UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

PhD School in Physics

Analisi dei Dati Sperimentali e confronto con Modelli Teorici

Dottorato di Ricerca in Fisica - Ciclo XXXI

Docente: Alexis Pompili

INFN Istituto Nazionale di Fisica Nucleare

Dipartimento Interateneo di Fisica di Bari

Parte teorica : **http://phdphysics.cloud.ba.infn.it/wp-content/uploads/2016/03/POMPILI-XXXI.pdf**

Parte pratica/esercitazione:

Esercitazione con *RooFit*

Dottorato di Ricerca in Fisica - Ciclo XXXI

Docente: Alexis Pompili

Si usa la macchina virtuale di ReCas: 212.189.205.223

Esercitazione n.1

Un manualetto essenziale di ROOT: www.l30-informatica.fisica.unimi.it/root.pdf

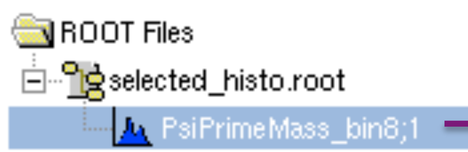
Innanzitutto eseguiamo il file di configurazione:

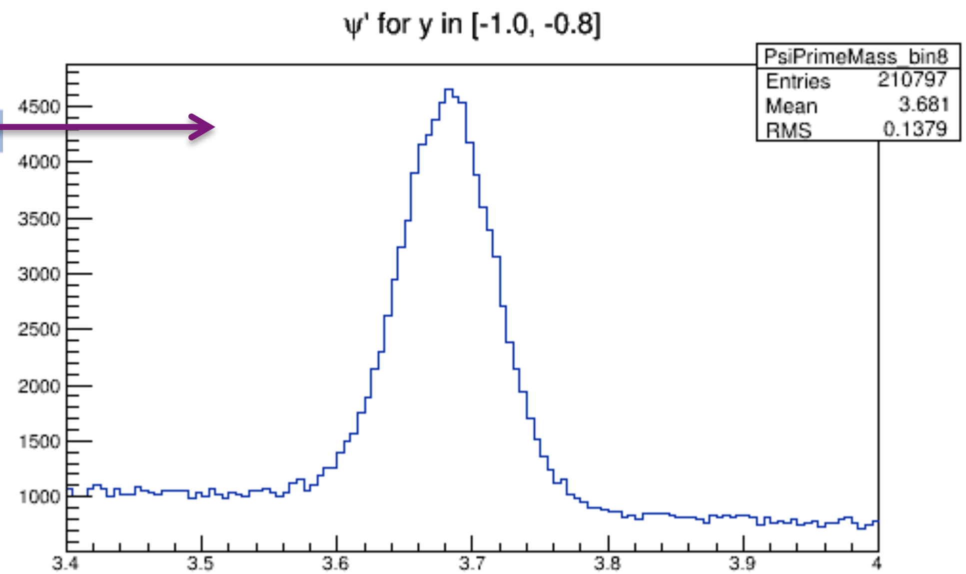
```
-bash-3.2$ source logincms_corso.sh
-bash-3.2$
```

Preliminarmente visualizziamo la distribuzione che deve essere interpolata:

```
[pompili@cmssusy esercitazione-5]$ root -l selected_histo.root
root [0]
Attaching file selected_histo.root as _file0...
root [1] TBrowser a
```

ROOT file di input





$m\left(\mu^+\mu^-\right)$

Si tratta dell'istogramma per la **massa invariante di una coppia di muoni**; il segnale (giaciente su un fondo combinatorio) rappresenta la particella $\psi'\left(\to \mu^+\mu^-\right)$

I candidati che entrano in questo plot sono caratterizzati da rapidita' $y_{\psi'} \in \left[-1.0,-0.8\right]$ [i dati sono di CMS], ma questo e' un dettaglio.

Per eseguire l'interpolazione basta fare (in ROOT) :

root [0] .x psiprime_fit.C

Macro file (C^{++} program)

Analizziamo la macro :

```
//////////////////////////////////
// run with root: .x psiPrime_fit.C
//////////////////////////////////
#include <vector>

gROOT->Reset();
gROOT->Clear();

using namespace RooFit;

void psiPrime_fit() {
  gROOT->ForceStyle();
  gStyle->SetTitleOffset(1.4, "Y");
  gStyle->SetOptFit(1);

  TFile* f1 = TFile::Open("./Select/selected_histo.root","read");

  TH1F* hPsiPrime;
  hPsiPrime = (TH1F*) f1->Get("PsiPrimeMass_bin8");

  TCanvas *myC = new TCanvas("myC","PsiPrimeMassPlot", 700, 700);

  Double_t xMin = hPsiPrime->GetXaxis()->GetXmin();
  Double_t xMax = hPsiPrime->GetXaxis()->GetXmax();
  Int_t nBins = hPsiPrime->GetNbinsX();

  RooRealVar xVar("xVar", "m(#mu^{+}#mu^{-}) [GeV/c^{2}]", xMin, xMax);
  xVar.setBins(nBins);

  RooDataHist* MuMuHist = new RooDataHist("#mu#mu_hist", hPsiPrime->GetTitle(), RooArgSet(xVar), Import(*hPsiPrime,kFALSE));
```

Allo scopo di usare il RooFit workspace

Apre rootupla esterna e ne prende l'istogramma d'interesse

Definisce variabile reale (massa invariante μμ) di RooFit: $m_{\mu\mu}$

Definisce istogramma di RooFit associato alla variabile reale precedentemente introdotta

modello per il segnale : PDF gaussiana

$$G_{SIG}(m_{\mu\mu})$$

```
RooRealVar mG("mean", "mean", 3.7, 3.67, 3.73);
RooRealVar sigma1("#sigma_{1}", "sigma1", 0.02, 0.001, 0.1);

RooGaussian sigPDF("sigPDF", "Signal", xVar, mG, sigma1);
```

un modello per il fondo (assunto lineare): polinomiale di ord.1 (con polinomi di Chebyshev) :

$$C_{BKG}(m_{\mu\mu})$$

```
RooRealVar c1("c_{1}", "c1", -0.1 ,-10, 10);
RooRealVar c2("c_{2}", "c2", -0.1 ,-10, 10);
RooChebychev bkgPDF("bkgPDF", "bkgPDF", xVar, RooArgSet(c1,c2));

RooRealVar nSig("nSig", "Number of signal candidates ", 2e+5, 1., 1e+6);
RooRealVar nBkg("nBkg", "Bkg component", 120e+3, 1., 1e+6);

RooAddPdf* totalPDF = new RooAddPdf("totalPDF", "totalPDF", RooArgList(sigPDF, bkgPDF), RooArgList(nSig, nBkg));

totalPDF->fitTo(*MuMuHist, Extended(kTRUE));
```

modello complessivo per segnale + fondo : combinazione lineare di segnale e fondo

$$n_{SIG} \cdot G_{SIG}(m_{\mu\mu}) + n_{BKG} \cdot C_{BKG}(m_{\mu\mu})$$

Qui viene eseguito il fit della distribuzione *binnata* della variabile

Per capire esattamente cosa significa *extended likelihood function in the case of binned data*, vedere G.Cowan 6.10 (e 6.9) !

A schermo si ottengono informazioni sul fit:

```
[pompili@cmssusy esercitazione-5]$ root -l
root [0] .x psiPrime_fit.C
```

Ottenendo ...

```
RooFit v3.56 -- 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
 ... ... ...
**********
**   13 **MIGRAD        3000           1
**********
FIRST CALL TO USER FUNCTION AT NEW START POINT, WITH IFLAG=4.
START MIGRAD MINIMIZATION.  STRATEGY  1.  CONVERGENCE WHEN EDM .LT. 1.00e-03
 ... ... ...
MIGRAD MINIMIZATION HAS CONVERGED.
MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=-2.0678e+06 FROM MIGRAD    STATUS=CONVERGED     198 CALLS         199 TOTAL
                    EDM=3.88053e-05    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   STEP         FIRST
 NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
  1  #sigma_{1}   3.56225e-02   2.17316e-04   3.67024e-03   6.06691e-01
  2  c_{1}       -1.88551e-01   5.22002e-03   5.14039e-04  -2.37797e+00
  3  c_{2}       -1.92706e-02   6.94904e-03   5.52338e-04   2.92534e+00
  4  mean         3.68091e+00   2.03170e-04   8.64724e-03  -5.38998e-01
  5  nBkg         1.11119e+05   4.70821e+02   1.16253e-03   3.01109e+00
  6  nSig         6.43078e+04   4.18322e+02   1.23552e-03  -9.41342e-02
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  6    ERR DEF=0.5
 4.723e-08  4.252e-08  6.647e-07 -1.472e-09 -4.920e-02  4.927e-02
 4.252e-08  2.725e-05 -3.769e-07 -1.218e-07 -8.827e-02  8.832e-02
 6.647e-07 -3.769e-07  4.829e-05  3.331e-08 -1.580e+00  1.580e+00
-1.472e-09 -1.218e-07  3.331e-08  4.128e-08  1.535e-03 -1.540e-03
-4.920e-02 -8.827e-02 -1.580e+00  1.535e-03  2.217e+05 -1.106e+05
 4.927e-02  8.832e-02  1.580e+00 -1.540e-03 -1.106e+05  1.750e+05
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2      3      4      5      6
       1  0.59608   1.000  0.037  0.440 -0.033 -0.481  0.542
       2  0.12945   0.037  1.000 -0.010 -0.115 -0.036  0.040
       3  0.59940   0.440 -0.010  1.000  0.024 -0.483  0.544
       4  0.12587  -0.033 -0.115  0.024  1.000  0.016 -0.018
       5  0.62392  -0.481 -0.036 -0.483  0.016  1.000 -0.561
       6  0.68387   0.542  0.040  0.544 -0.018 -0.561  1.000
```

risoluzione → $\hat{\sigma} \cong (35.62 \pm 0.22) MeV$

coefficienti di Cebyshev

massa → $\hat{m} \cong (3680.91 \pm 0.20) MeV$

 # candidati di fondo

 # candidati di segnale → $\hat{N}_{sig} \cong 64308 \pm 418$

```
**********
**   18 **HESSE        3000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=-2.0678e+06 FROM HESSE     STATUS=OK             40 CALLS          239 TOTAL
                     EDM=3.89377e-05    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                INTERNAL      INTERNAL
 NO.   NAME      VALUE            ERROR       STEP SIZE       VALUE
  1  #sigma_{1}   3.56225e-02   2.17932e-04   1.46809e-04  -3.05276e-01
  2  c_{1}       -1.88551e-01   5.22000e-03   2.05616e-05  -1.88562e-02
  3  c_{2}       -1.92706e-02   6.95669e-03   2.20935e-05  -1.92706e-03
  4  mean         3.68091e+00   2.03183e-04   3.45890e-04  -6.89538e-01
  5  nBkg         1.11119e+05   4.71806e+02   4.65010e-05  -8.91100e-01
  6  nSig         6.43078e+04   4.19277e+02   4.94208e-05  -1.05802e+00
                              ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  6    ERR DEF=0.5
 4.749e-08  4.178e-08  6.707e-07 -1.597e-09 -4.978e-02  4.978e-02
 4.178e-08  2.725e-05 -4.689e-07 -1.215e-07 -8.568e-02  8.568e-02
 6.707e-07 -4.689e-07  4.840e-05  3.327e-08 -1.589e+00  1.589e+00
-1.597e-09 -1.215e-07  3.327e-08  4.128e-08  1.556e-03 -1.556e-03
-4.978e-02 -8.568e-02 -1.589e+00  1.556e-03  2.226e+05 -1.115e+05
 4.978e-02  8.568e-02  1.589e+00 -1.556e-03 -1.115e+05  1.758e+05
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2      3      4      5      6
       1  0.59912   1.000  0.037  0.442 -0.036 -0.484  0.545
       2  0.12939   0.037  1.000 -0.013 -0.115 -0.035  0.039
       3  0.60057   0.442 -0.013  1.000  0.024 -0.484  0.545
       4  0.12644  -0.036 -0.115  0.024  1.000  0.016 -0.018
       5  0.62596  -0.484 -0.035 -0.484  0.016  1.000 -0.564
       6  0.68564   0.545  0.039  0.545 -0.018 -0.564  1.000
[#1] INFO:Minization -- RooMinuit::optimizeConst: deactivating const optimization
[#1] INFO:Plotting -- RooAbsPdf::plotOn(totalPDF) directly selected PDF components: (sigPDF)
[#1] INFO:Plotting -- RooAbsPdf::plotOn(totalPDF) indirectly selected PDF components: ()
[#1] INFO:Plotting -- RooAbsPdf::plotOn(totalPDF) directly selected PDF components: (bkgPDF)
[#1] INFO:Plotting -- RooAbsPdf::plotOn(totalPDF) indirectly selected PDF components: ()
Info in <TCanvas::Print>: png file ./Plots/PsiPrimeMassFit_alt.png has been created
```

Viene ricalcolata la matrice di covarianza.

I valori centrali delle stime dei parametri sono gli stessi ma viene raffinata la stima delle incertezze!

Il resto del codice serve per rappresentare l’istogramma e il risultato dell’interpolazione! (vedi slide seguente per il risultato)

```
RooPlot* xframe = xVar.frame();
xframe->SetTitle( hPsiPrime->GetTitle() );
xframe->SetYTitle("Candidates / 10 MeV/c^{2}");

MuMuHist->plotOn(xframe);
totalPDF->plotOn(xframe);

totalPDF->plotOn(xframe, Components(RooArgSet(sigPDF)), LineColor(kRed));

totalPDF->plotOn(xframe, Components(RooArgSet(bkgPDF)), LineColor(kGreen), LineStyle(kDashed) );

totalPDF->paramOn(xframe, Parameters(RooArgSet(mG,sigma1,nSig)), Layout(0.52,0.99,0.9)); //box con stime parametri

myC->cd();
xframe->Draw();

myC->SaveAs("./Plots/PsiPrimeMassFit_alt.png");

}
```

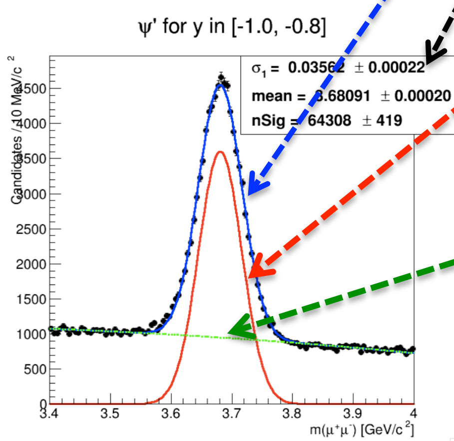
Definisce un *frame* a partire dalla variabile d’interesse

Rappresenta l’istogramma sul *frame*

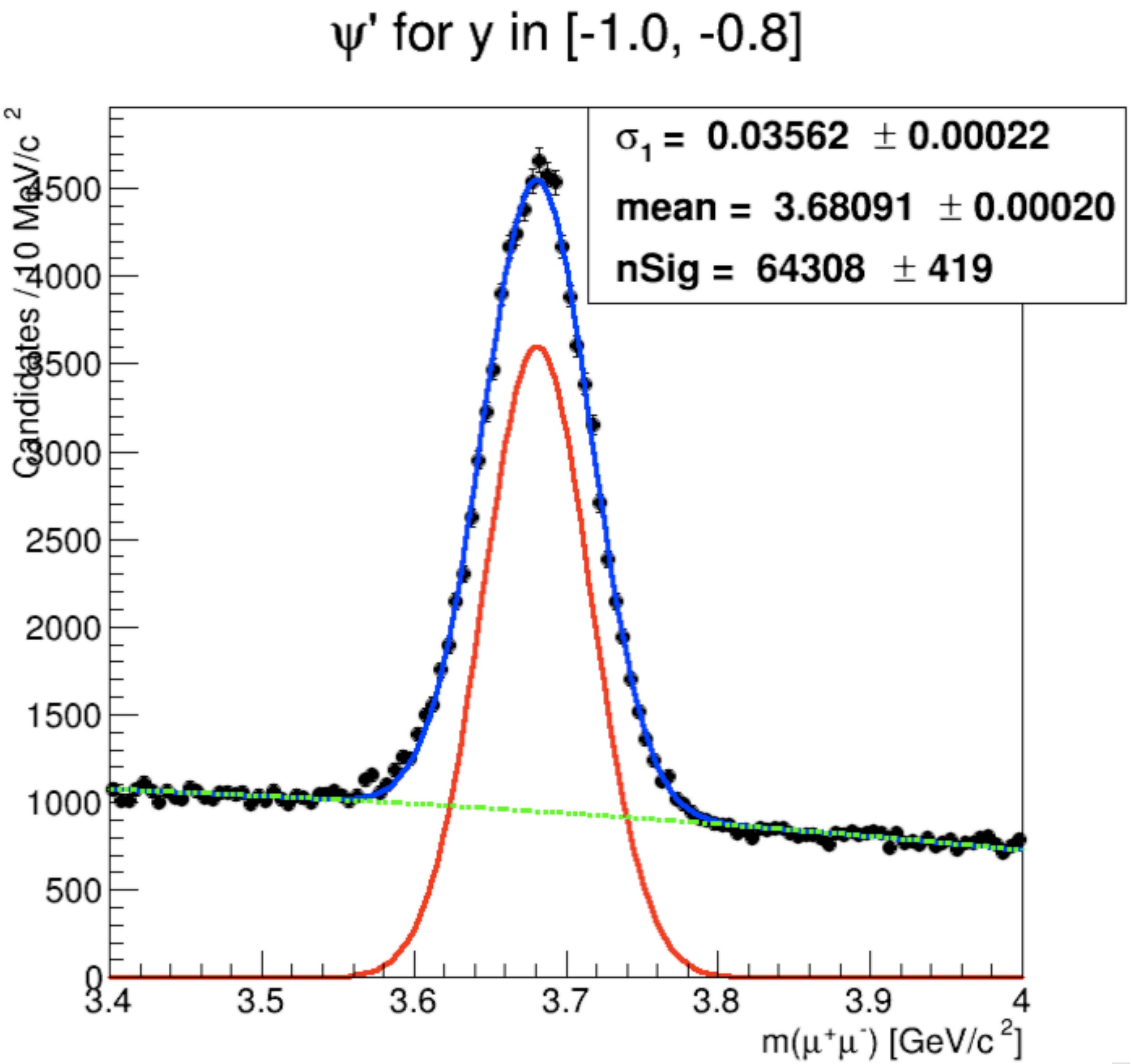
Rappresenta la funzione di fit sul *frame*

