Computationally efficient and well-suited for large datasets where marginal signal and background models in X are accurately known and can be well fitted. However, it assumes no correlations between X and Y in the joint PDF, leading to systematic biases if this assumption is violated.

In practice, the joint fit is preferred for precision analyses where correlations and model imperfections are critical, while the *sWeights* method is advantageous for exploratory studies or when computational resources are limited.

5 Conclusion

In this study, we compared the joint extended likelihood fit and the *sWeights* method for estimating the decay constant λ. The joint fit, while computationally intensive, provided unbiased estimates and accounted for correlations between variables X and Y. In contrast, the *sWeights* method, though computationally efficient, introduced a small systematic bias due to its decoupling of the fits. The results show that the joint fit is ideal for precision-focused tasks, whereas the *sWeights* method is more suitable for large datasets or computationally constrained scenarios.

References

[1] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

[2] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1968.

[3] Sheehan Olver and Alex Townsend. Fast inverse transform sampling in one and two dimensions. *arXiv preprint arXiv:1307.1223*, 2013.

[4] Jon Wakefield et al. *Bayesian and frequentist regression methods*, volume 23. Springer, 2013.

[5] Hans Dembinski, Piti Ongmongkolkul, Christoph Deil, David Menéndez Hurtado, Henry Schreiner, Matthew Feickert, Chris Burr, Fabian Rost, Alex Pearce, Lukas Geiger, et al. iminuit: Jupyter-friendly python interface for c++ minuit2. *Astrophysics Source Code Library*, pages ascl–2108, 2021.

[6] Hans Dembinski, Matthew Kenzie, Christoph Langenbruch, and Michael Schmelling. Custom orthogonal weight functions (cows) for event classification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1040:167270, 2022.

Appendix

A Declaration of Autogenerative Tools

In this project, I used GitHub Copilot as a coding assistant. I mainly used it to generate the plots in part c), the table in d), and the bias and variance plots in part e) and f). I also used it within my code to create comments, docstring my functions, debug and format markdown cells explaining the different sections in the notebook.

In regards to the report, I used the Overleaf AI which uses ChatGPT to help write the mathematics in latex format, and also used it to correctly format various sections of the report as I am quite new to LaTeX.