Development of an improved fitting routine. Classification and Reweighting of the $B \to K^* \mu\mu$ decay

Oliver Dahme

University of Zurich

o.dahme@cern.ch

June 16, 2018

Supervisors: Prof. Dr. Nicola Serra, Dr. Marcin Chzsaszcz

Overview

1. LHCb Detector
2. The $B_0 \to K^* \mu^+ \mu^-$ decay
3. Test of Classifiers
4. Reweighting
5. RooMCMarkovChain

The LHCb Detector

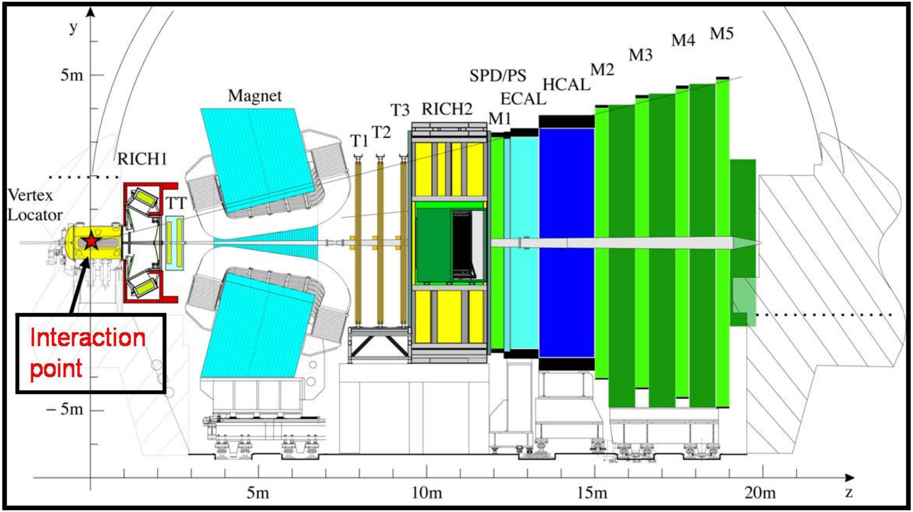

Figure: Detector overview

The $B_0 \to K^*\mu^+\mu^-$ decay

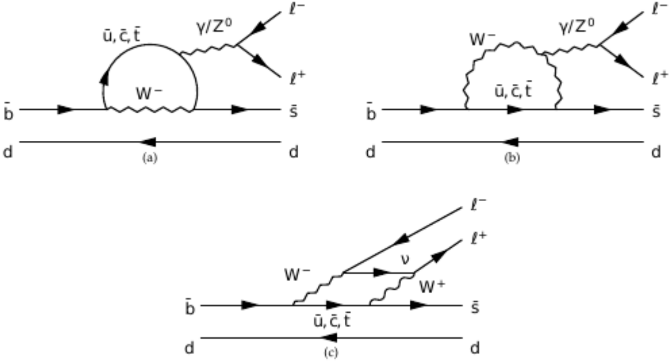

Figure: Feynman diagrams for decay $B(\bar{b}, d) \to K^*(\bar{s}, d)l^+\ l^-$ at lowest order

$$R_{K*0} = \frac{\mathcal{B}(B^0 \to K^{*0}\mu^+\mu^-)}{\mathcal{B}(B^0 \to K^{*0}J/\psi(\to \mu^+\mu^-))} \Bigg/ \frac{\mathcal{B}(B^0 \to K^{*0}e^+e^-)}{\mathcal{B}(B^0 \to K^{*0}J/\psi(\to e^+e^-))} \,,$$

$$R_{K*0} = \begin{cases} 0.66^{+0.11}_{-0.07}(\text{stat}) \pm 0.03(\text{syst})\text{for}0.045 < q^2 < 1.1 \text{ GeV}^2/c^4, \\ 0.69^{+0.11}_{-0.07}(\text{stat}) \pm 0.05(\text{syst})\text{for}1.1 < q^2 < 6.0 \text{ GeV}^2/c^4. \end{cases} \tag{1}$$

Kinematics

Test of Classifiers

Test of Classifiers

Monte Carlo Markov Chain

A Markov Chain is a random process which undergoes several states.
From each state there is a probability distribution to change into another state or to stay.
Most important is the asumption that every next step just depends on the current state.

