RooMCMarkovChain A METROPOLIS–HASTINGS ALGORITHM FOR THE ROOT FRAMEWORK*

Oliver Dahme

University of Zurich, Zurich, Switzerland
oliver.dahme@uzh.ch

(Received April 20, 2018)

This paper reports work done during an internship at the LHCb experiment at CERN. The task was to implement a Metropolis algorithm as a minimizer for negative log likelihood (nnl) fits into the ROOT Data analysis framework. The Metropolis algorithm is based on a MCMC which w.r.t. previously broadly used algorithm called Minuit can easily be scaled to multidimensional parameter space. Moreover, such kind of algorithms can easily be parallelized.

DOI:10.5506/APhysPolB.49.1097

1. Monte Carlo Markov Chain (MCMC)

1.1. Introduction

A Markov Chain is a sequence of random events $X_1, X_2, \ldots$, where the conditional distribution of X_{n+1} depends only on X_n. The space where the X_i take values is called the state space of the Markov chain, written $x_1, \ldots, x_n$. If the conditional distribution is independent of n, the Markov Chain has stationary transition probabilities. The joint distribution of a Markov chain is determined by:

— The marginal distribution of X_1 called the initial distribution;

— The conditional distribution of X_{n+1} called the transition distribution.

If the state space is finite, then the initial distribution can be associated with a vector $\lambda = (\lambda_1, \ldots, \lambda_n)$ defined by the following probability Pr:

$$\Pr(X_i = x_i) = \lambda_i \,. \tag{1}$$

* Presented at the Cracow Epiphany Conference on Advances in Heavy Flavour Physics, Kraków, Poland, January 9–12, 2018.

The transition probabilities can be written in terms of P matrices, whose elements p_{ij} are defined by

$$\Pr(X_{n+1} = x_j | X_n = x_i) = p_{ij} \tag{2}$$

that only holds if the state space is finite. Most Markov chains of interest in MCMC have an uncountable state space. The initial distribution then becomes an unconditional distribution and the transition probability distribution a conditional probability distribution.

1.2. Theory

Suppose one wishes to calculate the expectation value

$$\mu = E(g(X)) \,, \tag{3}$$

where g is a real-valued function on the state space. Often there is no exact method that can compute the expectation value. Suppose it is possible to simulate $X_1, X_2, \ldots$ independent identically distributed having the same distribution as X. Let us define an estimator

$$\hat{\mu}_n = \frac{1}{n} \Sigma_{i=1}^{n} g(X_i) \,. \tag{4}$$

Introducing the notation $Y_i = g(X_i)$, the Y_i are independent identically distributed with mean μ and variance

$$\sigma^2 = \mathrm{var}(g(X)) \,. \tag{5}$$

The $\hat{\mu}_n$ is the sample mean of the Y_i and according to the Central Limit Theorem

$$\hat{\mu}_n \approx N\left(\mu, \frac{\sigma^2}{n}\right) , \tag{6}$$

$$\hat{\sigma}_n^2 = \frac{1}{n} \Sigma_{i=1}^{n} (g(X_i) - \hat{\mu}_n)^2 \,. \tag{7}$$

1.3. Implementation

1.3.1. Basic

The most basic implementation of a Markov Chain Monte Carlo program follows the basic idea:

Algorithm 1 basic MCMC pseudo code

```
Initialize x
loop
    Generate random change to x
    Output x
```

It is important that x is the entire state of the program. If for example the movement of a particle is simulated, one could apply random changes to the velocity vector at each time step, but if sometimes the changes are not random, the process is not a Markov any more and all the mathematical formalism described in Sec. 1 does not hold any more. The state where random changes are applied to x is called an update mechanism. This report focuses on update mechanisms that preserve a specified distribution. That means the distribution before the update is the same after the update. From that, one can construct the Markov chains to sample that distribution. An update mechanism is called elementary, when the mechanism is not made up of parts. Since there is not much structure in Algorithm 1, most simulation can be fit into this format.

1.3.2. Metropolis–Hastings

Suppose that the specified distribution has unnormalized density h. This means that h is a positive constant multiplied by a probability density. Therefore, h is a non-negative function that integrates (for continuous states) or sums (for discrete states) to a value that is finite and non-zero. The Metropolis–Hastings update does the following:

— The current state x is proposed to move to y with a conditional probability density given x denoted as $q(x, \cdot)$;

— Calculate the Hastings ratio

$$r(x, y) = \frac{h(y)\, q(y, x)}{h(x)\, q(x, y)}; \tag{8}$$

— Accept the proposed, move to y with the probability

$$a(x, y) = \min(1, r(x, y)). \tag{9}$$

The Hastings ratio (Eq. (8)) is undefined if $h(x) = 0$, thus one has to ensure that $h(x) > 0$ in the initial state. If $h(y) = 0$, there is no problem since $r(x, y)$ also becomes zero and the probability to accept y just becomes zero. Note that the proposed y must satisfy $q(x, y) > 0$ because $q(x, \cdot)$ is the conditional density of y given x. Hence $h(x) > 0$, the denominator of the Hastings ratio is always non-zero and well-defined. The numerator does not have to be non-zero, because if $h(y) = 0$, then y is an impossible value of the desired distribution, and if $q(y, x) = 0$, then x is an impossible proposal when y is the current state.