Sovrappone sul *frame* la sola componente del segnale

Sovrappone sul *frame* la sola componente del fondo



... il plot nella canvas:



6.9 Extended maximum likelihood

Consider a random variable x distributed according to a p.d.f. $f(x;\boldsymbol{\theta})$, with unknown parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$, and suppose we have a data sample $x_1, \ldots, x_n$. It is often the case that the number of observations n in the sample is itself a Poisson random variable with a mean value ν. The result of the experiment can be defined as the number n and the n values $x_1, \ldots, x_n$. The likelihood function is then the product of the Poisson probability to find n, equation (2.9), and the usual likelihood function for the n values of x,

$$L(\nu, \boldsymbol{\theta}) = \frac{\nu^n}{n!} e^{-\nu} \prod_{i=1}^{n} f(x_i;\boldsymbol{\theta}) = \frac{e^{-\nu}}{n!} \prod_{i=1}^{n} \nu f(x_i;\boldsymbol{\theta}). \quad (6.33)$$

This is called the **extended likelihood function**. It is really the usual likelihood function, however, only now with the sample size n defined to be part of the result of the experiment. One can distinguish between two situations of interest, depending on whether the Poisson parameter ν is given as a function of $\boldsymbol{\theta}$ or is treated as an independent parameter.

Source: G.Cowan, Statistical Data Analysis, Clarendon Press – Oxford, 1998

6.10 Maximum likelihood with binned data

Consider $n_{\rm tot}$ observations of a random variable x distributed according to a p.d.f. $f(x;\boldsymbol{\theta})$ for which we would like to estimate the unknown parameter $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$. For very large data samples, the log-likelihood function becomes difficult to compute since one must sum $\log f(x_i;\boldsymbol{\theta})$ for each value x_i. In such cases, instead of recording the value of each measurement one usually makes a histogram, yielding a certain number of entries $\mathbf{n} = (n_1, \ldots, n_N)$ in N bins. The expectation values $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_N)$ of the numbers of entries are given by

$$\nu_i(\boldsymbol{\theta}) = n_{\rm tot} \int_{x_i^{\rm min}}^{x_i^{\rm max}} f(x;\boldsymbol{\theta})\,dx, \qquad (6.40)$$

where $x_i^{\rm min}$ and $x_i^{\rm max}$ are the bin limits. One can regard the histogram as a single measurement of an N-dimensional random vector for which the joint p.d.f. is given by a multinomial distribution, equation (2.6),

$$f_{\rm joint}(\mathbf{n};\boldsymbol{\nu}) = \frac{n_{\rm tot}!}{n_1! \ldots n_N!} \left(\frac{\nu_1}{n_{\rm tot}}\right)^{n_1} \cdots \left(\frac{\nu_N}{n_{\rm tot}}\right)^{n_N}. \qquad (6.41)$$

The probability to be in bin i has been expressed as the expectation value ν_i divided by the total number of entries $n_{\rm tot}$. Taking the logarithm of the joint p.d.f. gives the log-likelihood function,

$$\log L(\boldsymbol{\theta}) = \sum_{i=1}^{N} n_i \log \nu_i(\boldsymbol{\theta}), \qquad (6.42)$$

where additive terms not depending on the parameters have been dropped. The estimators $\hat{\boldsymbol{\theta}}$ are found by maximizing $\log L$ by whatever means available, e.g. numerically. In the limit that the bin size is very small (i.e. N very large) the likelihood function becomes the same as that of the ML method without binning (equation (6.2)). Thus the binned ML technique does not encounter any difficulties if some of the bins have few or no entries. This is in contrast to an alternative technique using the method of least squares discussed in Section 7.5.

Source: G.Cowan, Statistical Data Analysis, Clarendon Press Oxford, 1998

Come fa MINUIT a calcolare la matrice di covarianza?

6.6 Variance of ML estimators: the RCF bound

It turns out in many applications to be too difficult to compute the variances analytically, and a Monte Carlo study usually involves a significant amount of work. In such cases one typically uses the **Rao–Cramér–Frechet (RCF) inequality**, also called the **information inequality**, which gives a lower bound on an estimator's variance. This inequality applies to any estimator, not only those constructed from the ML principle. For the case of a single parameter θ the limit is given by

$$V[\hat{\theta}] \geq \left(1 + \frac{\partial b}{\partial \theta}\right)^2 \Big/ E\left[-\frac{\partial^2 \log L}{\partial \theta^2}\right], \qquad (6.16)$$

where b is the bias as defined in equation (5.4) and L is the likelihood function.

For the case of more than one parameter, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$, the corresponding formula for the inverse of the covariance matrix of their estimators $V_{ij} = \mathrm{cov}[\hat{\theta}_i, \hat{\theta}_j]$ is (assuming efficiency and zero bias)

$$(V^{-1})_{ij} = E\left[-\frac{\partial^2 \log L}{\partial \theta_i \partial \theta_j}\right]. \qquad (6.19)$$

$$b = E[\hat{\theta}] - \theta.$$

Source: G.Cowan, Statistical Data Analysis, Clarendon Press – Oxford, 1998

It turns out to be impractical in many situations to compute the RCF bound analytically, since this requires the expectation value of the second derivative of the log-likelihood function (i.e. an integration over the variable x). In the case of a sufficiently large data sample, one can estimate V^{-1} by evaluating the second derivative with the measured data and the ML estimates $\hat{\boldsymbol{\theta}}$:

$$(\widehat{V^{-1}})_{ij} = - \left. \frac{\partial^2 \log L}{\partial \theta_i \partial \theta_j} \right|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}} . \tag{6.21}$$

For a single parameter θ this reduces to

$$\widehat{\sigma^2}_{\hat{\theta}} = \left(-1 \Big/ \frac{\partial^2 \log L}{\partial \theta^2} \right) \bigg|_{\theta = \hat{\theta}} . \tag{6.22}$$

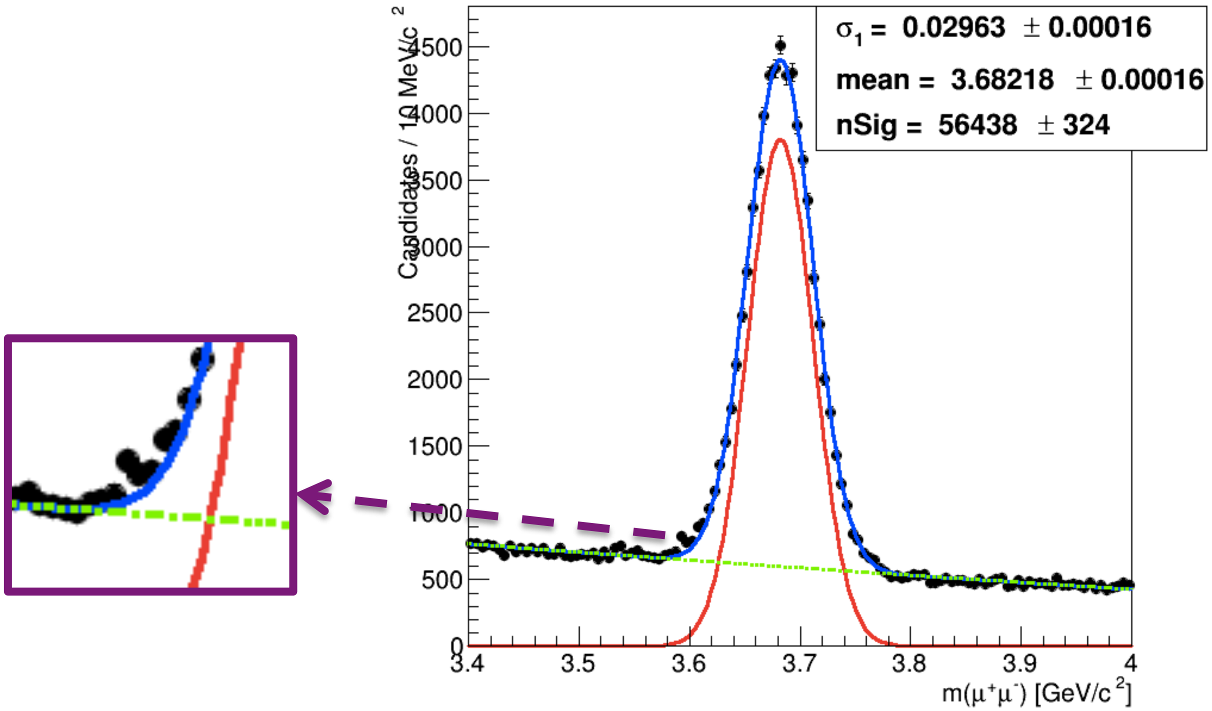
This is the usual method for estimating the covariance matrix when the likelihood function is maximized numerically.[1]

[1] For example, the routines MIGRAD and HESSE in the program MINUIT [Jam89, CER97] determine numerically the matrix of second derivatives of $\log L$ using finite differences, evaluate it at the ML estimates, and invert to find the covariance matrix.

Source: **G.Cowan, Statistical Data Analysis, Clarendon Press – Oxford, 1998**

Esercitazione n.1bis
approfondimento/prova pratica

Nella precedente esercitazione abbiamo interpolato la distribuzione di massa invariante *m*($\mu^+\mu^-$) contenuta nel file *psiprime_bin9_histo.root* :



Si noti che la coda a valori bassi di massa inv. per il picco di segnale non e' ben descritta dalla gaussiana.
In effetti e' preferibile – invece di una gaussiana - usare una singola funzione Crystal Ball, la quale integra una funzione gaussiana rappresentante la risoluzione sperimentale con una funzione potenza rappresentante la coda radiativa (dovuta a *bremsstrahlung interna*, un processo di QED con un muone che "emette" radiazione di stato finale).

Crystal ball function

The Crystal Ball function, named after the Crystal Ball Collaboration (hence the capitalized initial letters), is a probability density function commonly used to model various lossy processes in high-energy physics. It consists of a Gaussian core portion and a power-law low-end tail, below a certain threshold. The function itself and its first derivative are both continuous.

The Crystal Ball function is given by:

$$f(x;\alpha,n,\bar{x},\sigma) = N \cdot \begin{cases} \exp(-\frac{(x-\bar{x})^2}{2\sigma^2}), & \text{for } \frac{x-\bar{x}}{\sigma} > -\alpha \\ A \cdot (B - \frac{x-\bar{x}}{\sigma})^{-n}, & \text{for } \frac{x-\bar{x}}{\sigma} \leqslant -\alpha \end{cases}$$

where

$$A = \left(\frac{n}{|\alpha|}\right)^n \cdot \exp\left(-\frac{|\alpha|^2}{2}\right)$$

$$B = \frac{n}{|\alpha|} - |\alpha|$$

N is a normalization factor and α, n, x and σ are parameters which are fitted with the data.

Per avere un'idea dell'effetto dei due parametri descriventi la coda si osservi la seguente figura (tratta da *CMS AN-14-003*): →

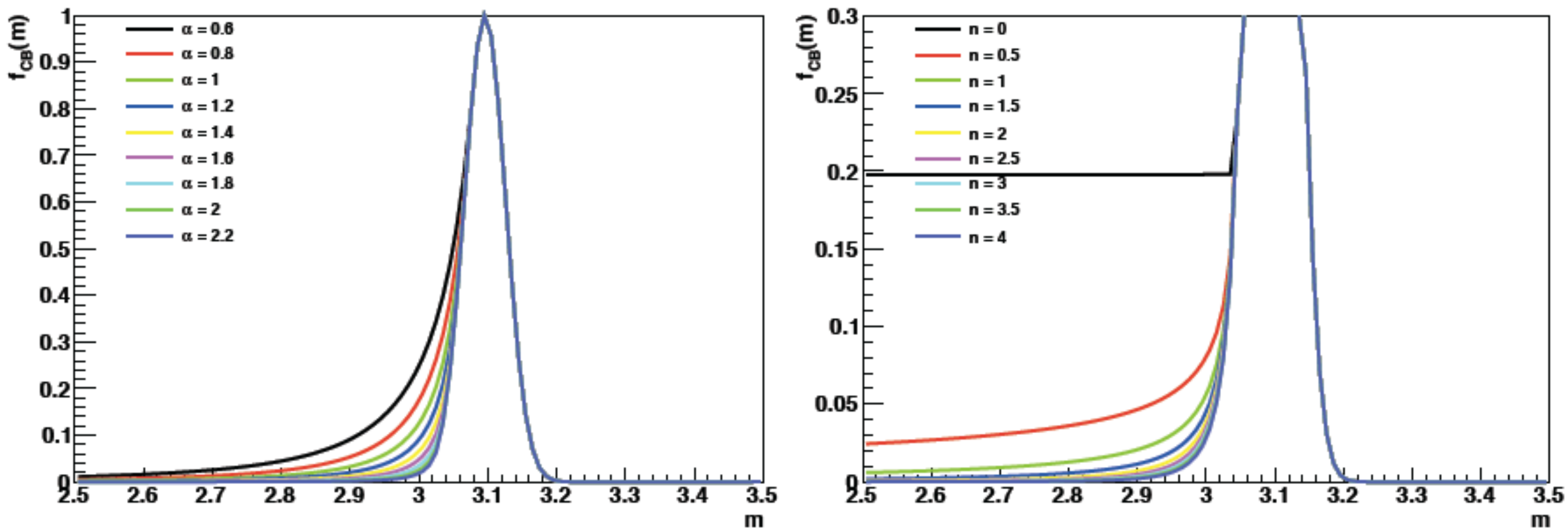


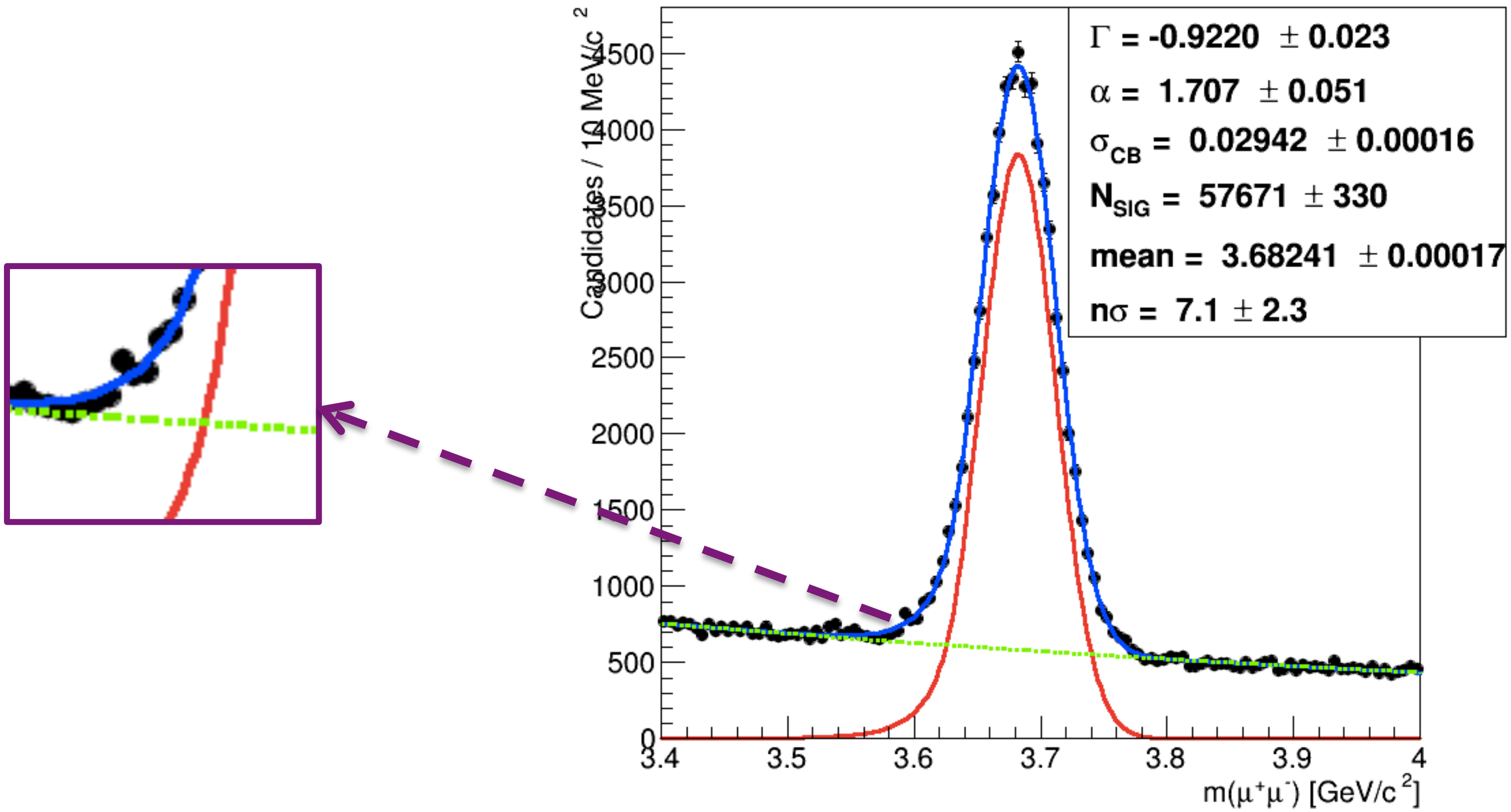
Figure 16: Shapes of the CB function for several different (n_{CB}, α_{CB}) values, fixing $n_{CB} = 2.5$ (left) or $\alpha_{CB} = 1.8$ (right).

