Exercise/Hands-on #5

Advanced fitting

Statistical Data Analysis for HEP

Prof. Alexis Pompili (University of Bari Aldo Moro)*

Erasmus+ Teaching Mobility Program / 16-20 October 2023 @ Sofia Physics Faculty

Note: This material has been revised/updated on the 26.11.2023
(the code may slightly differ from the one used in October,
however it is entrely provided in the final slides)

* alexis.pompili@ba.infn.it (or alexis.pompili@cern.ch)

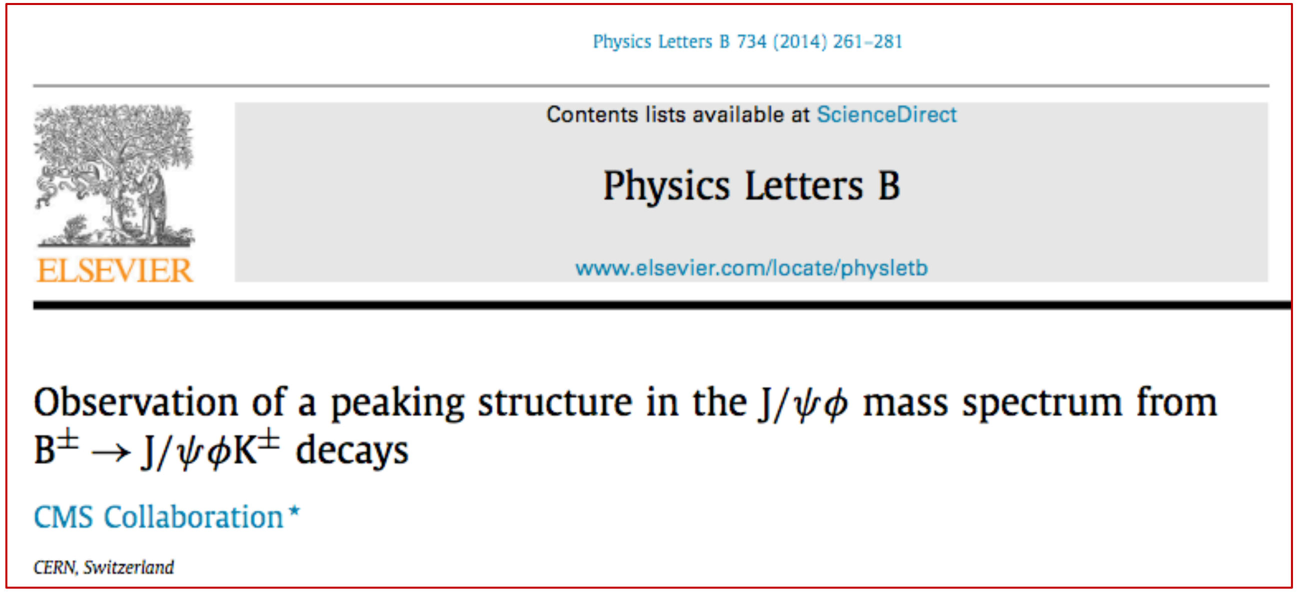
Bibliography

Inspired by part of the theory visualization & exercises by **Wouter Verkerke** :
https://indico.cern.ch/event/72320/contributions/2082589/attachments/1037201/1478048/roofit-intro-roostats-v11a.pdf
https://indico.cern.ch/event/305391/contributions/701304/attachments/580262/798889/Verkerke_Statistics_L2.pdf

See also :
- his slides for the Ferrara School 2009: https://www.nikhef.nl/~verkerke/ferrara)
- his slides for IN2P3 School 2014: https://indico.in2p3.fr/event/9742/contributions/50419/attachments/40828/50594/sos2014_systprof_v38.pdf

A good reference book is : **Luca Lista, Statistical methods for Data Analysis in Particle Physics**, Springer, 2^{nd} /3^{rd} Edition

In this Lab exercise we use CMS data used for the paper
https://doi.org/10.1016/j.physletb.2014.05.055


Physics Letters B 734 (2014) 261–281

Contents lists available at ScienceDirect

Physics Letters B

www.elsevier.com/locate/physletb

ELSEVIER

Observation of a peaking structure in the $J/\psi\phi$ mass spectrum from $B^{\pm} \to J/\psi\phi K^{\pm}$ decays

CMS Collaboration *

CERN, Switzerland

Retrieve binned data and plot

With reference to the code in the macro **yield.C** …

- Get the histogram of the **J/ψ**(μμ)**ϕ**(KK)**K** invariant mass **(**the signal represents the 3-body decay $B^{\pm} \rightarrow J/\psi\, \phi K^{\pm}$ **)** :

```
TFile f1("DatasetAandB_KaonTrackRefit_Bwin_new_21aug13.root","READ");
TH1D *hist = (TH1D*)f1.Get("myJpsiKKKmass_all");
```

- Declare & initialize the variable to represent the invariant mass and prepare the corresponding RooPlot pointer:

```
RooRealVar y("y","m(J/#psi #phi K)[GeV]",5.15,5.45);
RooPlot* yframe = y.frame("");
```

- Import the binned data by creating the RooDataHist object from the histogram and plot it:

```
RooDataHist BmassExt(hist->GetName(),hist->GetTitle(),RooArgSet(y),RooFit::Import(*hist, kFALSE));
BmassExt.plotOn(yframe);
//myC->cd();          // decomment to plot
//yframe->Draw();  // decomment to plot
```

Build the negative log-lokelihood (nll)

Build the model: - a gaussian for the signal (2 parameters: mass and width) ;
- a Chebyshev of 2nd order (2 parameters) for the background.

Based on these two PDFs, build the full PDF to make an **extended fit**:

```
RooRealVar nsig("nsig","n. of signal cands",2500.,2000.,3800.);
RooRealVar nbkg("nbkg","n. of bkg cands",2000.,0.,200000.);
//
RooAddPdf model_extended("model_extended","gauss+cheby EXT",RooArgList(gausse,chebye),RooArgList(nsig,nbkg));
```

Create a function object that represents the negative-log-likelihood (**nll**) …
… by using the method RooAbsPdf::**createNLL(**RooAbsData&**)**; the returned object is of type **RooAbsReal***

```
RooAbsReal* nll = model_extended.createNLL(BmassExt);
```

In this way we explicitely constructed the likelihood (function of PDF/data combination)
that can be used as any RooFit funcion object.

Note: likelihood can be created by a calculation that can be parallelized (suppose for instance on 4 cores):

```
RooAbsReal* nll = model_extended.createNLL(BmassExt,NumCPU(4));
```

MINUIT session

Let us invoke **MINUIT** to perform the binned extended fit.

First we can create a **MINUIT** minimizer object:

```
RooMinuit m(*nll);
```

Calling `MIGRAD` we get the **central values** (*best estimates*) for the parameters when convergence is reached:

```
m.migrad();
```

```
MIGRAD FAILS TO FIND IMPROVEMENT
MIGRAD MINIMIZATION HAS CONVERGED.
MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=-1.15354e+06 FROM MIGRAD    STATUS=CONVERGED     532 CALLS          533 TOTAL
                     EDM=0.00035446    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   STEP         FIRST
 NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
  1  c0e        -2.25841e-01   5.55732e-03   4.09669e-06  -6.98549e+02
  2  c1e        -1.08452e-02   6.02446e-03   4.04505e-06   1.24095e+03
  3  mge         5.27943e+00   5.31398e-04   7.92138e-02   6.65397e-02
  4  nbkg        9.55524e+04   3.48579e+02   2.32333e-03   6.58743e-01
  5  nsig        2.92321e+03   1.70487e+02   9.92199e-02  -9.14745e-03
  6  wge         9.48373e-03   5.96383e-04   6.96549e-02   1.25538e-01
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  6    ERR DEF=0.5
 3.088e-05  3.173e-08  1.521e-07 -8.477e-02  8.560e-02  2.114e-07
 3.173e-08  3.629e-05 -6.934e-08 -4.304e-01  4.346e-01  1.017e-06
 1.521e-07 -6.934e-08  2.835e-07 -2.503e-03  2.542e-03  1.914e-08
-8.477e-02 -4.304e-01 -2.503e-03  1.215e+05 -2.620e+04 -6.305e-02
 8.560e-02  4.346e-01  2.542e-03 -2.620e+04  2.942e+04  6.383e-02
 2.114e-07  1.017e-06  1.914e-08 -6.305e-02  6.383e-02  3.574e-07
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2      3      4      5      6
       1  0.10984   1.000  0.001  0.051 -0.044  0.090  0.064
       2  0.42489   0.001  1.000 -0.022 -0.205  0.421  0.282
       3  0.08630   0.051 -0.022  1.000 -0.013  0.028  0.060
       4  0.44039  -0.044 -0.205 -0.013  1.000 -0.438 -0.303
       5  0.71287   0.090  0.421  0.028 -0.438  1.000  0.622
       6  0.62527   0.064  0.282  0.060 -0.303  0.622  1.000
```