At this point, it is important to stress out that these properties are very fortunate since the Metropolis–Hastings update automatically does the right thing, almost surely rejecting such proposals. Therefore, it is not necessary to ensure that the proposals have to be possible values of the desired distribution. The only thing necessary is to assure that the unnormalized density function h works with every proposal and gives $h(y) = 0$ if y is an impossible proposal.

2. RooMCMarkovChain

This section focuses on the technical part of the implementation of the RooMCMarkovChain class into the ROOT data analysis framework. RooMCMarkovChain is variation of the RooMinuit class, which is part of the Roofit environment. It is made transparent to the user, in other words, the user can use the RooMCMarkovChain class as he did the RooMinuit class. The task of both is to fit a probability density function (pdf) to data. That is achieved by minimizing the negative log-likelihood function. RooMCMarkovChain uses a Metropolis–Hastings algorithm. As explained in Sec. 1.3.2, the Metropolis–Hastings is a Monte Carlo updater. In the following, it will be explained how RooMinuit uses this principle to minimize a negative log-likelihood function.

In the RooMCMarkovChain class, an event is a point on the nll. In the next step, a point in the vicinity of the current point is chosen as a proposal. The Hastings ratio (Eq. (8)) is the probability for this proposed point to be part of the Markov Chain or not. Here, the Hastings ratio is directly proportional to the difference between the nnl values of the current and the proposed point, while there is a higher probability for negative differences than for positive ones. It follows that the chain will end up in a minimum. After finding the minimum, it starts oscillating around it, that provides a good scan of the vicinity around the minimum. This scan is used to calculate an estimate on the errors of the parameters of interest. The exact description of the algorithm used can be found in [1].

The following subsections are meant as a code documentation, therefore, all the functions of the RooMCMarkovChain class are well-described and an example of their output is given. All examples are taken from a fit to 1000 simulated data points of a double "gaus" with this pdf

$$\mathrm{pdf}(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + f\,\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}\,, \tag{10}$$

where $\mu_1 = 4$, $\sigma_1 = 1$, $\mu_2 = -2$, $\sigma_2 = 1.5$ and the fraction $f = 0.5$.

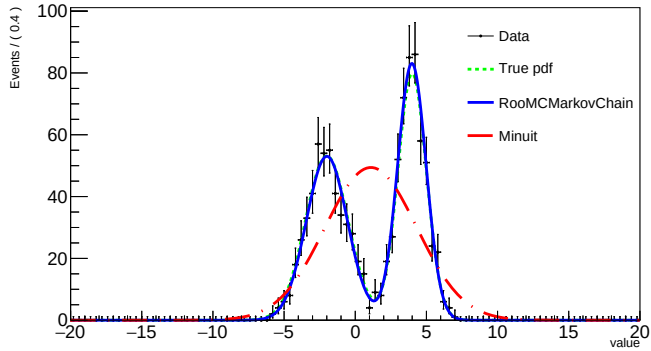

Fig. 1. (Color online) Fit of a double Gauss: The first Gauss has a mean of 4 and a sigma of 1, the second Gauss has a mean of −2 and a sigma of 1.5. The dotted/green line represents the true pdf. The solid/blue line has been fitted by the RooMCMarkovChain class, while the dashed/red line has been fitted by Minuit. For some unknown reason, Minuit sometimes fails to fit even such a simple pdf.