Inoltre, empiricamente, il fondo combinatorio della massa inv. dei due muoni risulta essere descritto da una funzione esponenziale, quindi con un parametro in meno rispetto alla polinomiale di ord.1!
Si puo' quindi provare anche a passare dall'esponenziale alla retta.

Si tenga conto che in *RooFit* si tratta di costruire modelli di segnale e fondo con funzioni del tipo *RooCBShape* e *RooExponetial* al posto di funzioni del tipo *RooGaussian* e *RooChebyshev* rispettivamente.

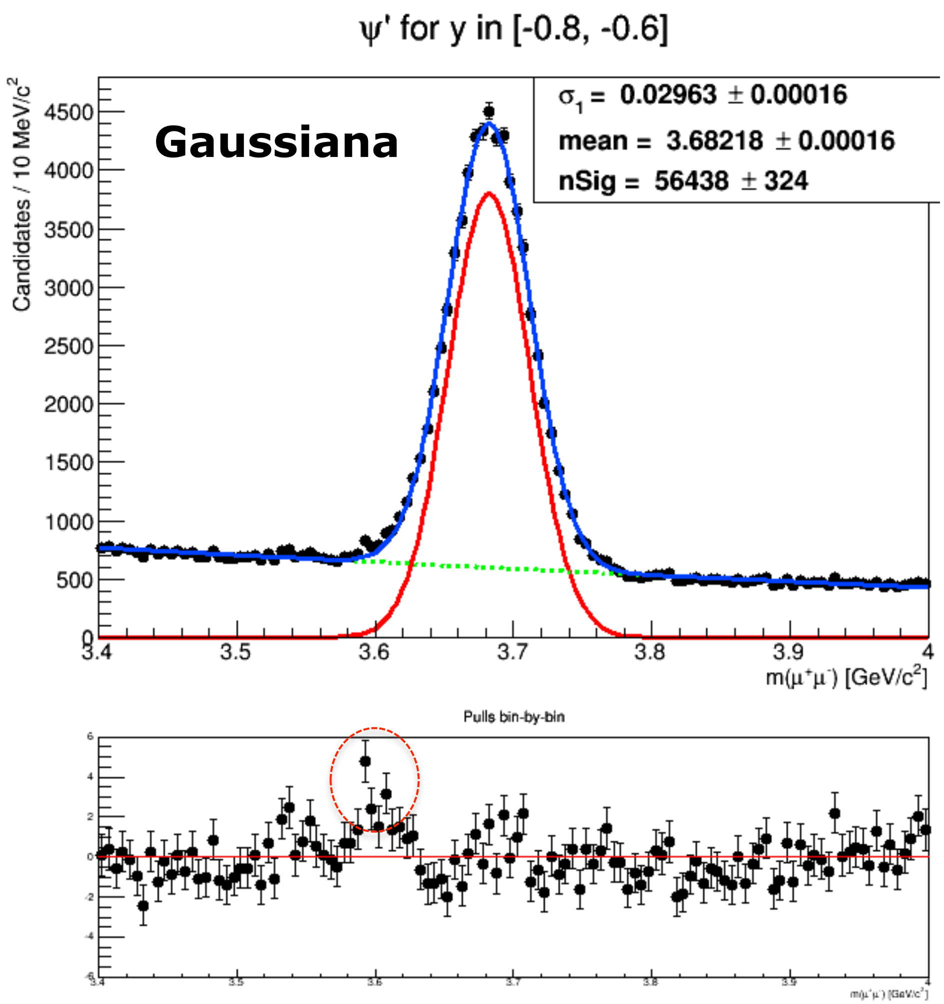
L'esercizio consiste quindi in:

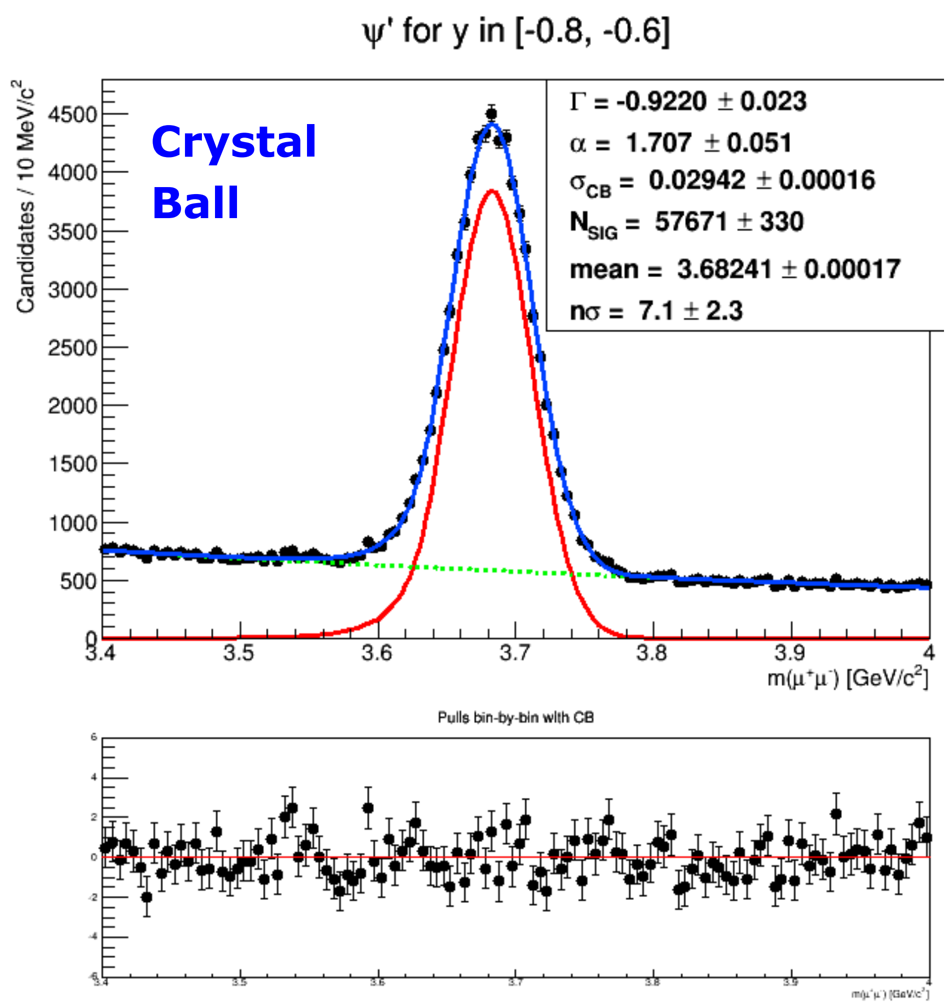
1) ripetere l'interpolazione usando, come modello per il segnale, una funzione Crystal Ball e, come modello per il fondo, la funzione esponenziale, ottenendo:



2) rifare l'interpolazione usando la polinomiale di ord.1 per il fondo, mentre si mantiene una Crystal Ball per il segnale.
Discutere come varia la stima dei candidati ψ' di segnale nei 3 fit.

In particolare usando il metodo delle pull per monitorare la *goodness-of-fit* si puo' osservare quale sia l'effetto nel passare dalla gaussiana alla CB:





I plot alla slide precedente si ottengono aggiungendo il codice relativo alle *pull* (vedi definizione alla slide successiva):

```
////////////////// goodness of fits with pulls for each bin :
//
RooPlot *framePull = xVar.frame();
framePull->SetTitle("Pulls bin-by-bin");
framePull->addObject( (TObject*)xframe->pullHist(), "p" ) ;
framePull->SetMinimum(-6);
framePull->SetMaximum(6);
//
myC->Divide(0,2);
myC->cd(2);
gPad->SetPad(0.,0.,1.,0.3);
framePull->Draw();
TLine *line = new TLine(3.4,0.,4.,0.);
line->SetLineColor(2);
line->Draw("same");
myC->cd(1);
gPad->SetPad(0.,0.3,1.,1.);
xframe->Draw();
//
```

Proprieta' delle pull:

1) **dai plot alla slide precedente si evince come l'errore sulla pull bin-by-bin sia unitario (il motivo e' spiegato alla slide successiva);**
2) **La proiezione sull'asse delle ordinate delle pull bin-by-bin deve fornire una distribuzione complessiva attesa essere una gaussiana standard (media 0 e varianza 1).**

Proprieta' delle pull:

Lo *scarto normalizzato* e' simile alla radice di un chi-quadrato corredato di segno (motivo per il quale tecnicamente e' uno "*pseudo chi-quadrato*"). Lo denoto con $\pm\sqrt{\chi^2}$

L'istogramma degli scarti normalizzati desidero che abbia lo stesso # di bin dell'istogramma della distribuzione di massa invariante.
E' necessario poi rappresentare lo scarto corredato dalla propria barra di errore!

$$\pm\sqrt{\chi^2}(i) = \frac{x_S^i - x_T^i}{\sigma_i} \equiv \frac{N_i - F_i}{\sqrt{N_i}}$$

- x_S^i: Valore sperimentale
- x_T^i: Valore atteso (teorico)
- σ_i: Incertezza associata al valore sperimentale

... essendo :
- N_i **= # candidati nel bin i-esimo dell'istogramma**
- F_i **= # candidati nel bin i-esimo atteso (assumendo corretto il modello di fit)**

L'incertezza (errore) sullo scarto normalizzato (per ogni bin) si calcola applicando l'usuale legge di propazazione degli errori casuali (tralascio l'indice per alleggerire la notazione):

$$\sigma^2_{\pm\sqrt{\chi^2}} = \left(\frac{d}{dN}\left(\frac{N-F}{\sqrt{N}}\right)\right)^2 \cdot \left(\sqrt{N}\right)^2 = \left(\frac{\sqrt{N} - \frac{1}{2\sqrt{N}}(N-F)}{N}\right)^2 \cdot N = \left(\frac{N - \frac{1}{2}(N-F)}{N\sqrt{N}}\right)^2 \cdot N = \left(\frac{1}{2}\left(\frac{N+F}{N}\right)\right)^2$$

In conclusione: $\sigma_{\pm\sqrt{\chi^2}} = \frac{1}{2}\frac{N+F}{N}$ **e, ad *alta statistica* (N grande) si ha:** $N \approx F \Rightarrow \sigma_{\pm\sqrt{\chi^2}} \approx 1$

Esercitazione n.2

In questa esercitazione impariamo a generare delle distribuzioni secondo un qualche modello teorico rappresentato dalla PDF e poi ne eseguiamo l'interpolazione.
Ovviamente si potra' verificare che le stime dei parametri restituite dal *fitter* saranno compatibili o molto simili ai valori usati per essi nella PDF al momento della generazione.

Si useranno i seguenti file:

- **Macro di RooFit: RooConvolutionExp.C**
- **File di configurazione : rootsource2.sh**

Creare inoltre - puramente per esigenza di ordine - due *sub-directory*:
- **txt_files**
- **plots**

Il file di configurazione permette:
- l'uso di una opportuna versione di ROOT presa da *afs* ;
- l'uso di un particolare compilatore (***gcc***), sempre preso da *afs*:

```
export ROOTSYS=/afs/cern.ch/sw/lcg/app/releases/ROOT/5.34.26/x86_64-slc5-gcc47-opt/root/
export PATH=$PATH:$ROOTSYS/bin:.
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib:.
source /afs/cern.ch/sw/lcg/app/releases/ROOT/5.34.26/x86_64-slc5-gcc47-opt/root/bin/thisroot.sh
source /afs/cern.ch/sw/lcg/external/gcc/4.7.2/x86_64-slc5-gcc47-opt/setup.sh
```

Per configurare l'ambiente di lavoro sulla macchina virtuale:

```
-bash-3.2$ source rootsource2.sh
-bash-3.2$ █
```

Per eseguire la macro di RooFit si lancia ROOT dopodiche' …

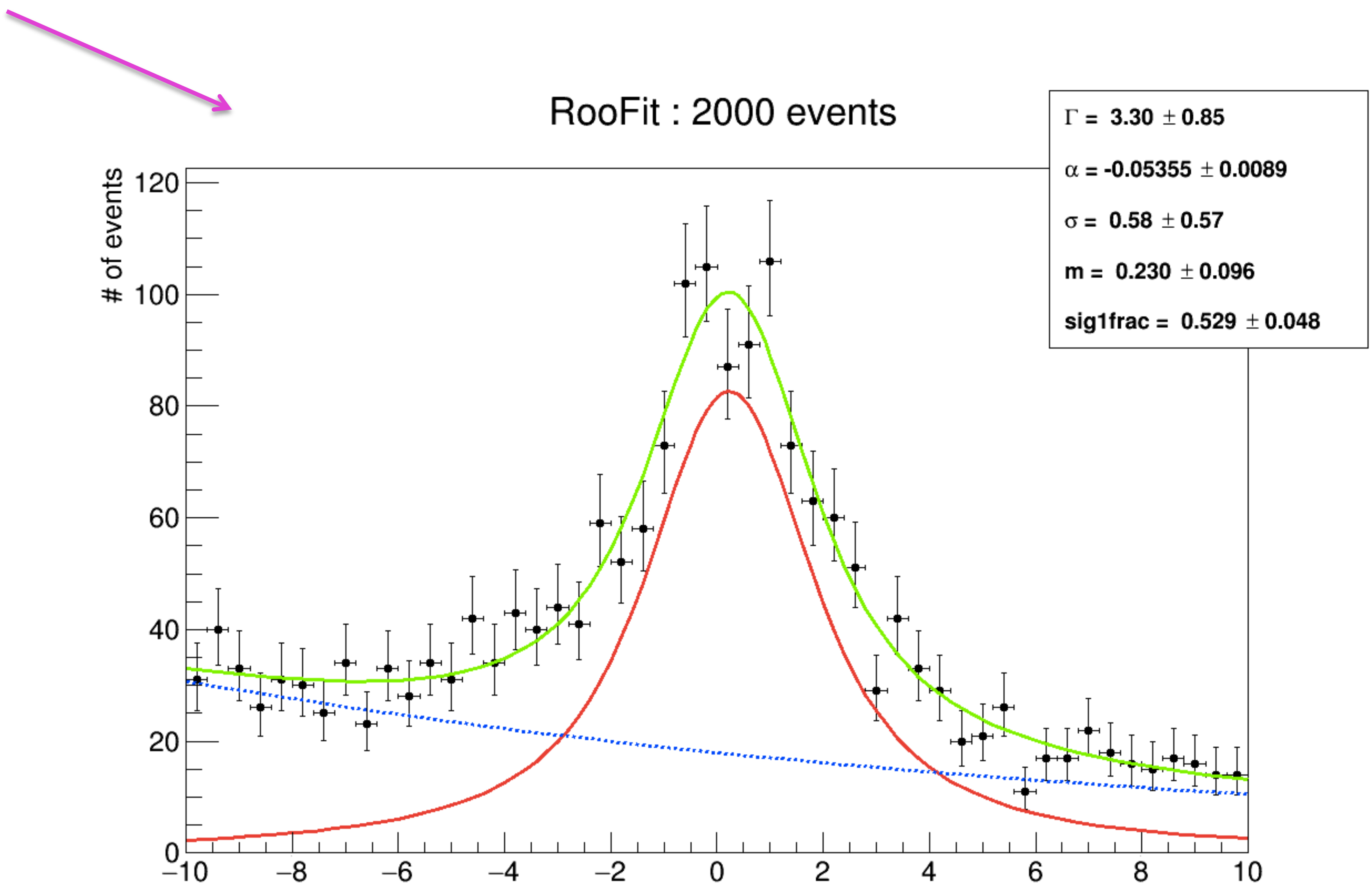
Root [0] .L RooConvolutionExp.C+
Root [1] RooConvolutionExp("#events",#bins)

P.es. #events=10000 e #bins=80 (ci impiega 130s)
oppure #events=100000 e #bins=120 (ci impiega 1250s)

N.B.: Il #bins viene fissato a soli fini di rappresentazione
(il fit e' *Unbinned Maximum Likelihood* !)

