Analysis Tools for MesonEx

- Application of RooFit

Derek Glazier
University of Glasgow

PWA11/ATHOS6, 2019
Centro Brasileiro de Pesquisas Físicas,
Rio de Janeiro

Goals

General

parameter estimation for all reaction cases → **Yields**
Polarisation Observables
Spin Density Matrix
Angular Moments
Amplitudes...

→ allow customised PDF classes
→ use of strings to define PDF

User friendly

→ Use ROOT Jupyter notebooks and/or PyROOT
→ Minimise amount of user code

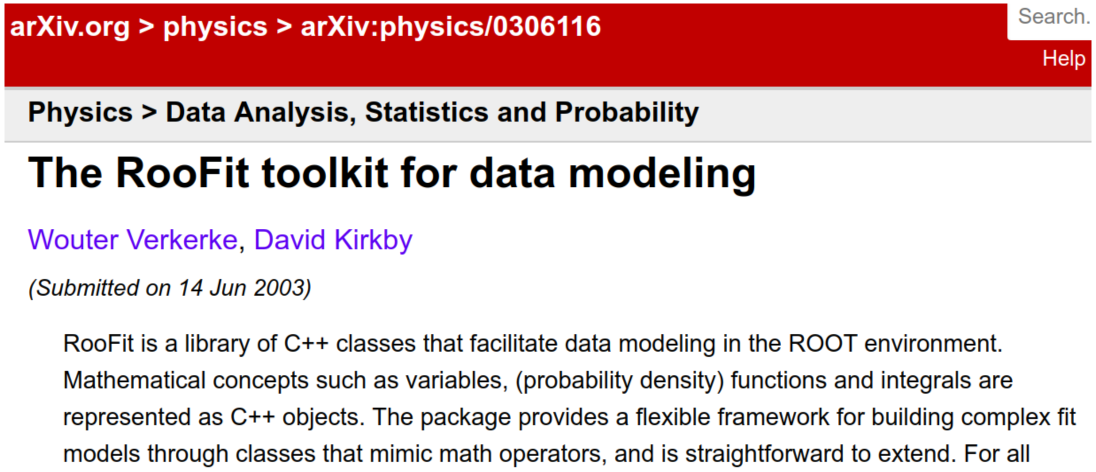
Fast Efficient calculation

→ Cache data and integrals
→ parallelise with via RooFit multicore Likelihood splitting
→ parallelise bins with ROOT PROOF multicore
→ parallelise on farm

Reliable results

Systematic uncertainties in extraction procedure?
→ ToyMC fits

RooFit and RooStats



arXiv.org > physics > arXiv:physics/0306116

Search.

Help

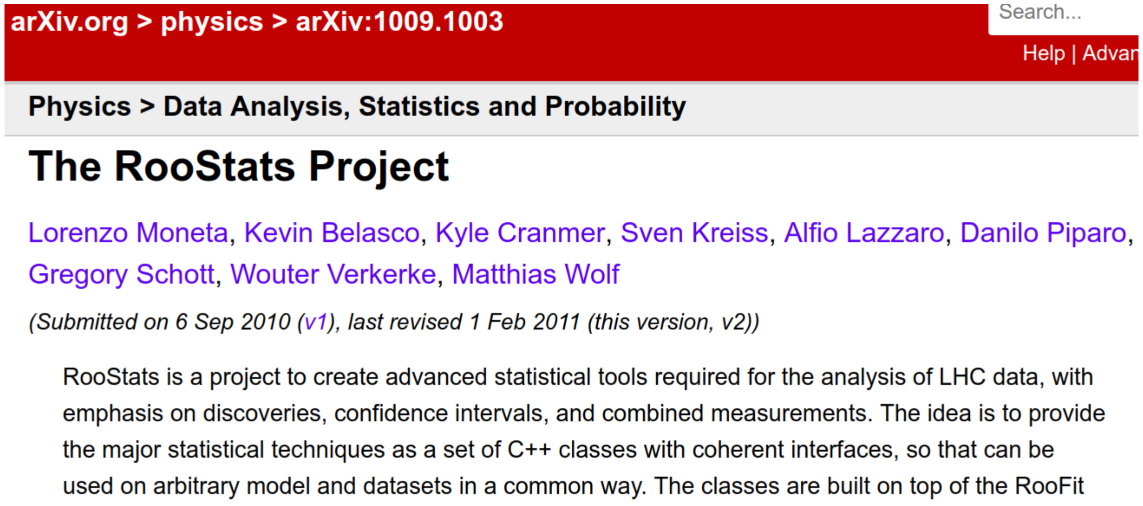
Physics > Data Analysis, Statistics and Probability

The RooFit toolkit for data modeling

Wouter Verkerke, David Kirkby

(Submitted on 14 Jun 2003)

RooFit is a library of C++ classes that facilitate data modeling in the ROOT environment. Mathematical concepts such as variables, (probability density) functions and integrals are represented as C++ objects. The package provides a flexible framework for building complex fit models through classes that mimic math operators, and is straightforward to extend. For all



arXiv.org > physics > arXiv:1009.1003

Search...

Help | Advan

Physics > Data Analysis, Statistics and Probability

The RooStats Project

Lorenzo Moneta, Kevin Belasco, Kyle Cranmer, Sven Kreiss, Alfio Lazzaro, Danilo Piparo, Gregory Schott, Wouter Verkerke, Matthias Wolf

(Submitted on 6 Sep 2010 (v1), last revised 1 Feb 2011 (this version, v2))

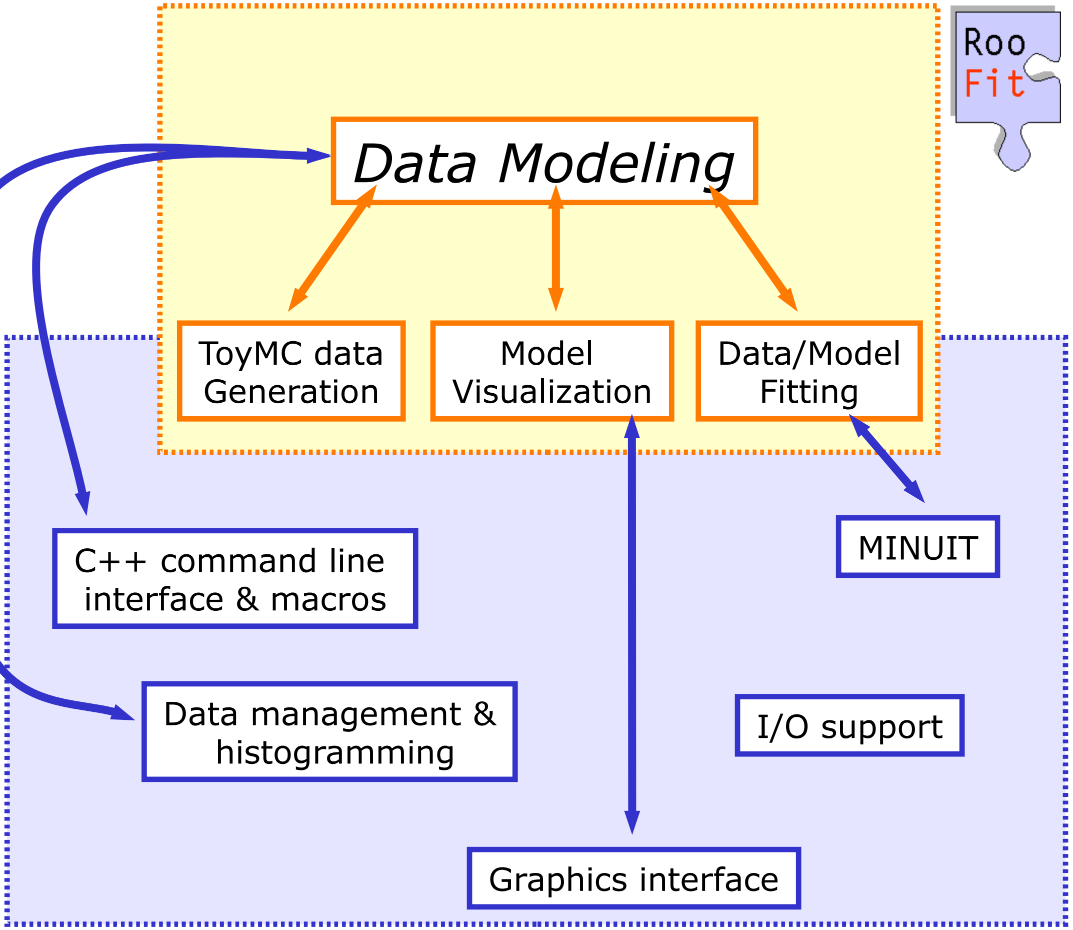
RooStats is a project to create advanced statistical tools required for the analysis of LHC data, with emphasis on discoveries, confidence intervals, and combined measurements. The idea is to provide the major statistical techniques as a set of C++ classes with coherent interfaces, so that can be used on arbitrary model and datasets in a common way. The classes are built on top of the RooFit

Used in many analysis in HEP

RooFit ~830 citations
RooStats ~670 citations

Optimised maximum likelihood fitter + lots of features

RooFit


Data Modeling
ToyMC data Generation
Model Visualization
Data/Model Fitting
C++ command line interface & macros
MINUIT
Data management & histogramming
I/O support
Graphics interface

ROOT

RooFit

- Mathematical objects are represented as C++ objects

Mathematical concept		RooFit class
variable	x	**RooRealVar**
function	$f(x)$	**RooAbsReal**
PDF	$f(x)$	**RooAbsPdf**
space point	$\vec{x}$	**RooArgSet**
integral	$\int_{x_{\min}}^{x_{\max}} f(x)dx$	**RooRealIntegral**
list of space points		**RooAbsData**
PDF summation	$f_1(x) + f_2(x)$	**RooAddPdf**
PDF product	$f_1(x) \cdot f_2(x)$	**RooProdPdf**

Adapted from Introduction to RooFit (W. Verkerke)

Model building – (Re)using standard components

- List of most frequently used pdfs and their factory spec

Gaussian	`Gaussian::g(x,mean,sigma)`
Breit-Wigner	`BreitWigner::bw(x,mean,gamma)`
Landau	`Landau::l(x,mean,sigma)`
Exponential	`Exponental::e(x,alpha)`
Polynomial	`Polynomial::p(x,{a0,a1,a2})`
Chebychev	`Chebychev::p(x,{a0,a1,a2})`
Kernel Estimation	`KeysPdf::k(x,dataSet)`
Poisson	`Poisson::p(x,mu)`
Voigtian (=BW$\otimes$G)	`Voigtian::v(x,mean,gamma,sigma)`

Application of RooFit

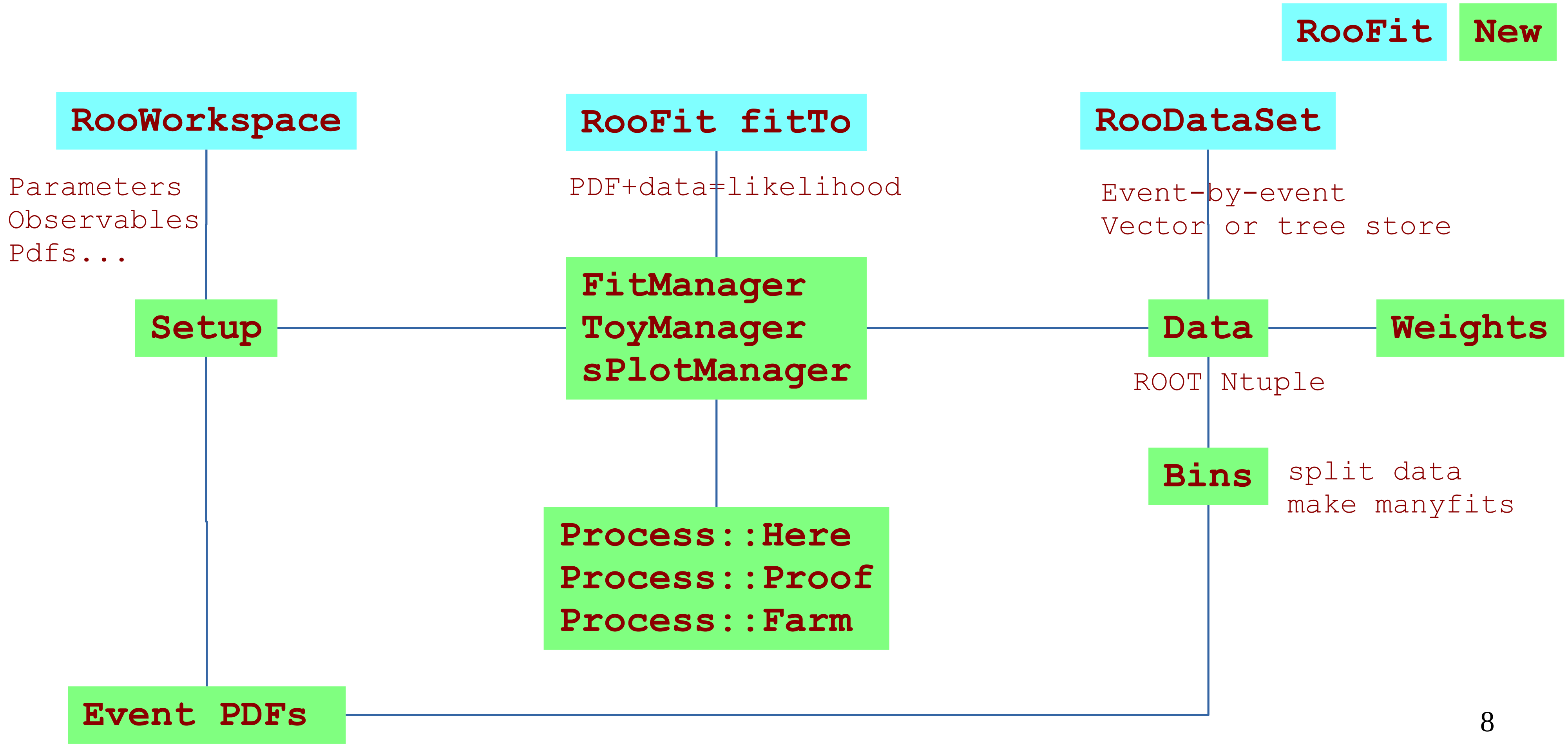
Is it useful for extracting parameters from fits to angular distributions of real experimental data ?

Arbitrary PDF	→ Skeleton Pdf code generator
	→ Define evaluate() function
	→ Use TFormula string→function→pdf
	→ Analytic or Numerical integrator
Signal / background	→ simultaneous fit to both $S(x)*f_S(y,\theta_S)+B(x)*f_B(y,\theta_B(x))$
	→ sPlot + weighted likelihood fits
Acceptance correction	→ **No, but could use weights**

Systematic uncertainties → ToyMC studies

Many fits to different bins → Not trivialy

Software Structure



Maximum Likelihood with acceptance

$$L(p) = \prod_i^N \frac{f(\tau_i : p)}{\int f(\tau_i : p)\, d\tau}$$

Likelihood = product Prob.
function f with observables τ and pars p

$$L_{acc}(p) = \prod_i^N \frac{f(\tau_i : p)\,\eta(\tau_i)}{\int f(\tau_i : p)\,\eta(\tau_i)\, d\tau}$$

With acceptance need product
of f and acceptance function

$$A(p) = \int f(\tau_i : p)\,\eta(\tau_i)\, d\tau$$

Probability Normalisation ∫

$$L_{acc}^{ext}(p) = \left[\frac{A(p)^N}{N!} e^{-A(p)}\right] \prod_i^N \frac{f(\tau_i : p)\,\eta(\tau_i)}{A(p)}$$

Extended ML with Poisson statistics

$$-\ln L_{acc}^{ext}(p) \propto -\sum_i^N \ln f(\tau_i : p)\,\eta(\tau_i) + A(p)$$

Minimise -ln(L)
Drop terms independent of p

$$= -\sum_i^N \ln f(\tau_i : p) \cancel{- \sum_i^N \ln \eta(\tau_i)} + A(p)$$

Drop acceptance in sum over events

$$\propto -\sum_i^N \ln f(\tau_i : p) + A(p)$$

Need to minimise this

$$A(p) \simeq \sum_j^M f(\tau_j : p)$$

Approximating A(p) as sum of
f over M accepted Monte-Carlo events

Simulated MC integrals for RooFit : RooHSEventsPDF

RooFit PDF classes require normalisation

-This can be done by users own method

-Or RooFit performs is own numerical integration(vegas)

RooHSEventsPDF calculates its own integral by summing over MC events – includes partial integration for plotting

Requires ROOT tree of reconstructed MC events with fit variables

Can give weights to MC events (e.g. better match non-fitted data and MC distributions, importance sampling)

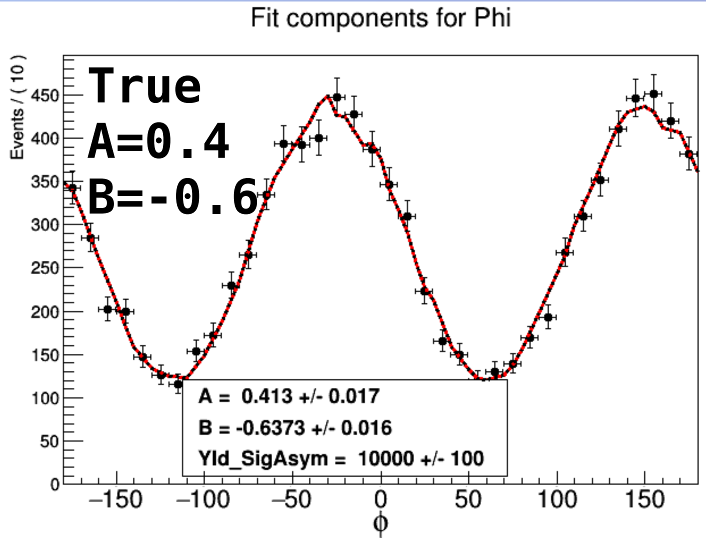
Additionally can generate events using true values if these are past through in the tree

- Evaluate function with generated truth, produce events with simulated reconstructed

Works any number of fit observables or parameters

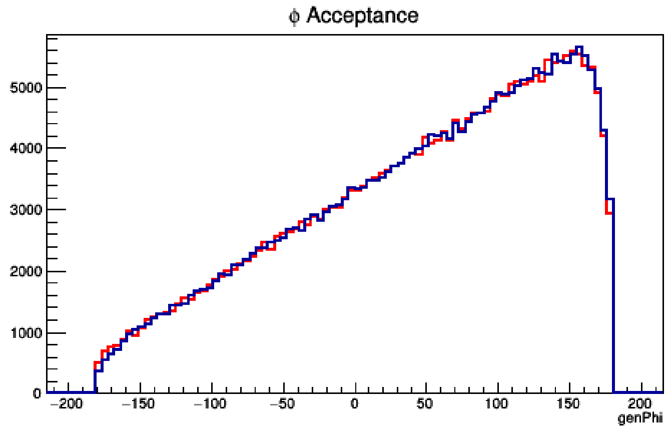
- i.e. given a Ndim Model it will fit for Mpar parameters accounting for detector acceptance effects

Fits with acceptance

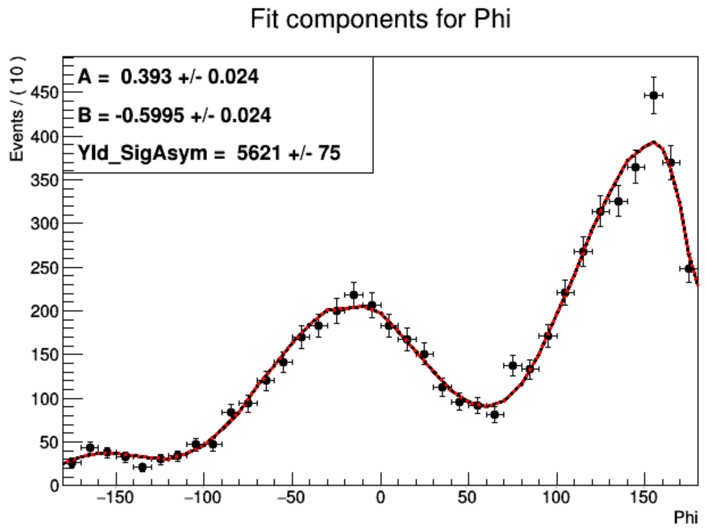


100% acceptance
PDF : 1 + Acos(2ϕ)+Bsin(2ϕ)

