Bayesian inference in rare processes—traditional and modern methods

Supervisor: Prof. Dr. A. Quadt, II. Physikalisches Institut, Georg-August-Universität Göttingen

E-mail: aquadt@uni-goettingen.de

Abstract: In comparing experimental data with theoretical prediction, Bayesian techniques are often employed to draw conclusions on model validity or estimate parameter values. An interface between two pieces of software designed for this purpose and utilising different algorithms has been developed. The first, the Bayesian Analysis Toolkit, relies primarily on the Metropolis-Hastings algorithm and Markov Chain Monte Carlo. The second, MultiNest, uses a new algorithm called nested sampling to evaluate the Bayesian evidence. The software is used in the analysis of the Gerda experiment, a search currently underway for the hypothesised rare process of neutrinoless double beta decay. The software is used to set limits on the half-life of the process and the neutrino mass as a function of the exposure—for an exposure of 100 kg yrs and a background rate of 10^{-3} counts/ kg keV yr, an upper limit on the mass of 120 meV could be set assuming no events were observed.

Except where specific reference is made to the work of others, this work is original and has not been already submitted either wholly or in part to satisfy a degree requirement at this or any other University.

Contents

1 Introduction

Confronted as modern physics is with a wealth of experimental data across diverse fields, Bayesian inference presents itself as a solution to the dual problems of model selection and parameter estimation. The object of such an analysis is, in short, the evaluation of the relative likelihoods of various models by comparison of predicted and measured values for relevant quantities and of the parameter ranges for any given model. The Bayesian Analysis Toolkit (BAT) [1] provides a realisation of this and utilises the traditional approach to such analysis, an implementation of the Metropolis-Hastings algorithm via Markov Chain Monte Carlo. The BAT calculates the posterior probability distribution (a degree of belief given

the available information) which is used in performing parameter estimation among other purposes (see section II). Though effective in this regard, the method does not itself evaluate the Bayesian evidence which requires integration over the parameter space. This evidence is required for model comparison based on absolute probabilities.

In this paper an interface between an alternative sampling algorithm, MultiNest and the BAT is used in the analysis of sparsely populated spectra. The MultiNest algorithm relies upon evaluation of the Bayesian evidence via nested sampling, also producing the posterior distribution in the process. In particular, it is highly efficient in the sampling of multimodal or degenerate distributions, a shortcoming of the MCMC method which proves slow to converge. Interfacing the two pieces of software allows an easy implementation of either approach on a given data set as well as the other external libraries with which the BAT is equipped (such as Cuba [2]).

Though the spectral analysis detailed here is general and may be applied to a variety of physics cases, the specific case of the Gerda experiment is discussed here, following previous analysis using standard MCMC [3]. The experiment searches for instances of neutrinoless double beta decay, which (if indeed realised in nature) is expected to be an extremely rare process. Discovery criteria (based on whether or not known phenomena sufficiently describe the data) are discussed and the sensitivity of the experiment evaluated. This is achieved by generating ensembles of Monte Carlo data and subjecting these to Bayesian analysis.

The paper is structured as follows. Section 2 provides an introduction to Bayesian inference in spectral analysis. The BAT and MCMC techniques are discussed briefly in section 3 followed by a similar overview of nested sampling and the MultiNest algorithm in section 4. In section 5 we detail the BAT-MultiNest interface, a major focus of this project. Section 6 describes the Gerda experiment and application of the software to this scenario. The results of said analysis are then presented. General conclusions are presented in section 7.

2 Bayesian inference in spectral analysis

Beginning with the famous theorem due to Bayes:

$$P(\boldsymbol{\lambda}, \boldsymbol{\nu}, H|\mathbf{D}) = \frac{P(\mathbf{D}|\boldsymbol{\lambda}, \boldsymbol{\nu}, H)P_0(\boldsymbol{\lambda}, \boldsymbol{\nu}, H)}{P(\mathbf{D})} \tag{2.1}$$

our objects are, for example, to determine whether an observed spectrum is due to signal and background processes or background alone, to evaluate the expected values of parameters should the signal be present and to set a limit on the signal contribution should none be observed. Here $P(\boldsymbol{\lambda}, \boldsymbol{\nu}, H|\mathbf{D})$ is known as the posterior probability distribution of a set of parameters $\boldsymbol{\lambda}$ and nuisance parameters $\boldsymbol{\nu}$ with a hypothesis or model H given data $\mathbf{D}$. This describes the state of knowledge after analysis. The prior $P_0(\boldsymbol{\lambda}, \boldsymbol{\nu}, H)$ gives the information present before analysis and describes an initial state of knowledge of the parameter ranges, model limitations etc. Finally we may define the likelihood $L = P(\mathbf{D}|\boldsymbol{\lambda}, \boldsymbol{\nu}, H)$ and evidence $Z = P(\mathbf{D})$, a normalisation factor over the parameter space. The evidence is given by

$$Z = \int L(\boldsymbol{\lambda}, \boldsymbol{\nu}) P_0 d\boldsymbol{\lambda} \tag{2.2}$$

This is usually ignored when inferring parameter values—in this case, sampling from a function proportional to the posterior suffices—but is required for model comparison. It should be noted first that the form of Bayes' theorem given above differs slightly for the case of parameter estimation (in which the model is fixed) and second that often some additional information is available which may constrain parameter values—this has been omitted above and in the following will be assumed implicitly present.

2.1 Hypothesis testing

We aim first to test the hypothesis that the observed spectrum is due to background only, denoted H_1. Its negation H_2 implies that a signal process is present—the two must satisfy

$$P(H_1|\mathbf{D}) + P(H_2|\mathbf{D}) = 1 \tag{2.3}$$

Since here all possible models can be enumerated (there are only two possibilities), both prior and likelihood may be calculated and a discovery criterion given. Using Bayes' theorem we may now write

$$P(\mathbf{D}) = P(\mathbf{D}|H_1)P_0(H_1) + P(\mathbf{D}|H_2)P_0(H_2) \tag{2.4}$$

We wish to express these probabilities in terms of an expected number of signal and/or background events, which we do as follows:

$$P(\mathbf{D}|H_1) = \int P(\mathbf{D}|B)P_0(B)dB \tag{2.5}$$

$$P(\mathbf{D}|H_2) = \int P(\mathbf{D}|S,B)P_0(B)P_0(S)dSdB \tag{2.6}$$

for expected numbers of events S and B. The choice of priors is influenced by previous experiments and knowledge.

We now wish to express the number of expected events in a given bin of the spectrum in terms of the given number of signal and/or background events. This is given by

$$\alpha_i(S,B) = S \cdot \int_{\Delta E_i} f_S(E)dE + B \cdot \int_{\Delta E_i} f_B(E)dE \tag{2.7}$$

for $f(E)$ the normalised shape of the signal/background spectrum and ΔE_i the width of the ith bin. If the number of events in a bin fluctuates according to a Poisson distribution about α_i and these fluctuations are assumed uncorrelated, we may write

$$P(\mathbf{D}|S,B) = \prod_{i=1}^{N} \frac{\alpha_i(S,B)^{n_i}}{n_i!} e^{-\alpha_i(S,B)} \tag{2.8}$$

for an observed number of events n_i and setting S to 0 in the background only case. Finally, we may obtain

$$P(H_1|\mathbf{D}) = \frac{[\int\{\prod \frac{\alpha_i^{n_i}}{n_i!}e^{-\alpha_i}\} \cdot P_0(B)dB]_{S=0} \cdot P_0(H_1)}{[\int\{\prod \frac{\alpha_i^{n_i}}{n_i!}e^{-\alpha_i}\} \cdot P_0(B)dB]_{S=0} \cdot P_0(H_1) + [\int\{\prod \frac{\alpha_i^{n_i}}{n_i!}e^{-\alpha_i}\} \cdot P_0(B) \cdot P_0(S)dSdB] \cdot P_0(H_2)} \tag{2.9}$$

Following similar previous analysis [3], we may utilise a suggested discovery criterion $P(H_1|\mathbf{D}) \leq 0.0001$ and a weaker evidence criterion $P(H_1|\mathbf{D}) \leq 0.01$. It is important that these values are defined beforehand (the experiment is carried out 'blind') in order to prevent bias in the analysis.