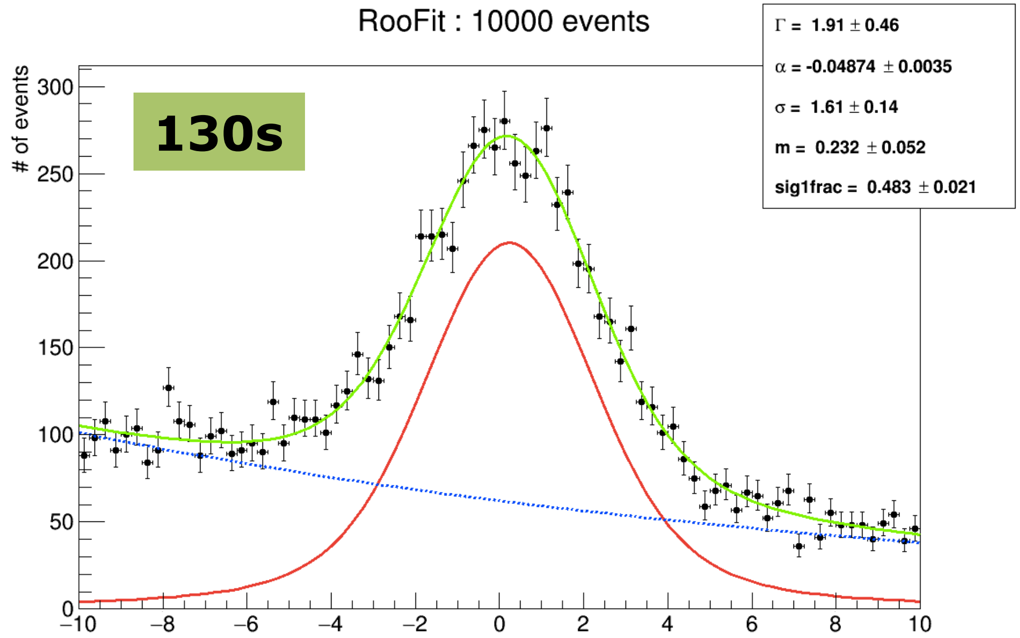
Si ottiene in ./plots **un file .eps/.png dal seguente contenuto:** →

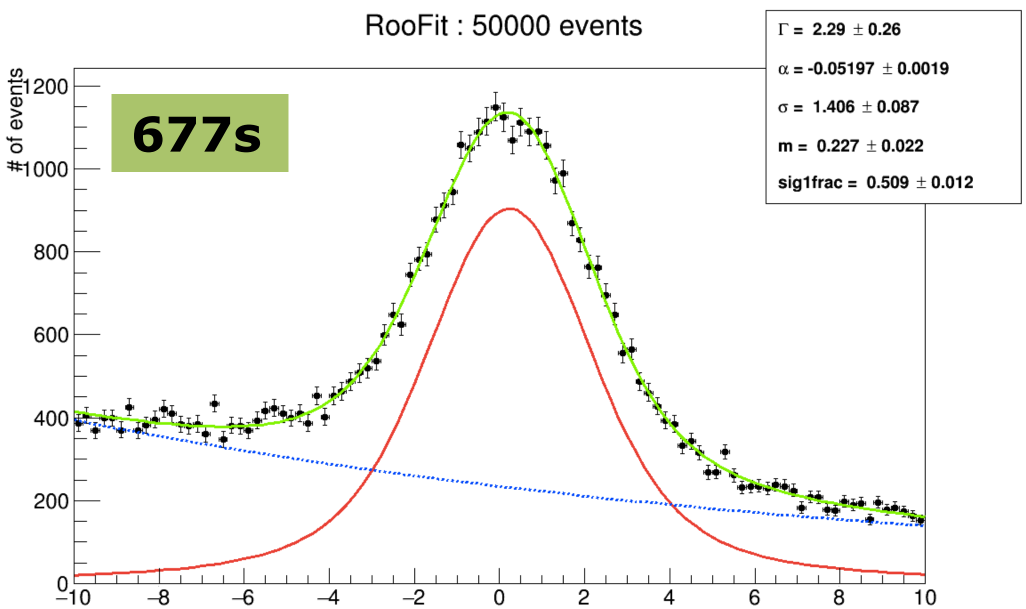
N.B.: Il fit non e' *Extended : #eventi lo decidete voi in generazione!*



Modello: il segnale (Breit-Wigner convoluta con una gaussiana di risoluzione sperimentale) "giace" su un fondo esponenziale.

**Si ottiene p.es.
nei 2 casi seguenti:**





E’ buona pratica confrontare il risultato dell’interpolazione con i valori dei parametri che sono stati messi in generazione. Si puo’ verificare come l’accordo aumenti all’aumentare del # di eventi generati!

Una tabellina dei tempi per valutare le prestazioni;
il tempo impiegato si riferisce al solo fit (ma in ogni caso
il tempo di generazione e' trascurabile rispetto a quello di fit):

#eventi	*RooFit*
10K	130s
50K	677s
100K	1250s

Ispezioniamo il codice della macro di *RooFit* (RooConvolutionExp.C)

```
#include "RooPolynomial.h"
#include "RooRealVar.h"
#include "RooBreitWigner.h"
#include "RooNumConvPdf.h"
#include "RooGaussian.h"
#include "RooExponential.h"
#include "RooDataSet.h"
#include "RooDataHist.h"
#include "RooAbsData.h"
#include "RooMinuit.h"
#include "RooPlot.h"
#include "RooChebychev.h"
#include "RooAddPdf.h"
#include "RooArgList.h"
#include "TH1F.h"
#include <vector>
#include "TCanvas.h"

#include <sys/time.h>
#include <sys/times.h>

using namespace RooFit; //Working in RooFit//

timeval startTime, stopTime, totalTime;
timeval startTimeRead, stopTimeRead, totalTimeRead;
clock_t startCPU, stopCPU;
clock_t startCPURead, stopCPURead;
tms startProc, stopProc;  //struct time intervals in clock ticks//
tms startProcRead, stopProcRead;

void RooConvolutionExp(TString argv,int bins=200) {

  int events = atoi(argv.Data()); //converte stringa "numero" in numero --
  TString name = "";
  switch (events)
        {
         case 100: name = "100";
           break;
         case 1000: name = "1k";
           break;
         case 10000: name = "10k";
           break;
         case 100000: name = "100k";
           break;
         case 500000: name = "500k";
           break;
         case 1000000: name = "1M";
           break;
         case 5000000: name = "5M";
           break;
         case 10000000: name = "10M";
           break;
         case 50000000: name = "50M";
           break;
         case 100000000: name = "100M";
           break;
           //
        default: name = argv;
           break;
        }

  char bufferstring[256];
```

```
char bufferstring[256];

RooRealVar xvar("xvar", "", -10, 10);
xvar.setBins(bins);

// Breit Wigner Signal //
RooRealVar mean("m", "mean", 0.2, -1, 1);                      //Breit Wigner mean//
RooRealVar gamma("#Gamma", "gamma", 2, 0.1, 5);                //Breit Wigner width//
RooBreitWigner signal("BW", "BW signal", xvar, mean, gamma); //Breit Wigner pdf//

// Gaussian Resolution Function //
RooRealVar zero("zero","Gaussian resolution mean", 0.);                  // offset from mean
RooRealVar sigma("#sigma", "sigma", 1.5, 0.1, 5);                        //Gaussian sigma//
RooGaussian resol("resol", "Gaussian resolution", xvar, zero, sigma); //Gaussian pdf//

// Background //
RooRealVar alpha("#alpha","Exponential Parameter",-0.05,-2.0,0.0);
RooExponential bkg("Bkg","Bkg",xvar,alpha);

// Gaussian + BW convolution //
RooNumConvPdf convolution("convolution", "BW (X) gauss", xvar, signal , resol);

// TotalPdf = Gaussian + Bkg //
RooRealVar sigfrac("sig1frac","fraction of component 1 in signal",0.5,0.,1.) ;
RooAddPdf total("totalPDF", "totalPDF", RooArgList(convolution, bkg),sigfrac);

cout <<"\nGenerating " <<name <<" events\n" <<endl ;

////////////////////////////////////////////////////////////////////////
// Generating data
////////////////////////////////////////////////////////////////////////

RooDataSet* data = total.generate(xvar,events);
//
sprintf(bufferstring,"./txt_files/%d_events.txt",events);
data->write(bufferstring);

cout <<"\nFitting " <<name <<" events\n" <<endl ;
```

Generazione secondo il modello (Pdf) *total*

Scrive la massa generata evento-per-evento nel file .txt esterno (ispezionare)

```
cout <<"\nFitting " <<name <<" events\n" <<endl ;

////////////////////////////////////////////////////////////////////////
// Fitting data
////////////////////////////////////////////////////////////////////////


RooAbsReal* nll = total.createNLL(*data);

//Declare null (pointer) and assign -log(Likelyhood) to it, Likelihood -> convolution and *data//
RooMinuit min(*nll);
gettimeofday(&startTime, NULL);
startCPU = times(&startProc);
//Migrad Fit
min.migrad();

stopCPU = times(&stopProc);
gettimeofday(&stopTime, NULL);

////////////////////////////////////////////////////////////////////////
// Fit result and data representation
////////////////////////////////////////////////////////////////////////

TCanvas *foo = new TCanvas("RooCanvas","Roofit Canvas", 1200, 800);

RooPlot *frame = xvar.frame("") ;
sprintf(bufferstring," RooFit : %d events",events);
frame->SetTitle(bufferstring) ;
frame->SetYTitle("# of events") ;
data->plotOn(frame);
total.plotOn(frame,LineColor(kGreen));
total.plotOn(frame,Components(RooArgSet(convolution)),LineColor(kRed));
total.plotOn(frame,Components(RooArgSet(bkg)),LineColor(kBlue),LineStyle(kDashed));
total.paramOn(frame, Layout(0.75,0.99,0.99));
frame->getAttText()->SetTextSize(0.028);

frame->Draw() ;
foo->SaveAs("plots/RooConvGen_"+name+".eps");
foo->SaveAs("plots/RooConvGen_"+name+".png");

// Print total fitting time
cout << "\n---------------------------------------------------------------" << endl ;
double myCPUc = (stopCPU - startCPU)*10000;
cout << "Total CPU time: " << (myCPUc / CLOCKS_PER_SEC);
cout << "\n---------------------------------------------------------------" << endl ;
cout << endl ;
}
```

PDF

dati generati (unbinned!!)

UML FIT

file esterni con il plot

NOTA ADDIZIONALE

Per esseri sicuri che generando 2 volte una distribuzione con lo stesso numero di eventi si ottengono, corrispondentemente, 2 distribuzioni diverse e' sufficiente aggiungere poche linee di codice, come di seguito spiegato.
L'idea e' di legare il **seme** ("seed") **usato dal generatore casuale** all'orario preso dal sistema operativo della macchina durante l'esecuzione della macro stessa.

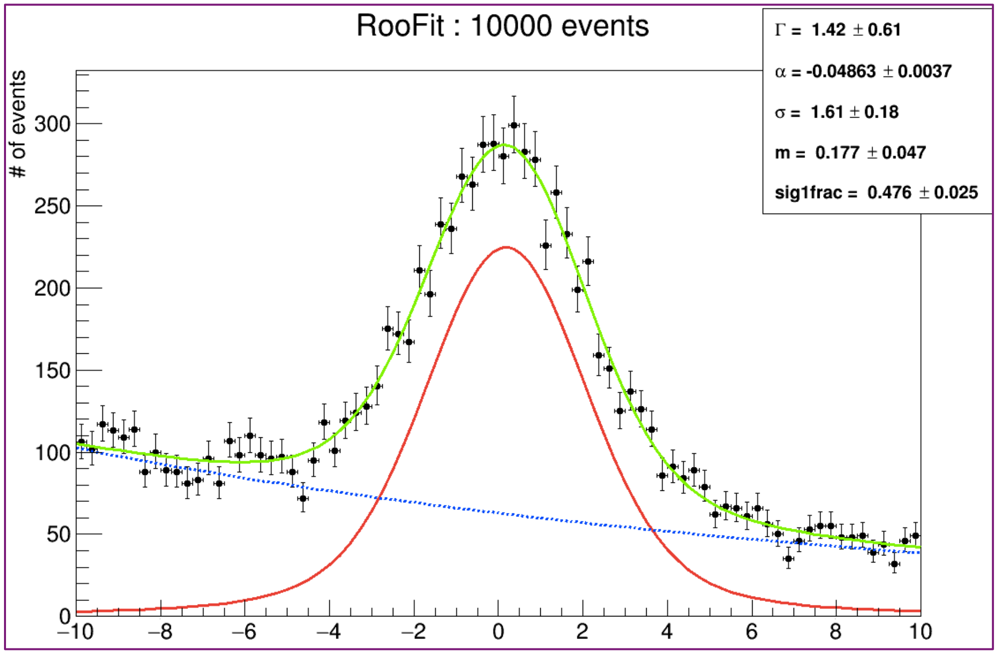
E' dunque sufficiente:

1) aggiungere il seguente include:

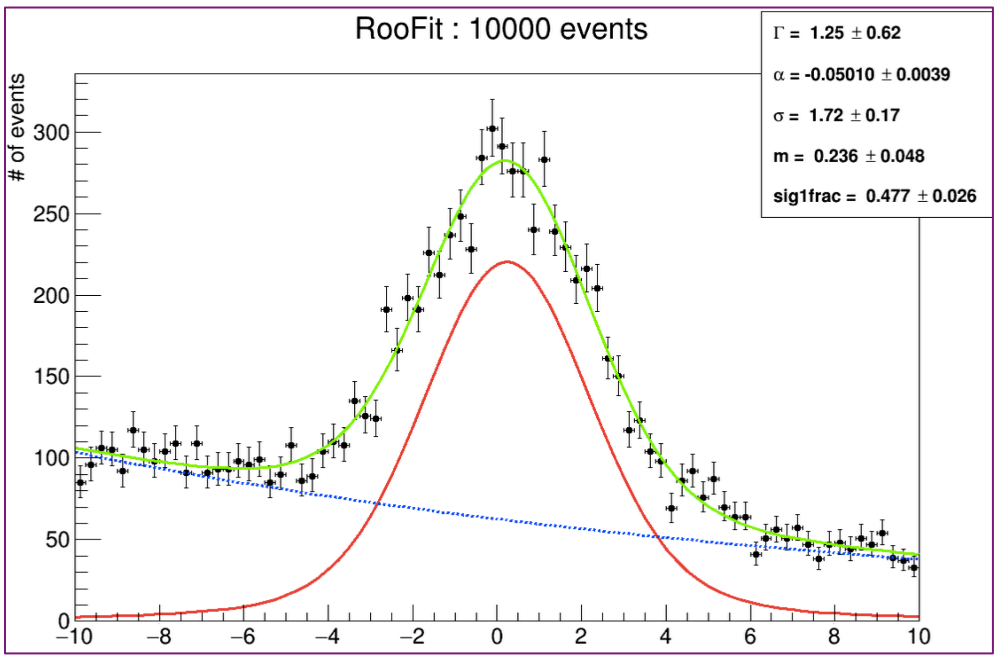
```
#include "RooRandom.h" // needed for Randomizer
```

2) generare il seme (intero di tipo *long int*) in base all'ora e darlo al generatore (con *SetSeed*):

```
/////////////////////////////////////////////////////////////////
// Generating data
/////////////////////////////////////////////////////////////////

timeval trand;
gettimeofday(&trand,NULL);

long int msRand = trand.tv_sec * 1000 + trand.tv_usec / 1000;
cout << "\n------" << endl;
cout << "msRand = " << msRand ;
cout << "\n------" << endl;
RooRandom::randomGenerator()->SetSeed(msRand);

RooDataSet* data = total.generate(xvar,events);
//
```




```
Generating 10k events

------
msRand = 1452353646588
------
```




```
Generating 10k events

------
msRand = 1452354081332
------
```

Approfondimenti:

1) Il metodo Monte Carlo e' spiegato bene e compattamente al capitolo 3 del testo di Cowan:

G.Cowan, Statistical Data Analysis, Clarendon Press – Oxford, 1998

Si noti che **RooFit usa l'** ***acceptance-rejection method*** (paragrafo 3.3)

2) Generare distribuzioni e' molto utile per usare la cosidetta ***MC toys technique.***