+

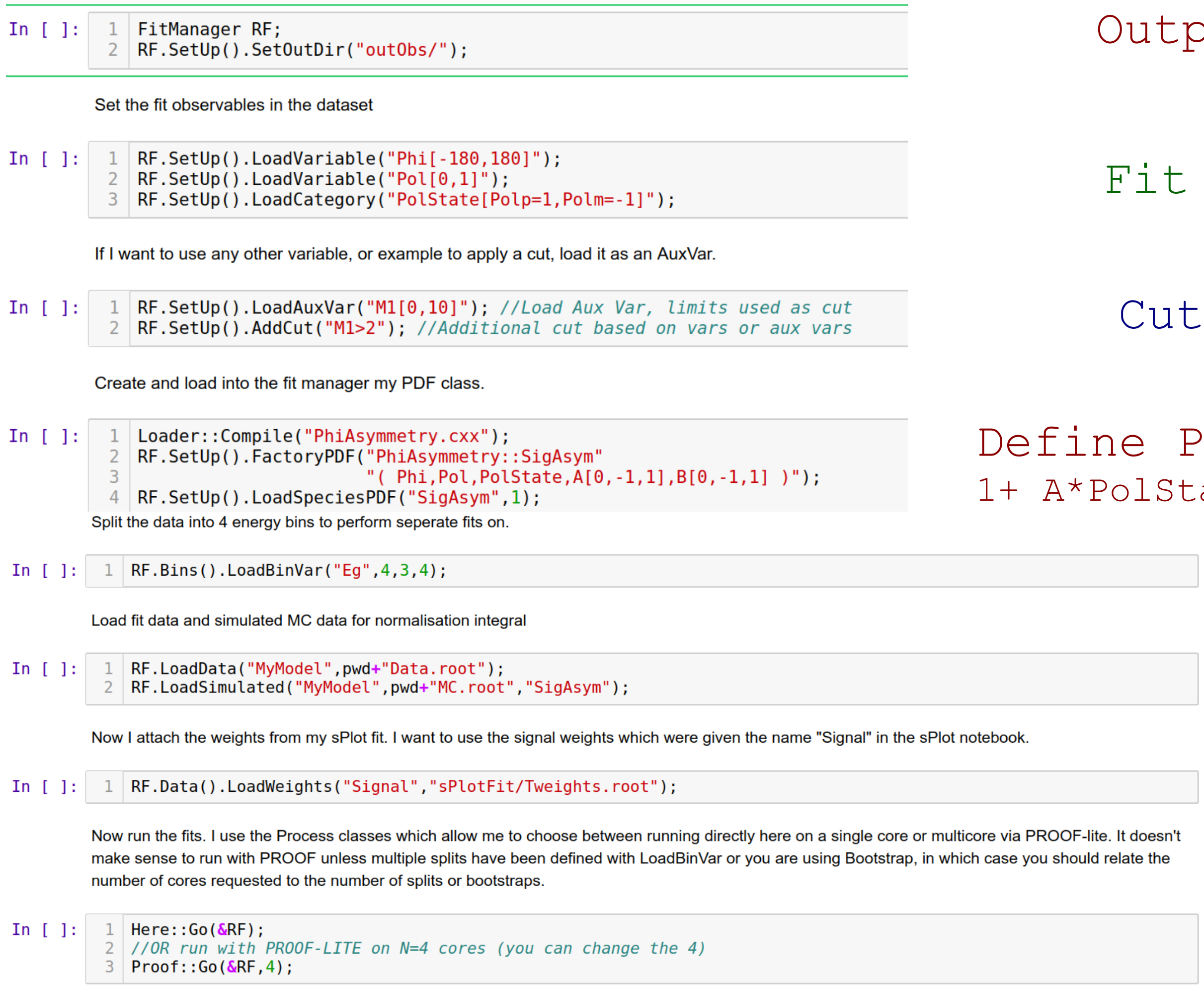


=



Plotting of fit function integrates over MC events

Example Fit in Jupyter



In [ ]:
```
FitManager RF;
RF.SetUp().SetOutDir("outObs/");
```

Output location

Set the fit observables in the dataset

In [ ]:
```
RF.SetUp().LoadVariable("Phi[-180,180]");
RF.SetUp().LoadVariable("Pol[0,1]");
RF.SetUp().LoadCategory("PolState[Polp=1,Polm=-1]");
```

Fit Observables

If I want to use any other variable, or example to apply a cut, load it as an AuxVar.

In [ ]:
```
RF.SetUp().LoadAuxVar("M1[0,10]"); //Load Aux Var, limits used as cut
RF.SetUp().AddCut("M1>2"); //Additional cut based on vars or aux vars
```

Cuts to other variables

Create and load into the fit manager my PDF class.

In [ ]:
```
Loader::Compile("PhiAsymmetry.cxx");
RF.SetUp().FactoryPDF("PhiAsymmetry::SigAsym"
                      "( Phi,Pol,PolState,A[0,-1,1],B[0,-1,1] )");
RF.SetUp().LoadSpeciesPDF("SigAsym",1);
```

Define PDF (here load a class)

1+ A*PolState*Pol*cos(2ϕ)+ B*PolState*Pol*sin(2ϕ)

Split the data into 4 energy bins to perform seperate fits on.

In [ ]:
```
RF.Bins().LoadBinVar("Eg",4,3,4);
```

Split data in bins of Eg

Load fit data and simulated MC data for normalisation integral

In [ ]:
```
RF.LoadData("MyModel",pwd+"Data.root");
RF.LoadSimulated("MyModel",pwd+"MC.root","SigAsym");
```

Set Fit and MC Integral data

Now I attach the weights from my sPlot fit. I want to use the signal weights which were given the name "Signal" in the sPlot notebook.

In [ ]:
```
RF.Data().LoadWeights("Signal","sPlotFit/Tweights.root");
```

Set Event weights

Now run the fits. I use the Process classes which allow me to choose between running directly here on a single core or multicore via PROOF-lite. It doesn't make sense to run with PROOF unless multiple splits have been defined with LoadBinVar or you are using Bootstrap, in which case you should relate the number of cores requested to the number of splits or bootstraps.

In [ ]:
```
Here::Go(&RF);
//OR run with PROOF-LITE on N=4 cores (you can change the 4)
Proof::Go(&RF,4);
```

Run fit here or on PROOF

sPlot

M. Pivk,F.R. Le Diberder,Nucl.Inst.Meth.A 555, 356-369, 2005

Given discriminatory PDF for signal and background calculates weight :

$$ {}_s\mathcal{P}_{\mathrm{n}}(y_e) = \frac{\sum_{j=1}^{\mathrm{N_S}} \mathbf{V}_{\mathrm{n}j} \mathrm{f}_j(y_e)}{\sum_{k=1}^{\mathrm{N_S}} N_k \mathrm{f}_k(y_e)} $$

N_s = Number of species

f_k = PDF for species k

N_k = Yield for species k

V = covariance matrix

Part of RooStats(used here)

Can include multiple signal
and background species

Can use directly in likelihood fits

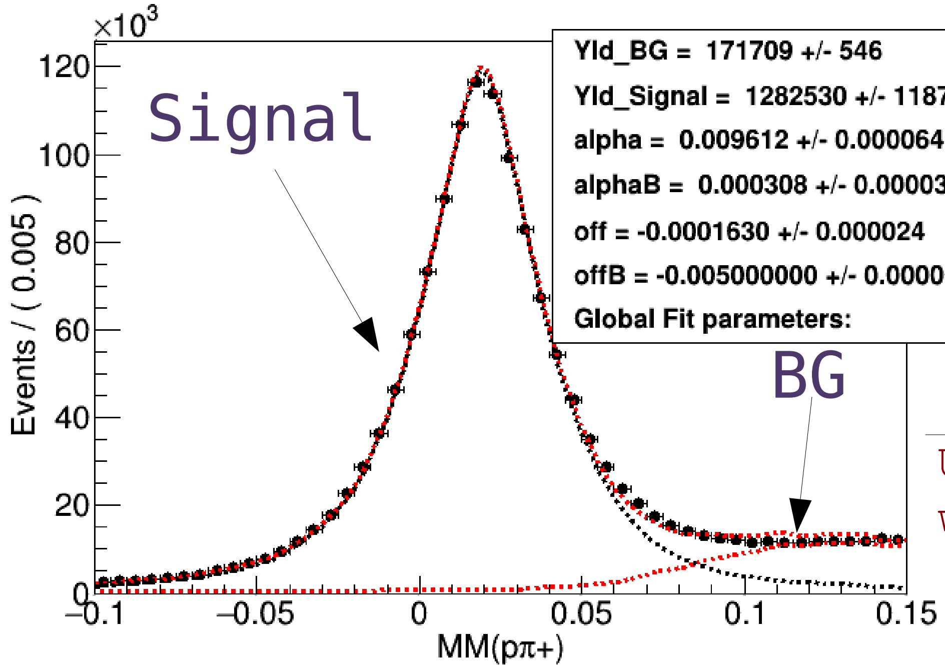
In this package running sPlot, same as running any fit

But only as good as fit model…

And Signal OR Background observable distributions cannot
correlate with discriminatory variables

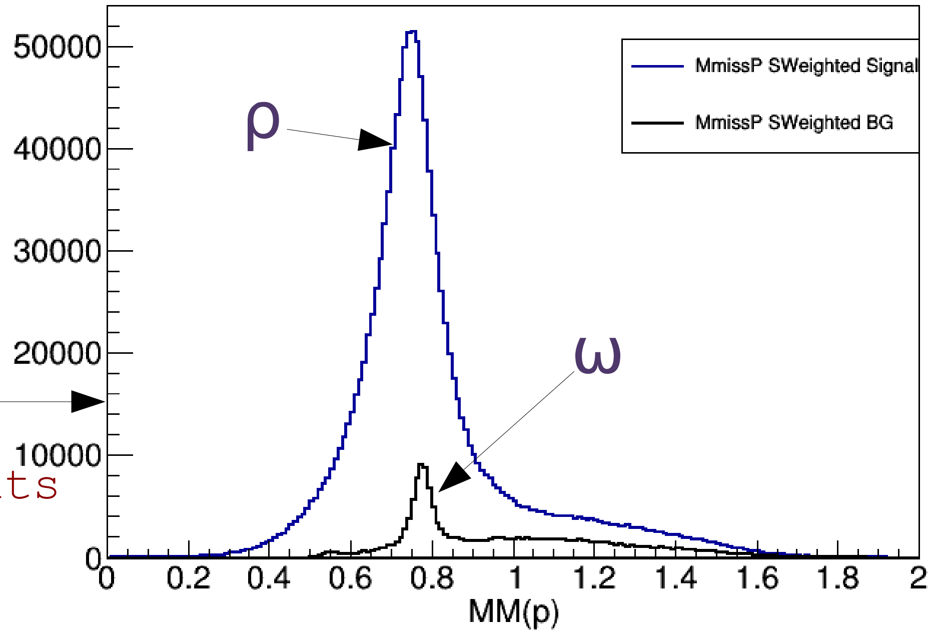
Example fit and disentangled distributions

$\gamma p \rightarrow p\pi+(\pi-)$ Models from simulated $\pi^+\pi^-p$ and $\pi^+\pi^-\pi^0p$ events



Yld_BG = 171709 +/- 546
Yld_Signal = 1282530 +/- 1187
alpha = 0.009612 +/- 0.000064
alphaB = 0.000308 +/- 0.000033
off = -0.0001630 +/- 0.000024
offB = -0.005000000 +/- 0.00000024
Global Fit parameters:

Use weights

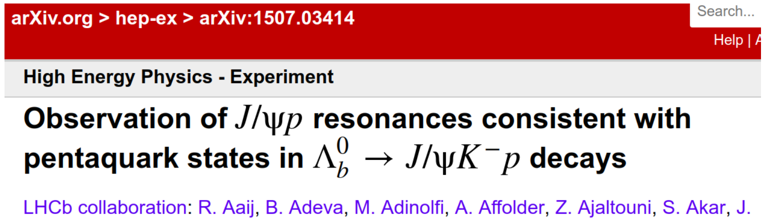


Note 2 fits required. First fixes alpha and off.
Second, only Yields free => Covariance matrix

Maximum likelihood with weights

Many analysis include additional factor to better approximate uncertainties

e.g



arXiv.org > hep-ex > arXiv:1507.03414

Search...

Help | A

High Energy Physics - Experiment

Observation of $J/\psi p$ resonances consistent with pentaquark states in $\Lambda_b^0 \to J/\psi K^- p$ decays

LHCb collaboration: R. Aaij, B. Adeva, M. Adinolfi, A. Affolder, Z. Ajaltouni, S. Akar, J.

$$-2\ln(L(p)) = -2\alpha\left[\sum_i \underset{\uparrow}{W_i} \ln(f(\tau_i, p)) + A(p)\right]$$

$$\alpha = \frac{\sum_i W_i}{\sum_i W_i^2}$$

Weights subtract off background contribution to likelihood

α term reduces gradient
Therefore increase uncertainty

RooFit with weights

$$-2\ln(L(p)) = -2\left[\sum_i W_i \ln(f(\tau, p)) + A(p)\right]$$ (1) Minimise this

$$-2\ln(L(p)) = -2\left[\sum_i W_i^2 \ln(f(\tau, p)) + A(p)\right]$$ (2) Also calculate this

$$G_{\mu\nu} = \sum_{i=1}^{N} w_i \frac{dp_i}{d\alpha_\mu} \frac{dp_i}{d\alpha_\nu}$$ covariance matrix for (1)

$$F_{\mu\nu} = \sum_{i=1}^{N} (w_i)^2 \frac{dp_i}{d\alpha_\mu} \frac{dp_i}{d\alpha_\nu}$$ covariance matrix for (2)

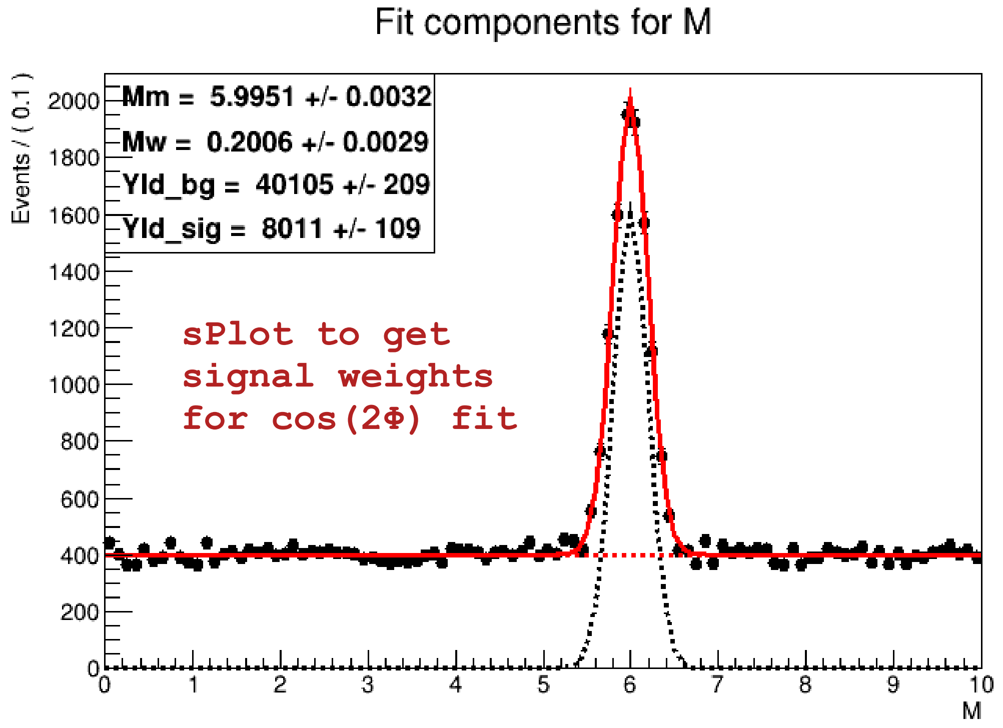
$$\boxed{\mathrm{COV}(\vec{\alpha}, \vec{\alpha}) = G^{-1}\, F\, G^{-1}}$$

errors scale with $\frac{\sum_i W_i^2}{\sum_i W_i}$

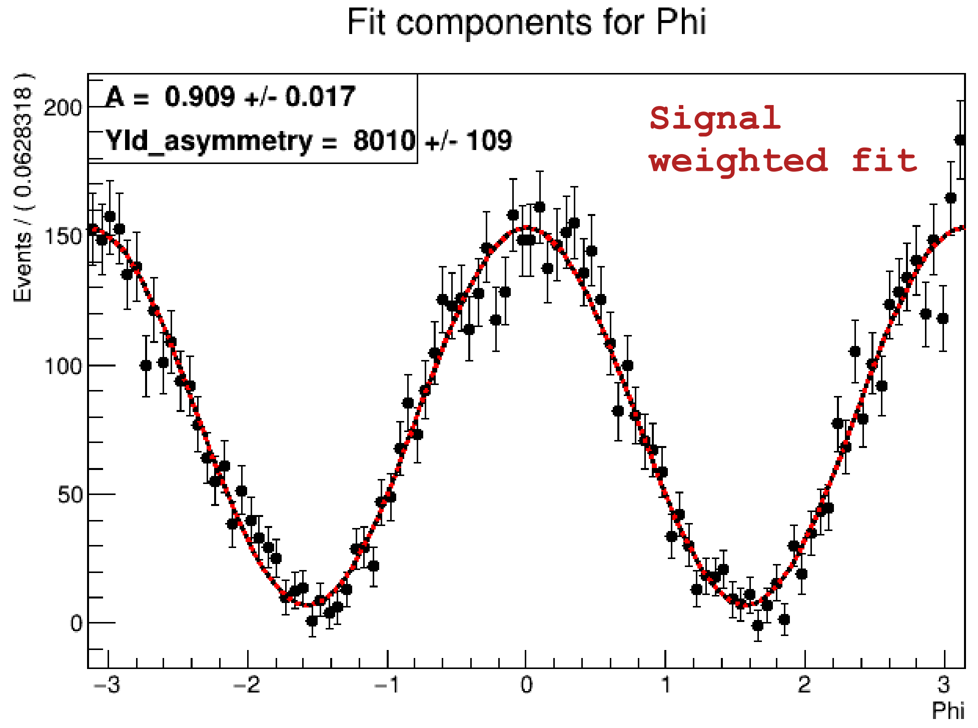
Taken from https://twiki.cern.ch/twiki/pub/Main/RooFit/baak_eventweights.pdf

Weighted Fit uncertainties validation

Toy Fits :	Signal 8k	f(M)= Gaussian(6,0.2)	g(Φ)= 1 + 0.92cos(2Φ)
	Backgr 40k	f(M)= Uniform	g(Φ)= Uniform

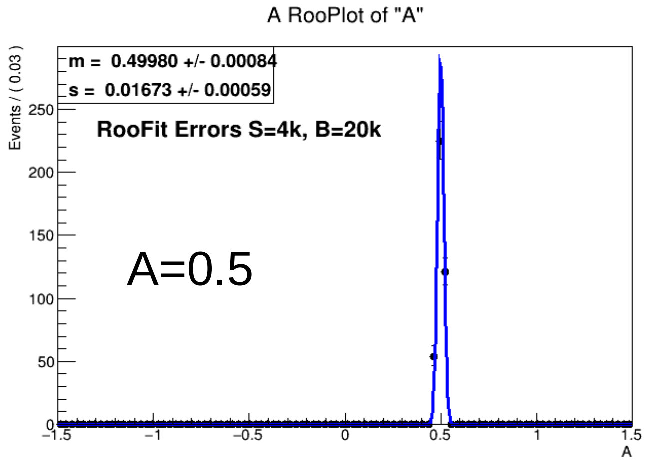


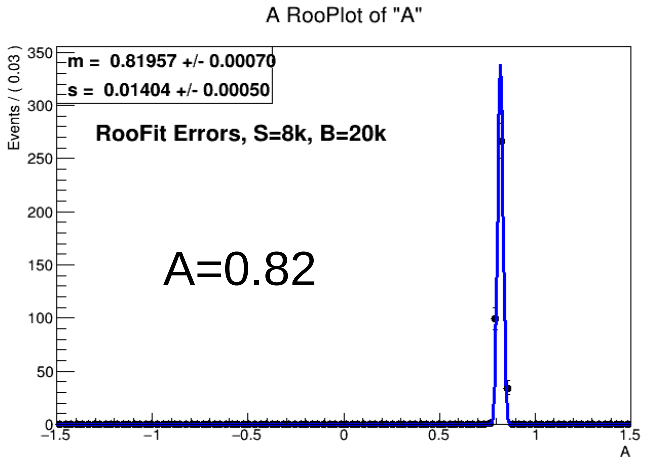
sPlot to get signal weights for cos(2Φ) fit