2.2 Parameter estimation

For a particular scenario it is possible either to estimate the value of a parameter (in this case, the signal contribution for the case of evidence) or to set limits on its value (in the case that no signal is observed). We first take the former case. Applying Bayes' theorem,

$$P(S, B|\mathbf{D}) = \frac{P(\mathbf{D}|S, B)P_0(S)P_0(B)}{\int P(\mathbf{D}|S, B)P_0(S)P_0(B)dSdB} \tag{2.10}$$

we marginalise with respect to B by integration:

$$P(S|\mathbf{D}) = \int P(S, B|\mathbf{D})dB \tag{2.11}$$

We may estimate the signal via the mode and calculate an associated uncertainty from appropriate probability intervals. The latter case is straightforward; a 90% probability limit S_{90} may be obtained via

$$\int_0^{S_{90}} P(S|\mathbf{D})dS = 0.90 \tag{2.12}$$

.

3 Markov Chain Monte Carlo and the Bayesian Analysis Toolkit

3.1 Markov Chain Monte Carlo

MCMC techniques have proved the most common means of sampling the posterior distribution. A Markov chain consists of a sequence of random variables X_n in which the X_{n+1} depends only on X_n, that is, the current state. It may be completely defined by a marginalised distribution for the initial probabilities of various states X_0 and a conditional transition probability $P(X_{n+1}|X_n)$ to obtain one state from another. Provided this transition probability is constant with n, the chain is ergodic and the probability to be in a given state stationary. The chain thus 'forgets' its initial state and converges on a stationary distribution (which in this case should be the posterior). To achieve this, the Metropolis-Hastings algorithm [4] is commonly applied as follows:
1. From a state $X_n = \mathbf{x}$, propose a new state $\mathbf{y}$ according to a symmetric proposal function $g(\mathbf{y}, \mathbf{x})$.
2. Calculate the ratio

$$R = \frac{f(\mathbf{y})}{f(\mathbf{x})}$$

where $f(\mathbf{x})$ is the desired limiting distribution and compare to a random number Q drawn from an uniform, flat distribution over [0,1]. If $Q < R$ set $X_{n+1} = \mathbf{y}$ else set $X_{n+1} = \mathbf{x}$.

The algorithm proves particularly useful in generating samples from distributions with no analytic form, and requires only that $f(\mathbf{x})$ can be calculated. The proposal function g may have any form; the chain will still converge correctly.

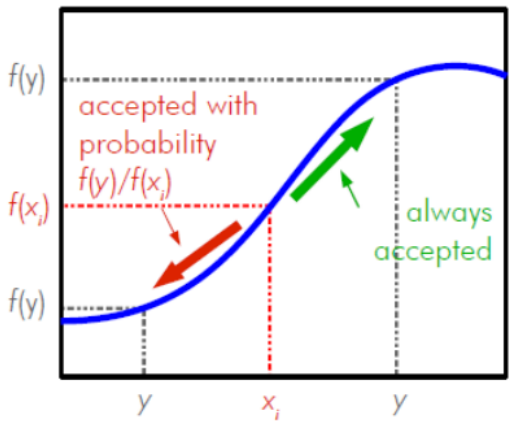

Figure 1: The Metropolis-Hastings algorithm results in a random walk biased towards higher probabilities, as shown in the figure. Step size should be chosen carefully; too small results in inefficient running, while too large can mean narrow modes are missed.

3.2 The Bayesian Analysis Toolkit

The BAT provides a particular implementation of the above and performs optimisation, marginalisation and integration where appropriate. It is C++ code available in the form of a library and has a class-based structure which allows models to be specified and numerical operations applied on them by the user. Although several algorithms for each aspect of functionality are possible, the primary engine runs an MCMC. A pre-run is first carried out to ensure convergence, followed by sampling and analysis runs. Optimisation is performed using the ROOT version of MINUIT. Where integration is required and no analytic expression for the evidence has been specified by the user, a numerical integration using the sampled mean algorithm (importance sampling optional) is performed. Interfaces to the CUBA library are also available. As output, the BAT produces ROOT trees which store the summary information and Markov chains, an ASCII file which contains the results of analysis and histograms of the marginalised distributions and correlations between parameters.

4 Nested sampling and MULTINEST

4.1 Nested sampling

While the MCMC method samples from the posterior distribution directly, the aim of nested sampling [5][6] is to calculate the evidence by sampling from the prior distribution. In this process the posterior distribution may be obtained as a by-product. In general, evaluation of the evidence requires a multi-dimensional integral over the entire prior density which becomes increasingly non-trivial with dimension (eq. 2.2). Replacing this with an integral over the 'prior volume', defined $dX = P_0(\boldsymbol{\lambda})d\boldsymbol{\lambda}$, we obtain

$$X(\Lambda) = \int_{\{\boldsymbol{\lambda}:L(\boldsymbol{\lambda})>\Lambda\}} P_0(\boldsymbol{\lambda})d\boldsymbol{\lambda} \tag{4.1}$$

an integral over a region of parameter space bounded by the likelihood contour Λ. For monotonically decreasing $L(X)$, equation 2.2 can then be rewritten as

$$Z = \int_0^1 L(X)dX \tag{4.2}$$

a one dimensional integral. For exactly known $L(X)$ the $L_i = L(X_i)$ may be evaluated for $0 < X_N < ... < X_0 = 1$ and the trapezium rule used to approximate the integral as

$$Z \approx \sum_{i=1}^{N} L_i w_i \tag{4.3}$$

where the weights $w_i = \frac{1}{2}(X_{i-1} - X_{i+1})$.

The nested sampling algorithm proceeds first by drawing N 'live' samples from the full prior P_0 and setting the initial prior volume X_0 to unity. At each iteration the sample with lowest likelihood L_i is removed from the set and replaced with another sample drawn from the prior satisfying $L > L_i$. The new likelihood contour contains a prior volume $X_i = t_i X_{i-1}$ for t_i distributed according to the largest of N samples drawn from [0,1], that is $P(t) = Nt^{N-1}$. This continues until the entire prior volume has been explored—the extent of the likelihood contour decreases with each iteration, hence *nested* sampling. It may be shown [5] that one may then take $X_i \approx \exp(-i/N)$. The stopping criterion is defined as the iteration at which contribution to the integral is less than some specified tolerance—the underestimate may be remedied using $\Delta Z_i = L_{max} X_i$. Having calculated the evidence, the posterior may be recovered using the discarded points weighted as

$$p_i = \frac{L_i w_i}{Z} \tag{4.4}$$

allowing relevant properties of the posterior to be calculated.