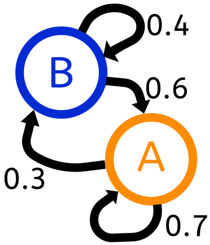

Figure: Example of states and probabilities of states in a Markov Chain

Metropolis-Hastings

- Current state x is proposed to move to y
- Calculate Hastings ratio:

$$r(x,y) = \frac{h(y) \cdot q(y,x)}{h(x) \cdot q(x,y)} \tag{2}$$

- Where $q(x,\cdot)$ is the conditional probability density and h is the unnormalized density of the specified distribution
- Accept the proposed move to y with the probability:

$$a(x,y) = min(1, r(x,y)). \tag{3}$$

The algorithm

The robust adaptive Metropolis process is defined recursively through

(R1) compute $Y_n := X_{n-1} + S_{n-1}U_n$, where $U_n \sim q$ is an independent random vector,

(R2) with probability $\alpha_n := \min\{1, \pi(Y_n)/\pi(X_{n-1})\}$ the proposal is accepted, and $X_n := Y_n$; otherwise the proposal is rejected and $X_n := X_{n-1}$, and

(R3) compute the lower-diagonal matrix S_n with positive diagonal elements satisfying the equation

$$S_n S_n^T = S_{n-1}\left(I + \eta_n(\alpha_n - \alpha_*)\frac{U_n U_n^T}{\|U_n\|^2}\right)S_{n-1}^T \tag{1}$$

where $I \in \mathbb{R}^{d\times d}$ stands for the identity matrix.

Figure: arXiv: 1011.4381v2

Here Y_n is the next state and X_{n-1} is the current state.
The S-maxtrix determs the direction and step size of the next step.

Example

fitting the folowing pdf:

$$\frac{1}{\sqrt{2\pi\sigma_1^2}}e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + f \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} \tag{4}$$

Example

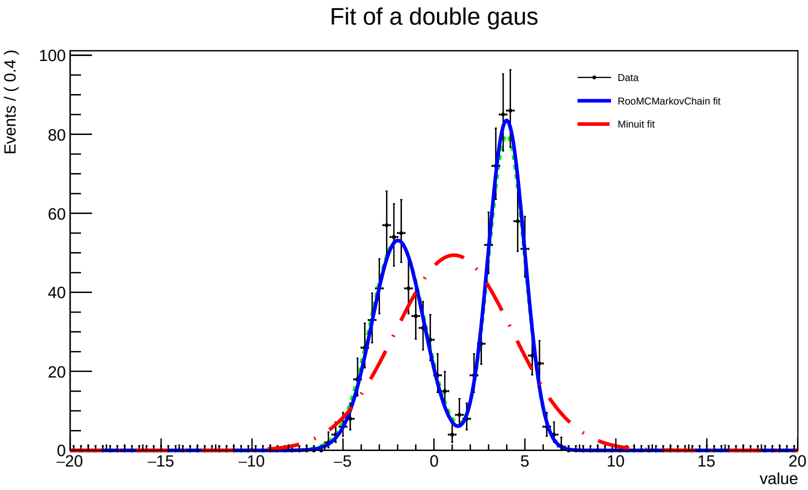

Figure: Fit of a double gaus

Features

- The constructor behaves like the RooMinuit constructor RooMCMC(RooAbsReal *negativeloglikelihood)
- then mcmc performes the walk and error calculation
- RooMCMarkovchain.**mcmc**(**int** npoints, **int** cutoff, **string** errorstrategy)
- there are two errorstategies: "gaus" for syemtric errors and "interval" for asymetric ones
- The terminal output is similar to the one of Minuit. It first prints the parameters with errors and then the correlation coefficients.

Features

- To look at the profile of the nll one can use:
- TGraph *profile = **getProfile**(**string** name, **bool** cutoff)
- with cutoff bool one can include or exclude the cutoff points

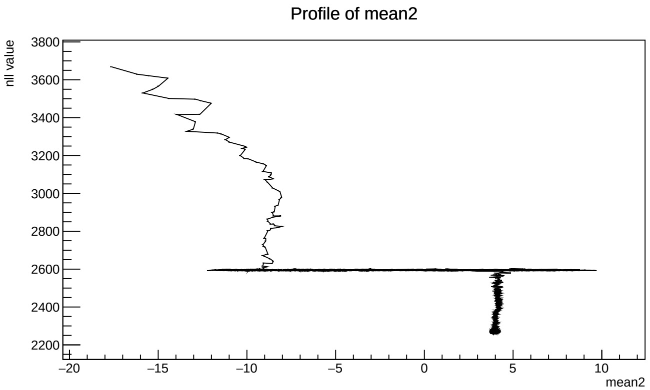

Figure: Profile from the mean of the second gaus

Features

- It is very important to look at the walk distribution, to check if the cutoff is well placed:
- TMultiGraph *walkdis = **getWalkDis**(**string** name, **bool** cutoff)