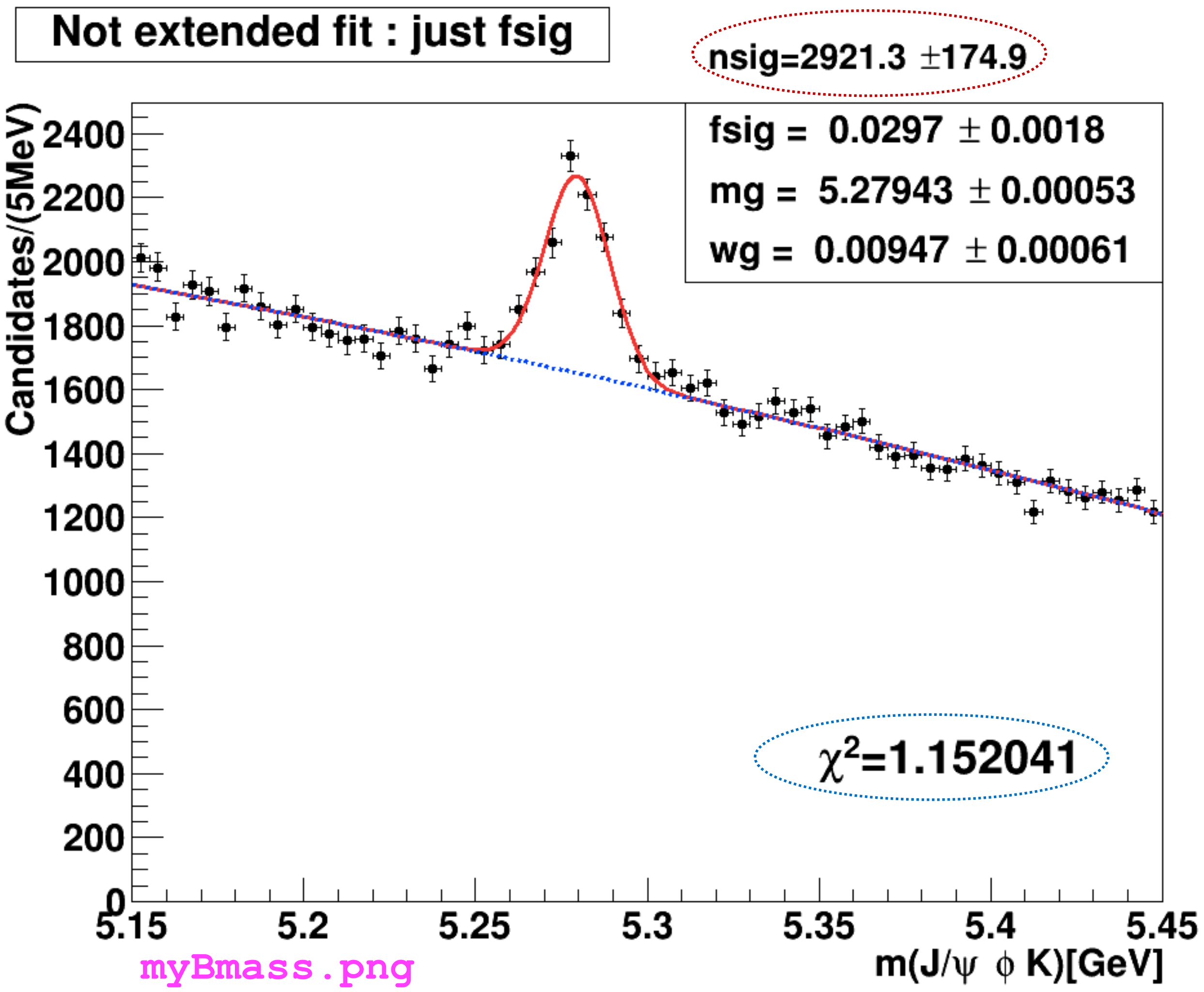
Parabolic uncertainties

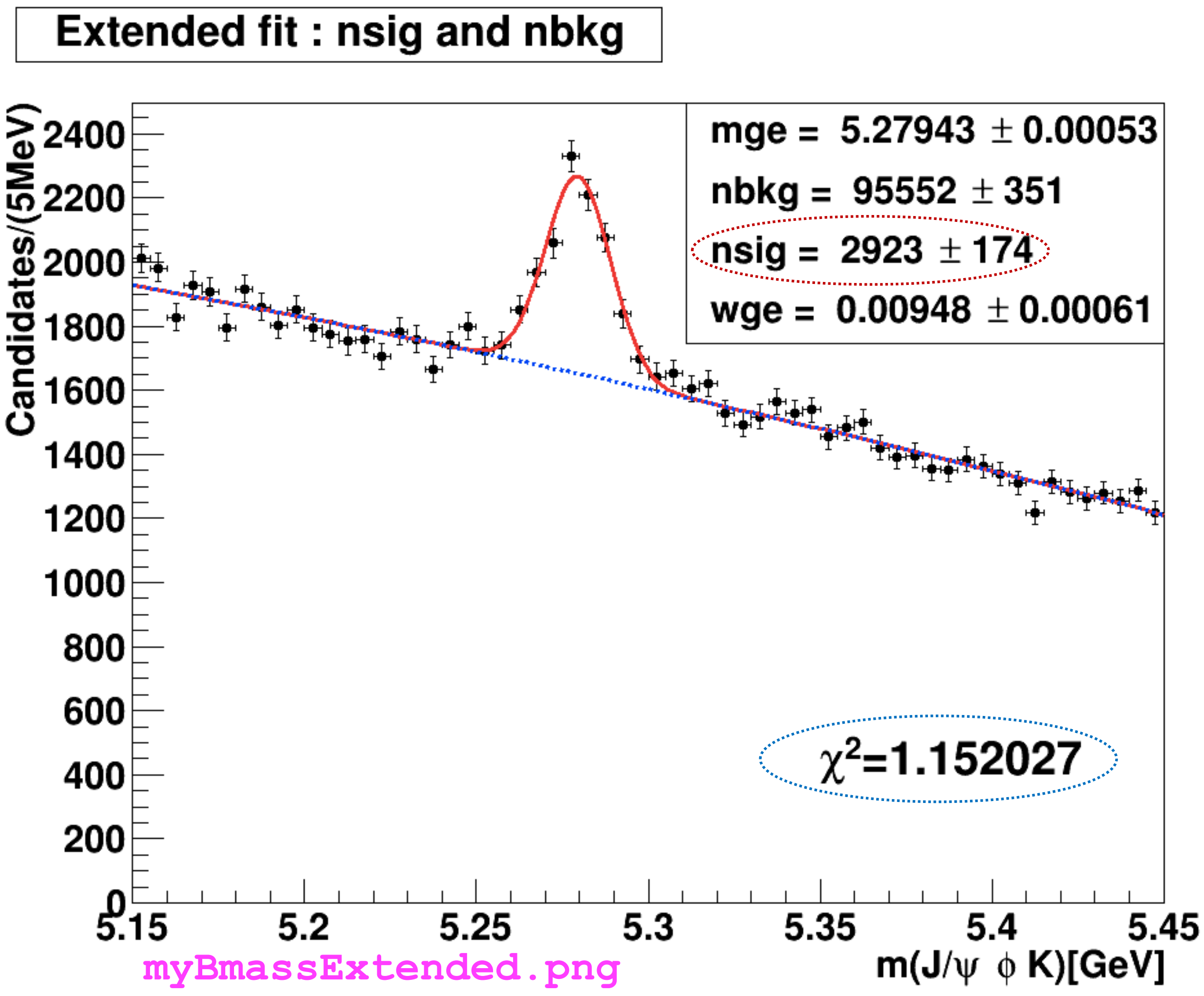
To recalculate the errors and the covariance matrix in an accurate way (still in parabolic assumption) we use `HESSE`, while central values (by `Migrad`) are conserved.