4.2 The MultiNest algorithm

A naïve implementation of the above in which samples were drawn from the full prior volume at each iteration would fail due to decreased acceptance rate as the likelihood contour shrinks. Following an ellipsoidal sampling method, MultiNest [7, 8] approximates the likelihood contour by one or more D-dimensional ellipsoids determined from the covariance matrix of live points and enlarged by some factor. The number of ellipsoids depends upon the number of distinct clusters of points and allows multimodal distributions and those with large degeneracies to be handled with ease. Figure 2 provides an illustration of the decomposition procedure at each iteration.

Given K ellipsoids at iteration i, points are drawn from the union of all ellipsoids such that a given ellipsoid is selected with a probability

$$p_k = \frac{V_k}{V_{\text{tot}}} \tag{4.5}$$

where $V_{\text{tot}} = \sum_{k=1}^{K} V_k$. Should the likelihood constraint be satisfied, the point is accepted with probability $1/q$ where q is the number of ellipsoids in which the point lies (this accounts for overlap) and the process repeated.

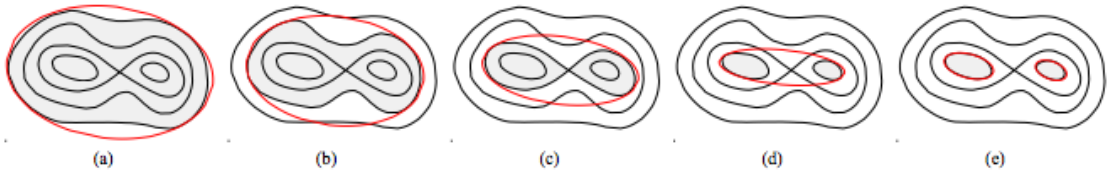

Figure 2: Illustration of the ellipsoidal sampling process. Stages (a) to (d) represent successive iterations and show the ellipsoidal approximations to the likelihood contours. The distribution shown is bimodal - it is clear by stage (d) that a single ellipsoid is a poor approximation. Stage (e) shows the increase in efficiency obtained by separating into two ellipsoids. From [8].

From version 3.0 of MultiNest onwards, the option for Importance Nested Sampling is also available—this decreases residual inefficiencies in the algorithm by utilising all points generated regardless of whether they satisfy the likelihood constraint. This removes the wasted computational cost in evaluating the likelihood of failed points at each iteration. Detailed discussion is found in reference [9]. A further advantage of nested sampling over MCMC is that it allows error estimation on the evidence with only a single run—this is achieved via summation of samples from the prior volume at each iteration. Again, further discussion (with particular reference to the INS case) may be found in [9].

5 The BAT-MultiNest interface

Advantages of the BAT package include its flexibility in phrasing models and data sets, its interface to pre-existing software used in high-energy physics (for example, ROOT) and its wide range of functionality. The package has been extended to include the MultiNest algorithm which can be run on BAT-defined models. This provides an efficient integration algorithm and alternative to the MCMC engine while compensating for certain shortcomings of that approach.

The BAT uses a C++ library based on ROOT (which is already prepared to deal with large data sets and graphical output) and implements models as user-defined classes on which different algorithms may then be run. The full source code and relevant documentation is available at `https://www.mppmu.mpg.de/bat/`. MultiNest is written in Fortran 90 but also provides a C/C++ interface which has been adapted for incorporation into the BAT. To this end additional classes BCMultinest.cxx and BCMultinestAdapter.cxx have been added, the former actually calling the Fortran routine and the latter adapting the likelihood function provided in the BAT model into the form required by MultiNest. A friend class of the adapter, BCIntegrate, is used to call the MultiNest integration (and all other integration, marginalisation routines etc.)—all user-defined model classes inherit from this allowing any desired operation to be called easily.

MultiNest allows the user to specify several options at runtime, including the tolerance (which determines the stopping point), the target efficiency, the number of live points etc. These are specified in a BAT model through use of a UserConfig object which allows

the desired settings to be applied while keeping fixed model-defined values that should not be altered (number of parameters, etc.). The full source code for the BCMultinestAdapter class is included in appendix A. This class accounts for several differences in the BAT and MultiNest likelihood functions. Firstly, the ellipsoidal sampling scheme detailed in section 4.2 requires a uniform prior from which to sample. Since the BAT allows a prior of any form to be specified, the likelihood is redefined as $L' = L \cdot P_0$ and this passed to MultiNest. This is most effective when the posterior distribution is dominated by the likelihood. In addition, MultiNest requires parameters to be specified on the unit hypercube. We now work with redefined priors $P'_0 = 1/V$ where V is the volume of the original parameter hyperrectangle—MultiNest returns the evidence $Z = Z' \cdot V$. Rescaling requires multiplication by the appropriate Jacobian (or addition in log-space). It is also possible to pass fixed (or 'nuisance') parameters which are scaled appropriately.

The adapter class also includes a dumper function which allows the MultiNest output to be written to a ROOT file for plotting. The code shown in the appendix includes examples of histogram plotting routines for each parameter and correlation plots for each combination of parameters using ROOT. This is primarily for demonstrative purposes—in the upcoming BAT release, functionality will be extended to allow existing BAT plotting and output routines to be applied to MultiNest output. Various other changes to the code that facilitate running can be found on the BAT website.

6 Applications: methods for assessing sensitivity

6.1 Neutrinoless double beta decay and the Gerda experiment

For certain isotopes a double beta decay (two simultaneous decays, $2\nu\beta\beta$) is permitted where single beta decay would otherwise be energetically forbidden. *Neutrinoless* double beta decay is a lepton-number violating second-order weak process potentially possible in extensions of the Standard Model. Such a process is possible only if the neutrino is a (massive) Majorana particle (that is, its own antiparticle). Figure 3 shows Feynman diagrams for the two processes.

The Gerda (Germanium Detector Array) experiment [11] employs high purity 86% ^{76}Ge enriched detectors to search for this rare process. The experimental signature of such a decay would be a single peak at the appropriate Q-value ($Q_{\beta\beta} = 2038.061 \pm 0.007$ keV, [12])—previous limits have been set by the Heidelberg-Moscow and the International Germanium Experiment collaborations, neither of which found evidence for the process [13, 14]. The current lower limits on the half-life are $T_{1/2} > 1.9 \cdot 10^{25}$ yr and $> 1.6 \cdot 10^{25}$ yr for the collaborations respectively. A claim by a part of the Heidelberg-Moscow collaboration for a reported $(28.75 \pm 6.86)0\nu\beta\beta$ decays, later strengthened by pulse shape information, was published giving a half life $T^{0\nu}_{1/2} = (1.19^{+0.37}_{-0.23}) \cdot 10^{25}$ yr [13]. This had not been examined until recently and the matter remains under debate [12, 15].

Data collection began in November 2011 and a total exposure of at least 100 kg years is anticipated during operation. A target background rate of 10^{-3}counts/(kg $\cdot$ keV $\cdot$ yr) has been set, which value we will use in the following analysis. The number of expected events

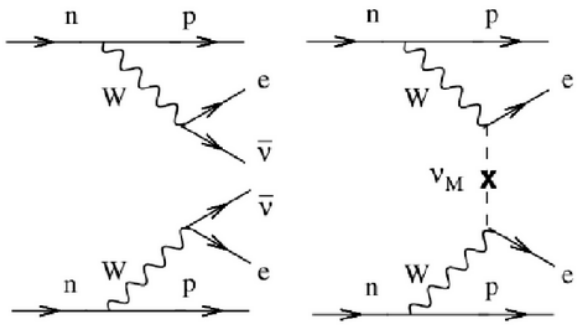

Figure 3: The diagram on the left shows a regular double beta decay, observed in eleven nuclei with a half-life in the range $10^{14} - 10^{24}$ yr [10]. The process on the right, $0\nu\beta\beta$, is possible only for a neutrino with Majorana mass component. Its half-life is dependent on the neutrino masses, mixing angles and CP phases.