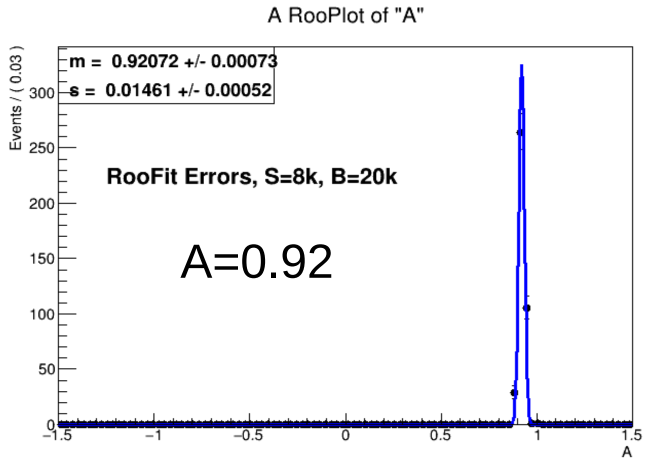


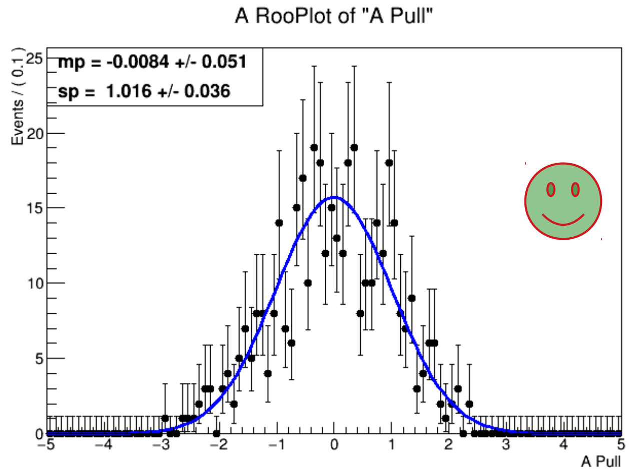
Signal weighted fit

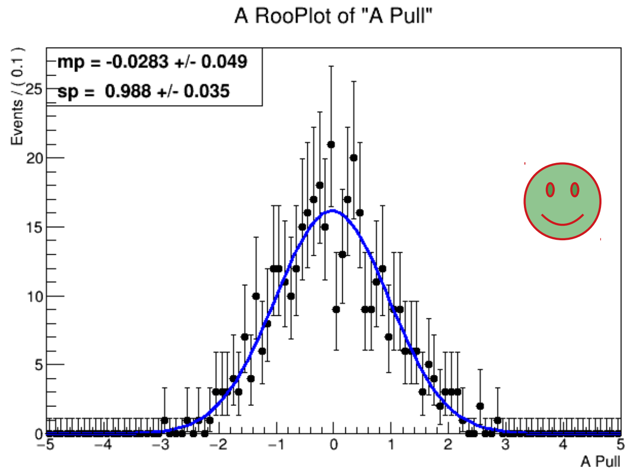
Weighted Fit uncertainties validation RooFit errors,400 fits, change A

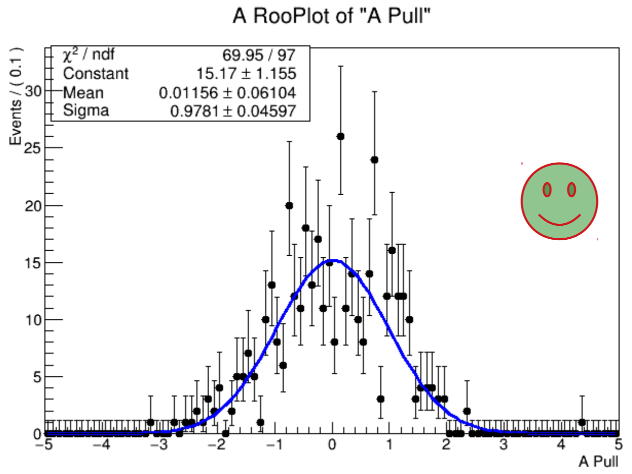




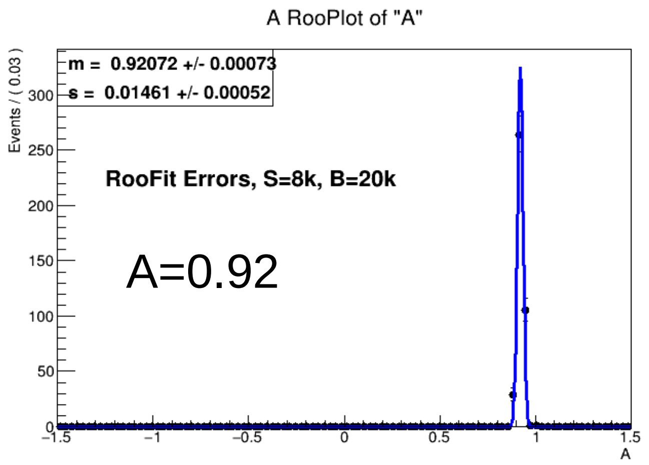


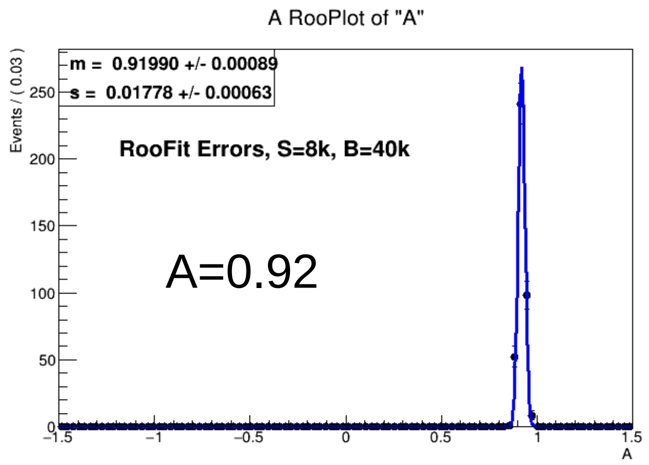


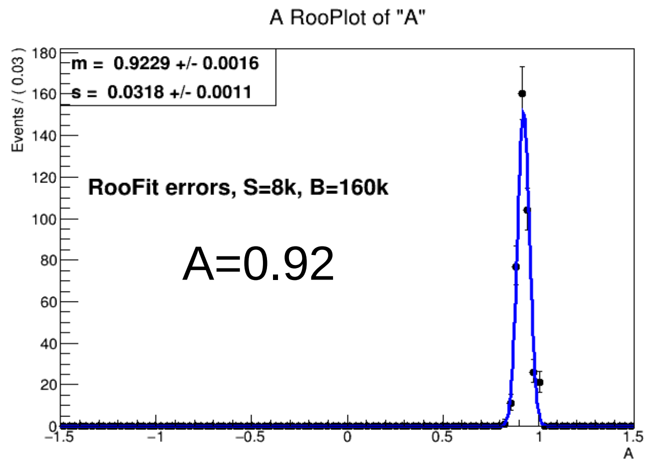


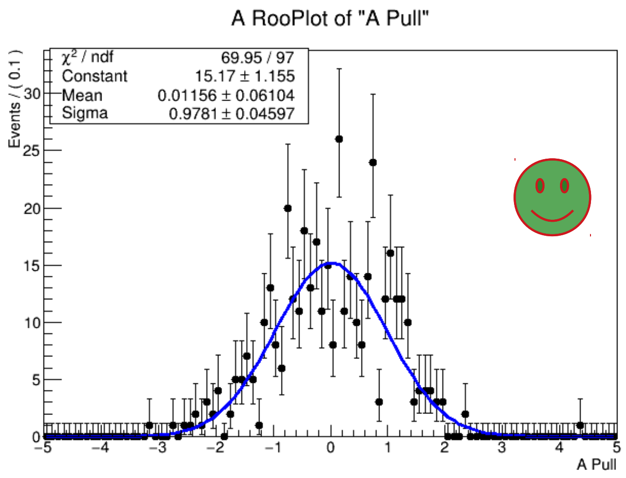


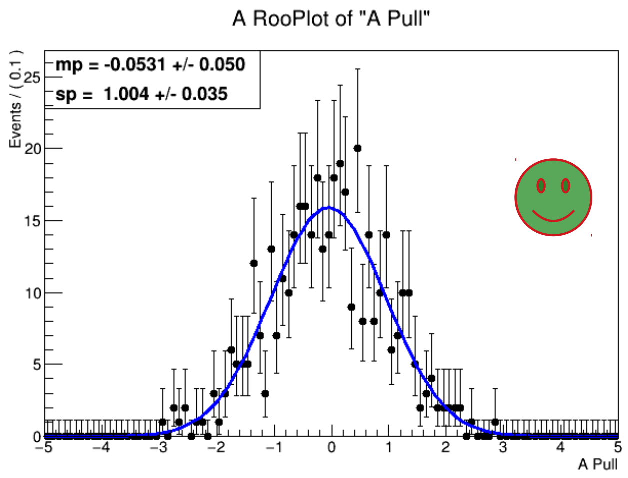
Weighted Fit uncertainties validation
RooFit errors,400 fits, change BG

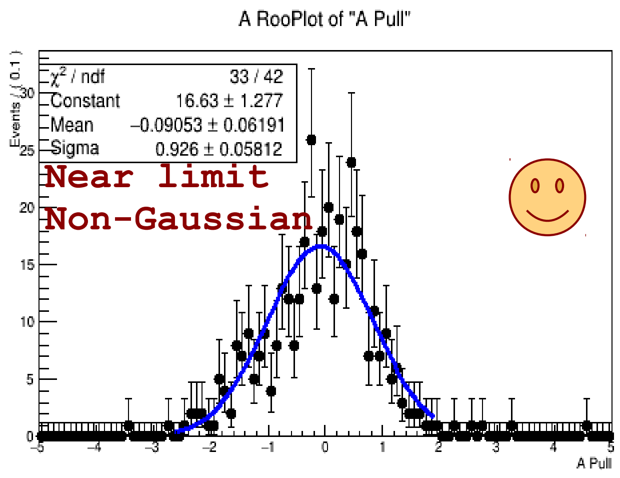






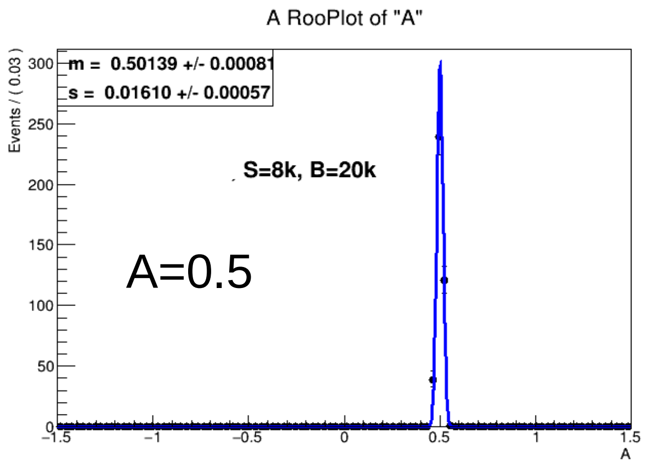


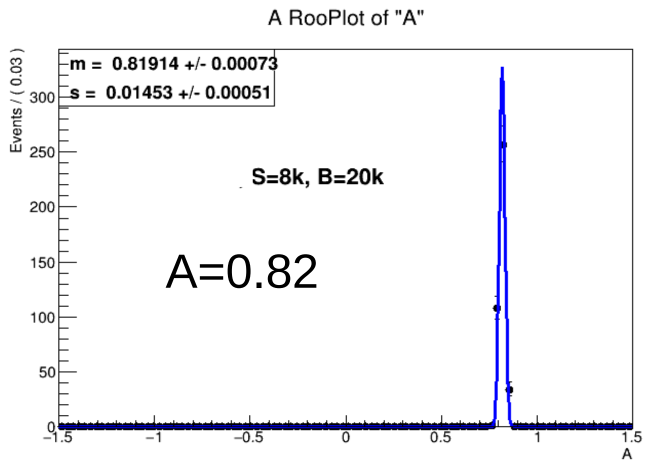


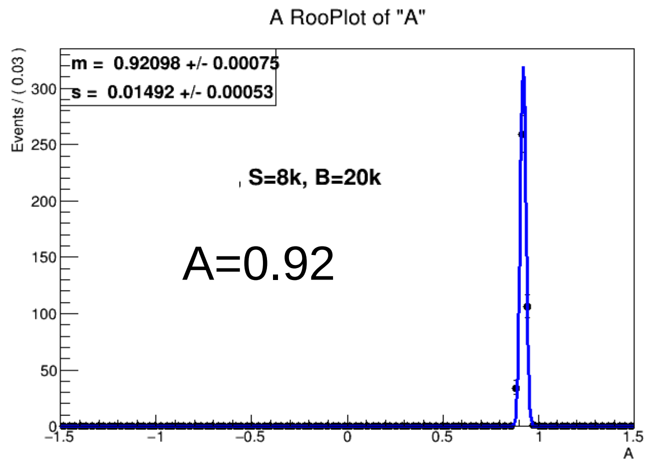


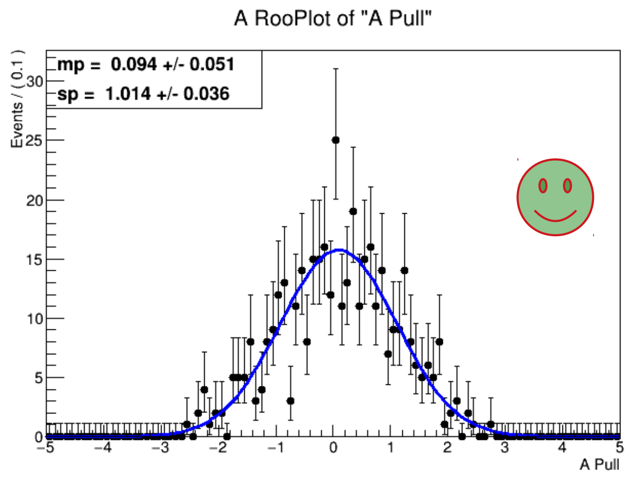
Near limit
Non-Gaussian

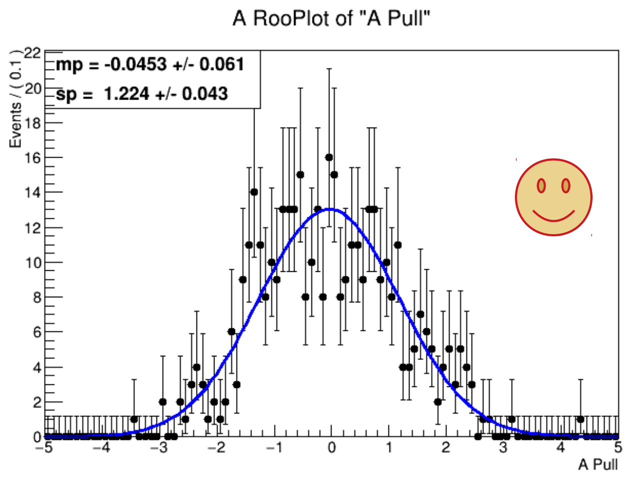
Weighted Fit uncertainties validation α-factor errors,400 fits, change A

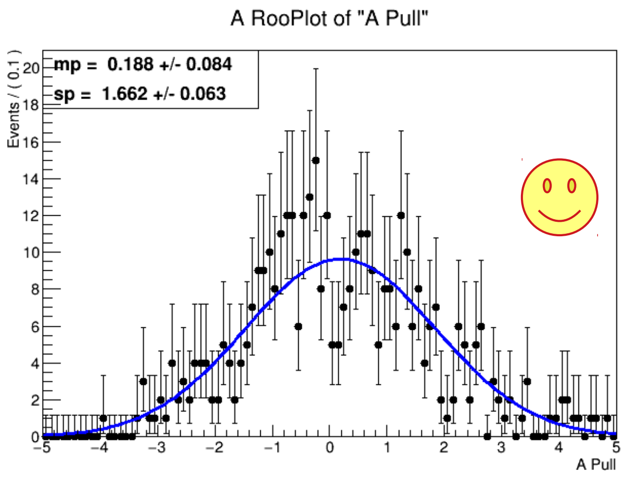












Uncertainities with acceptance

Uncertainties from simulated MC integral are not propogated to Minuit covariance matrix

Normally try to have x10 MC events ~ 1/√10 ~ 0.3 additional factor on statistical uncertainty

Can be estimated via boostrapping the MC events samples

For asymmetries MC statistical uncertainty can be removed by calculating

$$\sum_{i=0}^{Nacc} f(\tau,p,P=+1)+\sum_{i=0}^{Nacc} f(\tau,p,P=-1)$$

In the limit N(+)=N(-)=Nacc/2

As this will be constant, rather than

$$\sum_{i=0}^{Nacc/2} f(\tau,p,P=+1)+\sum_{i=Nacc/2}^{Nacc} f(\tau,p,P=-1)$$

With small corrections in the case of asymmetries in luminosity N(+)!=N(-)

Minimisers

It is straightforward to try different minimisers to fit the data

Minuit2 is used by default

To use Minuit :

```
    Fitter.SetMinimiser(new RooMinuit());
```

Or an MCMC implementation :

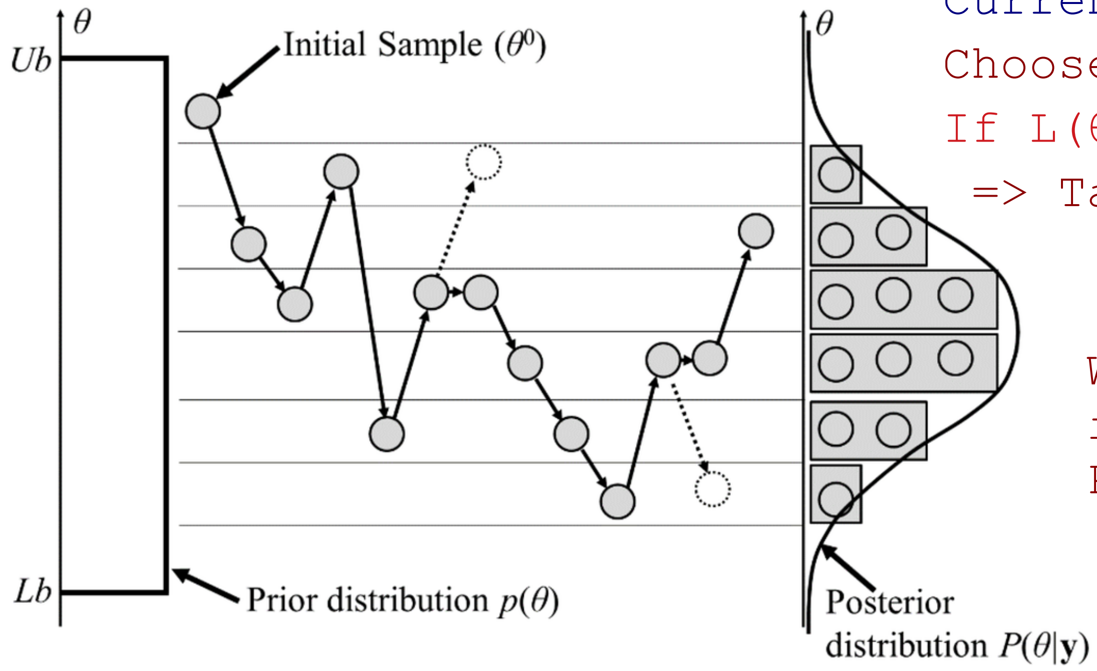
```
    Fitter.SetMinimiser(new RooMcmcSeq(30000,10000,1000));
```

Number of samples

Number of burnin

1/StepSize

MCMC with Metropolis-Hastings, for finding parameter values



Current parameters = θ_k

Choose θ′ from $q(\theta',\theta_k)$ (proposal func.)