Vedere per esempio:
http://roofit.sourceforge.net/docs/tutorial/fitgen/roofit_tutorial_fitgen.pdf

Una particolare applicazione dei MC toys si ha quando si vuole stimare il ***p-value*** di una distribuzione per determinare la **significativita' statistica di un segnale fisico.**
Vedere per esempio slide 5-6 del talk (@ ACAT2016)
https://indico.cern.ch/event/397113/contributions/1837858/attachments/1213108/1770056/pompili_acat16_final.pdf

Esercitazione n.3

Usando quanto finora imparato svolgere il seguente esercizio:

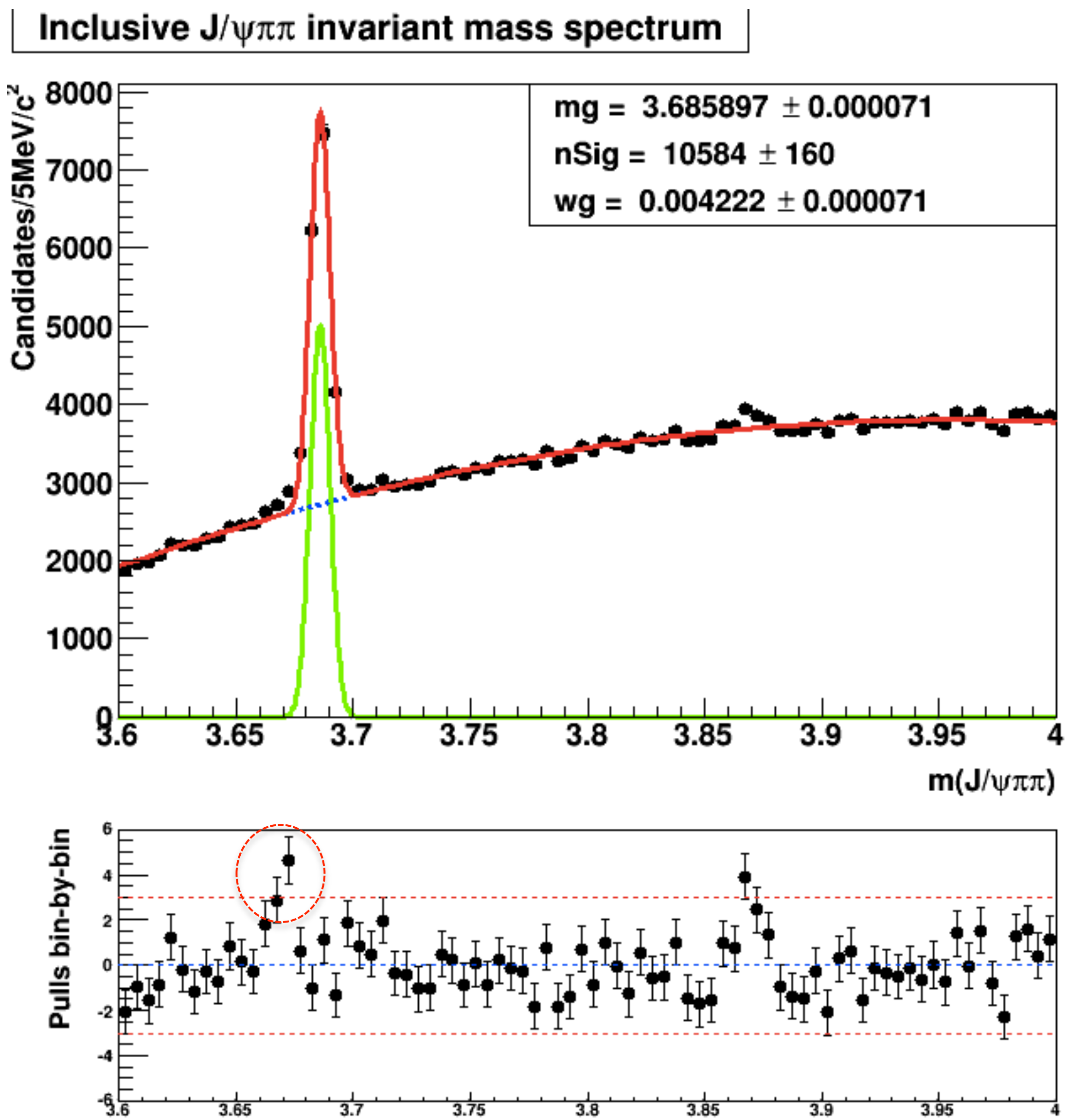
Interpolare mediante *RooFit* la distribuzione di massa invariante $m(\mu^+\mu^-\pi^+\pi^-)$ **(ottenuta fittando ad un vertice comune 2 muoni e 2 tracce e richiedendo il vincolo cinematico della massa della** J/ψ **per la coppia di muoni, in eventi del dataset 2011 dell'esperimento CMS) identificata, nel file** *esame-dec2014.root*, **con l'istogramma *PsiPrime_Mass_cut6* .**
Inizialmente partire con un semplice modello di fit e sulla base dell' andamento *bin-by-bin* della *pull* (*) raffinare via via l'interpolazione (**) sulla base di quanto sperimentato nelle esercitazioni. Discutere i(l) segnale(i) fisico(i) presente(i) individuandone le caratteristiche.

Suggerimenti:

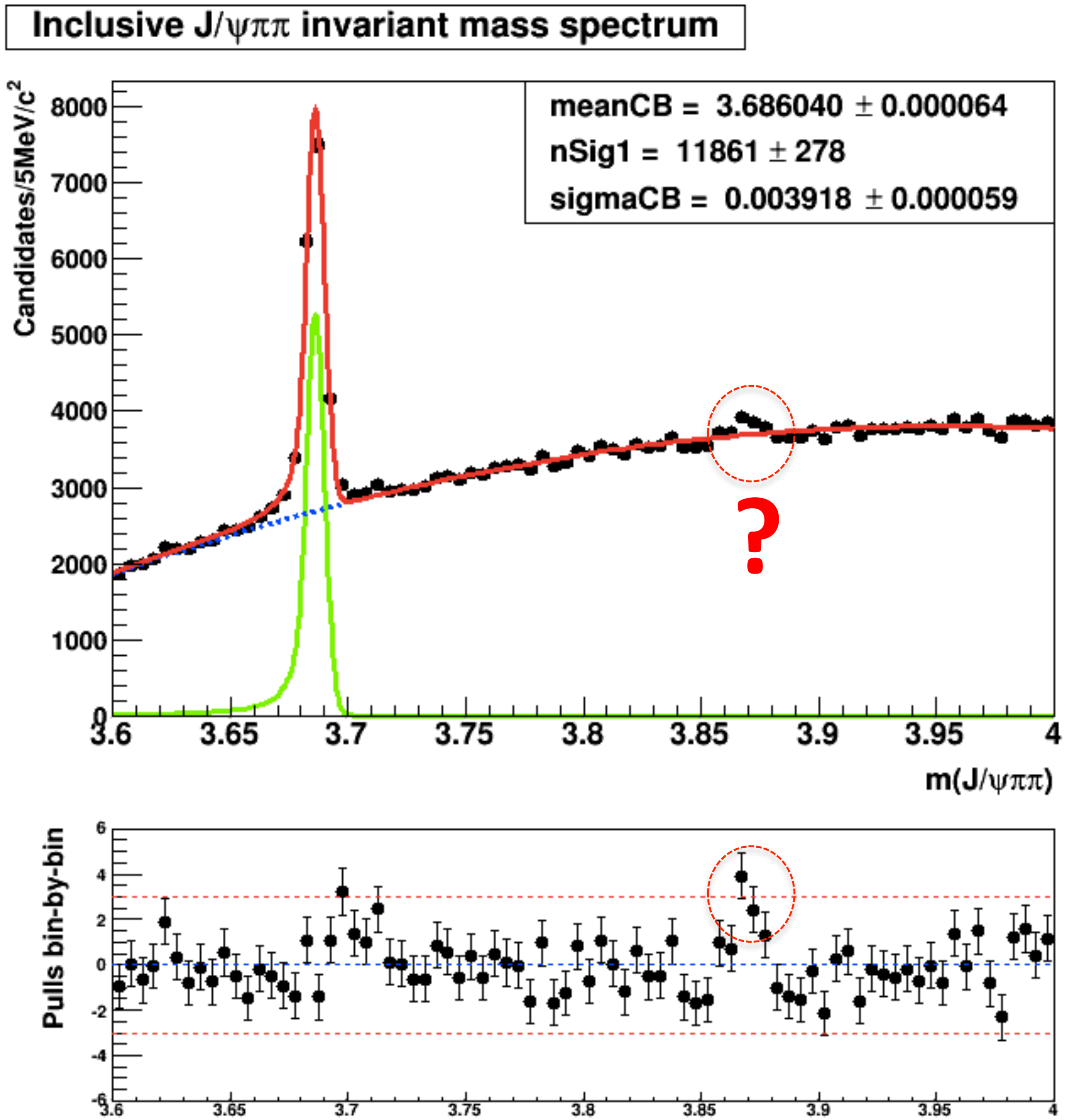
(*) usare: *xframe→pullHist()* **[vedere esercitazione 1bis]**
(**) usare: *PdfTotale→fitTo(nome_istogramma,Extended(kTRUE));*
avendo configurato come parametri il # di candidati di segnale e fondo

Domanda: come mai secondo voi la risoluzione in massa che caratterizza la larghezza del segnale della $\psi' \to J/\psi\,\pi\pi \to (\mu\mu)\pi\pi$ **risulta essere circa ¼ di quella del segnale** $\psi' \to \mu\mu$ **(vista nell'esercitazione 1).**

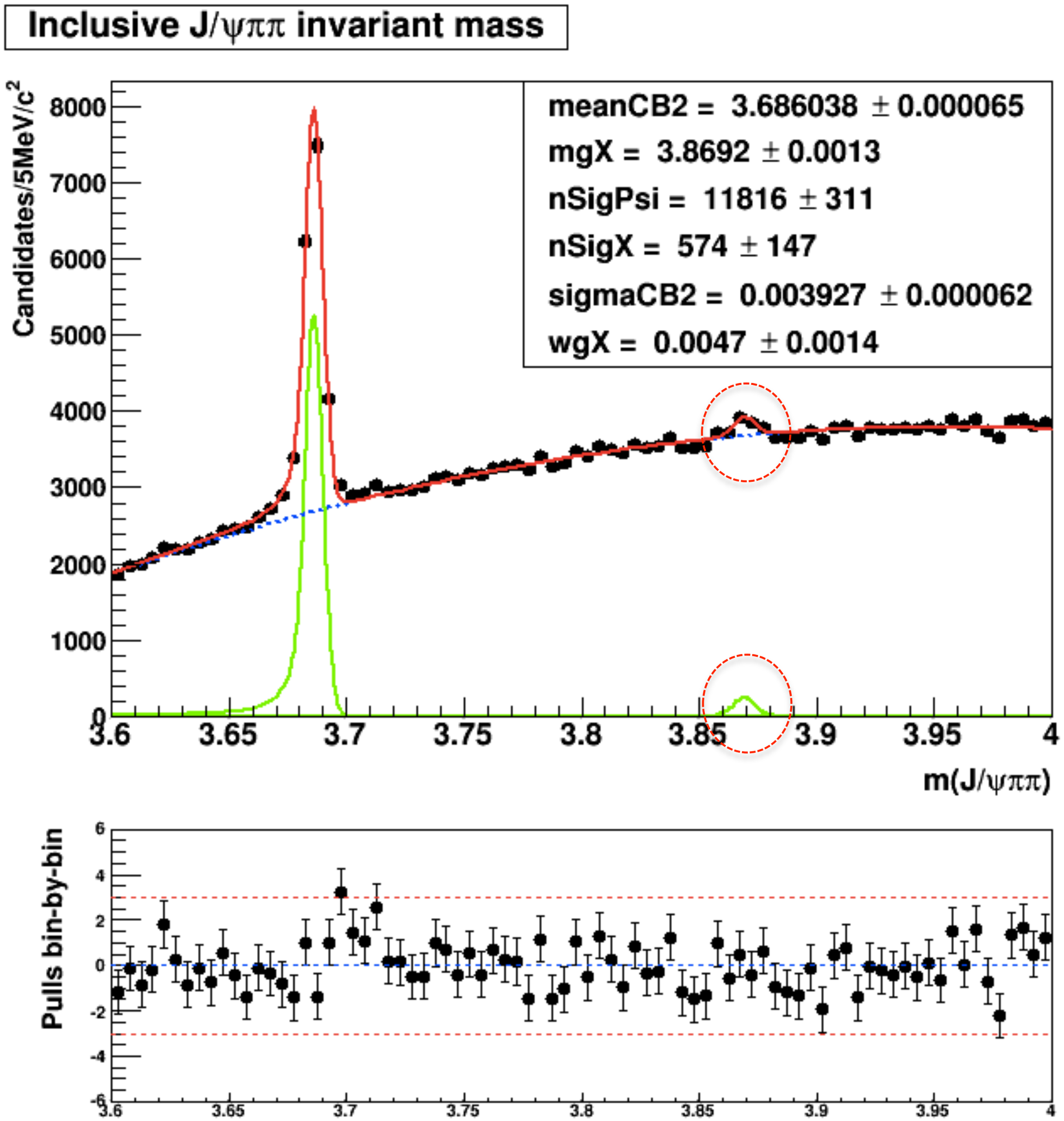
Primo tentativo di fit:

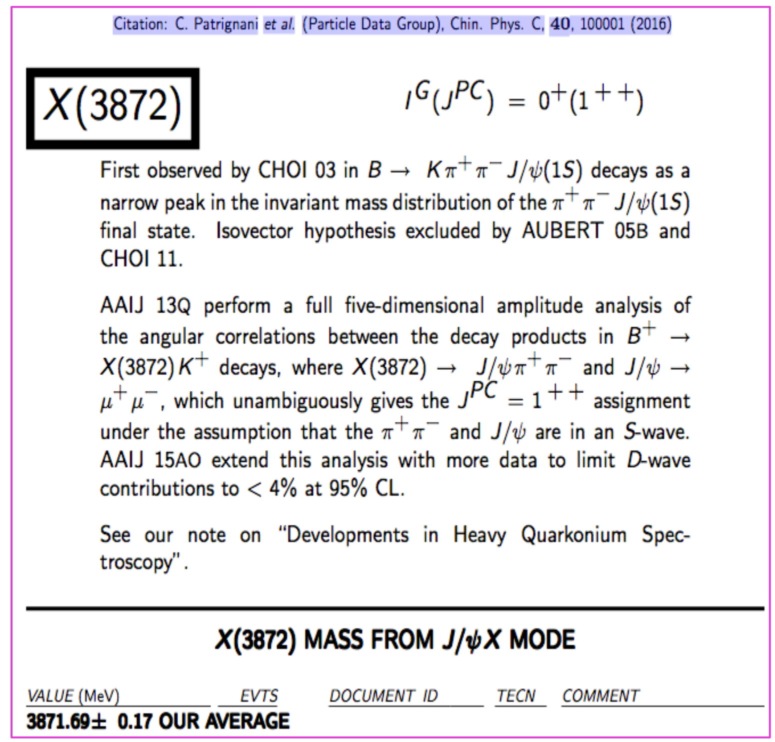


Perfezionando la descrizione della coda radiativa con una Crystal Ball:



Perfezionare introducendo un segnale aggiuntivo [quello della *X(3872)*]:





Citation: C. Patrignani *et al.* (Particle Data Group), Chin. Phys. C, **40**, 100001 (2016)

X(3872)

$I^G(J^{PC}) = 0^+(1^{++})$

First observed by CHOI 03 in $B \to K\pi^+\pi^- J/\psi(1S)$ decays as a narrow peak in the invariant mass distribution of the $\pi^+\pi^- J/\psi(1S)$ final state. Isovector hypothesis excluded by AUBERT 05B and CHOI 11.

AAIJ 13Q perform a full five-dimensional amplitude analysis of the angular correlations between the decay products in $B^+ \to X(3872) K^+$ decays, where $X(3872) \to J/\psi\pi^+\pi^-$ and $J/\psi \to \mu^+\mu^-$, which unambiguously gives the $J^{PC} = 1^{++}$ assignment under the assumption that the $\pi^+\pi^-$ and J/ψ are in an S-wave. AAIJ 15AO extend this analysis with more data to limit D-wave contributions to < 4% at 95% CL.

See our note on "Developments in Heavy Quarkonium Spectroscopy".