S_0 is related to the process half-life by

$$S_0 \approx \ln 2 \cdot \kappa M \epsilon_{\text{sig}} \frac{N_A t}{M_A T_{1/2}} \tag{6.1}$$

where M is the mass of germanium in grams, N_A is Avogadro's constant, t is the measuring time, $\kappa = 0.86$ is the enrichment factor, M_A is the atomic mass and ϵ_{sig} is the signal efficiency. Monte Carlo simulation predicts this to be ≈ 0.87. It is also possible to translate this into an effective Majorana mass via

$$\langle m_{\beta\beta} \rangle = \frac{(T_{1/2} G^{0\nu})^{-1/2}}{\langle M^{0\nu} \rangle} \tag{6.2}$$

for $G^{0\nu}$ a phase space factor (quoted in [16]) and $\langle M^{0\nu} \rangle$ the relevant matrix element (quoted in [17]).

6.2 Ensemble testing

In the following, the concept of ensemble testing is used—multiple instances of given conditions are simulated and distributions of variables of interest obtained, such as the number of instances in which the discovery criterion is satisfied. In general an independent Monte Carlo event generator could be used to produce a spectrum of interest—here this process is automated by the BAT. The number of signal events S_0 is set by the half-life of the process while the number of background events B_0 is directly related to the anticipated background index (taken as a fixed 10^{-3}) and the exposure. We consider a region of interest of ± 50 keV around the Q-value and assume a flat background spectrum, $f_B(E) = \text{const}$. We model the signal as a Gaussian centred on $Q_{\beta\beta}$ with width determined by the energy resolution of the detectors—we take this as 5 keV, giving $\sigma \approx 2.1$ keV.

Calculation of the Bayesian probabilities discussed in section 2 requires prior distributions to be specified. Here we set

$$P_0(H_1) = P_0(H_2) = 0.5 \tag{6.3}$$

which is justifiable on the grounds that there is little theoretical consensus on Majorana neutrinos, nor is there convincing existing experimental suggestion. Bearing this in mind, we may set a flat $P_0(S)$ (assuming H_2) over a range $S_{\max}$ defined such that equation 6.3 holds. Assuming a certain knowledge of the background contribution B, we set the prior as a Gaussian with $\mu_B = B_0$ and $\sigma_B = \frac{B_0}{2}$. We thus have

$$P_0(S) = \begin{cases} \frac{1}{S_{\max}} & 0 \leq S \leq S_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

$$P_0(B) = \begin{cases} \frac{e^{-((B-\mu_B)^2/2\sigma_B^2}}{\int_0^\infty e^{-((B-\mu_B)^2/2\sigma_B^2} dB} & B \geq 0 \\ 0 & B < 0 \end{cases} \tag{6.5}$$

6.3 Results and discussion

As an example of use of the interface, appendix B contains part of the source code of an implementation of the above. Examples of output are shown in figures 4 and 5. Figure 4 shows histograms produced by running MultiNest; top is shown the binned marginalised probability density ($P(S|\mathbf{D})$) for an ensemble generated with 20 signal and 10 background events (corresponding to an exposure of 100kg$\cdot$ yr under the assumptions stated previously). The observed mean is at 19.75 as expected given the number of signal events. Below is shown the distribution of $\ln P(H_1|\mathbf{D})$ for 100 ensembles generated under the same conditions; bottom appears the distribution of the raw (logarithmic) evidence. Figure 5 shows cases without signal; top left is shown the binned marginalised probability density ($P(S|\mathbf{D})$) for an ensemble with 10 background events.

Assuming no signal contribution present, $S_0 = 0$, ensembles are generated with the aim of setting limits on the half-life. The 90th percentile is extracted from the probability distribution for the signal for each ensemble and these histogrammed—the mean and root mean square values are then extracted. Figure 5 shows typical histograms under the settings discussed above with 100 ensembles—bottom left is shown the 90% credibility limits generated using MCMC, bottom right using MultiNest. A mean value for the limit of 4.55 is returned by both methods, with a standard deviation of 1.75 in the former case and 1.78 in the latter. A histogram of $P(H_1|\mathbf{D})$ for the ensembles is shown top right. The process is repeated for different values of the exposure and transformed into a limit on the half-life, producing the plot seen in figure 6. Below this is further translated to a limit on the Majorana mass.

Results for both MCMC and MultiNest running are shown. Evaluation of the root mean square at each point has shown the standard error to be negligible. It should be noted that the errors associated with algorithm performance (but which indicate a spread related to the data set itself) have not been included. The MultiNest and MCMC predictions therefore follow each other closely. In the case of no background one would expect the limit to scale linear with exposure—including a background contribution causes a slower increase on the limit as the possibility of a signal becomes more difficult to distinguish.

In addition, calculated values of the evidence are used to deduce the probabilities $P(H_2|\mathbf{D})$ for different values of the exposure—the plot is shown in figure 7. In this case

the MultiNest calculated evidence is compared to a sampled means integration for comparison. The expected decrease in probability with exposure is seen as increased numbers of background events make it harder to defend the hypothesis of signal presence. It should be noted that for no value of the exposure could evidence be claimed given the criteria defined earlier. It is possible to extract the errors associated in calculation of the evidence for each algorithm—these are not explicitly displayed as they are largely dependent on running settings which do not translate readily between methods. Suffice to say, for an importance nested MultiNest run at an efficiency setting of 0.8 with 1000 live points compared to a sampled mean run over 1000 iterations, the relative error in the evidence is typically improved by between 2 and 3 orders of magnitude. The error bars shown therefore represent only the standard error in averaging and do not reflect the proper weighting—given the relative magnitudes of the two, this has little impact.

Though convenient in demonstrating a physical application in which the small number of events makes a Bayesian approach particularly appropriate, the expected profile for the Gerda experiment is unimodal and non-degenerate. In this sense, this example does not demonstrate fully the advantages and capacity of MultiNest, which outperforms MCMC most visibly in such situations. Examples of such scenarios can be found for example in [7].

7 Conclusions

An interface between two commonly used programs for Bayesian inference, the BAT and MultiNest was presented and the software used in the analysis of sparse spectra, specifically for hypothesis testing. The search for neutrinoless double beta decay at the Gerda experiment was taken as a specific example and data generated under various conditions. The criterion for signal discovery was examined in cases with and without signal contribution: in the latter case, limits were set on the half-life of the process and the Majorana neutrino mass as a function of the exposure. The most recent values of the relevant matrix elements restrict the mass to be $<$320 meV for values of the exposure $>$10 kg yr, assuming a background rate of 10^{-3} counts/kg$\cdot$ keV$\cdot$ yr.

The performance of the two algorithms in this task was compared; the BAT-called MultiNest routine was found to reproduce closely the MCMC-derived limits. It was also found to evaluate the evidence to a precision greater than that possible using the BAT sampled means functionality by up to 3 orders of magnitude. Application of the new interface to physical problems featuring multimodal and highly degenerate distributions may serve to further highlight the advantages of the nested sampling algorithm over traditional MCMC. A comparison of the relative precision of MultiNest and the Cuba library functionality has also been made possible, and may prove useful in determining the suitability of each to a given type of problem.