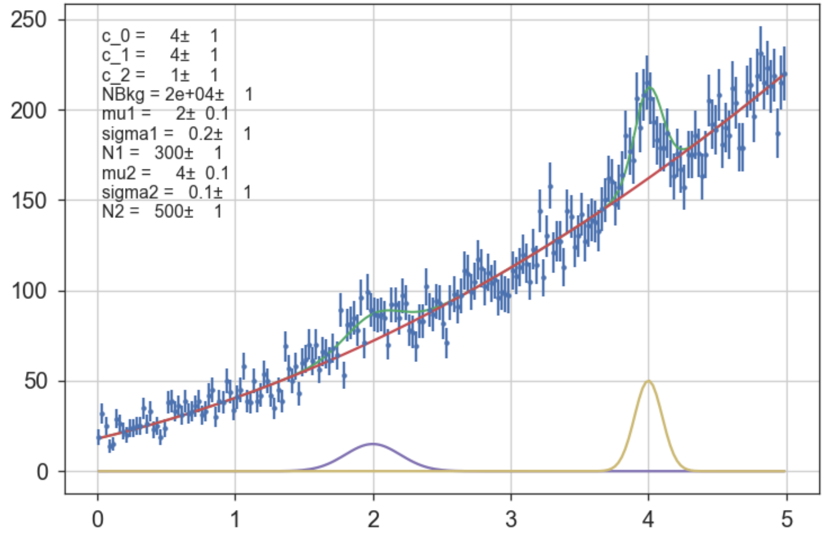
If $L(\theta')/L(\theta_k)$ > Uniform(0,1)

=> Take θ′ as next parameters sample

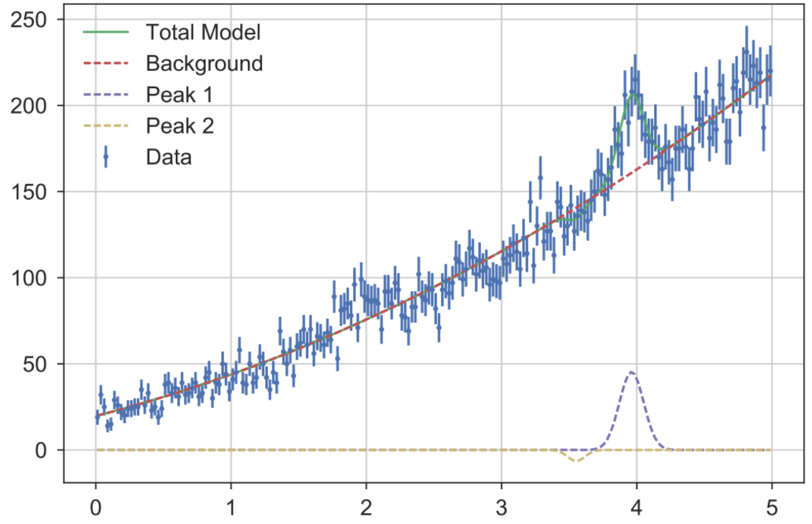
We use sequential proposal
i.e. change 1 parameter to
P += RandGaussian(P,P_{range}*StepSize)

MCMC Test Minuit fit

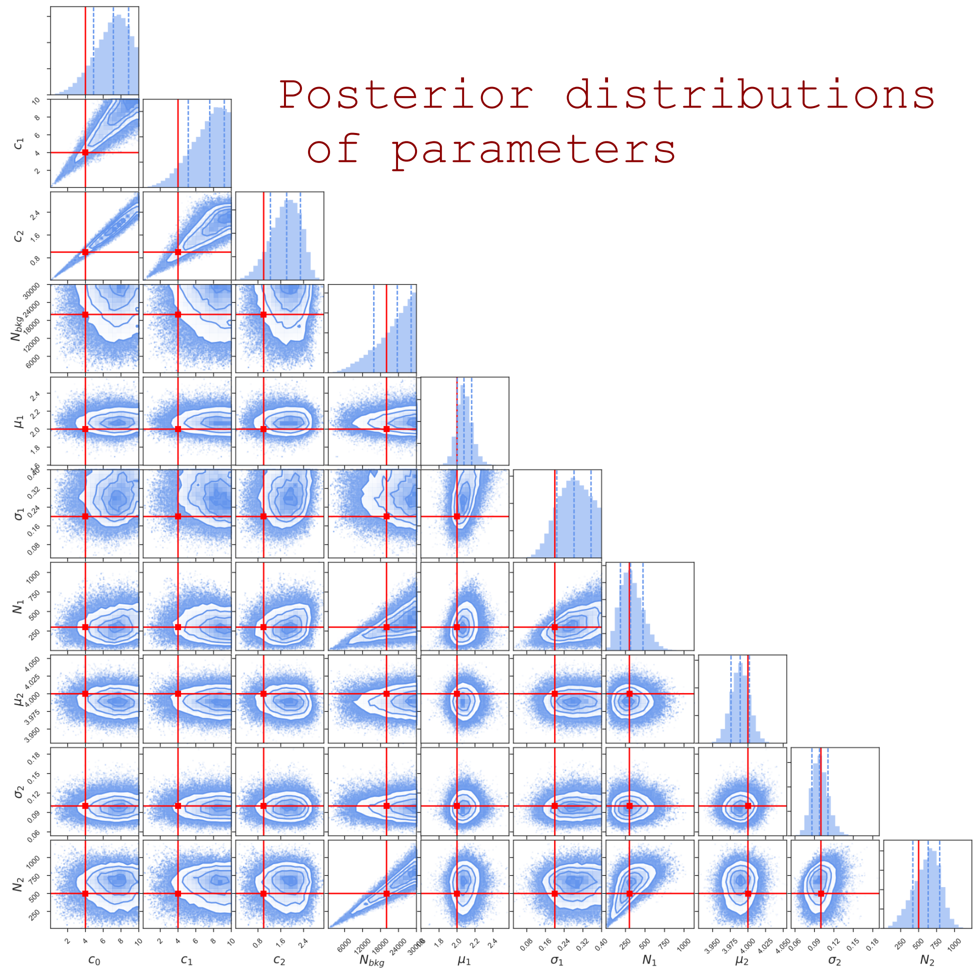
Generated Data with true PDFs



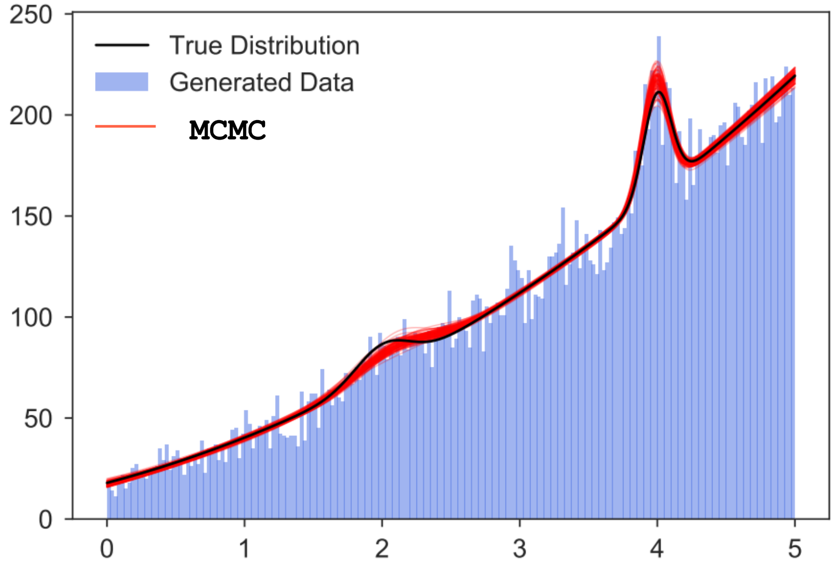
Minuit Fit results
Random initial parameters



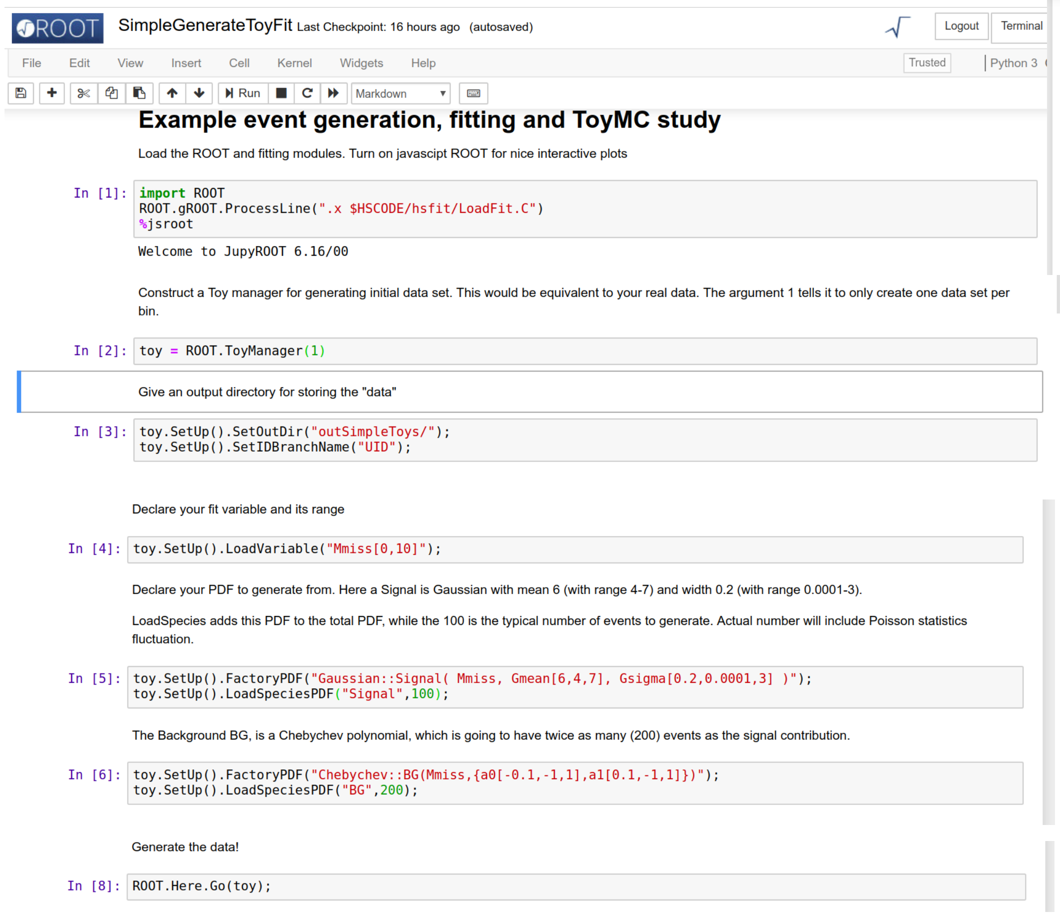
MCMC Test fit



Posterior distributions of parameters



Systematic Studies: ToyMC in Jupyter


ROOT
SimpleGenerateToyFit Last Checkpoint: 16 hours ago (autosaved)
Logout
Terminal
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

Example event generation, fitting and ToyMC study

Load the ROOT and fitting modules. Turn on javascipt ROOT for nice interactive plots

```
In [1]: import ROOT
        ROOT.gROOT.ProcessLine(".x $HSCODE/hsfit/LoadFit.C")
        %jsroot
```

```
Welcome to JupyROOT 6.16/00
```

Construct a Toy manager for generating initial data set. This would be equivalent to your real data. The argument 1 tells it to only create one data set per bin.

```
In [2]: toy = ROOT.ToyManager(1)
```

Give an output directory for storing the "data"

```
In [3]: toy.SetUp().SetOutDir("outSimpleToys/");
        toy.SetUp().SetIDBranchName("UID");
```

Declare your fit variable and its range

```
In [4]: toy.SetUp().LoadVariable("Mmiss[0,10]");
```

Declare your PDF to generate from. Here a Signal is Gaussian with mean 6 (with range 4-7) and width 0.2 (with range 0.0001-3).

LoadSpecies adds this PDF to the total PDF, while the 100 is the typical number of events to generate. Actual number will include Poisson statistics fluctuation.

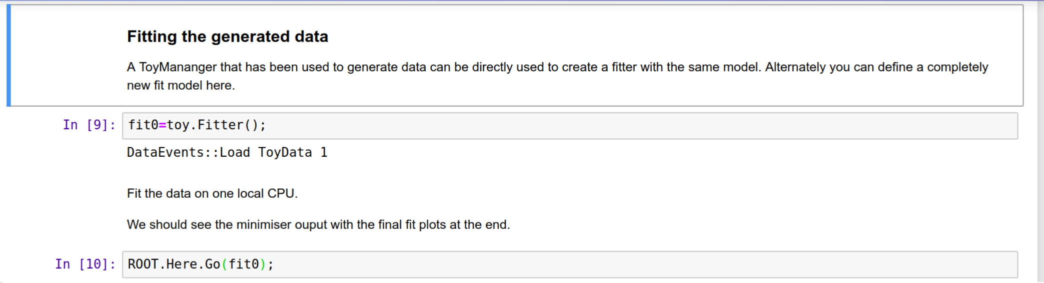
```
In [5]: toy.SetUp().FactoryPDF("Gaussian::Signal( Mmiss, Gmean[6,4,7], Gsigma[0.2,0.0001,3] )");
        toy.SetUp().LoadSpeciesPDF("Signal",100);
```

The Background BG, is a Chebychev polynomial, which is going to have twice as many (200) events as the signal contribution.

```
In [6]: toy.SetUp().FactoryPDF("Chebychev::BG(Mmiss,{a0[-0.1,-1,1],a1[0.1,-1,1]})");
        toy.SetUp().LoadSpeciesPDF("BG",200);
```

Generate the data!

```
In [8]: ROOT.Here.Go(toy);
```



Fitting the generated data

A ToyMananger that has been used to generate data can be directly used to create a fitter with the same model. Alternately you can define a completely new fit model here.

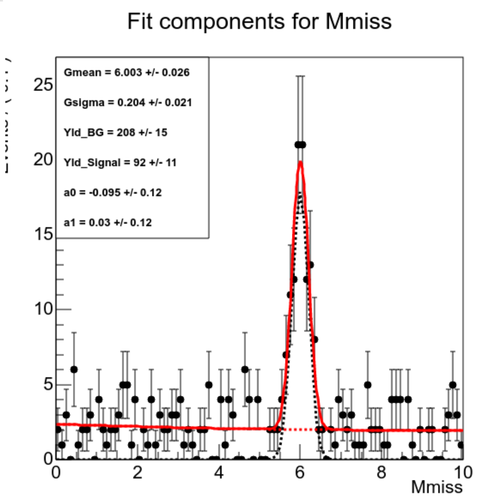
```
In [9]: fit0=toy.Fitter();
```

```
DataEvents::Load ToyData 1
```

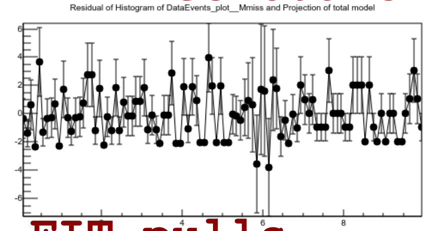
Fit the data on one local CPU.

We should see the minimiser ouput with the final fit plots at the end.

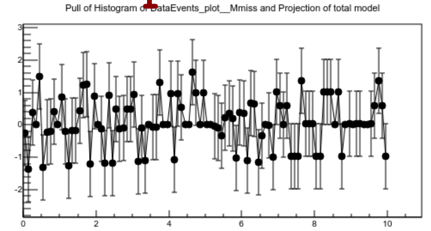
```
In [10]: ROOT.Here.Go(fit0);
```



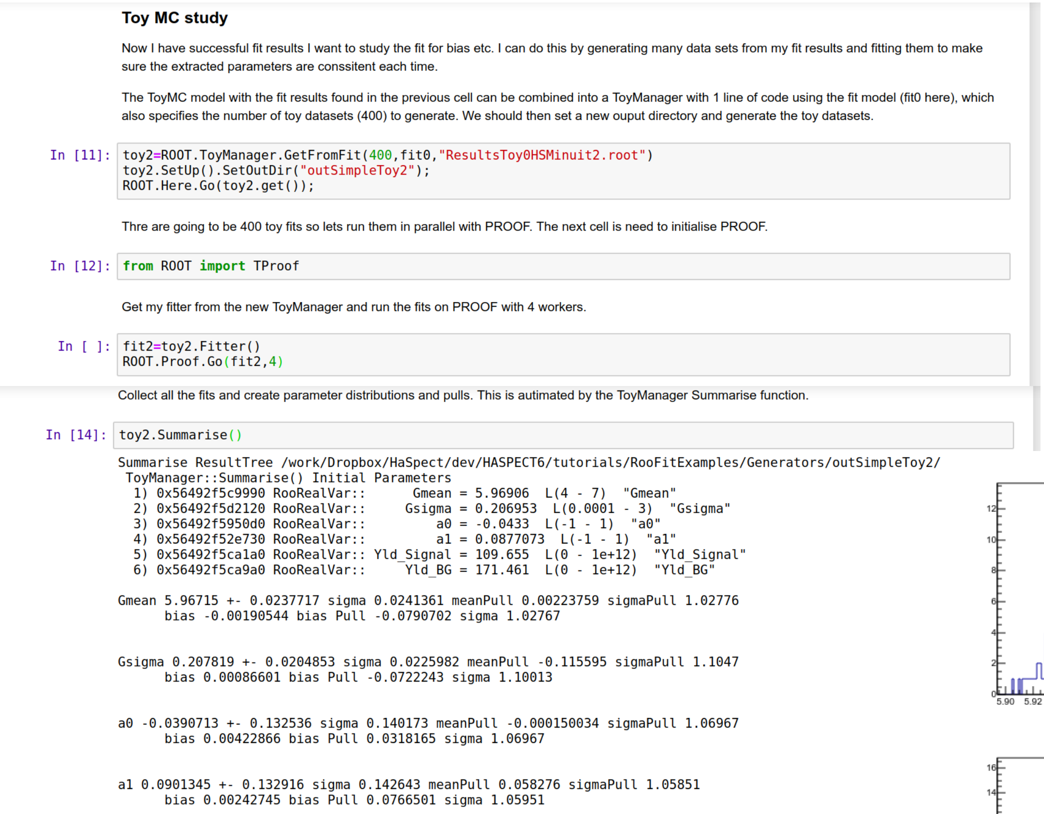
FIT residuals



FIT pulls



Example : ToyMC study in Jupyter



Toy MC study

Now I have successful fit results I want to study the fit for bias etc. I can do this by generating many data sets from my fit results and fitting them to make sure the extracted parameters are conssitent each time.

The ToyMC model with the fit results found in the previous cell can be combined into a ToyManager with 1 line of code using the fit model (fit0 here), which also specifies the number of toy datasets (400) to generate. We should then set a new ouput directory and generate the toy datasets.

In [11]:
```
toy2=ROOT.ToyManager.GetFromFit(400,fit0,"ResultsToy0HSMinuit2.root")
toy2.SetUp().SetOutDir("outSimpleToy2");
ROOT.Here.Go(toy2.get());
```

Thre are going to be 400 toy fits so lets run them in parallel with PROOF. The next cell is need to initialise PROOF.

In [12]:
```
from ROOT import TProof
```

Get my fitter from the new ToyManager and run the fits on PROOF with 4 workers.

In [ ]:
```
fit2=toy2.Fitter()
ROOT.Proof.Go(fit2,4)
```

Collect all the fits and create parameter distributions and pulls. This is autimated by the ToyManager Summarise function.