X(3872) MASS FROM J/ψX MODE

VALUE (MeV)	EVTS	DOCUMENT ID	TECN	COMMENT
3871.69± 0.17 OUR AVERAGE				

Ecco il codice in RooFit (*test.C*) per fare i 3 fit in sequenza:

```
#include <TROOT.h>
#include <TFile.h>
#include <TH1.h>
#include <TF1.h>
#include <TF2.h>
#include <TFormula.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <TProfile.h>
#include <TString.h>
#include <TLine.h>
#include <TPad.h>
#include <TMath.h>
#include <TLatex.h>
#include <TLegend.h>
#include <iostream>
#include <TColor.h>
#include "TAxis.h"

using namespace RooFit;

TStyle *myStyle= new TStyle("myStyle","myStyle");

/////////////////////////---inizio main  ///// to execute: .L test.C + main()

//void main(TString date, TString extens) {

void main() {
  //
  gROOT->SetStyle("Plain");
  gStyle->SetCanvasColor(0);
  gStyle->SetOptStat(10);
  //
  //gROOT->SetStyle("myStyle");
  //myStyle->SetFrameBorderMode(0); myStyle->SetCanvasBorderMode(0);
  //myStyle->SetPadBorderMode(0); myStyle->SetPadColor(0);
  //myStyle->SetStatColor(0); myStyle->SetFillColor(0);
  //myStyle->SetStatBorderStyle(1);
  //
  TCanvas* myC = new TCanvas("myC","Plots",700,700);
  myC->SetFrameFillColor(0);
  //myC->cd(1)->SetBottomMargin(0.41); myC->cd(1)->SetTopMargin(0.05) ;
  //
  ////////////////////////////////////////////////////////////////////////
  //
  TFile f1("./esame-dec2014.root","READ");
  TH1D *hist = (TH1D*)f1.Get("PsiPrime_Mass_cut6");
  //
  RooRealVar x("x","x",3.6,4.0);
  RooDataHist jpsipipi_mass(hist->GetName(),hist->GetTitle(),RooArgSet(x),RooFit::Import(*hist, kFALSE));
  //
  ////////////////////////////////////////////////////////////////////////
```

file esterno ← (TFile f1)

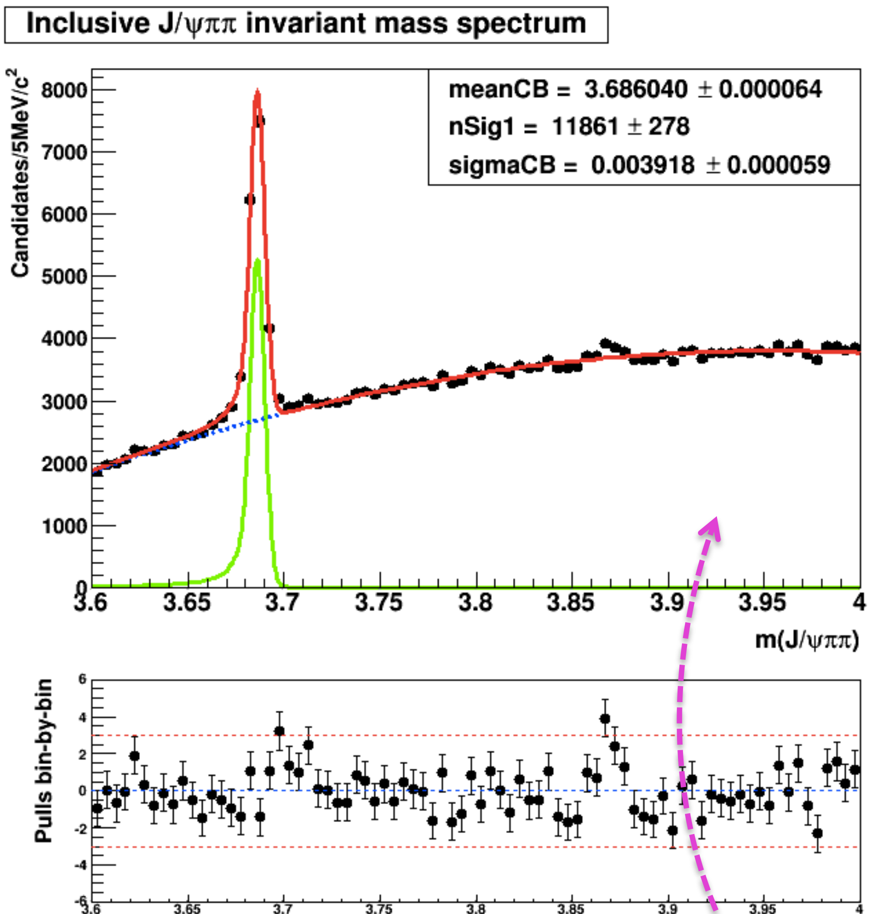
Primo fit:

```
//
RooPlot* xframe = x.frame(Title(""));
xframe->SetTitle("Inclusive J/#psi#pi#pi invariant mass spectrum");
xframe->SetTitleOffset(1.32,"y");
xframe->SetYTitle("Candidates/5MeV/c^{2}");
xframe->SetTitleOffset(1.26,"x");
xframe->SetXTitle("m(J/#psi#pi#pi)");
//
jpsipipi_mass.plotOn(xframe);
//xframe->Draw(); // to have immediately a first look to the histogram content
//
char *title[128]; jpsipipi_mass->SetTitle(*title); title ="";
//
////////////////////////////////////////////////// FIT
//
// signal
RooRealVar mg("mg","Gaussian's mean",3.685, 3.675, 3.695);
RooRealVar wg("wg","Gaussian's width",0.01, 0.001, 0.05);
RooGaussian myGauss("myGauss","Gauss(x,mg,wg,)",x,mg,wg);
//
// background
RooRealVar c0("c0","1st coeff",0.3,-100000,100000);
RooRealVar c1("c1","2nd coeff",-0.1,-100000,100000);
RooChebychev cheby("cheby","Chebyshev",x,RooArgList(c0,c1));
//
////RooRealVar c2("c2","3rd coeff",1.,-100000,100000);
////RooRealVar c3("c3","4th coeff",0.5,-1000,1000);
////RooChebychev cheby("cheby","Chebyshev",x,RooArgList(c0,c1,c2,c3));
//
// total pdf : f*gauss + (1-f)*cheby
//RooRealVar fsig("fsig","signal fraction",0.02,0.0,0.7);
//
RooRealVar nSig("nSig","Number of signal cands", 4e+5, 1.,1e+7);
RooRealVar nBkg("nBkg","Number of bkg componet", 120e+3, 1., 1e+8);
RooAddPdf* totalPdf = new RooAddPdf("totalPdf","totalPdf",RooArgList(myGauss,cheby),RooArgList(nSig,nBkg));
//

//
totalPdf->fitTo(jpsipipi_mass,Extended(kTRUE));
totalPdf->plotOn(xframe,RooFit::LineColor(kRed));
totalPdf->plotOn(xframe,RooFit::Components(RooArgSet(myGauss)), LineColor(kGreen));
totalPdf->plotOn(xframe,RooFit::Components(cheby),RooFit::LineStyle(kDashed));
// plot full fit again to make correct pulls
totalPdf->plotOn(xframe,RooFit::LineColor(kRed));
//totalPdf->paramOn(xframe);
totalPdf->paramOn(xframe, Parameters(RooArgSet(mg,wg,nSig)), Layout(0.45,0.9,0.9));
//
```

Rappresentazione primo fit:

```
//
RooPlot *framePull = x.frame("");
framePull->addObject((TObject*)xframe->pullHist(),"p");
framePull->SetTitle("");
 framePull->SetLabelSize(0.055,"y");
framePull->SetTitleSize(0.085,"y");
framePull->SetTitleOffset(0.35,"y");
framePull->SetYTitle("Pulls bin-by-bin");
framePull->SetLabelSize(0.055,"x");
framePull->SetXTitle(" ");
framePull->SetMinimum(-6.);
framePull->SetMaximum(6.);
//
myC->Divide(0,2);
myC->cd(2);
gPad->SetPad(0.,0.,1.,0.3);
//framePull->SetTitleOffset(1.25,"y");
//framePull->SetTitleSize(0.1,"y");
gStyle->SetLabelSize(0.06,"Y");
gStyle->SetTitleYSize(0.03);
framePull->Draw();
TLine* lineplus = new TLine(3.6,3.,4.,3.);
TLine* lineminus = new TLine(3.6,-3.,4.,-3.);
TLine* linezero = new TLine(3.6,0.,4.,0.);
lineplus->SetLineStyle(2);
lineplus->SetLineColor(2);
lineplus->Draw("same");
lineminus->SetLineStyle(2);
lineminus->SetLineColor(2);
lineminus->Draw("same");
linezero->SetLineStyle(2);
linezero->SetLineColor(4);
linezero->Draw("same");
myC->cd(1);
gPad->SetPad(0.,0.3,1.,1.);
xframe->Draw();
//
myC->SaveAs("./psiprime_gauss_cheby2.png");
//myC->Update();
delete myC;
```

Secondo fit:




```
/////////////////////////////////////////////////////////// NEW FIT
//
TCanvas* myC1 = new TCanvas("myC1","Plots",700,700);
myC1->SetFrameFillColor(0);
//
RooPlot* xframe1 = x.frame("");
xframe1->SetTitle("Inclusive J/#psi#pi#pi invariant mass spectrum");
xframe1->SetTitleOffset(1.32,"y");
xframe1->SetLabelSize(0.035,"y");
xframe1->SetTitleSize(0.037,"y");
xframe1->SetYTitle("Candidates/5MeV/c^{2}");
xframe1->SetTitleOffset(1.26,"x");
xframe1->SetXTitle("m(J/#psi#pi#pi)");
jpsipipi_mass.plotOn(xframe1);
//
// alternative (CB)
RooRealVar meanCB("meanCB", "meanCB", 3.685, 3.675, 3.695);
RooRealVar sigmaCB("sigmaCB", "sigmaCB", 0.0042222, 0.0004, 0.005);
RooRealVar alpha("alpha","alpha", 1.0, 0.00001, 10000.);
RooRealVar nCB("nCB","nCB", 1.0, 0.0001, 10000.);
//
RooCBShape myCB("myCB", "myCB", x, meanCB, sigmaCB, alpha, nCB);
//
RooRealVar nSig1("nSig1","Number of signal cands", 1e+4, 100.,1e+7);
RooRealVar nBkg1("nBkg1","Number of bkg componet", 2e+5, 1000., 1e+8);
RooAddPdf* totalPdf1 = new RooAddPdf("totalPdf1","totalPdf1",RooArgList(myCB,cheby),RooArgList(nSig1,nBkg1));
//
totalPdf1->fitTo(jpsipipi_mass,Extended(kTRUE));
totalPdf1->plotOn(xframe1,RooFit::LineColor(kRed));
totalPdf1->plotOn(xframe1,RooFit::Components(RooArgSet(myCB)), LineColor(kGreen));
totalPdf1->plotOn(xframe1,RooFit::Components(cheby),RooFit::LineStyle(kDashed));
// plot full fit again to make correct pulls
totalPdf1->plotOn(xframe1,RooFit::LineColor(kRed));
//totalPdf1->paramOn(xframe1); // non mettere proprio le stime dei parametri restituite dal fit
totalPdf1->paramOn(xframe1, Parameters(RooArgSet(meanCB,sigmaCB,nSig1)), Layout(0.45,0.9,0.9));
//
////////////////////////////////
//
RooPlot *framePull1 = x.frame("");
framePull1->addObject((TObject*)xframe1->pullHist(),"p");
framePull1->SetTitle("");
framePull1->SetLabelSize(0.055,"y");
framePull1->SetTitleSize(0.085,"y");
framePull1->SetTitleOffset(0.35,"y");
framePull1->SetYTitle("Pulls bin-by-bin");
framePull1->SetLabelSize(0.055,"x");
framePull1->SetXTitle(" ");
framePull1->SetMinimum(-6.);
framePull1->SetMaximum(6.);
//
myC1->Divide(0,2);
myC1->cd(2);
//
gPad->SetPad(0.,0.,1.,0.3);
framePull1->Draw();
TLine* lineplus1 = new TLine(3.6,3.,4.,3.);
TLine* lineminus1 = new TLine(3.6,-3.,4.,-3.);
TLine* linezero1 = new TLine(3.6,0.,4.,0.);
lineplus1->SetLineStyle(2);
lineplus1->SetLineColor(2);
lineplus1->Draw("same");
lineminus1->SetLineStyle(2);
lineminus1->SetLineColor(2);
lineminus1->Draw("same");
linezero1->SetLineStyle(2);
linezero1->SetLineColor(4);
linezero1->Draw("same");
//
myC1->cd(1);
gPad->SetPad(0.,0.3,1.,1.);
xframe1->Draw();
//
myC1->SaveAs("./psiprime_cb_cheby2.png");
```

Terzo fit:

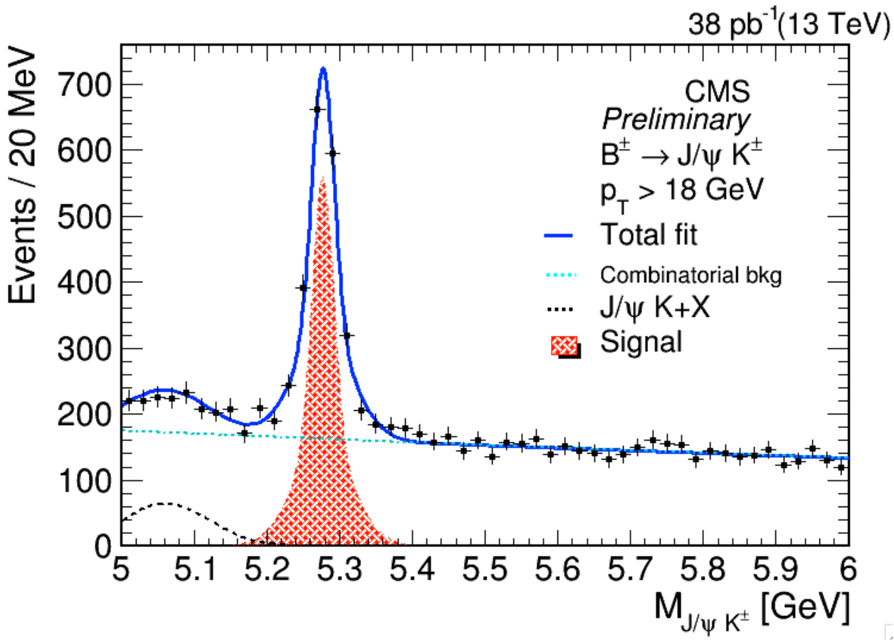
```
///////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////  NEW FIT
///////////////////////////////////////////////////////////////
//
gROOT->SetStyle("Plain");
gStyle->SetCanvasColor(0);
gStyle->SetOptStat(10);
//
TCanvas* myC2 = new TCanvas("myC2","Plots",700,700);
myC2->SetFrameFillColor(0);
//
RooPlot* xframe2 = x.frame(Title(""));
xframe2->SetTitle("Inclusive J/#psi#pi#pi invariant mass");
xframe2->SetTitleOffset(1.32,"y");
xframe2->SetLabelSize(0.035,"y");
xframe2->SetTitleSize(0.037,"y");
xframe2->SetYTitle("Candidates/5MeV/c^{2}");
xframe2->SetTitleOffset(1.26,"x");
xframe2->SetXTitle("m(J/#psi#pi#pi)");
jpsipipi_mass.plotOn(xframe2);
//
// signal
RooRealVar mgX("mgX","Gaussian's mean",3.868, 3.85, 3.88);
RooRealVar wgX("wgX","Gaussian's width",0.005, 0.002, 0.015);
//wgX.setConstant(kTRUE);
RooGaussian myGaussX("myGaussX","GaussX(x,mgX,wgX)",x,mgX,wgX);
//mgX.setConstant(kTRUE);
//
//RooRealVar meanCB2("meanCB2", "meanCB2", 3.685, 3.675, 3.695);
//RooRealVar sigmaCB2("sigmaCB2", "sigmaCB2", 0.004, 0.0001, 0.05);
//RooRealVar alpha2("alpha2","alpha2", 1.0, 0.00001, 10000.);
//RooRealVar nCB2("nCB2","nCB2", 1.0, 0.0001, 10000.);
//
RooRealVar meanCB2("meanCB2", "meanCB2", 3.686038, 3.68, 3.692);
RooRealVar sigmaCB2("sigmaCB2", "sigmaCB2", 0.003919, 0.001, 0.05);
RooRealVar alpha2("alpha2","alpha2", 1.41, 0.01, 10.);
RooRealVar nCB2("nCB2","nCB2", 1.64, 0.1, 10.);
// start fixing the CB parameters from previous fit and release them later (leave just # candidates free)
//meanCB2.setConstant(kTRUE);
//sigmaCB2.setConstant(kTRUE);
//alpha2.setConstant(kTRUE);
//nCB2.setConstant(kTRUE);
//
RooCBShape myCB2("myCB2", "myCB2", x, meanCB2, sigmaCB2, alpha2, nCB2);
//
RooRealVar nSigPsi("nSigPsi","Number of signal psi cands", 11858, 10000.,15000.); //start from previous plot to help
/////////////////////////////////////////nSigPsi.setConstant(kTRUE);
//
RooRealVar nSigX("nSigX","Number of signal X cands", 550, 350., 1500.);
//nSigX.setConstant(kTRUE);
//
////////RooAddPdf* totalSig2 = new RooAddPdf("totalSig2","totalSig2",RooArgList(myCB2,myGaussX),RooArgList(nSigPsi,nSigX));
//RooAddPdf totalSig2("totalSig2","totalSig2",RooArgList(myCB2,myGaussX),RooArgList(nSigPsi,nSigX));
//
//RooRealVar nSig2("nSig2","Number of total sig component", 2e+5, 1000., 1e+7);
RooRealVar nBkg2("nBkg2","Number of bkg component", 2e+5, 1000., 1e+8);

//RooAddPdf* totalPdf2 = new RooAddPdf("totalPdf2","totalPdf2",RooArgList(totalSig2,cheby),RooArgList(nSig2,nBkg2));
RooAddPdf* totalPdf2 = new RooAddPdf("totalPdf2","totalPdf2",RooArgList(myCB2,myGaussX,cheby),RooArgList(nSigPsi,nSigX,nBkg2));
// one shot fit !
//
gStyle->SetLineWidth(1); // cambia nulla
gStyle->SetFuncWidth(1);
//
totalPdf2->fitTo(jpsipipi_mass,Extended(kTRUE));
totalPdf2->plotOn(xframe2,RooFit::LineColor(kRed),RooFit::LineWidth(1));
// by default LineWidth is 3 pixels (somehow thick, while 1 is too subtle)
totalPdf2->plotOn(xframe2,RooFit::Components(RooArgSet(myGaussX)), LineColor(kGreen),RooFit::LineWidth(2));
totalPdf2->plotOn(xframe2,RooFit::Components(RooArgSet(myCB2)), LineColor(kGreen),RooFit::LineWidth(2));
totalPdf2->plotOn(xframe2,RooFit::Components(cheby),RooFit::LineStyle(kDashed),RooFit::LineWidth(2));
// plot full fit again to make correct pulls
totalPdf2->plotOn(xframe2,RooFit::LineColor(kRed),RooFit::LineWidth(2));
totalPdf2->paramOn(xframe2, Parameters(RooArgSet(meanCB2,sigmaCB2,nSigPsi,mgX,wgX,nSigX)), Layout(0.45,0.9,0.9));
```