2.1. mcmc

The RooMCMarkovChain::**mcmc**(**int** npoints, **int** cutoff, **string** errorstrategy) function does the main work of the class. It performs the MCMC to find the minimum. On the way, it saves all the accepted points. The value "npoints" is an integer, which declares the total number of accepted points the chain will have at the end. The value "cutoff" is an integer, which declares after how many points the cutoff is performed. Cutoff means that the points at the beginning of the chain until the cutoff are not included into the error calculation. That reduces the variance on the fitting parameters significantly. Moreover, one can choose between two errorstrategies: "gaus" assumes symmetric errors, while "interval" assumes non-Gaussian errors. The results of the fit are displayed in the terminal as shown in Fig. 2.

```
RooFit v3.60 -- Developed by Wouter Verkerke and David Kirkby
                Copyright (C) 2000-2013 NIKHEF, University of California & Stanford University
                All rights reserved, please read http://roofit.sourceforge.net/license.txt

Starting Monte Carlo Markov Chain Fit with 12000 points and cutoff after 7000 points
1% 2% 3% 4% 5% 6% 7% 8% 9% 10% 11% 12% 13% 14% 15% 16% 17% 18% 19% 20% 21% 22% 23% 24% 25% 26%
% 33% 34% 35% 36% 37% 38% 39% 40% 41% 42% 43% 44% 45% 46% 47% 48% 49% 50% 51% 52% 53% 54% 55% 5
 62% 63% 64% 65% 66% 67% 68% 69% 70% 71% 72% 73% 74% 75% 76% 77% 78% 79% 80% 81% 82% 83% 84% 85
91% 92% 93% 94% 95% 96% 97% 98% 99% 100%
NO.     NAME    VALUE           ERROR
1       frac    5.14084e-01     1.47307e-02
2       mean1   3.98243e+00     5.07205e-02
3       mean2    -1.99256e+00   7.51154e-02
4       sigma1  9.98988e-01     3.87089e-02
5       sigma2  1.44724e+00     5.86681e-02

CORRELATION COEFFICIENTS
NO.     1       2       3       4       5
1       1.000   -0.072  -0.043  0.095   -0.060
2       -0.072  1.000   0.131   -0.151  0.136
3       -0.043  0.131   1.000   -0.135  0.194
4       0.095   -0.151  -0.135  1.000   -0.171
5       -0.060  0.136   0.194   -0.171  1.000
```

Fig. 2. Terminal output of the mcmc function with errorstrategy set to "gaus".

2.2. getProfile

The RooMCMarkovChain::**getProfile**(**string** name, **bool** cutoff) function returns a profile of the nll for a certain parameter, which can be called by name. It does so by creating a TGraph and plots all the nll values of the walk in respect to the parameter. Moreover, it is possible to include or exclude the cutoff points. An example is given in Fig. 3

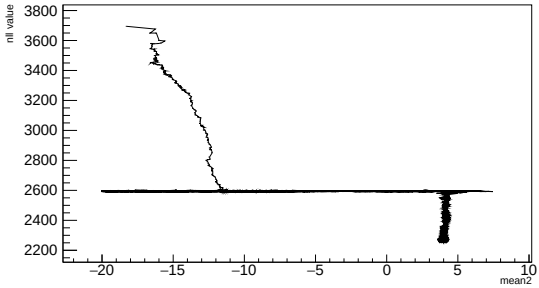

Fig. 3. Profile of the nll for the mean of the second Gaussian including the cutoff points. As one can see, the walk starts at −17 then finds a local minimum at the nll value of 2600, and then jumps out of it to find the global minimum at the true value of 4 for the mean of the second Gaussian.

2.3. getWalkDis

The RooMCMarkovChain::**getWalkDis**(**string** name, **bool** cutoff) function returns a TMultigraph pointer of the walk distribution of a parameter, which is called by name. It does so by creating two TGraphs: one with the points which had been cutoff and one with the included points. Besides, it adds a dotted line where the cutoff has been set. An example is given in Fig. 4.

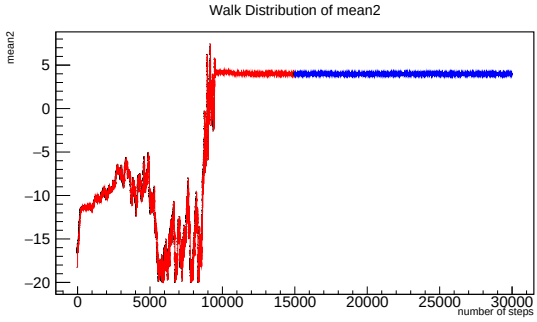

Fig. 4. (Color online) Walk Distribution of the mean of the second Gauss of a double Gauss fit. The true value is 4, while the starting point is −17. In light gray/red are the points which have been cut off and in dark gray/blue the rest points which are used for the error calculation, (see Sec. 2.4).

2.4. getWalkDisHis

The RooMCMarkovChain::**getWalkDisHis**(**string** name, **int** nbinsx, **bool** cutoff) returns a TH1F pointer with a histogram of the walk for a certain parameter, called by name. The number of bins for the histogram can be set by nbinsx. Cutoff points can be included or not. It does so by just adding all the points of the walk to a histogram. The main purpose is to look at the distribution of the points. This function is also used to calculate the symmetric errors of the parameter. An example is given in Fig. 5.