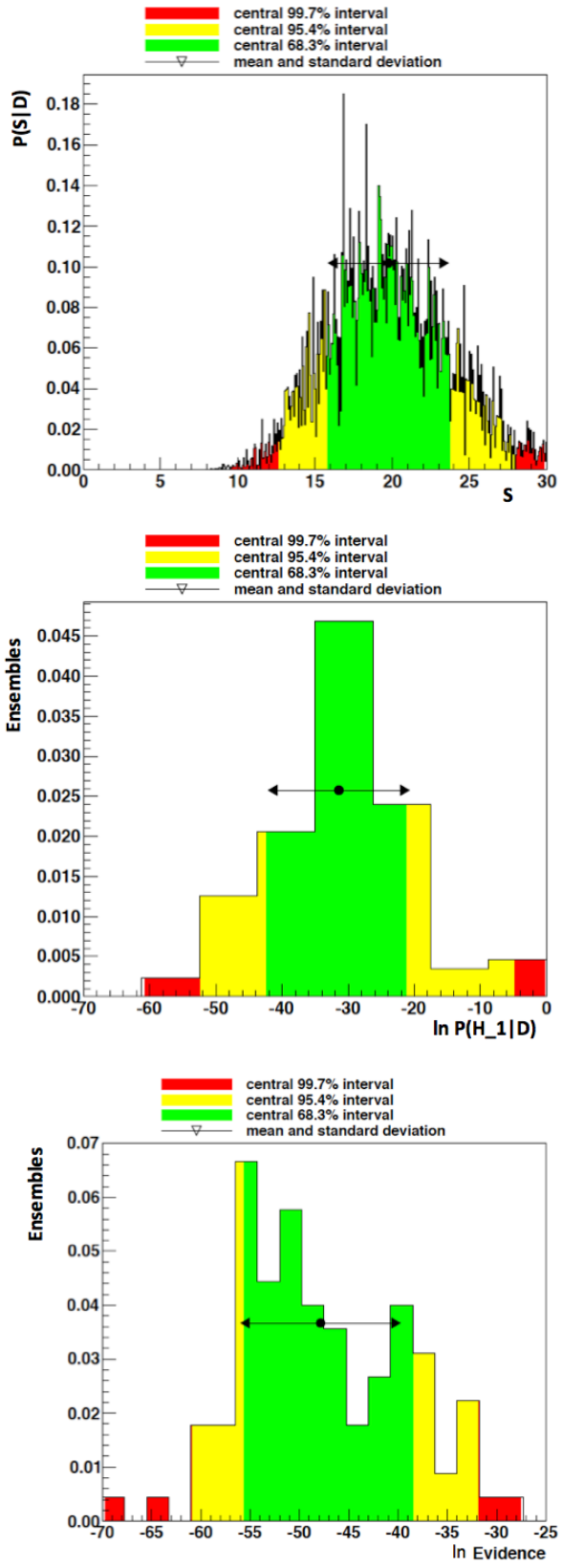

Figure 4: The probability density for an ensemble with 20 signal and 10 background events is shown (top). Below are shown the logarithms of the evidence (bottom) and marginalised probability (middle) for 100 such ensembles generated in this way.

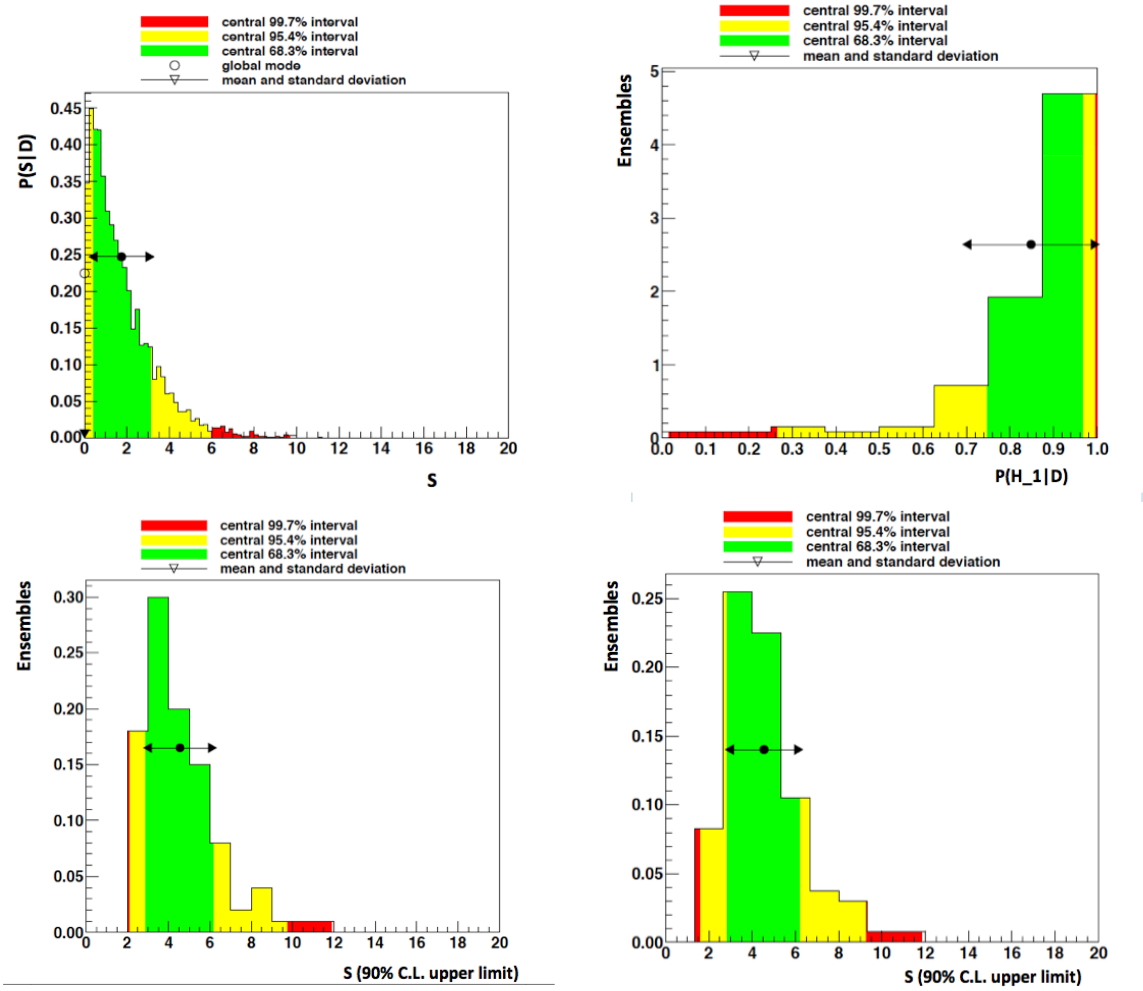

Figure 5: The probability density for an ensemble generated with no signal events and 20 background events is shown top left; top right gives the marginalised probability for 100 such ensembles. Bottom are shown the 90% credibility limits set on the signal by MCMC and MultiNest respectively.