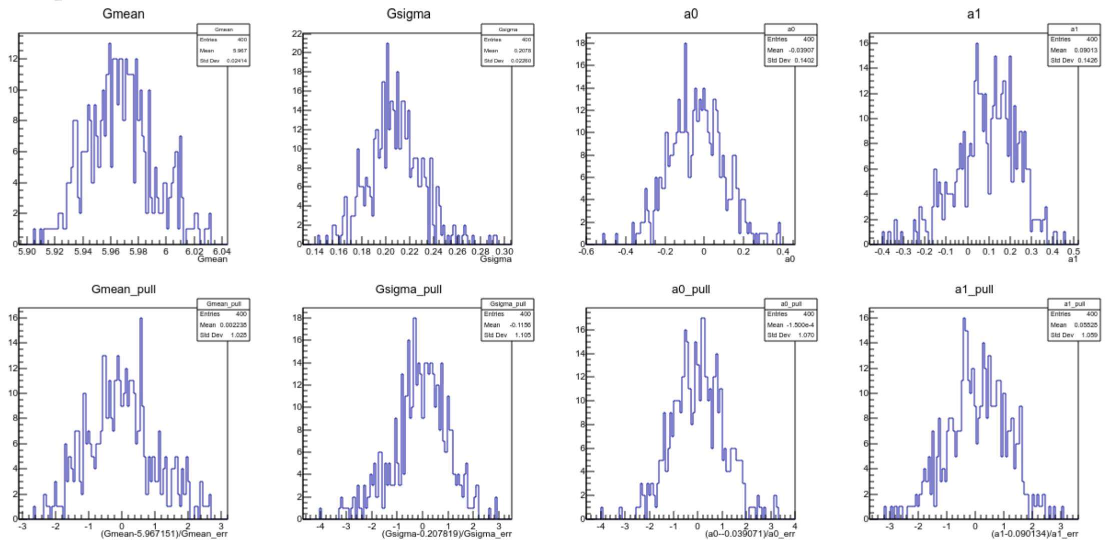
In [14]:
```
toy2.Summarise()
```

```
Summarise ResultTree /work/Dropbox/HaSpect/dev/HASPECT6/tutorials/RooFitExamples/Generators/outSimpleToy2/
 ToyManager::Summarise() Initial Parameters
  1) 0x56492f5c9990 RooRealVar::      Gmean = 5.96906  L(4 - 7)  "Gmean"
  2) 0x56492f5d2120 RooRealVar::     Gsigma = 0.206953  L(0.0001 - 3)  "Gsigma"
  3) 0x56492f5950d0 RooRealVar::         a0 = -0.0433  L(-1 - 1)  "a0"
  4) 0x56492f52e730 RooRealVar::         a1 = 0.0877073  L(-1 - 1)  "a1"
  5) 0x56492f5ca1a0 RooRealVar:: Yld_Signal = 109.655  L(0 - 1e+12)  "Yld_Signal"
  6) 0x56492f5ca9a0 RooRealVar::     Yld_BG = 171.461  L(0 - 1e+12)  "Yld_BG"

Gmean 5.96715 +- 0.0237717 sigma 0.0241361 meanPull 0.00223759 sigmaPull 1.02776
      bias -0.00190544 bias Pull -0.0790702 sigma 1.02767


Gsigma 0.207819 +- 0.0204853 sigma 0.0225982 meanPull -0.115595 sigmaPull 1.1047
      bias 0.00086601 bias Pull -0.0722243 sigma 1.10013


a0 -0.0390713 +- 0.132536 sigma 0.140173 meanPull -0.000150034 sigmaPull 1.06967
      bias 0.00422866 bias Pull 0.0318165 sigma 1.06967


a1 0.0901345 +- 0.132916 sigma 0.142643 meanPull 0.058276 sigmaPull 1.05851
      bias 0.00242745 bias Pull 0.0766501 sigma 1.05951
```

Get new ToyManager from previous fit

Run multicore on PROOF-lite
(or could use ROOT.Farm.Go(fit2))

Plot parameter distributions and pulls for the 400 toy fits



Speeding up likelihood calculation

Intrinsic RooFit optimisations implemented in RooNLLVar

- Variables not used by PDF are dropped
- PDF normalisation integrals only recalculated when range or parameter of that PDF changes
- Components with no or constant parameters are precalculated and cached
  → but you must construct your PDF carefully to benefit

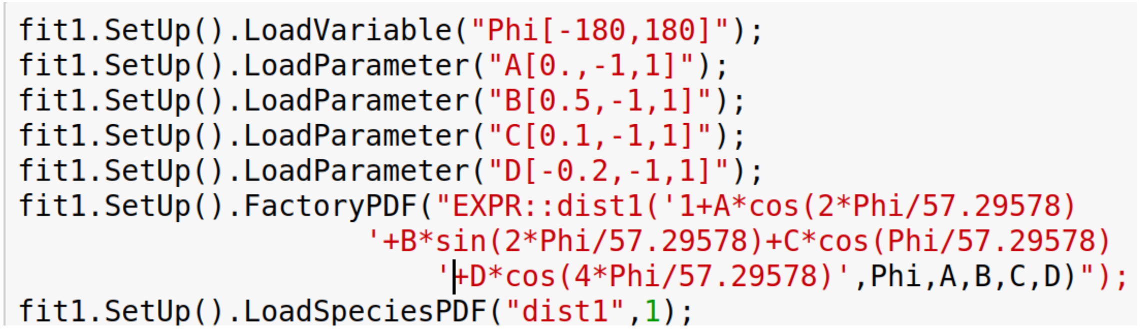
Fit with several paramteres but 1 component

Generate toy data (100k):

f(φ)=1 + 0.5*cos(2φ)

Fit with function :

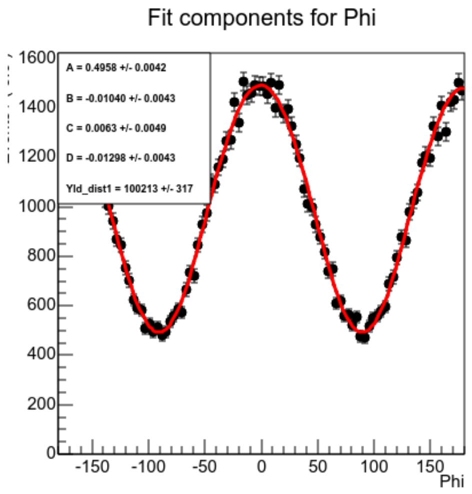
f(φ)=1 + A*cos(2φ)+ B*sin(2φ) + C*cos(φ) + D*cos(4φ)

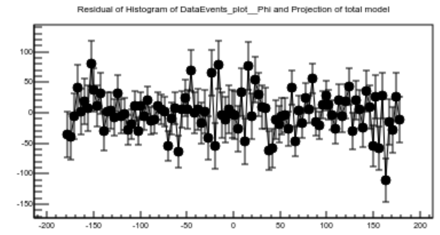


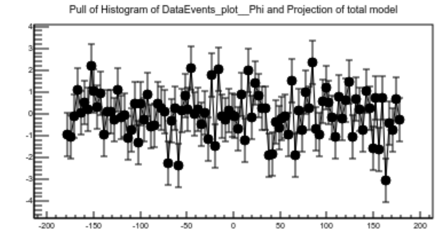
```
fit1.SetUp().LoadVariable("Phi[-180,180]");
fit1.SetUp().LoadParameter("A[0.,-1,1]");
fit1.SetUp().LoadParameter("B[0.5,-1,1]");
fit1.SetUp().LoadParameter("C[0.1,-1,1]");
fit1.SetUp().LoadParameter("D[-0.2,-1,1]");
fit1.SetUp().FactoryPDF("EXPR::dist1('1+A*cos(2*Phi/57.29578)
                            '+B*sin(2*Phi/57.29578)+C*cos(Phi/57.29578)
                               '+D*cos(4*Phi/57.29578)',Phi,A,B,C,D)");
fit1.SetUp().LoadSpeciesPDF("dist1",1);
```

Use RooFit Factory Expression Pdf, takes general formula string

Only has 1 component, so any time A,B,C,D change must recalculate for every event







This uses a numerical integration

Fit in 10.5 seconds

Fit with many components

Generate toy data (100k):

$f(\varphi)=1 + 0.5*\cos(2\varphi)$

Fit with function :

$f(\varphi)=1 + A*\cos(2\varphi)+ B*\sin(2\varphi)$
$+ C*\cos(\varphi) + D*\cos(4\varphi)$

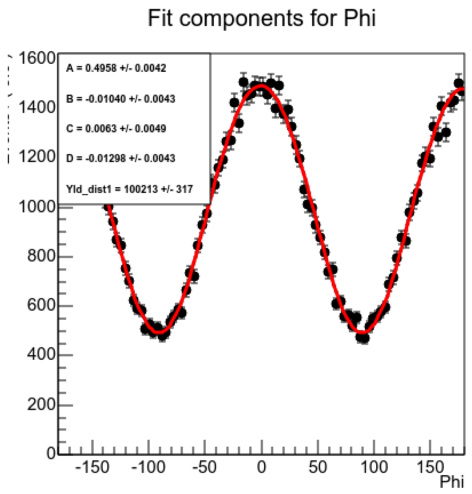
Use RooFit Factory Expression Pdf, takes general formula string

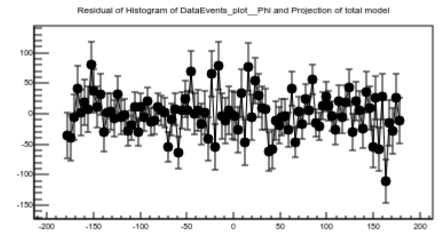
Define 4 sepearate terms which are Combined into 1 PDF

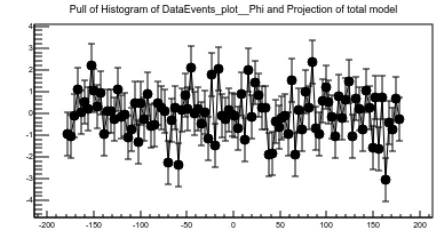
COS2,SIN2 etc do not depend on any parameters =>precalculated and cached

This uses a numerical integration

```
fit2.SetUp().LoadVariable("Phi[-180,180]");
fit2.SetUp().LoadParameter("A[0.,-1,1]");
fit2.SetUp().LoadParameter("B[0.5,-1,1]");
fit2.SetUp().LoadParameter("C[0.1,-1,1]");
fit2.SetUp().LoadParameter("D[-0.2,-1,1]");
fit2.SetUp().LoadFormula("COS2=cos(2*@Phi[]/57.29578)");
fit2.SetUp().LoadFormula("COS=cos(@Phi[]/57.29578)");
fit2.SetUp().LoadFormula("COS4=cos(4*@Phi[]/57.29578)");
fit2.SetUp().LoadFormula("SIN2=sin(2*@Phi[]/57.29578)");
fit2.SetUp().FactoryPDF("EXPR::dist2('1+A*COS2+B*SIN2-C*COS+D*COS4'
                        ,Phi,A,B,C,D,COS2,COS,COS4,SIN2)");
fit2.SetUp().LoadSpeciesPDF("dist2",1);
```







Fit in 4.5 seconds

Speeding up MC integrals

$$A(p)=\sum_{i=0}^{Nacc} f(\tau_i, p)$$

No factorisation, must recalculate every time one parameter changes

$$A(p)=\sum_{i=0}^{Nacc} P_0(p)*f_0(\tau_i) + P_1(p)*f_1(\tau_i) + P_2(p)*f_2(\tau_i) + \ldots$$

$$A(p)= P_0(p)\sum_{i=0}^{Nacc} f_0(\tau_i) + \ldots$$

Summations over $\boldsymbol{f_j}$s only needs done once and then cached = $\boldsymbol{F_j}$
i.e just 1 loop of MC events

Calculation of integral each step becomes very fast to calculate

$$A(p)=P_0(p)*F_0 + P_1(p)*F_1 + P_2(p)*F_2 + \ldots$$

Already done in dedicated Amplitude Analysis software

RooComponentsPDF

Inherits from RooHSEventsPDF

- loads MC integral events tree

Takes any number of parameters, functions, formula, …

- anything that is RooAbsReal

Sorts each component into parameter dependent/independent terms

Precalculates normalisation integral terms at start of fit

Can be used for any PDF that can be written as a sum of products with fit parameters and observable parts factorised :

$$P_0(p_a)*Q_0(p_b)*f_0(\tau_c)*g_0(\tau_d) + P_1(p_a)*Q_1(p_b)*f_1(\tau_c)*g_1(\tau_d) + \dots$$

Some subsets of $\boldsymbol{p}$

Some subsets of $\boldsymbol{\tau}$

Fit with RooComponentsPDF

Generate toy data (100k):

f(φ)=1 + 0.5*cos(2φ)

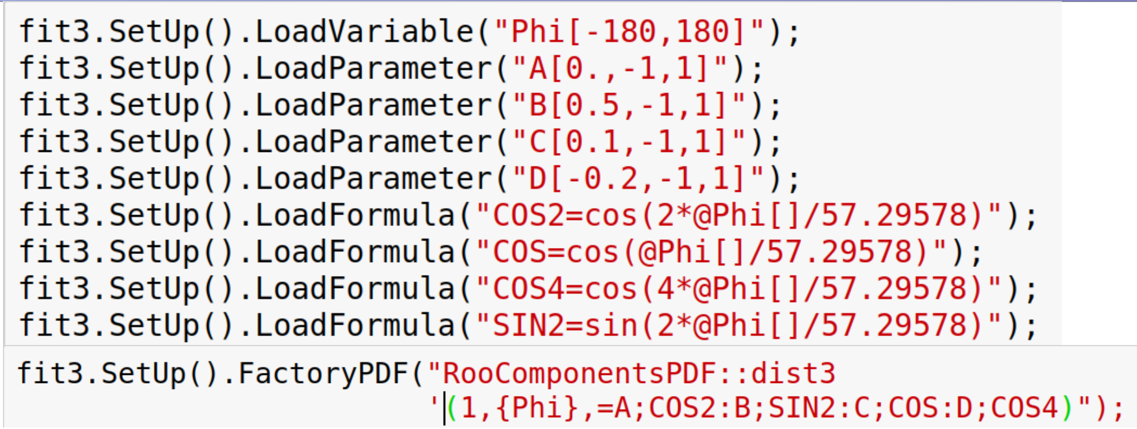
Fit with function :

f(φ)=1 + A*cos(2φ)+ B*sin(2φ)
+ C*cos(φ) + D*cos(4φ)

Use **RooComponentsPDF,**
give 100k MC events for integral

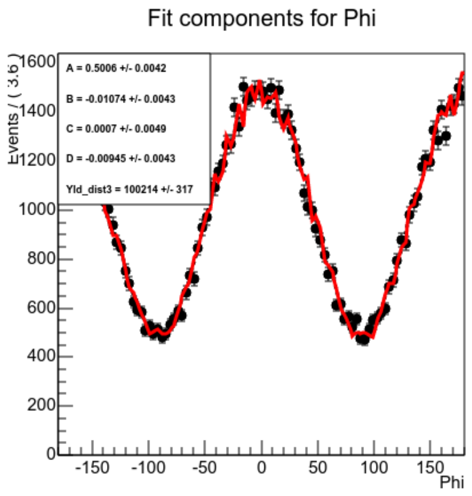
Define 4 sepearate terms which are
Combined into 1 PDF

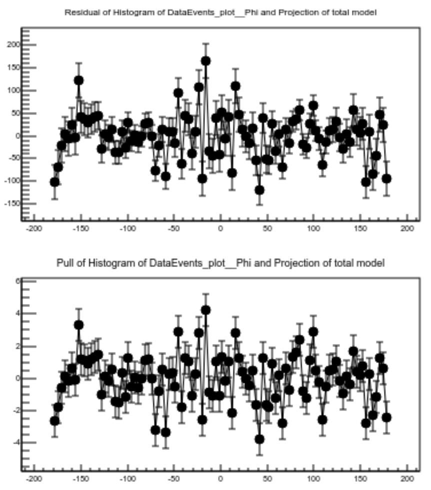
COS2,SIN2 etc do not depend on any
parameters =>precalculated and
cached



```
fit3.SetUp().LoadVariable("Phi[-180,180]");
fit3.SetUp().LoadParameter("A[0.,-1,1]");
fit3.SetUp().LoadParameter("B[0.5,-1,1]");
fit3.SetUp().LoadParameter("C[0.1,-1,1]");
fit3.SetUp().LoadParameter("D[-0.2,-1,1]");
fit3.SetUp().LoadFormula("COS2=cos(2*@Phi[]/57.29578)");
fit3.SetUp().LoadFormula("COS=cos(@Phi[]/57.29578)");
fit3.SetUp().LoadFormula("COS4=cos(4*@Phi[]/57.29578)");
fit3.SetUp().LoadFormula("SIN2=sin(2*@Phi[]/57.29578)");
```

```
fit3.SetUp().FactoryPDF("RooComponentsPDF::dist3
                          '(1,{Phi},=A;COS2:B;SIN2:C;COS:D;COS4)");
```





Fit in 2.2 seconds

Measuring hadron SDMEs

P. Pauli, Glasgow

Toy Example $\vec{\gamma}p \rightarrow XY^+$ with $X \rightarrow P^+P^-$

Use decay angles of X : θ, φ

Planar production angle relative to Linear polarisation Φ

Degree of linear polarisation P_γ

Fit function:

$$W_0 = \frac{1}{4\pi}\left[3\left(\frac{1}{2} - \rho^0_{11}\right)\sin^2(\theta) + \rho^0_{11}\left(1 + 3\cos^2(\theta)\right) - 2\sqrt{3}\left(\mathrm{Re}(\rho^0_{31})\cos(\varphi)\sin(2\theta) + \mathrm{Re}(\rho^0_{3-1})\cos(2\varphi)\sin^2(\theta)\right)\right]$$

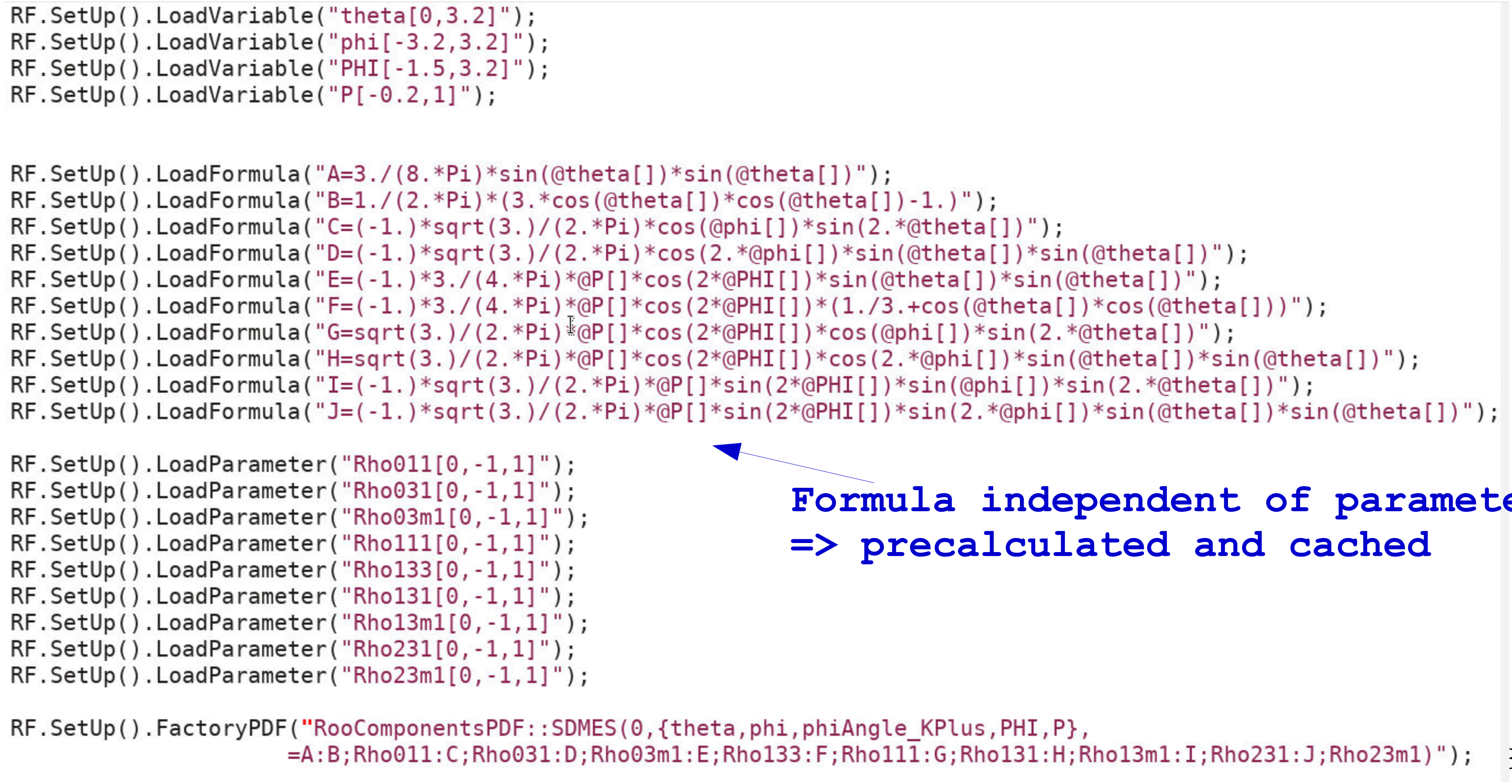
$$W_1 = \frac{1}{4\pi}\left[3\rho^1_{33}\sin^2(\theta) + \rho^1_{11}(1 + 3\cos^2(\theta)) - 2\sqrt{3}\left(\mathrm{Re}(\rho^1_{31})\cos(\varphi)\sin(2\theta) + \mathrm{Re}(\rho^1_{3-1})\cos(2\varphi)\sin^2(\theta)\right)\right]$$

$$W_2 = \frac{1}{4\pi}\left[2\sqrt{3}\left(\mathrm{Im}(\rho^2_{31})\sin(\varphi)\sin(2\theta) + \mathrm{Im}(\rho^2_{3-1})\sin(2\varphi)\sin^2(\theta)\right)\right]$$

$$\boxed{W = W_0 - P_\gamma\cos(2\Phi)W_1 - P_\gamma\sin(2\Phi)W_2}$$

with Φ being the angle between production plane and polarisation plane.