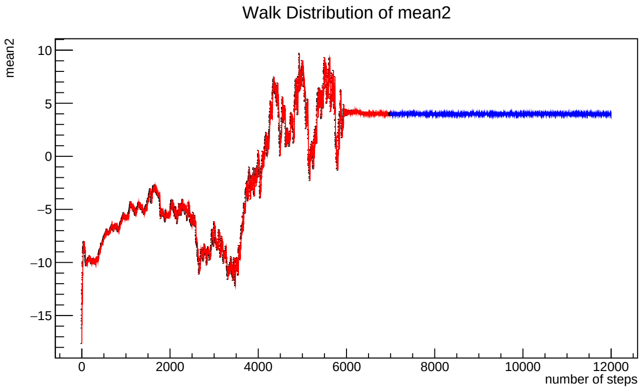

Figure: Walk distribution from the mean of the second gaus

Features

- It is also possible to get a histogram of the walk distribution, to check if the errors a symetric or asymetric.
- TH1F *walkdishis = **getWalkDisHis**(**string** name, **int** xbins, **bool** cutoff)

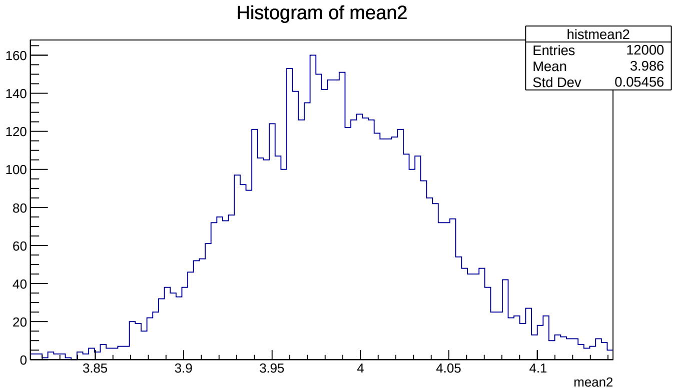

Figure: Histogram from the walk distribution of the mean of the second gaus

Features

- To check for correlations between parameters one can create a cornerplot between them.
- TH2D *corner = **getCornerPlot**(**string** name1, **string** name2, **int** nbinsx, **int** nbinsy, **bool** cutoff)

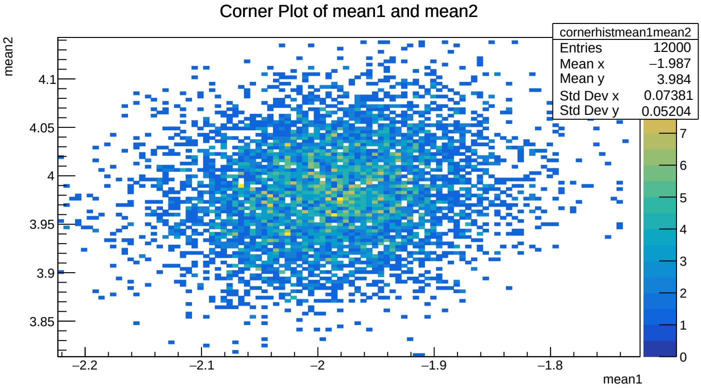

Figure: Cornerplot between the mean values of the two gaus.

Features

- Now to put everything together:
- **saveCornerPlotAs**(**string** picname)
- It saves a picture with a histogram of each parameter plus a correlation plot with each parameter pair.
- One can see directly if there is any correlation and if errors a symetric or asymetric

Features

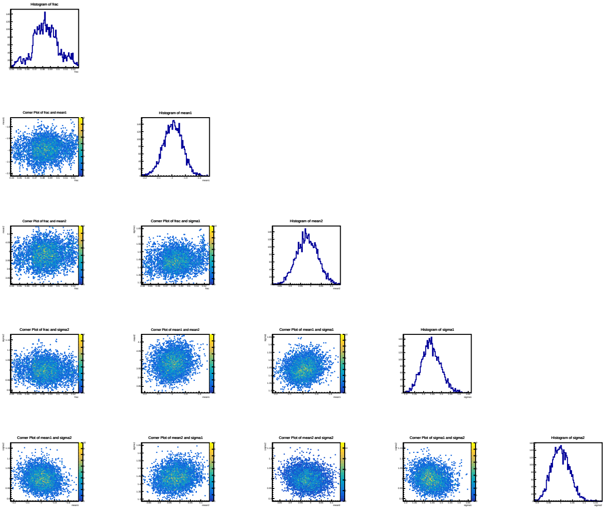

Figure: Corner Plot of the double gaus fit

Interested?

Pull Request: https://github.com/root-project/root/pull/1422

E-mail: o.dahme@cern.ch