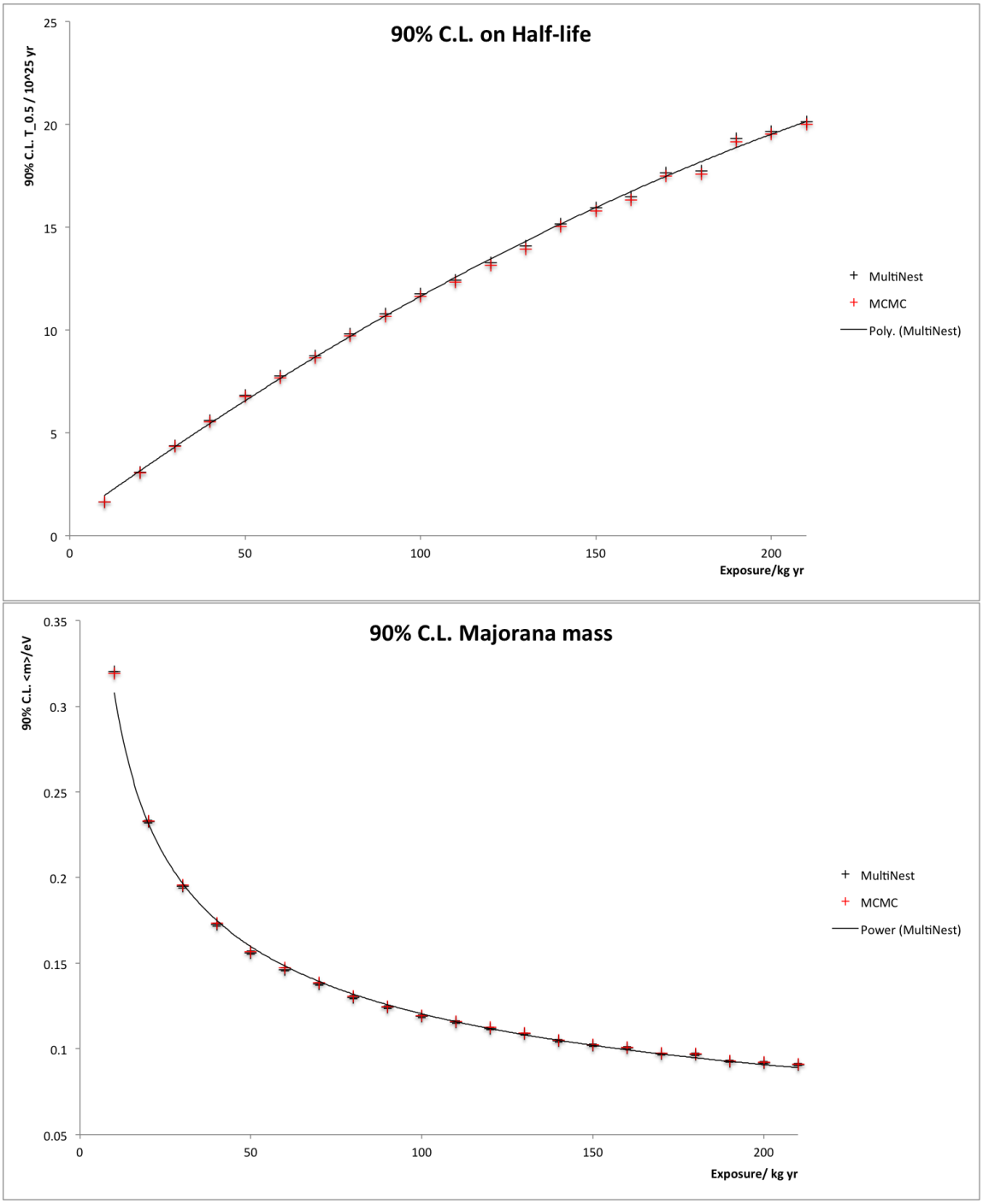

Figure 6: Top: the 90% probability lower limit on the process half-life as a function of exposure. Here a background rate of 10^{-3} counts/(kg keV yr), the target rate for GERDA, is assumed. A fit line has been added to guide the eye. Error bars are too small to be well distinguished and have been removed for clarity; it is apparent that MULTINEST and MCMC predictions are almost equivalent. Bottom: the half-life limit translated to a limit on the neutrino mass. A value of $\langle M^{0\nu} \rangle = 5.82$ has been assumed, following [17]. The phase space factor $G^{0\nu} = 0.1776$ has been used from [16]. The updated matrix elements provide a more tightly constrained upper limit on the mass than that calculated in [3] but this varies depending on the theoretical values used.