SDME code



```
RF.SetUp().LoadVariable("theta[0,3.2]");
RF.SetUp().LoadVariable("phi[-3.2,3.2]");
RF.SetUp().LoadVariable("PHI[-1.5,3.2]");
RF.SetUp().LoadVariable("P[-0.2,1]");


RF.SetUp().LoadFormula("A=3./(8.*Pi)*sin(@theta[])*sin(@theta[])");
RF.SetUp().LoadFormula("B=1./(2.*Pi)*(3.*cos(@theta[])*cos(@theta[])-1.)");
RF.SetUp().LoadFormula("C=(-1.)*sqrt(3.)/(2.*Pi)*cos(@phi[])*sin(2.*@theta[])");
RF.SetUp().LoadFormula("D=(-1.)*sqrt(3.)/(2.*Pi)*cos(2.*@phi[])*sin(@theta[])*sin(@theta[])");
RF.SetUp().LoadFormula("E=(-1.)*3./(4.*Pi)*@P[]*cos(2*@PHI[])*sin(@theta[])*sin(@theta[])");
RF.SetUp().LoadFormula("F=(-1.)*3./(4.*Pi)*@P[]*cos(2*@PHI[])*(1./3.+cos(@theta[])*cos(@theta[]))");
RF.SetUp().LoadFormula("G=sqrt(3.)/(2.*Pi)*@P[]*cos(2*@PHI[])*cos(@phi[])*sin(2.*@theta[])");
RF.SetUp().LoadFormula("H=sqrt(3.)/(2.*Pi)*@P[]*cos(2*@PHI[])*cos(2.*@phi[])*sin(@theta[])*sin(@theta[])");
RF.SetUp().LoadFormula("I=(-1.)*sqrt(3.)/(2.*Pi)*@P[]*sin(2*@PHI[])*sin(@phi[])*sin(2.*@theta[])");
RF.SetUp().LoadFormula("J=(-1.)*sqrt(3.)/(2.*Pi)*@P[]*sin(2*@PHI[])*sin(2.*@phi[])*sin(@theta[])*sin(@theta[])");

RF.SetUp().LoadParameter("Rho011[0,-1,1]");
RF.SetUp().LoadParameter("Rho031[0,-1,1]");
RF.SetUp().LoadParameter("Rho03m1[0,-1,1]");
RF.SetUp().LoadParameter("Rho111[0,-1,1]");
RF.SetUp().LoadParameter("Rho133[0,-1,1]");
RF.SetUp().LoadParameter("Rho131[0,-1,1]");
RF.SetUp().LoadParameter("Rho13m1[0,-1,1]");
RF.SetUp().LoadParameter("Rho231[0,-1,1]");
RF.SetUp().LoadParameter("Rho23m1[0,-1,1]");

RF.SetUp().FactoryPDF("RooComponentsPDF::SDMES(0,{theta,phi,phiAngle_KPlus,PHI,P},
                     =A:B;Rho011:C;Rho031:D;Rho03m1:E;Rho133:F;Rho111:G;Rho131:H;Rho13m1:I;Rho231:J;Rho23m1)");
```

Formula independent of parameters
=> precalculated and cached

RooComponentsPDF used for fast integration

SDME Pseudo data Fits

Acceptance : Holes in Lab θ
Reduced acceptance in decay θ

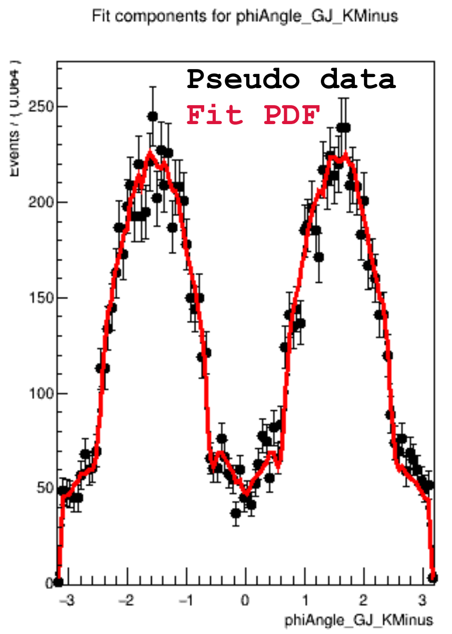
Number data events =13k MC Integral 100k

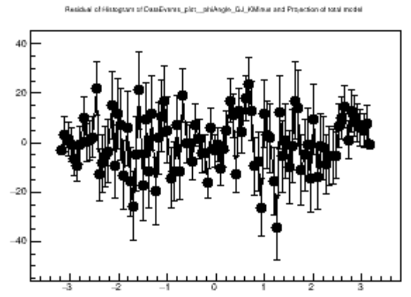
Run MCMC for 5000 posterior points (efficiency was ~3%)

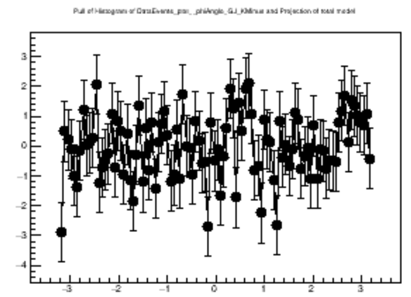
Old Time = 64,000 s RooComponentsPDF = 1,400 s ~factor 50 speed-up

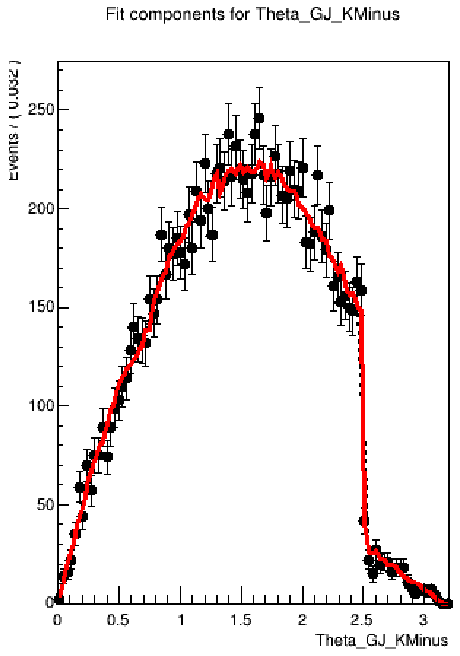
~23% is optimal so
Could be ~7x faster
Just change step size!

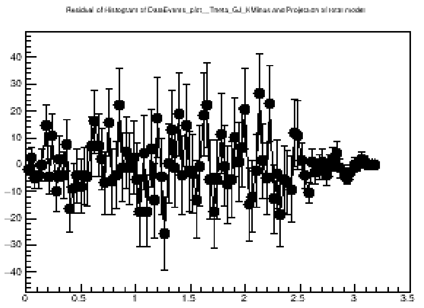
Data generated with random SDME values

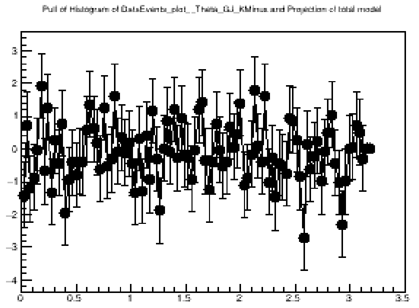








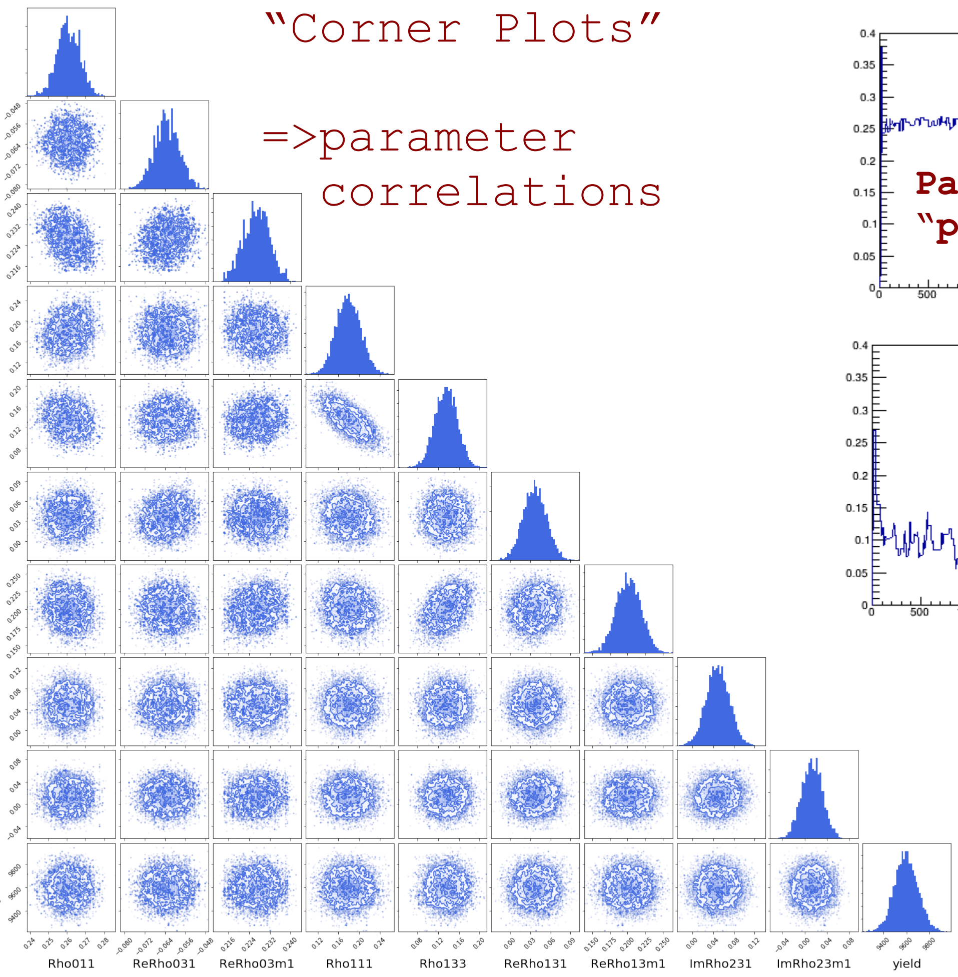


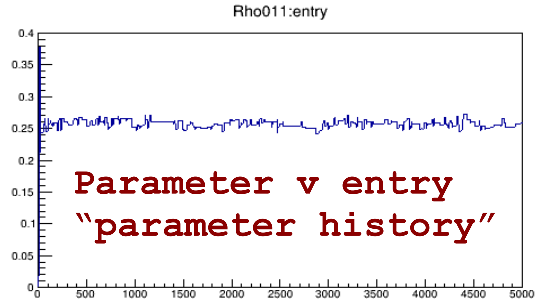


SDME MCMC diagnostics

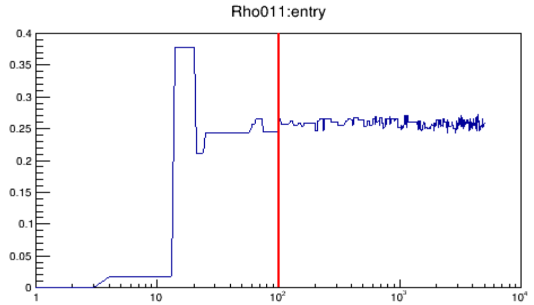
"Corner Plots"

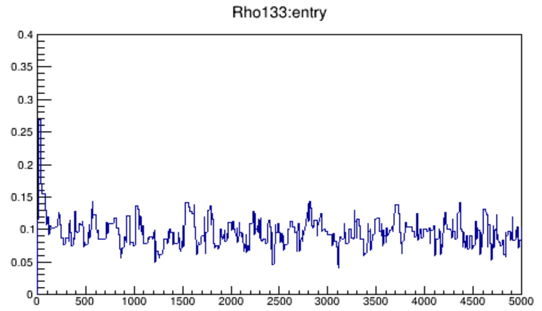
=>parameter correlations

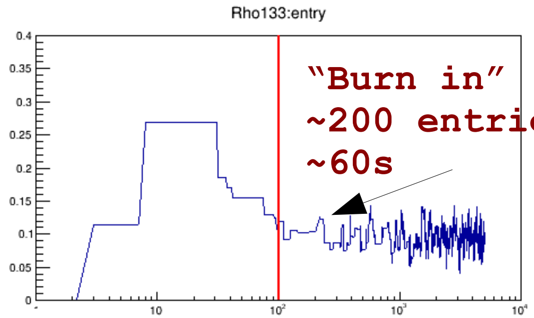




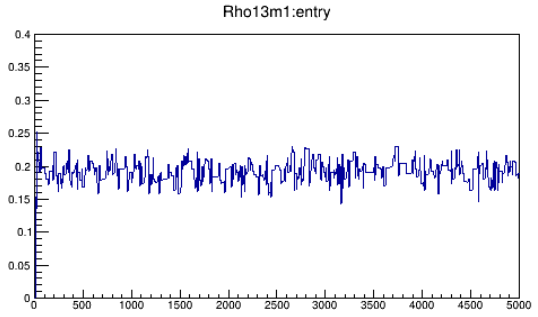
Parameter v entry "parameter history"

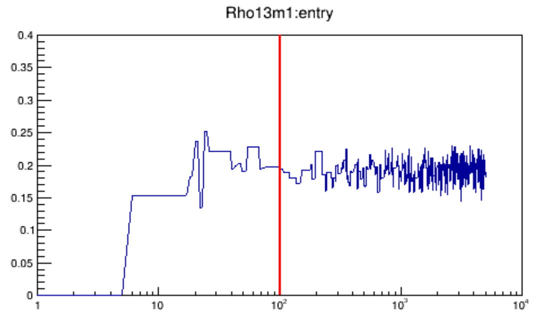






"Burn in" ~200 entries ~60s

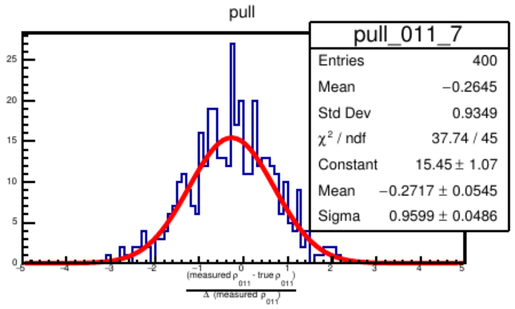




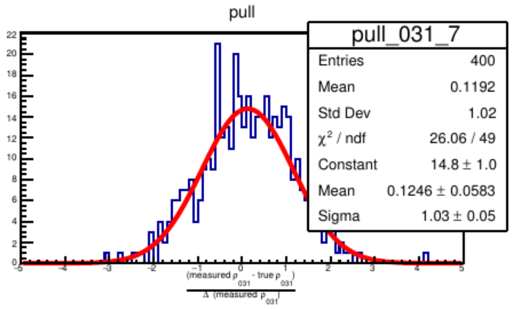
MCMC entry #

Log x scale

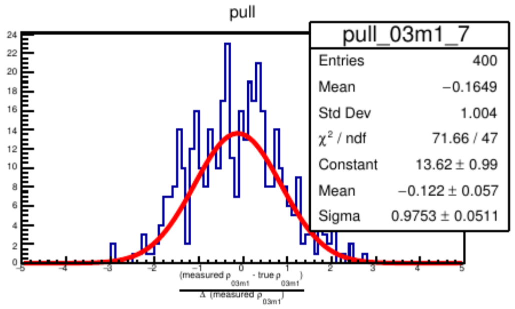
SDME Generate and MCMC 400 Toydatasets - Look for bias, correct uncertainties



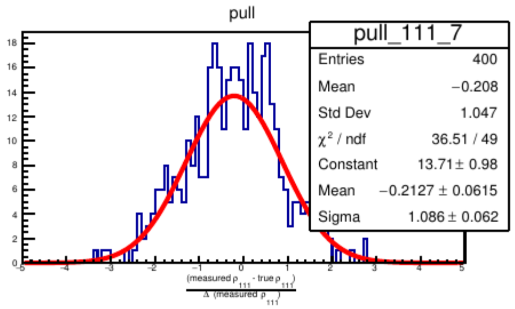
pull_011_7	
Entries	400
Mean	−0.2645
Std Dev	0.9349
χ^2 / ndf	37.74 / 45
Constant	15.45 ± 1.07
Mean	−0.2717 ± 0.0545
Sigma	0.9599 ± 0.0486



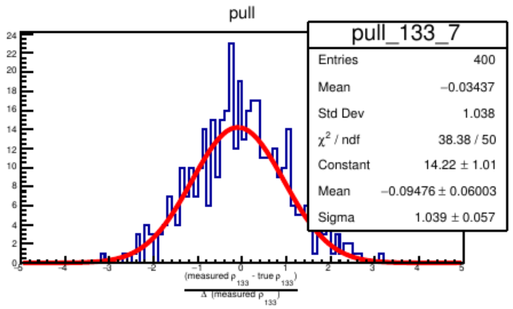
pull_031_7	
Entries	400
Mean	0.1192
Std Dev	1.02
χ^2 / ndf	26.06 / 49
Constant	14.8 ± 1.0
Mean	0.1246 ± 0.0583
Sigma	1.03 ± 0.05



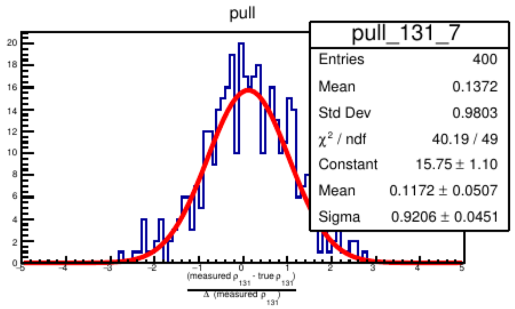
pull_03m1_7	
Entries	400
Mean	−0.1649
Std Dev	1.004
χ^2 / ndf	71.66 / 47
Constant	13.62 ± 0.99
Mean	−0.122 ± 0.057
Sigma	0.9753 ± 0.0511



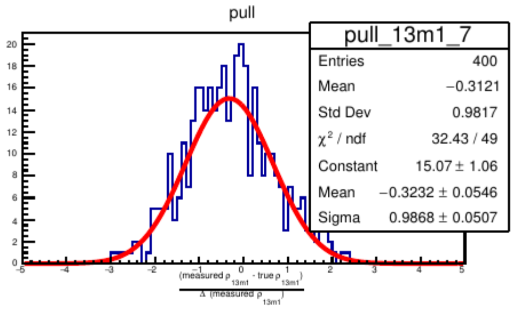
pull_111_7	
Entries	400
Mean	−0.208
Std Dev	1.047
χ^2 / ndf	36.51 / 49
Constant	13.71 ± 0.98
Mean	−0.2127 ± 0.0615
Sigma	1.086 ± 0.062



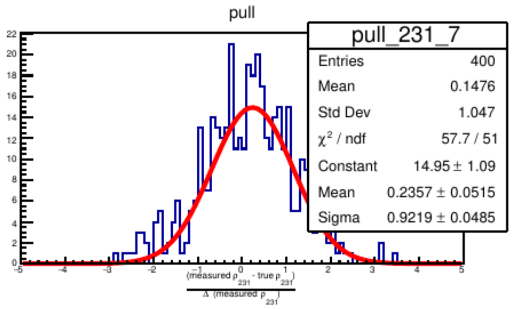
pull_133_7	
Entries	400
Mean	−0.03437
Std Dev	1.038
χ^2 / ndf	38.38 / 50
Constant	14.22 ± 1.01
Mean	−0.09476 ± 0.06003
Sigma	1.039 ± 0.057



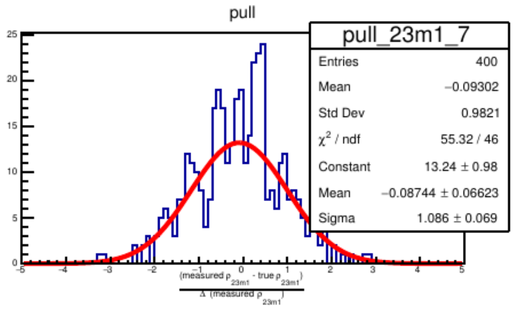
pull_131_7	
Entries	400
Mean	0.1372
Std Dev	0.9803
χ^2 / ndf	40.19 / 49
Constant	15.75 ± 1.10
Mean	0.1172 ± 0.0507
Sigma	0.9206 ± 0.0451



pull_13m1_7	
Entries	400
Mean	−0.3121
Std Dev	0.9817
χ^2 / ndf	32.43 / 49
Constant	15.07 ± 1.06
Mean	−0.3232 ± 0.0546
Sigma	0.9868 ± 0.0507