`m.hesse();` - - - - - - - - - - - - - - - - ►

```
**********
**   18 **HESSE        3000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=-1.15354e+06 FROM HESSE     STATUS=OK            40 CALLS          573 TOTAL
                    EDM=0.000362648    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   INTERNAL      INTERNAL
 NO.   NAME      VALUE            ERROR          STEP SIZE       VALUE
  1  c0e        -2.25841e-01   5.55902e-03   1.63867e-07  -2.25841e-04
  2  c1e        -1.08452e-02   6.05944e-03   1.61802e-07  -1.08452e-05
  3  mge         5.27943e+00   5.30310e-04   3.16855e-03   9.53869e+00
  4  nbkg        9.55524e+04   3.51136e+02   9.29332e-05  -4.44908e-02
  5  nsig        2.92321e+03   1.74070e+02   3.96880e-03  -6.25739e+00
  6  wge         9.48373e-03   6.08532e-04   2.78620e-03   2.50293e+01
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  6    ERR DEF=0.5
 3.090e-05  6.228e-08  1.532e-07 -8.983e-02  8.986e-02  2.249e-07
 6.228e-08  3.672e-05 -6.359e-08 -4.579e-01  4.580e-01  1.106e-06
 1.532e-07 -6.359e-08  2.823e-07 -2.776e-03  2.778e-03  1.440e-08
-8.983e-02 -4.579e-01 -2.776e-03  1.233e+05 -2.775e+04 -6.864e-02
 8.986e-02  4.580e-01  2.778e-03 -2.775e+04  3.069e+04  6.867e-02
 2.249e-07  1.106e-06  1.440e-08 -6.864e-02  6.867e-02  3.722e-07
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2      3      4      5      6
       1  0.11253   1.000  0.002  0.052 -0.046  0.092  0.066
       2  0.43584   0.002  1.000 -0.020 -0.215  0.431  0.299
       3  0.07496   0.052 -0.020  1.000 -0.015  0.030  0.044
       4  0.45348  -0.046 -0.215 -0.015  1.000 -0.451 -0.320
       5  0.72800   0.092  0.431  0.030 -0.451  1.000  0.643
       6  0.64446   0.066  0.299  0.044 -0.320  0.643  1.000
```

Extended vs not-extended fits: a comparison - I





Difference can be hardly appreciated: mass and width are about identical (see also next slide)! Also they have very similar $\tilde{\chi}^2_{fit}$

Extended fit has the advantage to provide directly as output also the number of B^+ candidates (`nsig`)

Extended vs not-extended fits: a comparison - II

NOT-extended

```
**********
**    9 **HESSE        2500
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=-119793 FROM HESSE     STATUS=OK             31 CALLS          359 TOTAL
                    EDM=1.33432e-05    STRATEGY= 1       ERROR MATRIX ACCURATE
 EXT PARAMETER                                INTERNAL      INTERNAL
 NO.   NAME        VALUE            ERROR       STEP SIZE       VALUE
  1  c0          -2.25828e-01   5.55891e-03   2.64024e-07  -2.25828e-04
  2  c1          -1.09184e-02   6.05941e-03   5.21376e-08  -1.09184e-05
  3  fsig         2.96640e-02   1.77569e-03   7.21237e-05  -1.91699e+00
  4  mg           5.27943e+00   5.30028e-04   1.01863e-03  -2.85771e+02
  5  wg           9.47402e-03   6.07750e-04   8.94182e-04   9.41424e+01
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  5    ERR DEF=0.5
 3.090e-05  6.145e-08 -9.119e-07  1.530e-07  2.245e-07
 6.145e-08  3.672e-05 -4.649e-06 -6.319e-08  1.105e-06
-9.119e-07 -4.649e-06  3.153e-06 -2.829e-08 -6.963e-07
 1.530e-07 -6.319e-08 -2.829e-08  2.820e-07  1.421e-08
 2.245e-07  1.105e-06 -6.963e-07  1.421e-08  3.712e-07
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2      3      4      5
       1  0.11250   1.000  0.002 -0.092  0.052  0.066
       2  0.43581   0.002  1.000 -0.432 -0.020  0.299
       3  0.69297  -0.092 -0.432  1.000 -0.030 -0.644
       4  0.07459   0.052 -0.020 -0.030  1.000  0.044
       5  0.64456   0.066  0.299 -0.644  0.044  1.000
**********
```

$m = (5279,43 \pm 0.53) MeV$ $\sigma = (9,4740 \pm 0,6078) MeV$

Extended

```
**   18 **HESSE        3000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=-1.15354e+06 FROM HESSE     STATUS=OK             40 CALLS          573 TOTAL
                    EDM=0.000362648    STRATEGY= 1       ERROR MATRIX ACCURATE
 EXT PARAMETER                                INTERNAL      INTERNAL
 NO.   NAME        VALUE            ERROR       STEP SIZE       VALUE
  1  c0e         -2.25841e-01   5.55902e-03   1.63867e-07  -2.25841e-04
  2  c1e         -1.08452e-02   6.05944e-03   1.61802e-07  -1.08452e-05
  3  mge          5.27943e+00   5.30310e-04   3.16855e-03   9.53869e+00
  4  nbkg         9.55524e+04   3.51136e+02   9.29332e-05  -4.44908e-02
  5  nsig         2.92321e+03   1.74070e+02   3.96880e-03  -6.25739e+00
  6  wge          9.48373e-03   6.08532e-04   2.78620e-03   2.50293e+01
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  6    ERR DEF=0.5
 3.090e-05  6.228e-08  1.532e-07 -8.983e-02  8.986e-02  2.249e-07
 6.228e-08  3.672e-05 -6.359e-08 -4.579e-01  4.580e-01  1.106e-06
 1.532e-07 -6.359e-08  2.823e-07 -2.776e-03  2.778e-03  1.440e-08
-8.983e-02 -4.579e-01 -2.776e-03  1.233e+05 -2.775e+04 -6.864e-02
 8.986e-02  4.580e-01  2.778e-03 -2.775e+04  3.069e+04  6.867e-02
 2.249e-07  1.106e-06  1.440e-08 -6.864e-02  6.867e-02  3.722e-07
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2      3      4      5      6
       1  0.11253   1.000  0.002  0.052 -0.046  0.092  0.066
       2  0.43584   0.002  1.000 -0.020 -0.215  0.431  0.299
       3  0.07496   0.052 -0.020  1.000 -0.015  0.030  0.044
       4  0.45348  -0.046 -0.215 -0.015  1.000 -0.451 -0.320
       5  0.72800   0.092  0.431  0.030 -0.451  1.000  0.643
       6  0.64446   0.066  0.299  0.044 -0.320  0.643  1.000
```

$m = (5279,43 \pm 0.53) MeV$ $\sigma = (9,4837 \pm 0,6085) MeV$

Asymmetric uncertainties

To get asymmetric error (central values and parabolic are the same) for a specific parameter, like `nsig` :

```
m.minos(nsig);
```

```
**********
**   23 **MINOS        3000          5
**********
FCN=-1.15354e+06 FROM MINOS     STATUS=SUCCESSFUL     132 CALLS          705 TOTAL
                     EDM=0.000362648    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                   PARABOLIC         MINOS ERRORS
 NO.   NAME      VALUE            ERROR      NEGATIVE      POSITIVE
  1  c0e        -2.25841e-01   5.55902e-03
  2  c1e        -1.08452e-02   6.05944e-03
  3  mge         5.27943e+00   5.30310e-04
  4  nbkg        9.55524e+04   3.51136e+02
  5  nsig        2.92321e+03   1.74070e+02  -1.74275e+02   1.76453e+02
  6  wge         9.48373e-03   6.08532e-04
                               ERR DEF= 0.5
RooRealVar::nsig = 2923.21 +/- (-174.275,176.453)  L(2000 - 3800)
```

To additionally print the result just do: `nsig.Print();`

To get asymmetric error **for all the parameters** :

```
m.minos();
```

```
**********
**   23 **MINOS        3000
**********
FCN=-1.15354e+06 FROM MINOS     STATUS=SUCCESSFUL     702 CALLS         1275 TOTAL
                     EDM=0.000362648    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                   PARABOLIC         MINOS ERRORS
 NO.   NAME      VALUE            ERROR      NEGATIVE      POSITIVE
  1  c0e        -2.25841e-01   5.55902e-03  -5.54067e-03   5.57848e-03
  2  c1e        -1.08452e-02   6.05944e-03  -6.12080e-03   6.00134e-03
  3  mge         5.27943e+00   5.30310e-04  -5.28542e-04   5.34442e-04
  4  nbkg        9.55524e+04   3.51136e+02  -3.50289e+02   3.52220e+02
  5  nsig        2.92321e+03   1.74070e+02  -1.74275e+02   1.76448e+02
  6  wge         9.48373e-03   6.08532e-04  -5.99390e-04   6.22600e-04
                               ERR DEF= 0.5
RooRealVar::nsig = 2923.21 +/- (-174.275,176.448)  L(2000 - 3800)
```

Note: asymmetric errors can slightly change if you execute `MINOS` for 1 or all parameters