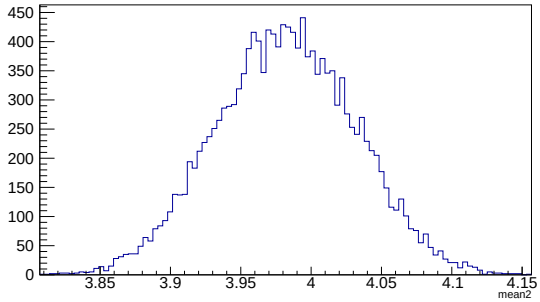

Fig. 5. Histogram of the walk for the mean of the second Gauss. As one can see, the true value 4 is in the error range around the mean of the histogram, since the values have a Gaussian distribution.

2.5. getCornerPlot

The RooMCMarkovChain::**getCornerPlot**(**string** name1, **string** name2, **int** nbinsx, **int** nbinsy, **bool** cutoff) function returns a TH2D pointer with a 2D histogram of two parameters, called by name1 and name2. The number of bins for the name1 parameter are set by nbinsx and for name2 by nbinsy. It does so just by adding all the points of the two parameters in a TH2D histogram. As always, the cutoff can be turned on or off. This plot could, for example, be used to look for correlations. An example is given in Fig. 6.

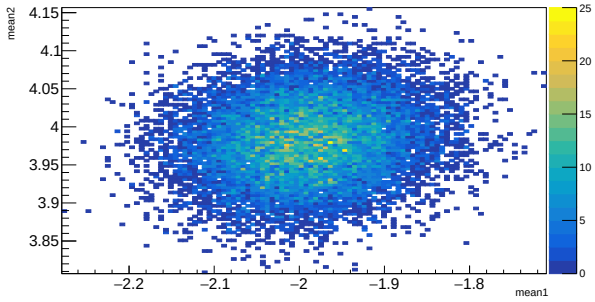

Fig. 6. Scatterplot of the two means of the double Gauss fit.

2.6. saveCornerPlotAs

The RooMCMarkovChain::**saveCornerPlotAs**(**string** picname) is the most complex function of the RooMCMarkovChain class. It creates a histogram of every parameter with getWalkDisHis and a corner plot with every pair of parameters. It can be used to see any correlations between the parameters. The histograms can be used to see graphically if a parameter has an asymmertric error or if it has a Gaussian distribution. Picname defines the name of the output file. An example is given in Fig. 7.

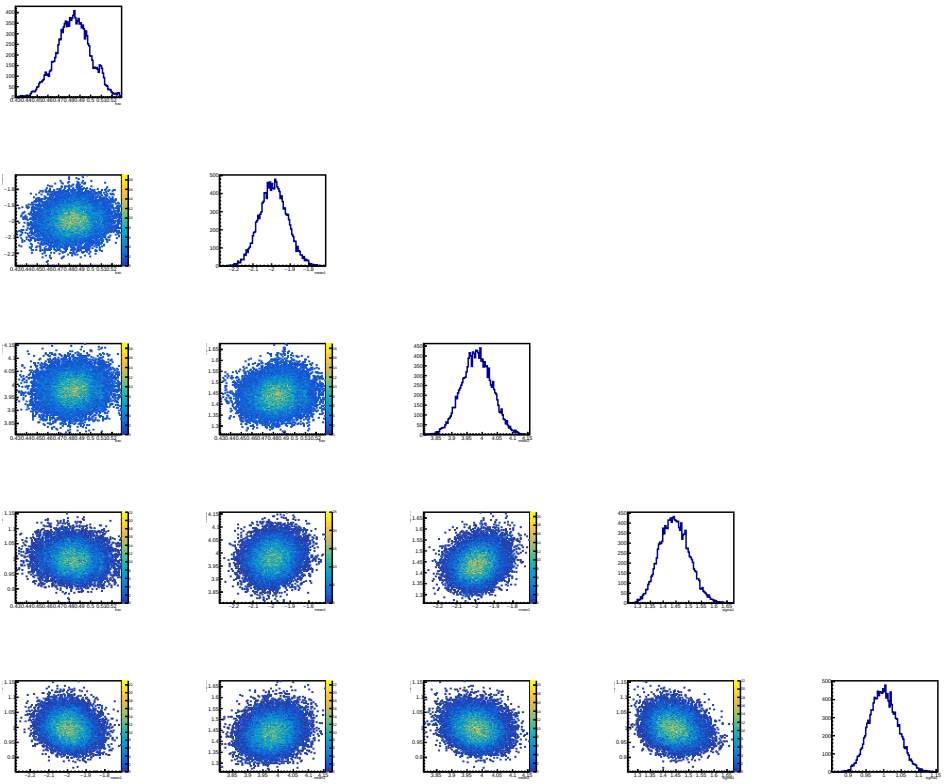

Fig. 7. Scatter plots between all the parameters of the double Gauss fit and the walk distributions of every parameter.

REFERENCES

[1] M. Vihola, *Stat. Comput.* **27**, 997 (2012).

[2] http://www.physik.uzh.ch/~odahme/example_RooMCMarkovChain/