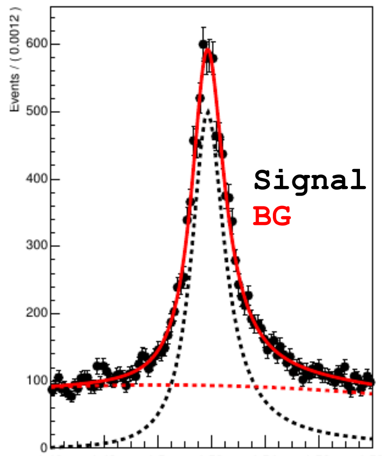


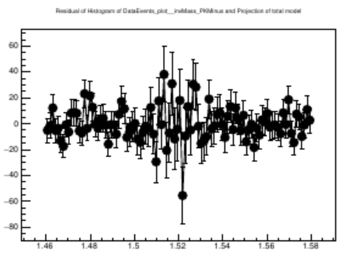
pull_231_7	
Entries	400
Mean	0.1476
Std Dev	1.047
χ^2 / ndf	57.7 / 51
Constant	14.95 ± 1.09
Mean	0.2357 ± 0.0515
Sigma	0.9219 ± 0.0485

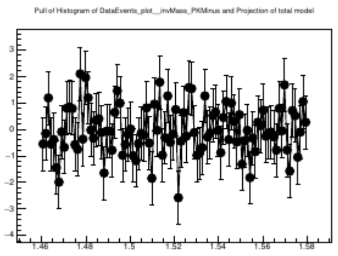


pull_23m1_7	
Entries	400
Mean	−0.09302
Std Dev	0.9821
χ^2 / ndf	55.32 / 46
Constant	13.24 ± 0.98
Mean	−0.08744 ± 0.06623
Sigma	1.086 ± 0.069

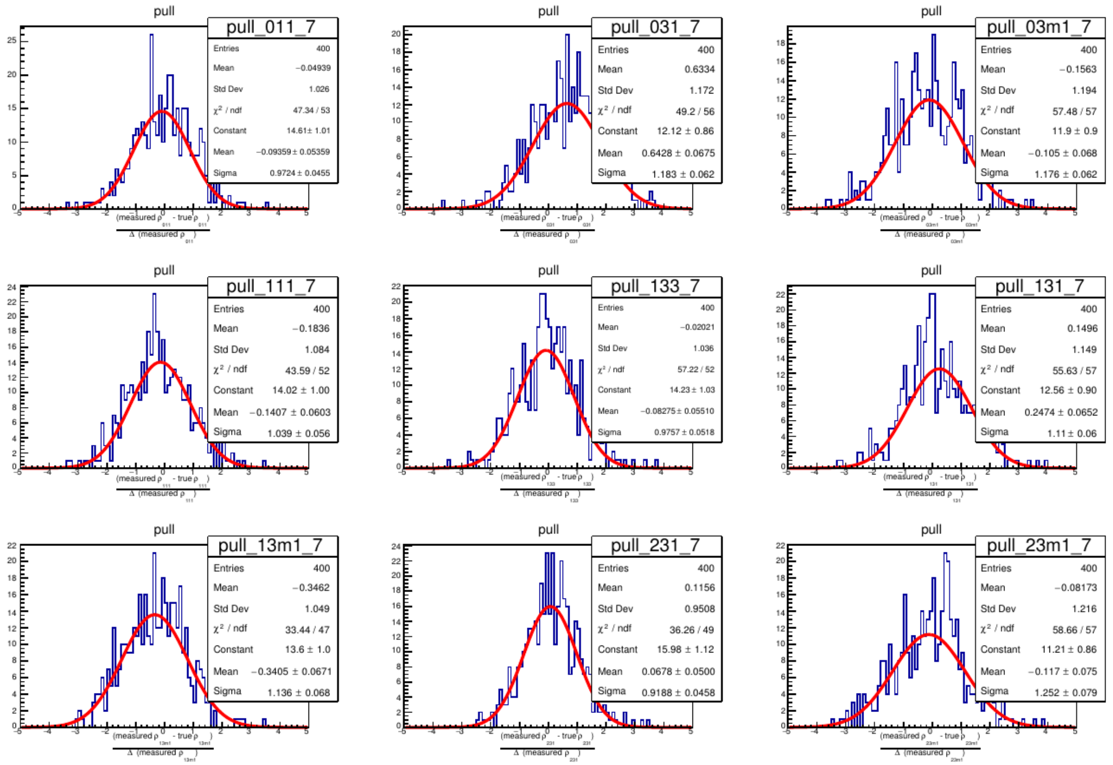
SDME Generate and MCMC 400 Toydatasets - Include background and sWeights







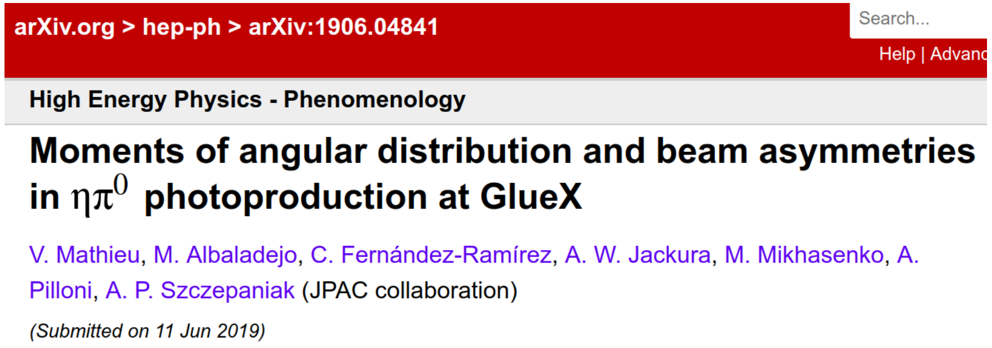
MCMC more robust than Minuit with weights



Fit to pseudo discriminatory variable
i.e. different PDF for
sig and BG
Use weights in toy fits
Look for distortions due to background subtraction

Would like this to be standard procedure

JPAC, Two Meson Photoproduction : Moments



arXiv.org > hep-ph > arXiv:1906.04841

Search...

Help | Advanc

High Energy Physics - Phenomenology

Moments of angular distribution and beam asymmetries in $\eta\pi^0$ photoproduction at GlueX

V. Mathieu, M. Albaladejo, C. Fernández-Ramírez, A. W. Jackura, M. Mikhasenko, A. Pilloni, A. P. Szczepaniak (JPAC collaboration)

(Submitted on 11 Jun 2019)

Directly relate Y^M_L moments to partial waves :

$$H^0(00) = H^1(00) + 2\left[|P_1^{(+)}|^2 + |D_1^{(+)}|^2 + |D_2^{(+)}|^2\right] ,$$

$$H^1(00) = 2\left[|S_0^{(+)}|^2 + |P_0^{(+)}|^2 + |D_0^{(+)}|^2\right] ,$$

$$H^0(10) = H^1(10) + \frac{4}{\sqrt{5}}\,\mathrm{Re}(P_1^{(+)} D_1^{(+)*}) ,$$

$$H^1(10) = \frac{4}{5\sqrt{3}}\left[2\sqrt{5}\,\mathrm{Re}(P_0^{(+)} D_0^{(+)*}) + 5\,\mathrm{Re}(S_0^{(+)} P_0^{(+)*})\right] ,$$

$$I(\Omega,\Phi) = I^0(\Omega) - P_\gamma I^1(\Omega)\cos(2\Phi) - P_\gamma I^2(\Omega)\sin(2\Phi)$$

$$I^0(\Omega) = \sum_L \sum_{M=0}^{M\le L} \sqrt{\left(\frac{2L+1}{4\pi}\right)}(2-\delta_{M,0})\,H^0(L,M)\,\Re[Y_L^M(\Omega)]$$

$$I^1(\Omega) = -\sum_L \sum_{M=0}^{M\le L} \sqrt{\left(\frac{2L+1}{4\pi}\right)}(2-\delta_{M,0})\,H^1(L,M)\,\Re[Y_L^M(\Omega)]$$

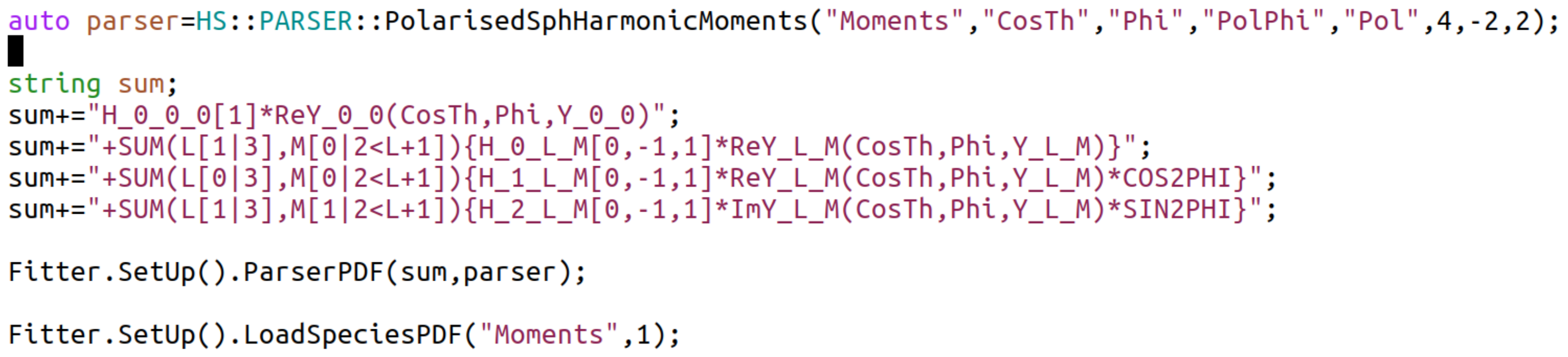
$$I^2(\Omega) = 2\sum_L \sum_{M=0}^{M\le L} \sqrt{\left(\frac{2L+1}{4\pi}\right)}\,\Im[H^2(L,M)]\,\Im[Y_L^M(\Omega)]$$

Fit Function
= sum of products
=>RooComponentsPDF

Just requires RooHSSphHarmonic class

Experimental : Parsers

Parsers → Create string for configuring RooComponentsPDF



```
auto parser=HS::PARSER::PolarisedSphHarmonicMoments("Moments","CosTh","Phi","PolPhi","Pol",4,-2,2);

string sum;
sum+="H_0_0_0[1]*ReY_0_0(CosTh,Phi,Y_0_0)";
sum+="+SUM(L[1|3],M[0|2<L+1]){H_0_L_M[0,-1,1]*ReY_L_M(CosTh,Phi,Y_L_M)}";
sum+="+SUM(L[0|3],M[0|2<L+1]){H_1_L_M[0,-1,1]*ReY_L_M(CosTh,Phi,Y_L_M)*COS2PHI}";
sum+="+SUM(L[1|3],M[1|2<L+1]){H_2_L_M[0,-1,1]*ImY_L_M(CosTh,Phi,Y_L_M)*SIN2PHI}";

Fitter.SetUp().ParserPDF(sum,parser);

Fitter.SetUp().LoadSpeciesPDF("Moments",1);
```

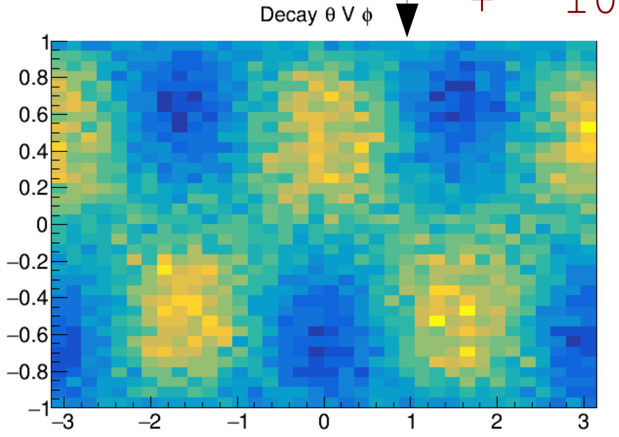
Becomes function with 22 $H^i(L,M)$ fit parameters,

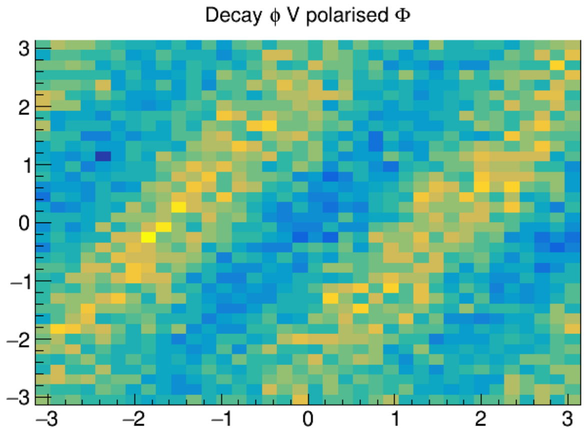
Objects of RooHSSphHarmonicRe

**RooComponentsPDF::Moments(0,{CosTh,Phi,PolPhi,Pol},=H_0_0_0[1];ReY_0_0:
H_0_1_0[0,-1,1];ReY_1_0:H_0_1_1[0,-1,1];ReY_1_1:H_0_2_0[0,-1,1];ReY_2_0:
H_0_2_1[0,-1,1];ReY_2_1:H_0_2_2[0,-1,1]; ReY_2_2:H_0_3_0[0,-1,1];ReY_3_0:
H_0_3_1[0,-1,1];ReY_3_1:H_0_3_2[0,-1,1];ReY_3_2:H_1_0_0[0,-1,1]; ReY_0_0;COS2PHI:
H_1_1_0[0,-1,1];ReY_1_0;COS2PHI:H_1_1_1[0,-1,1];ReY_1_1;COS2PHI:
H_1_2_0[0,1,1];ReY_2_0;COS2PHI:H_1_2_1[0,-1,1];ReY_2_1;COS2PHI:
H_1_2_2[0,-1,1];ReY_2_2;COS2PHI:H_1_3_0[0,-1,1]; ReY_3_0;COS2PHI:
H_1_3_1[0,1,1];ReY_3_1;COS2PHI:H_1_3_2[0,1,1];ReY_3_2;COS2PHI:
H_2_1_1[0,-1,1];ImY_1_1;SIN2PHI:H_2_2_1[0,-1,1];ImY_2_1;SIN2PHI:
H_2_2_2[0,-1,1];ImY_2_2;SIN2PHI:H_2_3_1[0,1,1];ImY_3_1;SIN2PHI:H_2_3_2[0,-1,1];ImY_3_2;SIN2PHI)**

Fit Pseudo Data with Moments PDF

Pseudo Data 100K events
+ 100K MC events





L<4,M<3 22 paramters

Fit with Minuit

→ 40s

Fit with MCMC

2000 samples

→ 260s 18%accept

L<5,M<4 36 params.

Fit with Minuit

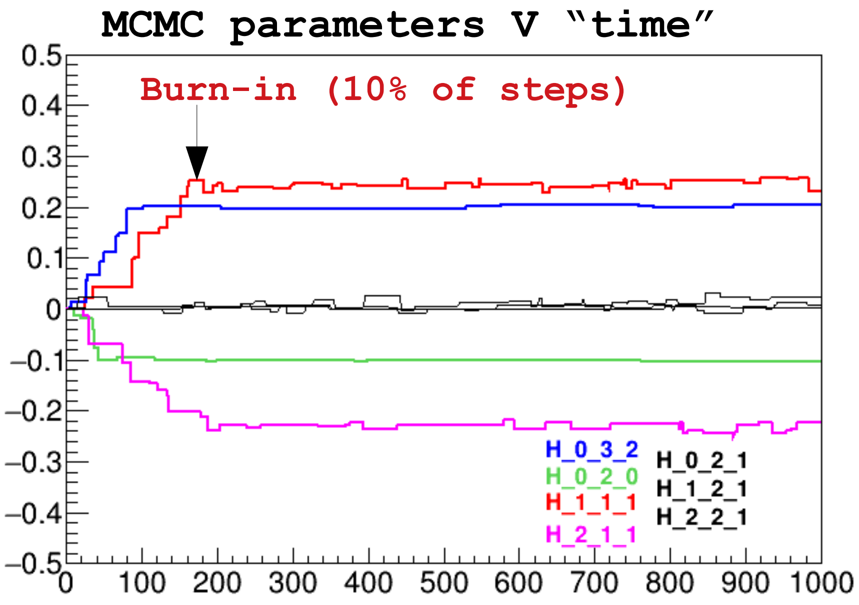
→ 130s

Fit with MCMC

2000 samples

→ 360s 18%accept

MCMC scales better with parameter #



Proof of principal : Fitting Amplitudes

RooFit does not directly support complex numbers

- Try expanding intensities with amplitudes so only deal with real valued numbers and functions

$$\left(\sum AB\right)^2 = \sum_i F(A_i,A_i)F(B_i,B_i)+2\sum_i \sum_j^{i<j}[F(A_i,A_j)F(B_i,B_j)-G(A_i,A_j)G(B_i,B_j)]$$

$$F(C,D)=\Re(C)\Re(D)+\Im(C)\Im(D) \quad G(C,D)=\Im(C)\Re(D)-\Re(C)\Im(D)$$

Define RooFit function for F and G which take 4 real numbers

These numbers can be real and imaginery parts of complex parameters or functions (e.g. Y^M_L)

$$\left(\sum h_L^M Y_L^M\right)^2 = F(h_0^0,h_0^0)F(Y_0^0,Y_0^0) + F(h_1^0,h_1^0)F(Y_1^0,Y_1^0)\ldots$$
$$+ 2F(h_0^0,h_1^0)F(Y_0^0,Y_1^0) - 2G(h_0^0,h_1^0)G(Y_0^0,Y_1^0) +\ldots$$

Sum of products of F and G functions → RooComponentsPDF

JPAC, Two Meson Photoproduction: Amps

Amplitudes given as

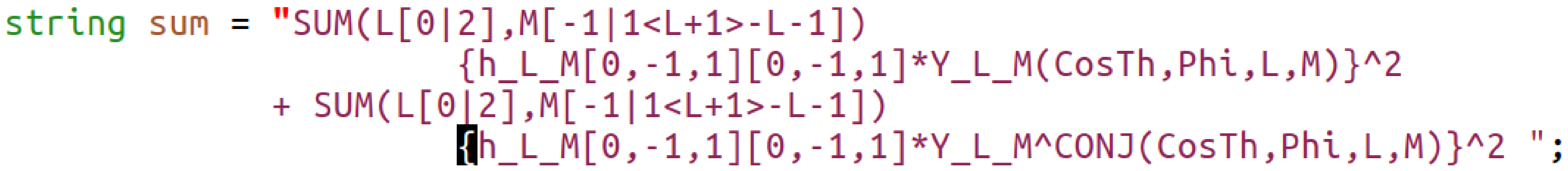
$$U_k^{(\epsilon)}(\Omega) = \sum_{\ell,m} [\ell]^{(\epsilon)}_{m;k} Y_\ell^m(\Omega) \ ,$$

$$\widetilde{U}_k^{(\epsilon)}(\Omega) = \sum_{\ell,m} [\ell]^{(\epsilon)}_{m;k} \left[Y_\ell^m(\Omega)\right]^* \ .$$

$$I^0(\Omega) = \kappa \sum_{\epsilon.k} |U_k^{(\epsilon)}(\Omega)|^2 + |\widetilde{U}_k^{(\epsilon)}(\Omega)|^2 \ ,$$