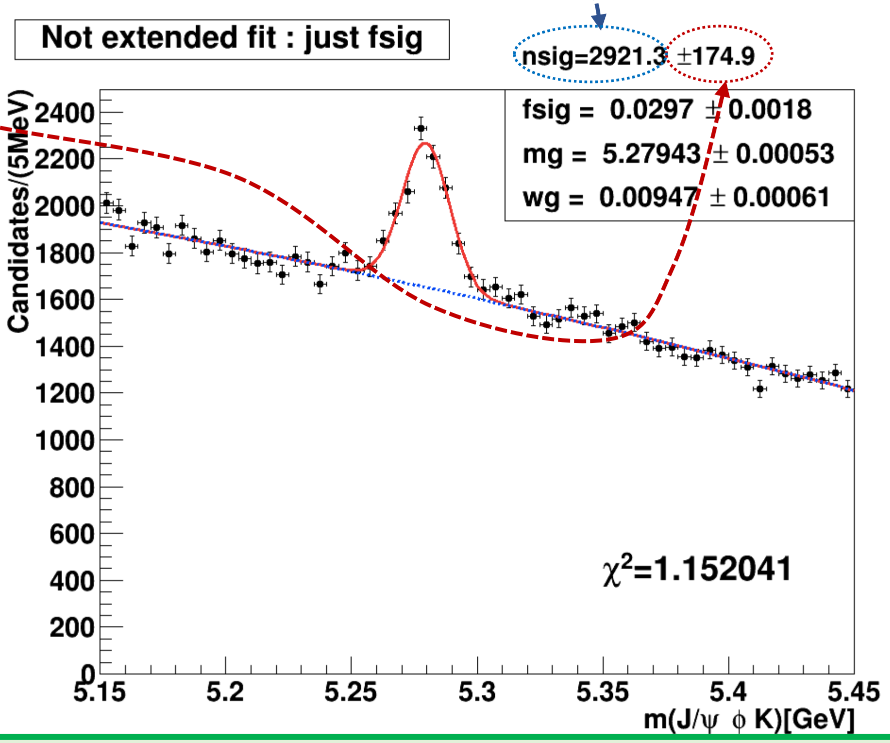
(in this case only … upper uncertainty changes)

How-to-calculate the # of signal candidates in a not extended fit

After executing the *not extended* fit …

```
// double cands = fsig.getVal()*myEntries; // in case you want to use it later as a variable
//
cout << "\n # of entries = " << myEntries << " of which # signal candidates is = " << fsig.getVal()*myEntries << " +/- " << fsig.getError()*myEntries << endl;

// -- let me write the # of candidates estimated by the fit (via fsig):
TLatex* myLatCands = new TLatex(5.318,2600.,Form("nsig=%.1f",fsig.getVal()*myEntries));
TLatex* myLatCands1 = new TLatex(5.38,2600.,Form("#pm%.1f",fsig.getError()*myEntries)); // it rounds as expected
myLatCands->SetTextSize(0.038);
myLatCands1->SetTextSize(0.038);
xframeChi2->addObject(myLatCands);
xframeChi2->addObject(myLatCands1);
```



How-to-calculate the normalized chi-squared $\widetilde{\chi}^2_{fit}$ - I

In the code of the macro, I propose two different ways to extract the $\widetilde{\chi}^2_{fit}$ of the binned ML fit.
We apply both for the *not extended* fit. We will prefer the 2nd approach and choose it for the *extended* fit.

```
// Note: will try two approaches
//
// -- 1st approach
//
// --https://root.cern.ch/doc/master/classRooPlot.html
//   Syntax: chiSquare (const char *pdfname, const char *histname, int nFitParam=0) const
//   Description : Calculate and return reduced chi-squared between a curve and a histogram.
//   Syntax: chiSquare (int nFitParam=0) const
//   Description: Shortcut for RooPlot::chiSquare(const char* pdfname, const char* histname, int nFitParam=nullptr)
RooPlot* xframeChi2 = x.frame("");
Bmass.plotOn(xframeChi2);                        // histogram (type RooDataHist)
model.plotOn(xframeChi2,RooFit::LineColor(kRed)); // curve
//
RooArgSet* floatPars = model.getParameters(Bmass);
int numFreeParams = floatPars->getSize();
cout << "\n # of free fit params is = " << numFreeParams << endl;      (A)
//
// normalized chiSquared is given by chi2 divided by ndof = (# bins of the fit range - #free params)
// though, the following is given already normalized!
Double_t chi2Norm = xframeChi2->chiSquare(numFreeParams);
cout << "\n the Chi2 for the not-extended fit is = " << chi2Norm << endl;      (B)
//
```

How-to-calculate the normalized chi-squared $\tilde{\chi}^2_{fit}$ - II

```
// -- 2nd approach
//
RooAbsReal* chi2 = model.createChi2(Bmass,Range(begin,end));
// if extended  ... add Extended(true): createChi2(Bmass,Range(begin,end),Extended(true))
// because in this way the prediction of the total number of events is taken from the extended pdf and not from the histogram
cout << "\n chi2 (not normalized) is = " << chi2->getVal() << endl;            (C)
// maybe this is the best way but be careful with the parameters you pass to the createChi2 function
// since it does not divide by ndf, i.e. the number of non-zero bins minus the number of parameters;
// you have to do this manually afterwards!
//
int nDOF = nBins - numFreeParams;
cout << "\n nDOF is = " << nDOF << endl;                  (D)
double chi2NormCalc = (chi2->getVal()) / nDOF;
cout << "\n chi2 (normalized by manual calculation) is = " << chi2NormCalc << endl;     (E)
//
// Still, this method doesn't get it right if your pdf is continuous,
// because it just takes the pdf value in the bin center.
// If this is a problem for you, you can consider wrapping the pdf in a RooBinSamplingPdf - to be explored -
// which turns the continuous pdf into a binned pdf where the values for each bin are obtained by integration.
// However, this can be important only in case of steep functions
// where value at the center of the bin and integal over the bin may differ; this is not the case here!
```

this is done for the extended fit !

```
# of free extended-fit params is = 6

nDOF is = 54

chi2 (normalized by manual calculation) is = 1.15203
```

We get on screen@ execution time:

```
# of free fit params is = 5                                     (A)

the Chi2 for the not-extended fit is = 1.158                    (B)
---------------------------------------------------------------
chi2 (not normalized) is = 63.3622                              (C)

nDOF is = 55                                                    (D)

chi2 (normalized by manual calculation) is = 1.15204            (E)
```

(A), (B): 1st approach

(C), (D), (E): 2nd approach

To print on the plot:

```
TLatex* myLatChi2 = new TLatex(5.35,400.,Form("#chi^{2}=%f",chi2NormCalc));
```

Correlation Matrix

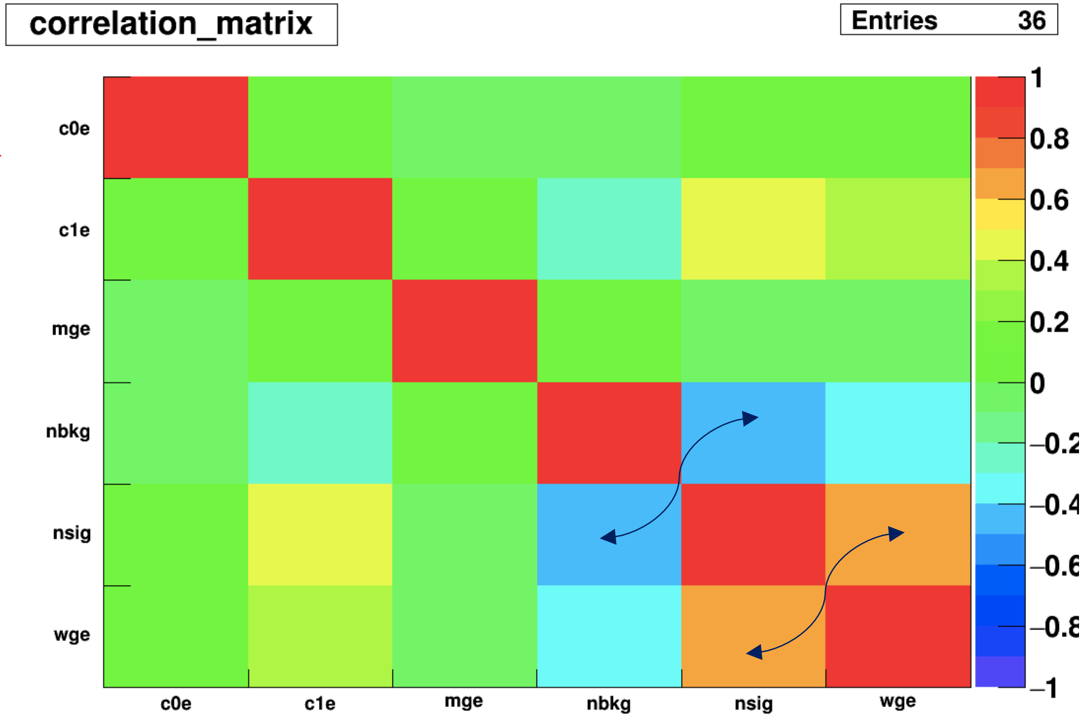
It is possible to save the status of the fit, including the information about the convariance matrix:

```
RooFitResult* fitres = m.save();
```