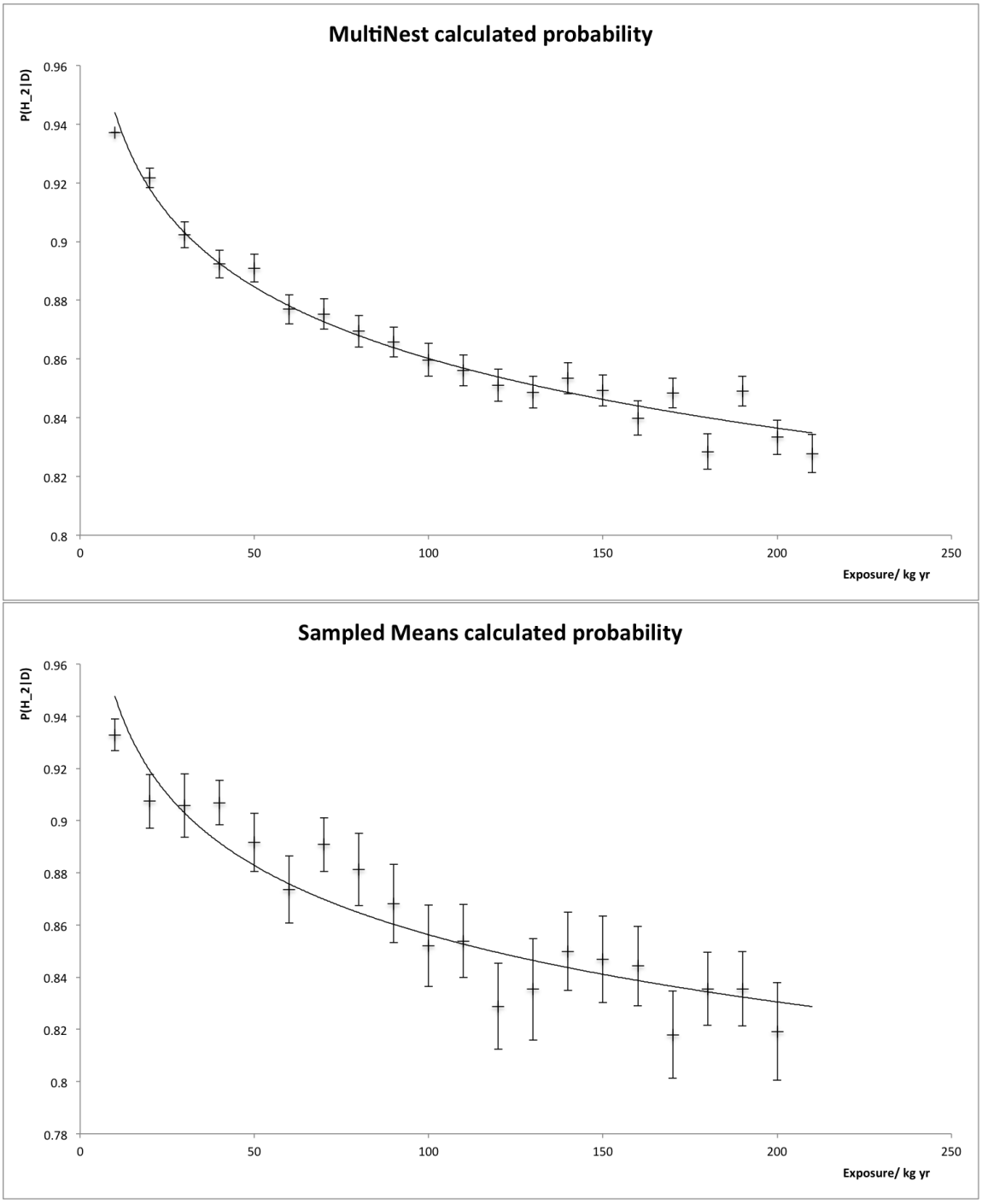

Figure 7: Top: the probability $P(H_2|\mathbf{D})$ generated by MULTINEST assuming signal presence. A trend line has been added to guide the eye. As the exposure increases, the probability drops off—in no case can evidence for the process be claimed. Bottom: as above, generated by a sample means procedure. Note the greater variability about the trend line compared to MULTINEST predictions.

Appendices

A BCMultinestAdapter

The source code BCMultinestAdapter.cxx is displayed below. MULTINEST functionality in the BAT also requires BCMultinest.cxx, an adapted form of the C/C++ interface provided in the MULTINEST distribution.

```
#include "BCMultinestAdapter.h"
#include "BCIntegrate.h"
#include "BCParameter.h"
#include "TFile.h"
#include "TTree.h"
#include <math.h>
#include <cassert>
#include <iostream>
#include <fstream>

#include "BCH1D.h"
#include "BCLog.h"
#include <TH1D.h>
#include "BCH2D.h"
#include <TH2D.h>

BCMultinestAdapter::BCMultinestAdapter()
{
}

void BCMultinestAdapter::logLikelihood(double * Cube, int &
   ndim, int & npars, double & lnew, void * context) {
  BCIntegrate * local_this = static_cast<BCIntegrate *>(context
     );

   assert(size_t(ndim) == local_this->fParameters.Size());

     std::vector<double> scaled_parameters;
    double jacobian = 1.0;
  // rescale parameter from unit hypercube and update values
  for (int i = 0 ; i < npars ; ++i) {
      BCParameter * p = local_this->fParameters[i];
      if(p->Fixed()==false){
      double range = p->GetRangeWidth();
      Cube[i] = p->GetLowerLimit() + Cube[i] * range;
```

```
      scaled_parameters.push_back(Cube[i]);
      jacobian *= range;
      }
      else{
        Cube[i]=p->GetFixedValue();
        scaled_parameters.push_back(Cube[i]);
      }
  }

  lnew = local_this->LogEval(scaled_parameters);

  //Multiply by prior volume
  lnew += log(jacobian);
}

void BCMultinestAdapter::rootDumper(int &nSamples, int &nlive,
   int &nPar, double **physLive, double **posterior,
               double **paramConstr, double &maxLogLike,
                  double &logZ, double &logZerr, void *context
                  )
{
  BCIntegrate * local_this = static_cast<BCIntegrate *>(context
     );

TFile f("multinestoutput.root", "RECREATE");
        TTree tree("MultinestSamples", "MultinestSamples");

        std::vector<double> sample(nPar, 0.0);
        double weight;

        assert(size_t(nPar) == local_this->fParameters.Size());

        // setup branches, one for each parameter, and one for
           the posterior weight
        for (int i = 0; i < nPar ; ++i)
          tree.Branch(local_this->fParameters[i]->GetName().
             data(), &sample[i],  std::string(local_this->
             fParameters[i]->GetName()+std::string("/D")).c_str
             ()\
);
        tree.Branch("weight", &weight, "weight/D");

        // extract one sample at a time from Fortran order, see
```

```
    eggbox.cc dumper example
ofstream file;
file.open ("samples.txt");
for (int i = 0; i < nSamples ; ++i)
  {
        for (int j = 0; j < nPar ; ++j)
        {
                sample[j] = posterior[0][j * nSamples +
                    i];
                file<< j <<" "<<sample[j]<<std::endl;
        }
        weight = posterior[0][(nPar + 1) * nSamples + i
           ];
        file<<weight<<std::endl;
        tree.Fill();
}
file.close();

//Output quantiles to file; fill histograms
ofstream qfile;
qfile.open ("quantiles.txt");
for (int j = 0; j < nPar ; ++j) {
  double min = local_this->fParameters[j]->
     GetLowerLimit();
  double max = local_this->fParameters[j]->
     GetUpperLimit();
  TH1D * Hist = new TH1D(Form("Hist_%i", BCLog::
     GetHIndex()), "", 500, min, max);
  for (int i = 0; i < nSamples ; ++i)
    {
      Hist->Fill(posterior[0][j * nSamples + i],
         posterior[0][(nPar + 1) * nSamples + i]);
    }
    BCH1D * bchist = new BCH1D(Hist);
  bchist->Print(Form("%s_multinest.pdf",local_this->
     fParameters[j]->GetName().data()));
  double quantile = bchist->GetQuantile(0.9);
  if(j!=0){
  qfile<<quantile<<std::endl;
  }
}
qfile.close();
```

```
        //Loop over all combinations of two parameters and fill
            histograms
        for (int j=0; j< nPar-1; ++j) {
          for (int k=j+1; k< nPar; ++k){
          double min_1, min_2, max_1, max_2;
          TH2D * Hist2 = new TH2D(Form("Hist_%i", BCLog::
              GetHIndex()), "", 500, min_1, max_1, 500, min_2 ,
              max_2);
          min_1 = local_this->fParameters[j]->GetLowerLimit();
          min_2 = local_this->fParameters[j+k]->GetLowerLimit()
              ;
          max_1 = local_this->fParameters[j]->GetUpperLimit();
          max_2 = local_this->fParameters[j+k]->GetUpperLimit()
              ;
          for (int i = 0; i < nSamples ; ++i)
            {
              Hist2->Fill(posterior[0][j * nSamples + i],
                 posterior[0][(j+k) * nSamples+i],posterior
                 [0][(nPar + 1) * nSamples + i]);
            }
          BCH2D * bchist2 = new BCH2D(Hist2);
          bchist2->Print(Form("2D_%i_%i.pdf", j, k));
          }
        }
        tree.Write();
        f.Flush();
        //Output evidence on screen
        local_this->evidence = logZ;
        double ev = exp(logZ);
        std::cout<<"Evidence "<<ev<<std::endl;
        return;
}
```