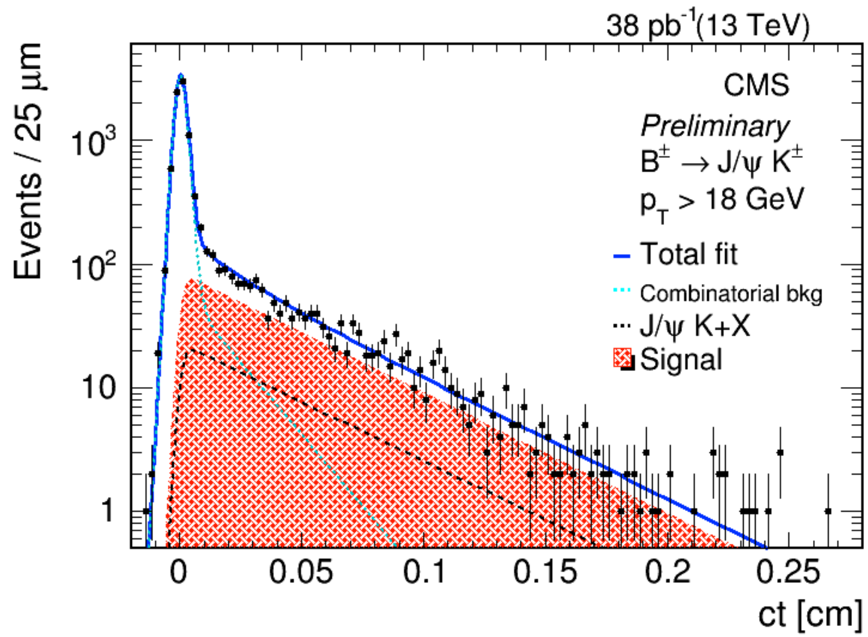
Rappresentazione terzo fit:

```
  //
  RooPlot *framePull2 = x.frame("");
  framePull2->addObject((TObject*)xframe2->pullHist(),"p");
  framePull2->SetTitle(""); // elimina titolo
  framePull2->SetLabelSize(0.055,"y");
  framePull2->SetTitleSize(0.085,"y");  // ingrandisce ma sposta anche verso sinistra
  framePull2->SetTitleOffset(0.35,"y"); // risposta a destra
  framePull2->SetYTitle("Pulls bin-by-bin");
  framePull2->SetLabelSize(0.055,"x");
  framePull2->SetXTitle(" ");  //framePull2->SetXTitle("m(J/#psi#pi#pi)"); // pleonastico
  framePull2->SetMinimum(-6.);
  framePull2->SetMaximum(6.);
  //
  myC2->Divide(0,2);
  myC2->cd(2);
  //
  gPad->SetPad(0.,0.,1.,0.3);
  framePull2->Draw();
  //
  TLine* lineplus2 = new TLine(3.6,3.,4.,3.);
  TLine* lineminus2 = new TLine(3.6,-3.,4.,-3.);
  TLine* linezero2 = new TLine(3.6,0.,4.,0.);
  lineplus2->SetLineStyle(2);
  lineplus2->SetLineColor(2);
  lineplus2->Draw("same");
  lineminus2->SetLineStyle(2);
  lineminus2->SetLineColor(2);
  lineminus2->Draw("same");
  linezero2->SetLineStyle(2);
  linezero2->SetLineColor(4);
  linezero2->Draw("same");
  //
  myC2->cd(1);
  gPad->SetPad(0.,0.3,1.,1.);
  xframe2->Draw();
  //
  myC2->SaveAs("./psiprime_cb_cheby2_x3872.png");
  //myC2->Clear();
  delete myC2;
  //
  ///////////////////////////////////////////////
  //
  f1.Close();
  f1.Delete();
  //
  gROOT->Reset();
  gROOT->Clear();
  //
}
```

Esercitazione n.4

In questa esercitazione impareremo ad eseguire interpolazioni bidimensionali. Nello specifico produrremo il seguente tipo di plot:





Questi sono *plot* di CMS con i primi dati a $\sqrt{s} = 13 TeV$
ma non sono stati oggetto di pubblicazione poiche' sono stati ottenuti con un trigger inclusivo di J/ψ e non con un trigger J/ψ displaced (meno fondo!).
Sono inoltre ottenuti con un campione di dati piccolissimo (primissimi dati del Run-II).

Si vuole interpolare contemporaneamente due osservabili:

- la massa invar. $J/\psi(\mu^+\mu^-)K^\pm$ con segnale del mesone *B*+ ($B^\pm \to J/\psi K^\pm$)

- il tempo proprio del suddetto spettro

Il fine e' la stima della vita media del mesone *B*+.

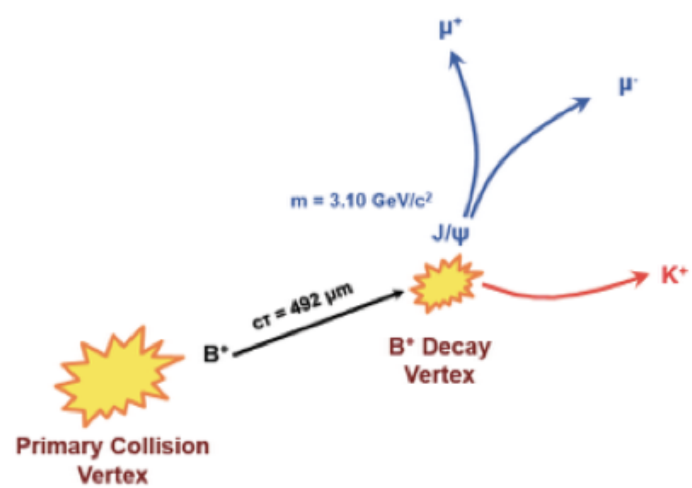
Si ricordi che:

tempo di volo ***distanza di volo***

***tempo proprio* :** $$t = \frac{t_{LAB}}{\gamma} = \frac{1}{\gamma}\cdot\frac{l_{DEC}}{\beta c} \quad\Rightarrow\quad ct = \frac{l_{DEC}}{\beta\gamma} \equiv \frac{l_{DEC}}{\beta\gamma}\cdot\frac{m_{B^+}^{PDG}}{m_{B^+}^{PDG}} = m_{B^+}^{PDG}\cdot\frac{l_{DEC}}{p_{B^+}}$$

Quindi, a seconda che la distanza di volo sia 3D o nel piano trasverso, si ha:

$$ct = m_{B^+}^{PDG}\cdot\frac{l_{DEC}}{p_{B^+}} = m_{B^+}^{PDG}\cdot\frac{l_{DEC}^{\perp}}{p_{B^+}^{\perp}}$$

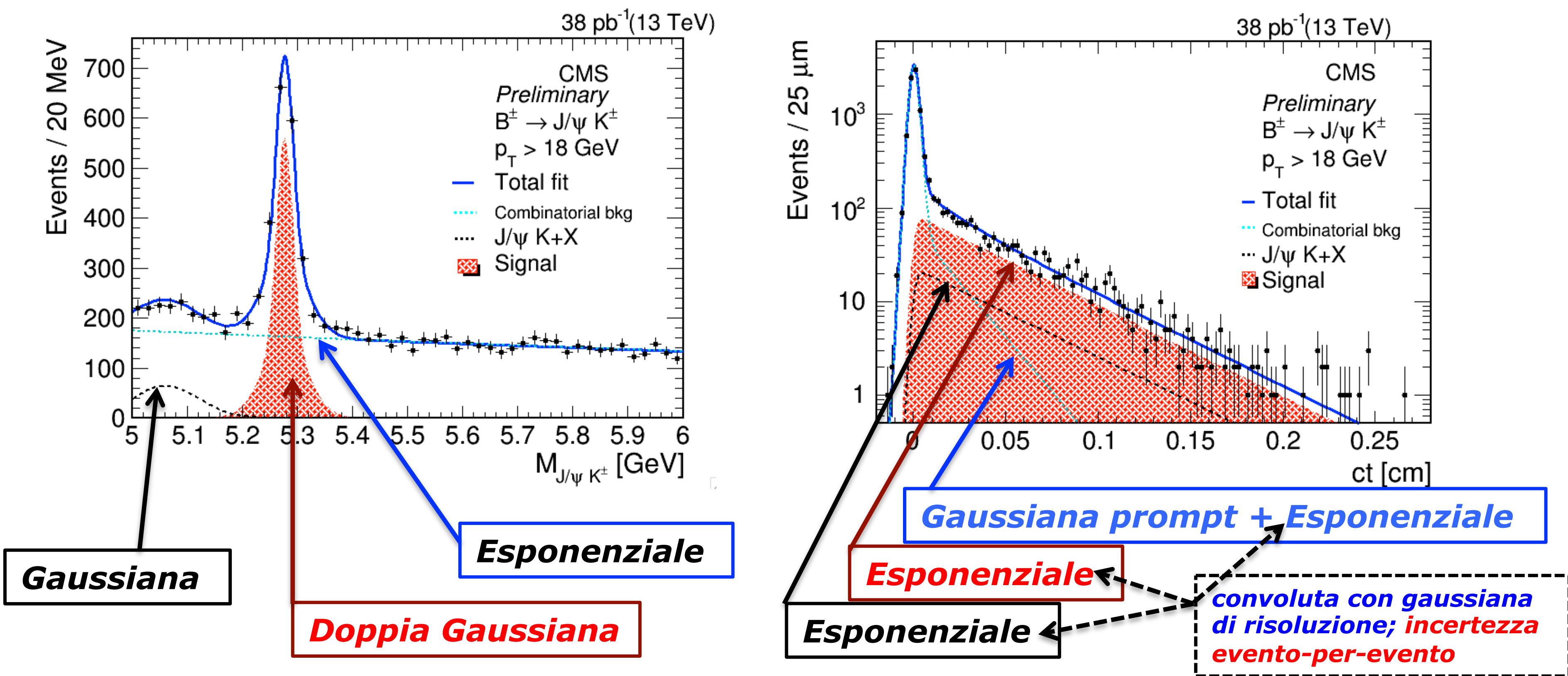


Si ricordi che, indicata con τ la vita media, si ha, per il B+: $c\tau_{B^+} \cong 492\mu m$

Prima di passare al dettaglio implementativo in RooFit cerchiamo di capire il modello fisico che definiremo per l'interpolazione.

Segnale : decadimenti $B^{\pm} \to J/\psi K^{\pm}$

Fondo :
- **combinatorio (dominato da prompt J/ψ + traccia random)**
- **fisico (decadimenti del tipo $B \to J/\psi K + X$ con X non ricostruito)**



RooFit *macro* per il fit bidimensionale: *myfitter2d.cc*

```
#include <TStyle.h>
#include <TAxis.h>
#include <TLatex.h>
#include <TPaveText.h>
#include <TFile.h>
#include <TTree.h>
#include <TCanvas.h>
#include <TNtupleD.h>
#include <TH1D.h>
//
#include <RooRealVar.h>
#include <RooDataSet.h>
#include <RooGaussian.h>
#include <RooChebychev.h>
#include <RooExponential.h>
#include <RooAddPdf.h>
#include <RooProdPdf.h>
#include <RooDecay.h>
#include <RooGaussModel.h>
#include <RooAddModel.h>
#include <RooPlot.h>
//
#include "myloop.h"
#include "plotDressing2D.h"

using namespace RooFit;

// General fitting options
#define NUMBER_OF_CPU       1
#define DO_MINOS            kTRUE
// 0 - w/o DISPLAY
// 1 - w/  DISPLAY
#define DISPLAY             1

#define MASS_MIN            5.0
#define MASS_MAX            6.0
#define MASS_PEAK           BP_MASS
#define SOURCE              "myloop.root"
```

Typical service ROOT classes included

RooFit classes to build fit mode included

Inclusione di file esterni

Istruzione necessaria per usare RooFit

Intervallo di valore di massa & massa nominale del B+ (PDG) [definita in *myloop.h*]

Rootupla di input: *myloop.root*

plotDressing2D.h **definisce opzioni per la canvas e i plot**

myloop.h **definisce la classe per leggere la rootupla in input !**
Viene generata con i comandi *makeClass* (o *makeSelector*) di ROOT.

Nel caso specifico si tratta della classe *ReducedBranches* :

```
class ReducedBranches{
    public:
    int      run;
    int      event;

    int      type;  // B hadron information
    double   mass;
    double   pt;
    double   eta;
    double   phi;
    double   y;
    double   vx;
    double   vy;
    double   vz;
    double   lxy;
    double   lxyz;
    double   errxy;
    double   errxyz;
    double   vtxprob;
    double   cosalpha2d;
    double   cosalpha3d;
    double   ctau2d;
    double   ctau3d;
    double   ctau2derr;
    double   ctau3derr;

    double   ujmass; // dimuon information
    double   ujpt;
    double   ujeta;
    double   ujphi;
    double   ujy;
    double   ujvtxprob;

    double   tktkmass; // ditrack information
    double   tktkpt;
    double   tktketa;
    double   tktkphi;
    double   tktky;
    double   tktkvtxprob;
    double   tktklxy;
    double   tktklxyz;
    double   tktkerrxy;
    double   tktkerrxyz;
    double   tktkblxy;
    double   tktkblxyz;
    double   tktkberrxy;
    double   tktkberrxyz;

    int      mu1idx;
    double   mu1pt;
    double   mu1eta;
    double   mu1phi;
    int      mu2idx;
    double   mu2pt;
    double   mu2eta;
    double   mu2phi;

    int      tk1idx;
    double   tk1pt;
    double   tk1eta;
    double   tk1phi;
    int      tk2idx;
    double   tk2pt;
    double   tk2eta;
    double   tk2phi;

    int nhltbook; // triggers
    int hltbook[N_HLT_BOOKINGS];

    void regTree(TTree *root){
        root->Branch("run",&run,"run/I");
        root->Branch("event",&event,"event/I");
        root->Branch("type",&type,"type/I");
        root->Branch("mass",&mass,"mass/D");
        root->Branch("pt",&pt,"pt/D");
        root->Branch("eta",&eta,"eta/D");
        root->Branch("phi",&phi,"phi/D");
        root->Branch("y",&y,"y/D");
        ⋮
```

Nella prima parte riportata c'e' l'insieme delle dichiarazioni delle variabili contenute nella rootupla (che puo' essere ispezionata nel solito modo: con il *TBrowser*).