It is possible to visualize the correlation matrix:

```
gStyle->SetPalette(1); //- for better color choice
fitres->correlationHist()->Draw("colz");
```

myCorrelationMatrix.png



Note:
- anticorrelation between `nsig` and `nbkg` as expected
- correlation between `nsig` and width of Gaussian `wge`

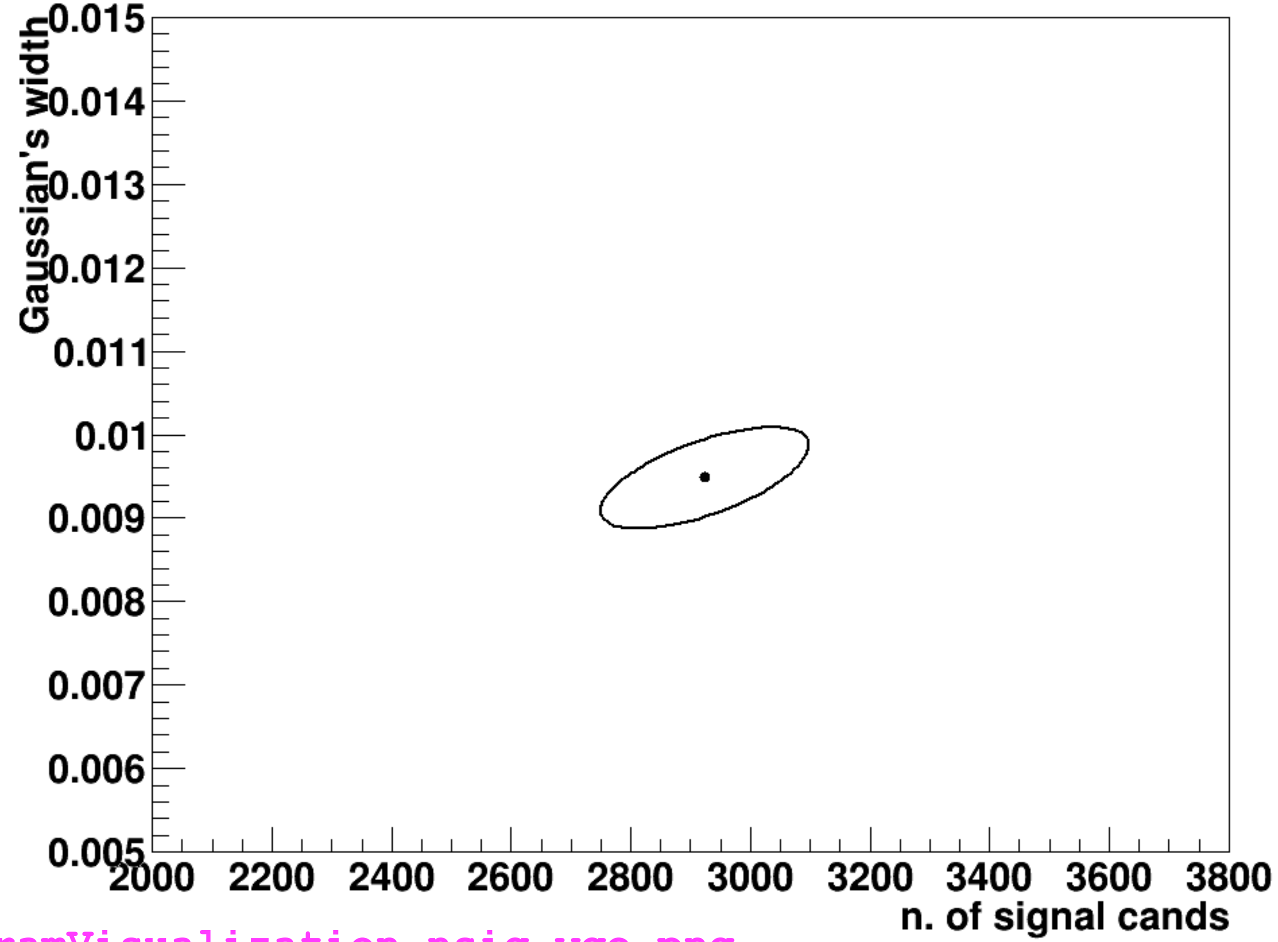
Note: in general, if correlations are very strong (i.e. > 0.9) the model may become unstable and it may be worthwhile to fix one of the parameters in the fit.
If the strong correlation is between two nuisance parameters, this is not a problem. Instead, when a parameter-of-interest is correlated with a nuisance one, it must be avoided to fix the nuisance parameter because the risk is to strongly under-estimate the uncertainty on the physical parameter!

Visualization of correlated errors - I

It is also possible to visualize errors & correlation matrix elements:

```
RooPlot* paramFrame = new RooPlot(nsig,wge);
fitres->plotOn(paramFrame,nsig,wge);
paramFrame->SetTitleOffset(1.38,"Y");
paramFrame->Draw();
myC->SaveAs("./pdfParamVisualization_nsig_wge.png");
```

where this example shows a certain level of correlation.

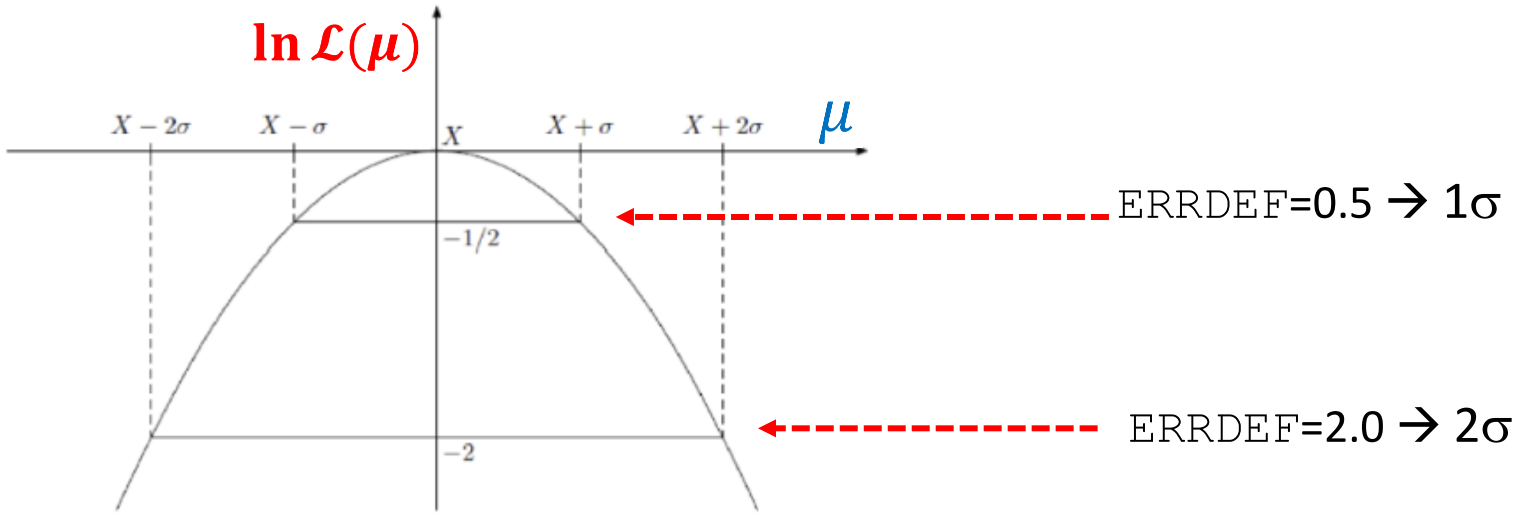


pdfParamVisualization_nsig_wge.png

Visualization of correlated errors - II

But why ERRDEF=0.5 and 2.0 are considered? This is a reminder.