B Implementing the MultiNest interface

Below is shown the file used to run analysis using MultiNest on data created beforehand using a ROOT macro. The MCMC case with sampled mean normalisation is implemented similarly—this serves to show the versatility of the BAT in working with pre-defined models.

```
#include <BAT/BCLog.h>
#include <BAT/BCAux.h>
#include <BAT/BCSummaryTool.h>
```

```
#include <BAT/BCMTFAnalysisFacility.h>
#include <BAT/BCMTF.h>
#include <BAT/BCMTFChannel.h>


#include "BAT/BCH1D.h"
#include <TFile.h>
#include <TH1D.h>
#include <TTree.h>
#include <fstream>
#include <math.h>
#include <stdlib.h>
#include <iostream>
int main(int argc, char* argv[])
{
   // ---- set style and open log files ---- //

   // set nicer style for drawing than the ROOT default
   BCAux::SetStyle();

   // open log file
   BCLog::OpenLog("log.txt");
   BCLog::SetLogLevel(BCLog::detail);

   // ---- read histograms from a file ---- //

   // open file
   std::string fname = "templates.root";
   TFile * file = new TFile(fname.c_str(), "READ");

   // check if file is open
   if (!file->IsOpen()) {
      BCLog::OutError(Form("Could not open file %s.",fname.
         c_str()));
      BCLog::OutError("Run macro CreateHistograms.C in Root to
         create the file.");
      return 1;
   }

   //open evidence output
   ofstream efile;
   efile.open ("evidenceout.txt");
```

```
//open quantile output
//ofstream outfile;
//outfile.open ("90quantilesMultinest.txt");
ofstream outfile("90quantilesMultinest.txt", std::ios::app);
ofstream probability("Multinestprob.txt", std::ios::app);

// read template histograms
TH1D hist_signal      = *((TH1D *) file->Get("hist_sgn"));
     // signal template
TH1D hist_background  = *((TH1D *) file->Get("hist_bkg"));
     // background template

int ensembles=700;
int i=0;
std::vector<double> evidence, evidence_null, marginalized;

//evidence histogram
double min=-210, max=-185.0;
TH1D* ehist = new TH1D("evidence", "Evidence;Ensembles",
   100, min, max);

//marginalized histogram
double min_m=0, max_m=1.0;
TH1D* mhist = new TH1D("marginalized", "Ensembles;␣p(H|Data)
   ", 10, min_m, max_m);

//90% limit histogram
double min_n=0, max_n=12;
TH1D* nhist = new TH1D("90%␣Upper␣C.L.", "Ensembles;S(90%␣
   Upper␣C.L.)", 20, min_n, max_n);

// loop over ensembles
while (i< ensembles)
  {
    double events;
    events = atof(argv[1]);
// read data histograms
    TH1D hist_data       = *((TH1D *) file->Get(Form("
       hist_data_%i",i))); // data for channel 1

// ---- perform fitting ---- //

// create new fitter object
```

```
    BCMTF * m = new BCMTF("SingleChannelMTF");
    BCMTF * n = new BCMTF("SingleChannelMTFNull");

// set the required precision of the MCMC (kLow, kMedium,
   kHigh)
// the higher the precision the longer the MCMC run
   //m->MCMCSetPrecision(BCEngineMCMC::kHigh);
   //n->MCMCSetPrecision(BCEngineMCMC::kHigh);

// add channels
   m->AddChannel("channel1");
   n->AddChannel("channel1");


// add processes
   m->AddProcess("background", 0., 30);
   m->AddProcess("signal", 0., 20.);
   n->AddProcess("background", 0., 30);

// set data
   m->SetData("channel1", hist_data);
   n->SetData("channel1", hist_data);
// set template and histograms
   m->SetTemplate("channel1", "signal",     hist_signal,
      0.87);
   m->SetTemplate("channel1", "background", hist_background,
       1.0);
   n->SetTemplate("channel1", "background", hist_background,
       1.0);
// set priors
   m->SetPriorGauss("background", events, events*0.5);
   m->SetPriorConstant("signal");
   n->SetPriorGauss("background", events, events*0.5);

// run Multinest

   ifstream infile;
   double data;
   efile<<m->IntegrateMultinest()<<std::endl;
   evidence.push_back(m->IntegrateMultinest());

   //Get quantiles and fill
     infile.open("quantiles.txt");
```

```
      infile>>data;
      nhist->Fill(data);
      outfile<<data<<std::endl;

   //ehist->Fill(evidence[i]);
   evidence_null.push_back(n->IntegrateMultinest());

     //Option for log;
   //marginalized.push_back(log(exp(evidence_null[i])/(exp(
      evidence_null[i])+exp(evidence[i]))));
   marginalized.push_back(exp(evidence_null[i])/(exp(
      evidence_null[i])+exp(evidence[i])));
   efile<<marginalized[i]<<std::endl;
   mhist->Fill(marginalized[i]);

   std::cout<<"RUN"<<i<<std::endl;
i++;
delete m;
delete n;

}

// ---- clean up ---- //

// close log file
BCLog::CloseLog();

//Write evidence histogram
BCH1D * bchist = new BCH1D(ehist);
bchist->Print(Form("MultinestEvidence.pdf"));

//Write marginalised histogram
BCH1D * bchist_2 = new BCH1D(mhist);
bchist_2->Print(Form("MultinestMarginalized.pdf"));
double meanprob = bchist_2->GetMean();
double rmsprob = bchist_2->GetRMS();
probability<<meanprob<<"\t"<<rmsprob<<std::endl;


//Write 90% C.L. histogram
BCH1D * bchist_3 = new BCH1D(nhist);
bchist_3->Print(Form("Multinest90CL.pdf"));
double mean = bchist_3->GetMean();
```

```
    double rms = bchist_3->GetRMS();
    std::cout<<mean<<"\t"<<rms<<std::endl;
    outfile<<mean<<"\t"<<rms<<std::endl;
    // close files
    efile.close();
    outfile.close();
    probability.close();

    return 1;
}
```

References

[1] A. Caldwell, D. Kollar, and K. Kroeninger, Computer Physics Communications **180**, 2197 (2009).

[2] T. Hahn, Nucl.Instrum.Meth. **A559**, 273 (2006), arXiv:hep-ph/0509016 [hep-ph] .

[3] A. Caldwell and K. Kroeninger, Phys. Rev. D **74**, 092003 (2006).

[4] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, The Journal of Chemical Physics **21**, 1087 (1953).

[5] J. Skilling, AIP Conference Proceedings **735**, 395 (2004).

[6] J. Skilling, Bayesian Anal. **1**, 833 (2006).

[7] F. Feroz and M. Hobson, Mon.Not.Roy.Astron.Soc. **384**, 449 (2008), arXiv:0704.3704 [astro-ph] .

[8] F. Feroz, M. Hobson, and M. Bridges, Mon.Not.Roy.Astron.Soc. **398**, 1601 (2009), arXiv:0809.3437 [astro-ph] .

[9] F. Feroz, M. Hobson, E. Cameron, and A. Pettitt, (2013), arXiv:1306.2144 [astro-ph.IM] .

[10] Adare, A. et al (PHENIX Collaboration), Phys. Rev. C **81**, 034911 (2010).

[11] S. S. et al., Nuclear Physics B - Proceedings Supplements **145**, 242 (2005), proceedings of the Neutrino Oscillation Workshop.

[12] M. Agostini *et al.* (GERDA Collaboration), (2013), arXiv:1307.4720 [nucl-ex] .

[13] H. Klapdor-Kleingrothaus, I. Krivosheina, A. Dietz, and O. Chkvorets, Physics Letters B **586**, 198 (2004).

[14] Aalseth, C. E. et al. ((IGEX Collaboration)), Phys. Rev. D **65**, 092007 (2002).

[15] H. V. Klapdor-Kleingrothaus, I. V. Krivosheina, and S. N. Karpov, (2013), arXiv:1308.2524 [hep-ex] .

[16] J. Kotila and F. Iachello, Phys. Rev. C **85**, 034316 (2012).

[17] P. Bhupal Dev, S. Goswami, M. Mitra, and W. Rodejohann, (2013), arXiv:1305.0056 [hep-ph] .