A mano l'analista puo' aggiungere altre dichia-razioni di visibilita' (*scope*) generale, come, p.es., delle costanti :

```
#define MUON_MASS     0.10565837
#define PION_MASS     0.13957018
#define KAON_MASS     0.493677
#define KSHORT_MASS   0.497614
#define KSTAR_MASS    0.89594
#define PHI_MASS      1.019455
#define JPSI_MASS     3.096916
#define PSI2S_MASS    3.686109
#define PROTON_MASS   0.938272046
#define LAMBDA_MASS   1.115683
#define BP_MASS       5.27926
#define B0_MASS       5.27958
#define BS_MASS       5.36677
#define BC_MASS       6.2756
#define LAMBDAB_MASS  5.6195
```

La prima parte di *myfitter2d.cc* legge la rootupla per ricavare terne di valori (una terna per candidato B^+). La terna consiste nei valori di : 1) *mass*a, 2) tempo proprio (*ct*), 3) errore su tempo proprio (*cterr*).

```
void myfitter2d()
{
  // define variables: mass, proper time and error on proper tim:
  RooRealVar mass("mass","mass",MASS_MIN,MASS_MAX);
  RooRealVar ct("ct","ct",-0.02,0.28);
  RooRealVar cterr("cterr","cterr",0.0001,0.008);

  //outout
  TFile *fout = new TFile("myfitter2d.root","recreate");      // output file
  TNtupleD *_nt = new TNtupleD("_nt","_nt","mass:ct:cterr"); // output ntuple

  // input
  TFile *fin = new TFile(SOURCE);
  TTree *tin = (TTree*)fin->Get("ntkp");

  // setting up rootuple for reading
  ReducedBranches br;
  br.setbranchadd(tin);

  // reading rootuple
  for (int evt=0;evt<tin->GetEntries();evt++)
    {
     tin->GetEntry(evt);
     //
     // cuts to select events/cands
     if (br.hltbook[HLT_Dimuon16_Jpsi_v1]!=1) continue;
     if (br.vtxprob<=0.15) continue;
     if (br.tk1pt<=2.0) continue;
     //
     // filling the 3D vector in the output ntuple
     double var[3];
     var[0] = br.mass;
     var[1] = br.ctau2d;
     var[2] = br.ctau2derr;
     _nt->Fill(var);
    }
  //
  fin->Close();
  //
  // the dataset contains only the 3 variables of interest
  RooDataSet *data = new RooDataSet("data","data",_nt,RooArgSet(mass,ct,cterr));
  //
  ///////////////////////////////////////////////////////////////////
  //
```

Rootupla di ouput: *myfitter2D.root*

Rootupla di input

Ulteriore selezione di eventi/candidati

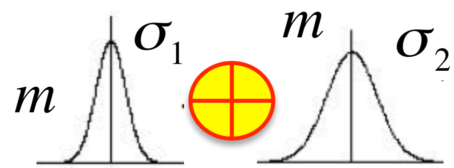
La ntupla nel file di output viene riempita

RooDataSet con la terna di variabili

Costruzione della PDF di segnale :

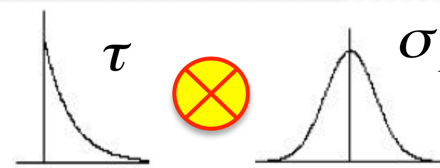
Doppia Gaussiana con media comune (e larghezza diversa);
Trattasi di ***somma (RooAddPdf) con coefficiente pari alla "frazione"***

```
//
// signal PDF
//===========
//
// double gaussian for the signal in mass
//
RooRealVar m_mean("m_mean","m_mean",MASS_PEAK,MASS_MIN,MASS_MAX);
RooRealVar m_sigma1("m_sigma1","m_sigma1",0.016,0.001,0.045);
RooRealVar m_sigma2("m_sigma2","m_sigma2",0.035,0.001,0.090);
RooRealVar m_fraction("m_fraction","m_fraction",0.5);
//
RooGaussian m_gaussian1("m_gaussian1","m_gaussian1",mass,m_mean,m_sigma1);
RooGaussian m_gaussian2("m_gaussian2","m_gaussian2",mass,m_mean,m_sigma2);
//
RooAddPdf pdf_m_signal("pdf_m_signal","pdf_m_signal",RooArgList(m_gaussian1,m_gaussian2),RooArgList(m_fraction));
//
// exponential convoluted with gausian resolution for the signal in ct
//
RooRealVar res_sig_mean("res_sig_mean","res_sig_mean",0.0,-1.,1.);
RooRealVar res_sig_sigma("res_sig_sigma","res_sig_sigma",1.0,0.3,2.0);
//
RooGaussModel res_signal("res_signal","res_signal",ct,res_sig_mean,res_sig_sigma,cterr);
//
RooRealVar ctau("ctau","ctau",0.04911,0.010,0.090);
RooDecay pdf_t_signal("pdf_t_signal","pdf_t_signal",ct,ctau,res_signal,RooDecay::SingleSided);
//
// bidimensional signal pdf
//
RooProdPdf pdf_signal("pdf_signal","pdf_signal", RooArgSet(pdf_m_signal, pdf_t_signal));
//
```



coefficiente :
$cG_1 + (1-c)G_2$

S σ_i



$\sigma_{RES} = S\sigma_i$

PDF bidimensionale del segnale (*RooProdPdf* delle due PDF per le due variabili)

Esponenziale convoluta con gaussiana di risoluzione sperimentale in tempo proprio

Scale Factor **to take into account eventual systematic under/over-estimation of proper-time event-by-event error**

class RooDecay: public RooAbsAnaConvPdf

Single or double sided decay function that can be analytically convolved with any RooResolutionModel implementation

Function Members (Methods)

public:

```
         virtual  ~RooDecay ()
    static TClass*  Class ()
  virtual TObject*  clone (const char* newname) const
 virtual Double_t  coefficient (Int_t basisIndex) const
      virtual void  generateEvent (Int_t code)
     virtual Int_t  getGenerator (const RooArgSet& directVars, RooArgSet& generateVars, Bool_t staticInitOK = kTRUE) const
   virtual TClass*  IsA () const
        RooDecay&  operator= (const RooDecay&)
                    RooDecay ()
                    RooDecay (const RooDecay& other, const char* name = 0)
                    RooDecay (const char* name, const char* title, RooRealVar& t, RooAbsReal& tau, const RooResolutionModel&
                        model, RooDecay::DecayType type)
      virtual void  ShowMembers (TMemberInspector& insp) const
      virtual void  Streamer (TBuffer&)
              void  StreamerNVirtual (TBuffer& ClassDef_StreamerNVirtual_b)
```

protected:

Data Members

public:

```
static RooDecay::DecayType  DoubleSided
static RooDecay::DecayType  Flipped
static RooDecay::DecayType  SingleSided
```

protected:

```
                Int_t  _basisExp
         RooRealProxy  _t
         RooRealProxy  _tau
  RooDecay::DecayType  _type
```

Documentazione:
- **roofit.sourceforge.net**
- **https://root.cern.ch/root/html/**

RooGaussModel
res_signal

Il costruttore della classe *RooDecay* e':

RooDecay (const char* name, const char* title, RooRealVar& t, RooAbsReal& tau, const RooResolutionModel& model, RooDecay::DecayType type)

SingleSided   **variabile *ct***   **parametro τ**

class RooGaussModel: public RooResolutionModel

Class RooGaussModel implements a RooResolutionModel that models a Gaussian distribution. Object of class RooGaussModel can be used for analytical convolutions with classes inheriting from RooAbsAnaConvPdf

Function Members (Methods)

public:

```
virtual              ~RooGaussModel ()
void                 advertiseAymptoticIntegral (Bool_t flag)
void                 advertiseFlatScaleFactorIntegral (Bool_t flag)
virtual Double_t     analyticalIntegral (Int_t code, const char* rangeName) const
virtual Int_t        basisCode (const char* name) const
static TClass*       Class ()
virtual TObject*     clone (const char* newname) const
virtual void         generateEvent (Int_t code)
virtual Int_t        getAnalyticalIntegral (RooArgSet& allVars, RooArgSet& analVars, const char* rangeName = 0) const
virtual Int_t        getGenerator (const RooArgSet& directVars, RooArgSet& generateVars, Bool_t staticInitOK = kTRUE) const
virtual TClass*      IsA () const
RooGaussModel&       operator= (const RooGaussModel&)
                     RooGaussModel ()
                     RooGaussModel (const RooGaussModel& other, const char* name = 0)
                     RooGaussModel (const char* name, const char* title, RooRealVar& x, RooAbsReal& mean, RooAbsReal& sigma)
                     RooGaussModel (const char* name, const char* title, RooRealVar& x, RooAbsReal& mean, RooAbsReal& sigma,
                        RooAbsReal& msSF)
                     RooGaussModel (const char* name, const char* title, RooRealVar& x, RooAbsReal& mean, RooAbsReal& sigma,
                        RooAbsReal& meanSF, RooAbsReal& sigmaSF)
virtual void         ShowMembers (TMemberInspector& insp) const
virtual void         Streamer (TBuffer&)
void                 StreamerNVirtual (TBuffer& ClassDef_StreamerNVirtual_b)
```

protected:

```
static complex<Double_t>  evalCerf (Double_t swt, Double_t u, Double_t c)
static complex<Double_t>  evalCerfApprox (Double_t swt, Double_t u, Double_t c)
complex<Double_t>         evalCerfInt (Double_t sign, Double_t wt, Double_t tau, Double_t umin, Double_t umax, Double_t c) const
virtual Double_t          evaluate () const
```

Data Members

•
•
•

Uno dei costruttori della classe *RooGaussModel* e' :

RooGaussModel (const char* name, const char* title, RooRealVar& x, RooAbsReal& mean, RooAbsReal& sigma, RooAbsReal& msSF)

ct $\overline{t}_{RES}$ S

cterr

Costruzione della PDF del fondo combinatorio (traccia random):

Vita media *del fondo combinatorio*

Si ricorre alla stessa funzione di risoluzione gaussiana usata per il segnale: *res_signal*

```
//
// combinatorial background PDF (prompt or non-prompt J/psi + random track)
//===================================================================
//
// exponential for the combinatorial background in mass
//
RooRealVar m_par1("m_par1","m_par1",-0.3,-2.,+2.);
RooExponential pdf_m_combinatorial("pdf_m_combinatorial","pdf_m_combinatorial",mass,m_par1);
//
// exponential convoluted with gaussian resolution for the non-prompt background in ct
//
RooRealVar ctau_nonprompt("ctau_nonprompt","ctau_nonprompt",0.0500, 0.0010, 0.1000);
RooDecay pdf_t_nonprompt("pdf_t_nonprompt","pdf_t_nonprompt",ct,ctau_nonprompt,res_signal,RooDecay::SingleSided);
//
// Sum of gaussian resolution function (res_signal) for prompt background in ct and the previous exponential for NP-bkg
//
RooRealVar prompt_fraction("prompt_fraction","prompt_fraction",0.5,0.0,1.0);
//
RooAddPdf pdf_t_combinatorial("pdf_t_combinatorial","pdf_t_combinatorial",RooArgList(res_signal,pdf_t_nonprompt),RooArgList(prompt_fraction));
//
// bidimensional combinatorial-bkg pdf
//
RooProdPdf pdf_combinatorial("pdf_combinatorial","pdf_combinatorial",RooArgSet(pdf_m_combinatorial,pdf_t_combinatorial));
//
```

Costruzione della PDF del fondo fisico:

Si ricorre alla stessa funzione di risoluzione gaussiana usata per il segnale ed il fondo combinatorio NP:

res_signal

```
//
// B->J/psi+track+X background PDF
//==============================
//
// single gaussian for the physical background in mass
//
RooRealVar m_jpsix_mean("m_jpsix_mean","m_jpsix_mean",5.1,5.0,5.3);
RooRealVar m_jpsix_sigma("m_jpsix_sigma","m_jpsix_sigma",0.05,0.01,0.10);
RooGaussian pdf_m_jpsix("pdf_m_jpsix","pdf_m_jpsix",mass,m_jpsix_mean,m_jpsix_sigma);
//
// exponential convoluted with gaussian resolution for the physical background in ct
//
RooRealVar ctau_jpsix("ctau_jpsix","ctau_jpsix",0.0500, 0.0010, 0.1000);
RooDecay pdf_t_jpsix("pdf_t_jpsix","pdf_t_jpsix",ct,ctau_jpsix,res_signal,RooDecay::SingleSided);
//
// bidimensional physical-bkg pdf
//
RooProdPdf pdf_jpsix("pdf_jpsix","pdf_jpsix",RooArgSet(pdf_m_jpsix, pdf_t_jpsix));
//
```

Vita media *del fondo fisico*

Costruzione del modello 2D complessivo (segnale+2fondi) :

```
//
// FULL MODEL (SIGNAL + 2 BKGS)
//
// define coefficients for addition of the 3 pdfs
//
RooRealVar n_signal("n_signal","n_signal",n_signal_initial,0.,data->sumEntries());
RooRealVar n_combinatorial("n_combinatorial","n_combinatorial",n_combinatorial_initial,0.,data->sumEntries());
RooRealVar n_jpsix("n_jpsix","n_jpsix",200.,0.,data->sumEntries());
//
RooAddPdf model("model","model",
              RooArgList(pdf_signal, pdf_combinatorial, pdf_jpsix),
              RooArgList(n_signal, n_combinatorial, n_jpsix));
//
```

c_1 c_2 c_3

```
RooAddPdf is an efficient implementation of a sum of PDFs of the form

 c_1*PDF_1 + c_2*PDF_2 + ... c_n*PDF_n

or

 c_1*PDF_1 + c_2*PDF_2 + ... (1-sum(c_1...c_n-1))*PDF_n

The first form is for extended likelihood fits, where the
expected number of events is Sum(i) c_i. The coefficients c_i
can either be explicitly provided, or, if all components support
extended likelihood fits, they can be calculated the contribution
of each PDF to the total number of expected events.

In the second form, the sum of the coefficients is enforced to be one,
and the coefficient of the last PDF is calculated from that condition.
```

Il # di candidati di segnale e di fondo combinatorio vengono in precedenza dichiarati ed inizializzati

```
// initialization
//
double n_signal_initial = data->sumEntries(TString::Format("abs(mass-%g)<0.015",MASS_PEAK))
- data->sumEntries(TString::Format("abs(mass-%g)<0.030&&abs(mass-%g)>0.015",MASS_PEAK,MASS_PEAK));
//
double n_combinatorial_initial = data->sumEntries() - n_signal_initial;
//
```

Interpolazione (e plotting) !

Extended(kTRUE) ? NON SERVE !

```
//
// finally go for fitting !
//
model.fitTo(*data,Minos(DO_MINOS),NumCPU(NUMBER_OF_CPU),Offset(kTRUE));
//
// go to display plots with fits superimposed on data distributions
#if DISPLAY
//
// Display mass plots
//------------------
//
TCanvas *c1 = canvasDressing("c1");
//
RooPlot* frame_m = mass.frame();
//
TH1D* histo_data_m = (TH1D*)data->createHistogram("histo_data_m", mass, Binning(50,mass.getMin(),mass.getMax()));
//
   •
   •
   •
//
// Display c*proper-time plots
//----------------------------
//
TCanvas *c2 = canvasDressing("c2");
//
RooPlot* frame_t = ct.frame();
//
TH1D* histo_data_t = (TH1D*)data->createHistogram("histo_data_t", ct, Binning(120,ct.getMin(),ct.getMax()));
//
   •
   •
   •
```

Nota bene: il fit e' automaticamente del tipo EXTENDED !

Infatti:

- If **RooAddPdf** is given N coefficients instead of N-1 fractions
  → **RooAddPdf** is automatically extended
  → coefficients represent the expected #events for each PDF comp.

[da: http://roofit.sourceforge.net/docs/tutorial/intro/roofit_tutorial_intro.pdf]

Per eseguire la macro: ***.x myfitter2d.cc***

Oltre ad ottenere il plot gia' mostrato inizialmente,
si provi a commentare il risultato del fit.

```
**********
**   23 **MINOS        7500
**********
FCN=-2992.64 FROM MINOS     STATUS=SUCCESSFUL    4324 CALLS        6110 TOTAL
                     EDM=5.11676e-05    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                  PARABOLIC         MINOS ERRORS
 NO.   NAME      VALUE            ERROR      NEGATIVE      POSITIVE
  1  ctau         4.44363e-02   1.25997e-03  -1.23372e-03   1.28786e-03
  2  ctau_jpsix   4.56622e-02   2.73766e-03  -2.61995e-03   2.87062e-03
  3  ctau_nonprompt   1.86478e-02   1.67187e-03  -1.73138e-03   1.71269e-03
  4  m_jpsix_mean   5.06123e+00   6.43382e-03  -7.33523e-03   5.85437e-03
  5  m_jpsix_sigma   6.06444e-02   5.98874e-03  -5.38280e-03   6.85032e-03
  6  m_mean       5.27737e+00   6.90799e-04  -6.93813e-04   6.88152e-04
  7  m_par1      -2.69325e-01   4.14541e-02  -4.14491e-02   4.14688e-02
  8  m_sigma1     4.09537e-02   3.02175e-03  -2.71070e-03   3.87860e-03
  9  m_sigma2     1.50309e-02   6.81301e-04  -6.68188e-04   6.96192e-04
 10  n_combinatorial   7.67237e+03   9.21780e+01  -9.20165e+01   9.23685e+01
 11  n_jpsix      4.13610e+02   2.59086e+01  -2.54124e+01   2.64232e+01
 12  n_signal     1.54505e+03   4.63785e+01  -4.56911e+01   4.72517e+01
 13  prompt_fraction   9.41981e-01   4.97410e-03  -5.06926e-03   4.87957e-03
 14  res_sig_mean   2.27907e-01   1.68816e-02  -1.68765e-02   1.68933e-02
 15  res_sig_sigma   1.28223e+00   1.33976e-02  -1.32768e-02   1.35262e-02
                              ERR DEF= 0.5
```