Well, do not forget that a PDF can be converted into a Likelihood function $\mathcal{L}$ by "exchanging" the vector of observations $\vec{x}$ with the vector of parameters $\vec{\theta}$!
For only one parameter, say μ , the likelihood is a function of it, namely $\mathcal{L}(\mu)$, and **$\ln \mathcal{L}(\mu)$ is a parabola!**

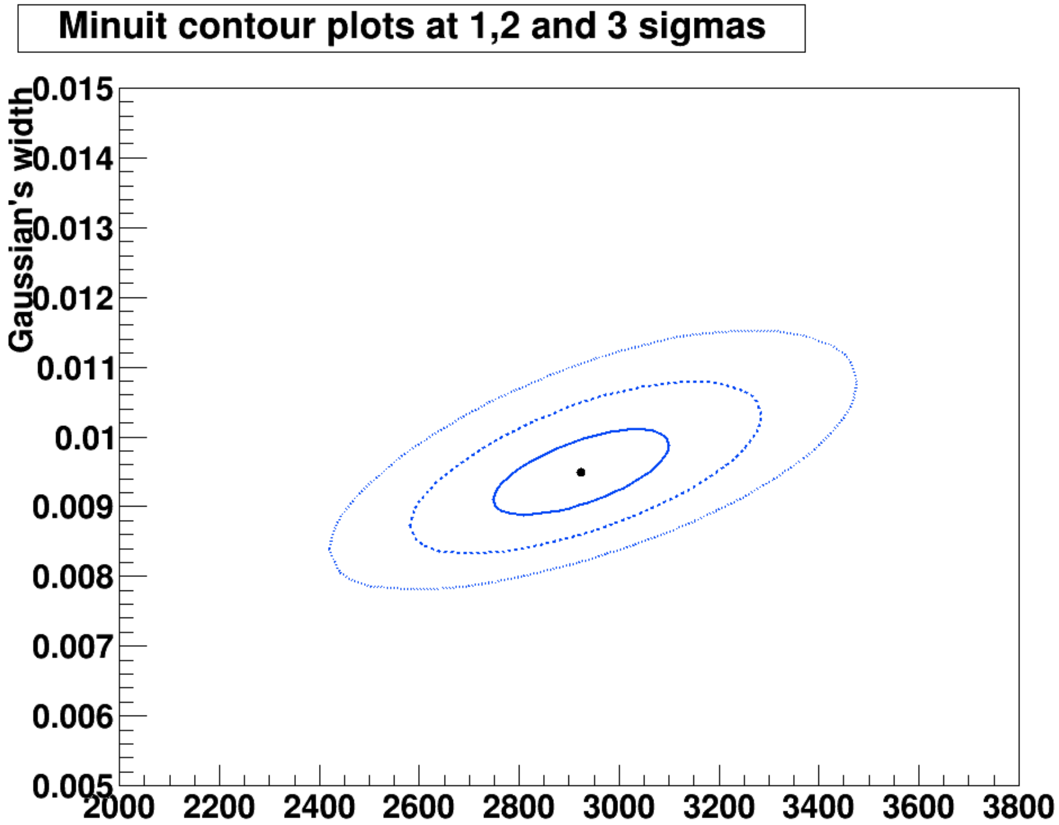


Note : if you put the "-" in front of it, thus getting the neg-log-likelihood, $-\ln \mathcal{L}(\mu)$, the parabola changes sign and "points" upwards instead of downwards.

Extension: Now suppose we've 2 parameters of interest; in this case you can imagine a paraboloid instead of a parabola with different aperture when projecting in 1-dim. The "multivariate" uncertainty is then represented by an elliptic contour.

Visualization of correlated errors - III

```
RooPlot* contourFrame = m.contour(nsig,wge,1,2,3,0,0,0); // gives the 3 contours obtained for 1, 2 and 3 sigmas
contourFrame->SetTitle("Minuit contour plots at 1,2 and 3 sigmas");
contourFrame->SetTitleOffset(1.38,"Y");
contourFrame->Draw();
myC->SaveAs("./paramContours_nsig_wge.png");
```



paramContours_nsig_wge.png

It starts the MNCONTOUR calculation of 50 point on three contours (for ERRDEF = 0.5, 2.0 and 4.5). Each point is identified by a pair of values of parameters 5 (nsig) and 6 (wge) on the scatter plot.

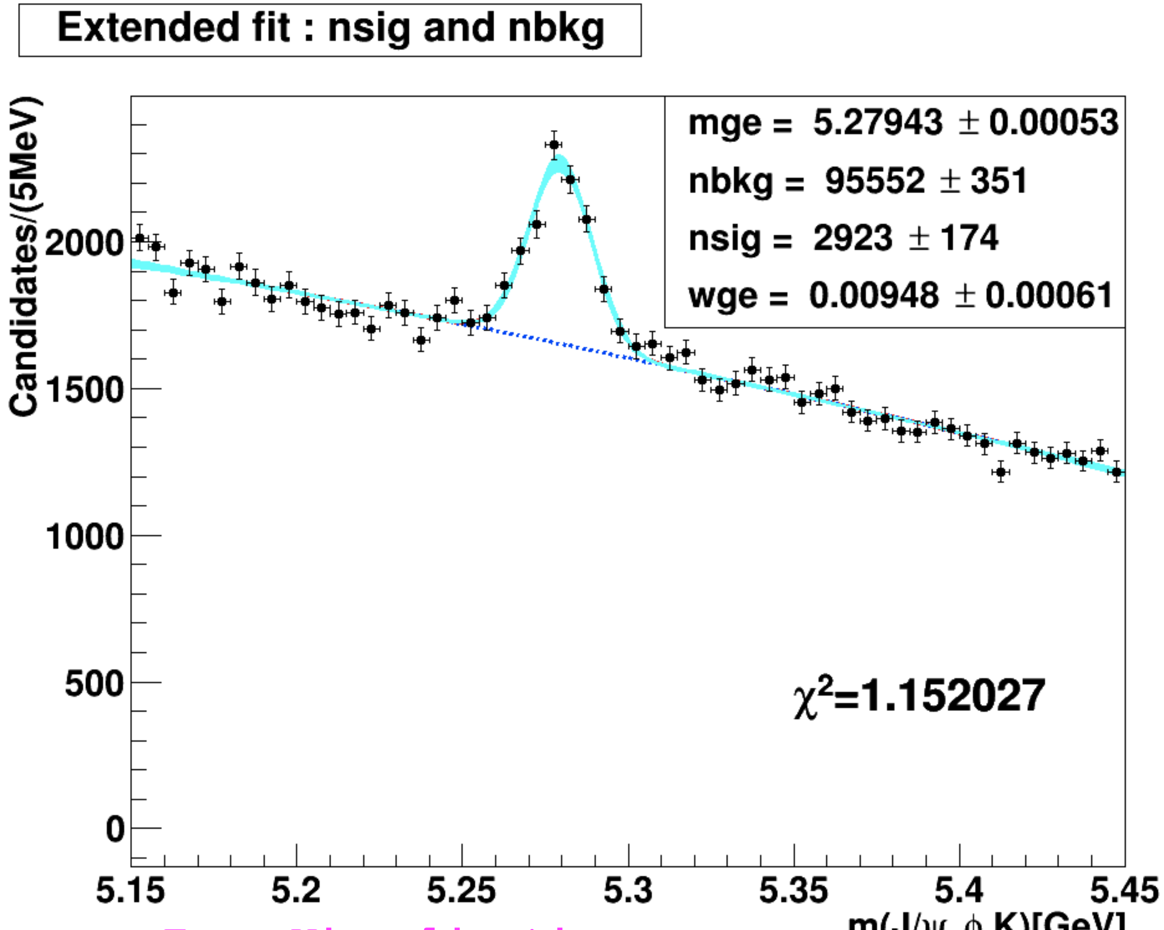
As you can check … the three sets of 50 pairs of values are printed on the screen @ execution time.

Note: the contour with ERRDEF=0.5 is the same one obtained earlier with a different command.

Visualization of the fit uncertainty

It is possible to propagate the errors (stored in the covariance matrix of a fit result) to a PDF projection:

```
model_extended.plotOn(yframe, VisualizeError(*fitres));
yframe->Draw();
```



myErrorVisualization.png

To get the points' errors over the cyan shadowed region describing the uncertainty we need to add the following two lines (to get the "trick" done):

```
BmassExt.plotOn(yframe);
yframe->Draw("Esame");
```

Visualization of the fit log-likelihood function and of the Profile Likelihood ratio

We can obtain the best estimate for **nsig** & the **MINOS** uncertainty **corresponding** to the interval provided by the PL ratio :

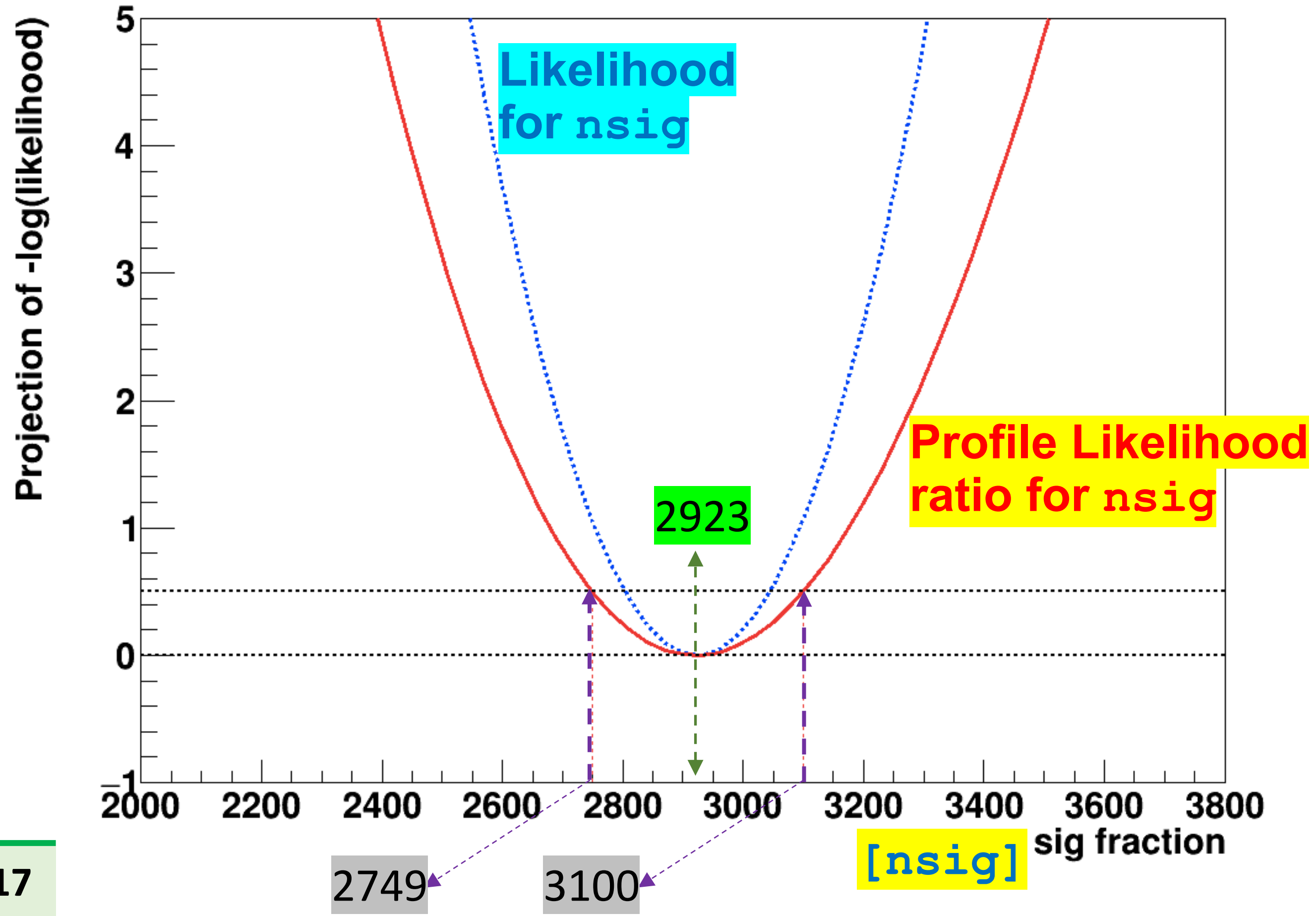
```
// plot the likelihood as a function of the parameter of interest (here nsig):
RooPlot* nsig_frame = nsig.frame(RooFit::Bins(60),RooFit::Range(2000,3800));
nll->plotOn(nsig_frame,RooFit::ShiftToZero(),RooFit::LineStyle(kDashed),LineColor(kBlue));
//
// make the Profile Likelihood ratio (that can be represented as a regular RooFit function)
RooAbsReal* pll_nsig = nll->createProfile(nsig);
pll_nsig->plotOn(nsig_frame,RooFit::ShiftToZero(),LineColor(kRed));
nsig_frame->SetMinimum(-1);
nsig_frame->SetMaximum(5);
nsig_frame->Draw();
```

From MIGRAD: 2923.2

From MINOS: $(2923.2) - 174.3 + 176.4$
(slightly asymmetric)

Overall interval: $\cong [2749, 3100]$

myLikelihood.png



Code of the macro yield.C - I

```
///////////////////////
// To run it:
// root> .L yield.C
// root> main()
//
//////////////////////

#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"
#include <TPaveLabel.h>

using namespace RooFit;

////////////////////////////////---inizio main

int main() {

  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(10);
  gStyle->SetTitleOffset(1.2,"");
  //
  TCanvas* myC = new TCanvas("myC","Plots",1000, 800);
  //
  ///////////////////////////////////////////////////
  //
  TFile f1("DatasetAandB_KaonTrackRefit_Bwin_new_21aug13.root","READ");
  TH1D *hist = (TH1D*)f1.Get("myJpsiKKKmass_all");
  // in alternativa c'e' anche l'istogramma myJpsiKKKmass_tight
  //
  //-- # of bins:
  int nBins = hist->GetNbinsX();
  int entries = hist->GetEntries(); // this containes also overflow so I recalculate by hand as follows:
  int myEntries;
  for (int l=0;l<60;l++)
     {
      myEntries += hist->GetBinContent(l+1);
     }
  //
  ///////////////////////////////////////////////////
```

Code of the macro yield.C - II

```
/////////////////////////////////////////////////////
//
RooRealVar x("x","m(J/#psi #phi K)[GeV]",5.15,5.45);
RooDataHist Bmass(hist->GetName(),hist->GetTitle(),RooArgSet(x),RooFit::Import(*hist, kFALSE));
//
Float_t begin = 5.15;
Float_t end = 5.45;
//
RooPlot* xframe = x.frame("");
Bmass.plotOn(xframe);
////
myC->cd();
//
// -- SIGNAL
RooRealVar mg("mg","Gaussian's mean",5.28,5.275,5.285);
RooRealVar wg("wg","Gaussian's width",0.010,0.005,0.015);
RooGaussian gauss1("gauss1","Gauss(x,mg,wg)",x,mg,wg);
// -- BKG
RooRealVar c0("c0","1st coeff",0.5,-1000.,1000.);
RooRealVar c1("c1","2nd coeff",-0.5,-1000.,1000.);
//--RooRealVar c2("c2","3rd coeff",0.1,-1000.,1000.); // 2nd order degree is enough here
RooChebychev cheby("cheby","Chebyshev",x,RooArgList(c0,c1)); // 2 coeff. means 2nd order polynominal
//
// -- TOTAL pdf : f*gauss1 + (1-f)*cheby
RooRealVar fsig("fsig","narrow fraction",0.05,0.0,1.0);
RooAddPdf model("model","gauss1+cheby",RooArgList(gauss1,cheby),fsig); // configured in this way this is not extended
//
// -- Execute FIT
// model.fitTo(Bmass,RooFit::Minos(kTRUE)); // let us give explicitely also the FitRange:
model.fitTo(Bmass,RooFit::Minos(kTRUE),Range(begin,end));
model.plotOn(xframe,RooFit::LineColor(kRed));
model.plotOn(xframe,RooFit::Components(cheby),RooFit::LineStyle(kDashed));
model.paramOn(xframe, Parameters(RooArgSet(mg,wg,fsig)), Layout(0.53,0.9,0.9)); // 3rd is up
//
// double cands = fsig.getVal()*myEntries; // in case you want to use it later as a variable
//
cout << "\n # of entries = " << myEntries << " of which # signal candidates is = " << fsig.getVal()*myEntries << " +/- " << fsig.getError()*myEntries << endl;
//
xframe->SetTitle("Not extended fit : just fsig and (1-fsig)");
xframe->Draw();
//
```

Code of the macro yield.C - III

```
//
// -- calculate a chiSquared (from a curve and a histogram in a RooPlot)
//======================================================================
//
// Note: will try two approaches
//
// -- 1st approach
//
// --https://root.cern.ch/doc/master/classRooPlot.html
//   Syntax: chiSquare (const char *pdfname, const char *histname, int nFitParam=0) const
//   Description : Calculate and return reduced chi-squared between a curve and a histogram.
//   Syntax: chiSquare (int nFitParam=0) const
//   Description: Shortcut for RooPlot::chiSquare(const char* pdfname, const char* histname, int nFitParam=nullptr)
RooPlot* xframeChi2 = x.frame("");
Bmass.plotOn(xframeChi2);                        // histogram (type RooDataHist)
model.plotOn(xframeChi2,RooFit::LineColor(kRed)); // curve
//
RooArgSet* floatPars = model.getParameters(Bmass);
int numFreeParams = floatPars->getSize();
cout << "\n # of free fit params is = " << numFreeParams << endl;
//
// normalized chiSquared is given by chi2 divided by ndof = (# bins of the fit range - #free params)
// though, the following is given already normalized!
Double_t chi2Norm = xframeChi2->chiSquare(numFreeParams);
cout << "\n the Chi2 for the not-extended fit is = " << chi2Norm << endl;
//
```

Code of the macro yield.C - IV

```
//
// -- 2nd approach
//
RooAbsReal* chi2 = model.createChi2(Bmass,Range(begin,end));
// if extended  ... add Extended(true): createChi2(Bmass,Range(begin,end),Extended(true))
// because in this way the prediction of the total number of events is taken from the extended pdf and not from the histogram
cout << "\n chi2 (not normalized) is = " << chi2->getVal() << endl;
// maybe this is the best way but be careful with the parameters you pass to the createChi2 function
// since it does not divide by ndf, i.e. the number of non-zero bins minus the number of parameters;
// you have to do this manually afterwards!
//
int nDOF = nBins - numFreeParams;
cout << "\n nDOF is = " << nDOF << endl;
double chi2NormCalc = (chi2->getVal()) / nDOF;
cout << "\n chi2 (normalized by manual calculation) is = " << chi2NormCalc << endl;
//
// Still, this method doesn't get it right if your pdf is continuous,
// because it just takes the pdf value in the bin center.
// If this is a problem for you, you can consider wrapping the pdf in a RooBinSamplingPdf - to be explored -
// which turns the continuous pdf into a binned pdf where the values for each bin are obtained by integration.
// However, this can be important only in case of steep functions
// where value at the center of the bin and integral over the bin may differ; this is not the case here!
//
model.plotOn(xframeChi2,RooFit::Components(cheby),RooFit::LineStyle(kDashed));
model.paramOn(xframeChi2, Parameters(RooArgSet(mg,wg,fsig)), Layout(0.53,0.9,0.9));
//
TLatex* myLatChi2 = new TLatex(5.35,400.,Form("#chi^{2}=%f",chi2NormCalc));
xframeChi2->addObject(myLatChi2);
//
// -- let me write the # of candidates estimated by the fit (via fsig):
TLatex* myLatCands = new TLatex(5.318,2600.,Form("nsig=%.1f",fsig.getVal()*myEntries));
TLatex* myLatCands1 = new TLatex(5.38,2600.,Form("#pm%.1f",fsig.getError()*myEntries)); // it rounds as expected
myLatCands->SetTextSize(0.038);
myLatCands1->SetTextSize(0.038);
xframeChi2->addObject(myLatCands);
xframeChi2->addObject(myLatCands1);
//
xframeChi2->SetTitle("Not extended fit : just fsig");
xframeChi2->SetYTitle("Candidates/(5MeV)");
xframeChi2->SetTitleOffset(1.32,"Y");
xframeChi2->Draw();
//
myC->SaveAs("./myBmass.png");
//gSystem->Sleep(20000);
//
myC->Update();
myC->cd();
//
```

Code of the macro yield.C - V

```
/////////////////////////////////////////// EXTENDED FIT :
//
myC->Divide(1,1);
//
RooRealVar y("y","m(J/#psi #phi K)[GeV]",5.15,5.45);
RooPlot* yframe = y.frame("");
//
RooDataHist BmassExt(hist->GetName(),hist->GetTitle(),RooArgSet(y),RooFit::Import(*hist, kFALSE));
BmassExt.plotOn(yframe);
//myC->cd();        // decomment to plot
//yframe->Draw();   // decomment to plot
//
RooRealVar mge("mge","Gaussian's mean",5.28,5.275,5.285);
RooRealVar wge("wge","Gaussian's width",0.01,0.005,0.015);
RooGaussian gausse("gausse","Gauss(y,mge,wge)",y,mge,wge);
//
RooRealVar c0e("c0e","1st coeff",0.5,-1000,1000);
RooRealVar c1e("c1e","2nd coeff",-0.5,-1000,1000);
//--RooRealVar c1e("c2e","3rd coeff",-0.5,-1000,1000)
//
RooChebychev chebye("chebye","Chebyshev",y,RooArgList(c0e,c1e));
//
RooRealVar nsig("nsig","n. of signal cands",2500.,2000.,3800.);
RooRealVar nbkg("nbkg","n. of bkg cands",2000.,0.,200000.);
//
RooAddPdf model_extended("model_extended","gauss+cheby EXT",RooArgList(gausse,chebye),RooArgList(nsig,nbkg));
//
RooAbsReal* nll = model_extended.createNLL(BmassExt);
RooMinuit m(*nll);
m.migrad();
m.hesse();
// m.minos(nsig);  // get asymmetric just for parameter "nsig"
m.minos();  // get asymmetric for all parameters
//
nsig.Print();
//
RooArgSet* floatParsExt = model_extended.getParameters(BmassExt);
int numFreeParamsExt = floatParsExt->getSize();
cout << "\n # of free extended-fit params is = " << numFreeParamsExt << endl;
//
int nDOFExt = nBins - numFreeParamsExt;
cout << "\n nDOF is = " << nDOFExt << endl;
RooAbsReal* chi2Ext = model_extended.createChi2(BmassExt,Range(begin,end));
double chi2NormExtCalc = (chi2Ext->getVal()) / nDOF;
cout << "\n chi2 (normalized by manual calculation) is = " << chi2NormExtCalc << endl;
//
//
RooFitResult* fitres = m.save();
gStyle->SetPalette(1); //- for better color choice
fitres->correlationHist()->Draw("colz");
myC->SaveAs("./myCorrelationMatrix.png");
myC->Update();
myC->cd();
//
model_extended.plotOn(yframe,RooFit::LineColor(kRed));
model_extended.plotOn(yframe,RooFit::Components(chebye),RooFit::LineStyle(kDashed));
model_extended.paramOn(yframe, Parameters(RooArgSet(mge,wge,nsig,nbkg)), Layout(0.53,0.9,0.9));
yframe->SetTitle("Extended fit : nsig and nbkg");
//
TLatex* myLatExt = new TLatex(5.35,400.,Form("#chi^{2}=%f",chi2NormExtCalc));
yframe->addObject(myLatExt);
yframe->SetYTitle("Candidates/(5MeV)");
yframe->SetTitleOffset(1.32,"Y");
yframe->Draw();
//
myC->SaveAs("./myBmassExtended.png");
myC->Update();
```

Code of the macro yield.C - VI

```
//
// -- fit is done but now we want to derive more info from the RooFitResult object
//
model_extended.plotOn(yframe, VisualizeError(*fitres));
yframe->Draw();
BmassExt.plotOn(yframe);
yframe->Draw("Esame");
myC->SaveAs("./myErrorVisualization.png");
myC->Update();
myC->cd();
//
//--
////RooAbsPdf* paramPDF = fitres->createHessePdf(RooArgSet(nsig,wge)); //not working
//
RooPlot* paramFrame = new RooPlot(nsig,wge);
fitres->plotOn(paramFrame,nsig,wge);
paramFrame->SetTitleOffset(1.38,"Y");
paramFrame->Draw();
myC->SaveAs("./pdfParamVisualization_nsig_wge.png"); // it gives just a visualization with the 1sigma ellipse
myC->Update();
myC->cd();
//
// -- the following is more useful than the previous
//
RooPlot* contourFrame = m.contour(nsig,wge,1,2,3,0,0,0); // gives the 3 contours obtained for 1, 2 and 3 sigmas
contourFrame->SetTitle("Minuit contour plots at 1,2 and 3 sigmas");
contourFrame->SetTitleOffset(1.38,"Y");
contourFrame->Draw();
myC->SaveAs("./paramContours_nsig_wge.png");
myC->Update();
myC->cd();
//
```

Code of the macro yield.C - VII

```
  //
  //////////////// now plot Likelihood and Profile Likelihood Ratio functions :
  //
  myC->Divide(1,1);
  //
  // plot the likelihood as a function of the parameter of interest (here nsig):
  RooPlot* nsig_frame = nsig.frame(RooFit::Bins(60),RooFit::Range(2000,3800));
  nll->plotOn(nsig_frame,RooFit::ShiftToZero(),RooFit::LineStyle(kDashed),LineColor(kBlue));
  //
  // make the Profile Likelihood ratio (that can be represented as a regular RooFit function)
  RooAbsReal* pll_nsig = nll->createProfile(nsig);
  pll_nsig->plotOn(nsig_frame,RooFit::ShiftToZero(),LineColor(kRed));
  nsig_frame->SetMinimum(-1);
  nsig_frame->SetMaximum(5);
  nsig_frame->Draw();
  //
  TLine *line0 = new TLine(2000,0,3800,0);
  line0->SetLineColor(1);
  line0->SetLineWidth(2);
  line0->SetLineStyle(2);
  line0->Draw("same");
  //
  TLine *line05 = new TLine(2000,0.5,3800,0.5);
  line05->SetLineColor(1);
  line05->SetLineWidth(2);
  line05->SetLineStyle(2);
  line05->Draw("same");
  //
  TLine *lineN1 = new TLine(3100,-1.,3100,0.5);
  lineN1->SetLineColor(2);
  lineN1->SetLineWidth(1);
  lineN1->SetLineStyle(2);
  lineN1->Draw("same");
  //
  TLine *lineN2 = new TLine(2749,-1.,2749,0.5);
  lineN2->SetLineColor(2);
  lineN2->SetLineWidth(1);
  lineN2->SetLineStyle(2);
  lineN2->Draw("same");
  //
  myC->SaveAs("./myLikelihood.png");
  myC->Update();
  myC->cd();
  //
  delete myC;
  //
  gROOT->Reset();
  gROOT->Clear();
  //
return 0;
}
```