Use “Parser” to code as

L<2 M=-1,0,1 <L

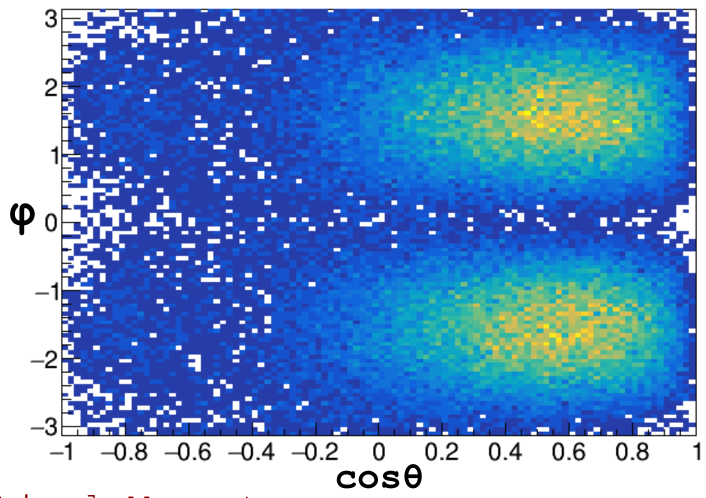


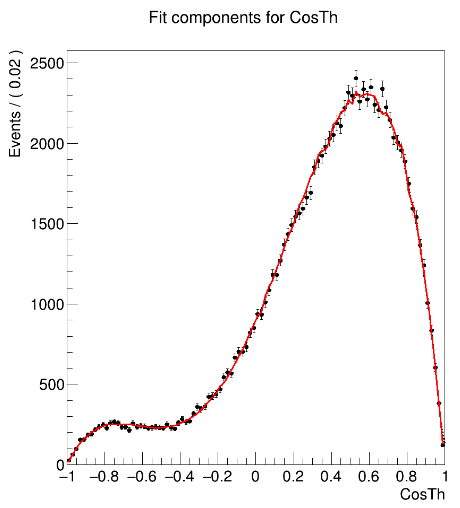
```
string sum = "SUM(L[0|2],M[-1|1<L+1>-L-1])
                 {h_L_M[0,-1,1][0,-1,1]*Y_L_M(CosTh,Phi,L,M)}^2
             + SUM(L[0|2],M[-1|1<L+1>-L-1])
                 {h_L_M[0,-1,1][0,-1,1]*Y_L_M^CONJ(CosTh,Phi,L,M)}^2 ";
```

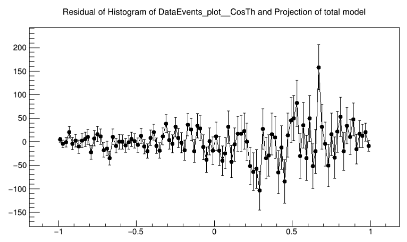
Real and imaginery fit parameters

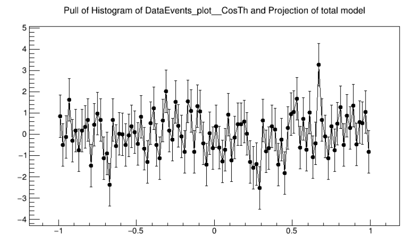
Complex spherical harmonics

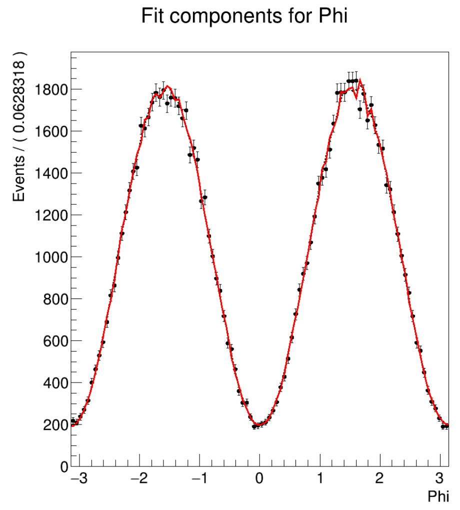
Pseudo data, Spherical Harmonic Moments Fit to data generated with amplitudes

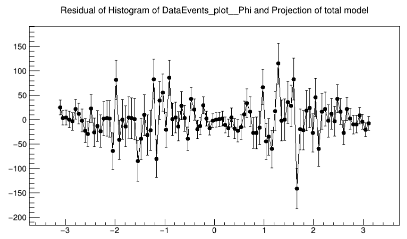


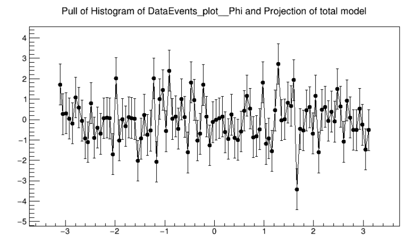






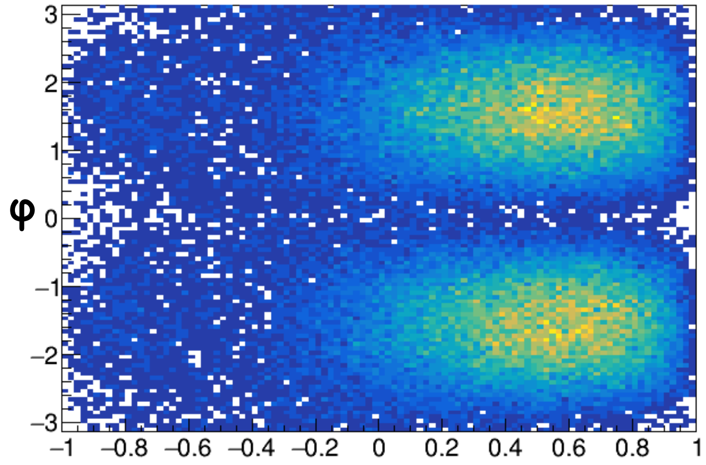




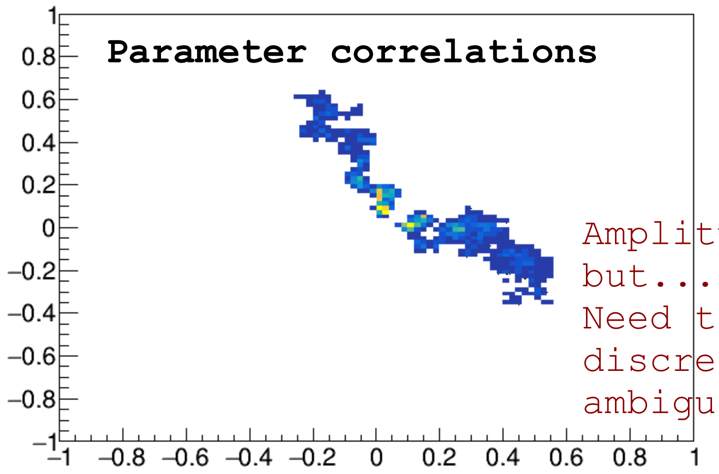


Analytical Moments from Amplitudes	Fit Results
(L,M) = (1,0): H0 = 0.3578 ;	H_0_1_0 = 0.358112 +/- 0.00797507
(L,M) = (1,1): H0 = 0.0000 ;	H_0_1_1 = -0.000740184 +/- 0.000722244
(L,M) = (2,0): H0 = -0.0629 ;	H_0_2_0 = -0.0623866 +/- 0.000474004
(L,M) = (2,1): H0 = 0.0000 ;	H_0_2_1 = -0.000551358 +/- 0.000478794
(L,M) = (2,2): H0 = -0.1680 ;	H_0_2_2 = -0.169541 +/- 0.000885905
(L,M) = (3,0): H0 = -0.1533 ;	H_0_3_0 = -0.153232 +/- 0.00448687
(L,M) = (3,1): H0 = 0.0000	H_0_3_1 = -0.000258408 +/- 0.000334288
(L,M) = (3,2): H0 = -0.1400 ;	H_0_3_2 = -0.140019 +/- 0.00137683
(L,M) = (3,3): H0 = 0.0000 ;	H_0_3_3 = 0.000338522 +/- 0.000556094
(L,M) = (4,0): H0 = -0.0762 ;	H_0_4_0 = -0.0765479 +/- 0.000712912
(L,M) = (4,1): H0 = 0.0000 ;	H_0_4_1 = 0.00024877 +/- 0.00030687
(L,M) = (4,2): H0 = -0.0602 ;	H_0_4_2 = -0.0605708 +/- 0.000410417
(L,M) = (4,3): H0 = 0.0000 ;	H_0_4_3 = -0.000109336 +/- 0.000434857
(L,M) = (4,4): H0 = 0.0000 ;	H_0_4_4 = 0.000138865 +/- 0.000480976

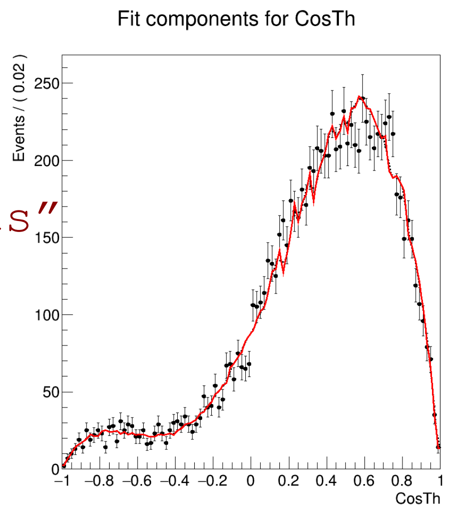
Pseudo data, Amplitudes Fit to data generated with amplitudes

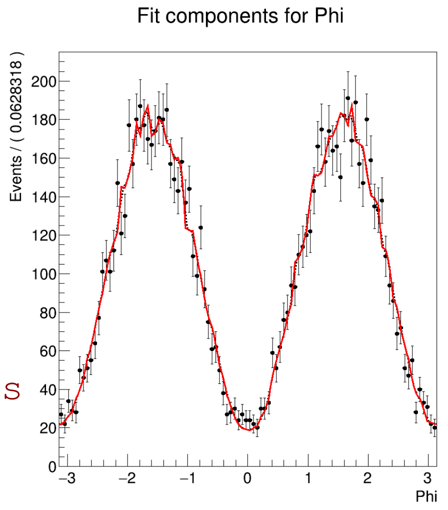


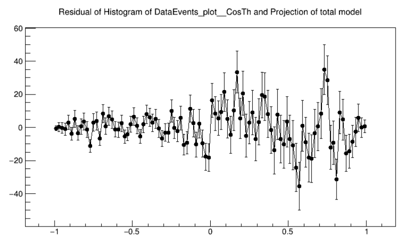
MCMC "Fits"

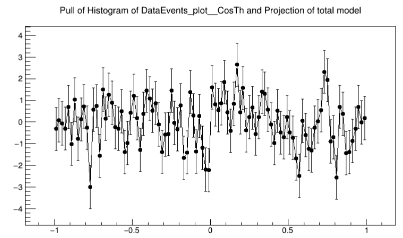


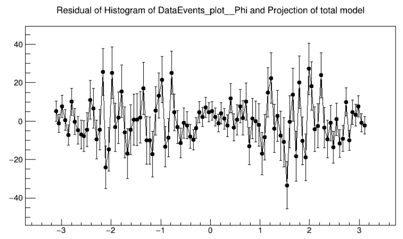
Amplitude fit works, but...
Need to account for discrete and continous ambiguities

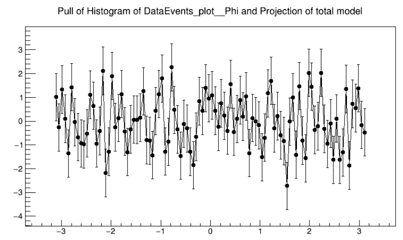












Summary

We are developing general data fitting tools based on RooFit

Extend RooFit by adding PDF class with normalisation integral calculated from simulated MC events

Extend this by adding further PDF with cached MC integrals

Fits can be performed via “user friendly” Juptyer notebooks

Applying ToyMC studies to fits with signal and background is a good way to investigate systematic effects in the parameter extraction

Tools will be used with MesonEx experiment

Already being applied to CLAS, GLUEX, CB@MAMI data

```
1_Y_2_0:_CSST3_h_2_-1_h_2_1;_CSST3_Y_2_-1_Y_2_1:_CSST3_h_2_0_h_0_0;_CSST3_Y_2_0_Y_0_0:_CSST3_h_2_0_h_1_-1;_CSST3_Y_2_0_Y_1_-
1:_CSST3_h_2_0_h_1_0;_CSST3_Y_2_0_Y_1_0:_CSST3_h_2_0_h_1_1;_CSST3_Y_2_0_Y_1_1:_CSST3_h_2_0_h_2_-1;_CSST3_Y_2_0_Y_2_-
1:_CSST3_h_2_0_h_2_1;_CSST3_Y_2_0_Y_2_1:_CSST3_h_2_1_h_0_0;_CSST3_Y_2_1_Y_0_0:_CSST3_h_2_1_h_1_-1;_CSST3_Y_2_1_Y_1_-
1:_CSST3_h_2_1_h_1_0;_CSST3_Y_2_1_Y_1_0:_CSST3_h_2_1_h_1_1;_CSST3_Y_2_1_Y_1_1:_CSST3_h_2_1_h_2_-1;_CSST3_Y_2_1_Y_2_-
1:_CSST3_h_2_1_h_2_0;_CSST3_Y_2_1_Y_2_0:_CSST_h_0_0_h_0_0;_CSST_Y_0_0_Y_0_0_CONJ:_CSST_h_0_0_h_1_-1;_CSST_Y_0_0_Y_1_-
1_CONJ:_CSST_h_0_0_h_1_0;_CSST_Y_0_0_Y_1_0_CONJ:_CSST_h_0_0_h_1_1;_CSST_Y_0_0_Y_1_1_CONJ:_CSST_h_0_0_h_2_-1;_CSST_Y_0_0_Y_2_-
1_CONJ:_CSST_h_0_0_h_2_0;_CSST_Y_0_0_Y_2_0_CONJ:_CSST_h_0_0_h_2_1;_CSST_Y_0_0_Y_2_1_CONJ:_CSST_h_1_-1_h_0_0;_CSST_Y_1_-1_Y_0_0_CONJ:_CSST_h_1_-
1_h_1_-1;_CSST_Y_1_-1_Y_1_-1_CONJ:_CSST_h_1_-1_h_1_0;_CSST_Y_1_-1_Y_1_0_CONJ:_CSST_h_1_-1_h_1_1;_CSST_Y_1_-1_Y_1_1_CONJ:_CSST_h_1_-1_h_2_-
1;_CSST_Y_1_-1_Y_2_-1_CONJ:_CSST_h_1_-1_h_2_0;_CSST_Y_1_-1_Y_2_0_CONJ:_CSST_h_1_-1_h_2_1;_CSST_Y_1_-
1_Y_2_1_CONJ:_CSST_h_1_0_h_0_0;_CSST_Y_1_0_Y_0_0_CONJ:_CSST_h_1_0_h_1_-1;_CSST_Y_1_0_Y_1_-
1_CONJ:_CSST_h_1_0_h_1_0;_CSST_Y_1_0_Y_1_0_CONJ:_CSST_h_1_0_h_1_1;_CSST_Y_1_0_Y_1_1_CONJ:_CSST_h_1_0_h_2_-1;_CSST_Y_1_0_Y_2_-
1_CONJ:_CSST_h_1_0_h_2_0;_CSST_Y_1_0_Y_2_0_CONJ:_CSST_h_1_0_h_2_1;_CSST_Y_1_0_Y_2_1_CONJ:_CSST_h_1_1_h_0_0;_CSST_Y_1_1_Y_0_0_CONJ:_CSST_h_1_1_h_1_
-1;_CSST_Y_1_1_Y_1_-1_CONJ:_CSST_h_1_1_h_1_0;_CSST_Y_1_1_Y_1_0_CONJ:_CSST_h_1_1_h_1_1;_CSST_Y_1_1_Y_1_1_CONJ:_CSST_h_1_1_h_2_-1;_CSST_Y_1_1_Y_2_-
1_CONJ:_CSST_h_1_1_h_2_0;_CSST_Y_1_1_Y_2_0_CONJ:_CSST_h_1_1_h_2_1;_CSST_Y_1_1_Y_2_1_CONJ:_CSST_h_2_-1_h_0_0;_CSST_Y_2_-1_Y_0_0_CONJ:_CSST_h_2_-
1_h_1_-1;_CSST_Y_2_-1_Y_1_-1_CONJ:_CSST_h_2_-1_h_1_0;_CSST_Y_2_-1_Y_1_0_CONJ:_CSST_h_2_-1_h_1_1;_CSST_Y_2_-1_Y_1_1_CONJ:_CSST_h_2_-1_h_2_-
1;_CSST_Y_2_-1_Y_2_-1_CONJ:_CSST_h_2_-1_h_2_0;_CSST_Y_2_-1_Y_2_0_CONJ:_CSST_h_2_-1_h_2_1;_CSST_Y_2_-
1_Y_2_1_CONJ:_CSST_h_2_0_h_0_0;_CSST_Y_2_0_Y_0_0_CONJ:_CSST_h_2_0_h_1_-1;_CSST_Y_2_0_Y_1_-
1_CONJ:_CSST_h_2_0_h_1_0;_CSST_Y_2_0_Y_1_0_CONJ:_CSST_h_2_0_h_1_1;_CSST_Y_2_0_Y_1_1_CONJ:_CSST_h_2_0_h_2_-1;_CSST_Y_2_0_Y_2_-
1_CONJ:_CSST_h_2_0_h_2_0;_CSST_Y_2_0_Y_2_0_CONJ:_CSST_h_2_0_h_2_1;_CSST_Y_2_0_Y_2_1_CONJ:_CSST_h_2_1_h_0_0;_CSST_Y_2_1_Y_0_0_CONJ:_CSST_h_2_1_h_1_
-1;_CSST_Y_2_1_Y_1_-1_CONJ:_CSST_h_2_1_h_1_0;_CSST_Y_2_1_Y_1_0_CONJ:_CSST_h_2_1_h_1_1;_CSST_Y_2_1_Y_1_1_CONJ:_CSST_h_2_1_h_2_-1;_CSST_Y_2_1_Y_2_-
1_CONJ:_CSST_h_2_1_h_2_0;_CSST_Y_2_1_Y_2_0_CONJ:_CSST_h_2_1_h_2_1;_CSST_Y_2_1_Y_2_1_CONJ:_CSST3_h_0_0_h_1_-1;_CSST3_Y_0_0_Y_1_-
1_CONJ:_CSST3_h_0_0_h_1_0;_CSST3_Y_0_0_Y_1_0_CONJ:_CSST3_h_0_0_h_1_1;_CSST3_Y_0_0_Y_1_1_CONJ:_CSST3_h_0_0_h_2_-1;_CSST3_Y_0_0_Y_2_-
1_CONJ:_CSST3_h_0_0_h_2_0;_CSST3_Y_0_0_Y_2_0_CONJ:_CSST3_h_0_0_h_2_1;_CSST3_Y_0_0_Y_2_1_CONJ:_CSST3_h_1_-1_h_0_0;_CSST3_Y_1_-
1_Y_0_0_CONJ:_CSST3_h_1_-1_h_1_0;_CSST3_Y_1_-1_Y_1_0_CONJ:_CSST3_h_1_-1_h_1_1;_CSST3_Y_1_-1_Y_1_1_CONJ:_CSST3_h_1_-1_h_2_-1;_CSST3_Y_1_-1_Y_2_-
1_CONJ:_CSST3_h_1_-1_h_2_0;_CSST3_Y_1_-1_Y_2_0_CONJ:_CSST3_h_1_-1_h_2_1;_CSST3_Y_1_-
1_Y_2_1_CONJ:_CSST3_h_1_0_h_0_0;_CSST3_Y_1_0_Y_0_0_CONJ:_CSST3_h_1_0_h_1_-1;_CSST3_Y_1_0_Y_1_-
1_CONJ:_CSST3_h_1_0_h_1_1;_CSST3_Y_1_0_Y_1_1_CONJ:_CSST3_h_1_0_h_2_-1;_CSST3_Y_1_0_Y_2_-
1_CONJ:_CSST3_h_1_0_h_2_0;_CSST3_Y_1_0_Y_2_0_CONJ:_CSST3_h_1_0_h_2_1;_CSST3_Y_1_0_Y_2_1_CONJ:_CSST3_h_1_1_h_0_0;_CSST3_Y_1_1_Y_0_0_CONJ:_CSST3_h_1
_1_h_1_-1;_CSST3_Y_1_1_Y_1_-1_CONJ:_CSST3_h_1_1_h_1_0;_CSST3_Y_1_1_Y_1_0_CONJ:_CSST3_h_1_1_h_2_-1;_CSST3_Y_1_1_Y_2_-
1_CONJ:_CSST3_h_1_1_h_2_0;_CSST3_Y_1_1_Y_2_0_CONJ:_CSST3_h_1_1_h_2_1;_CSST3_Y_1_1_Y_2_1_CONJ:_CSST3_h_2_-1_h_0_0;_CSST3_Y_2_-
1_Y_0_0_CONJ:_CSST3_h_2_-1_h_1_-1;_CSST3_Y_2_-1_Y_1_-1_CONJ:_CSST3_h_2_-1_h_1_0;_CSST3_Y_2_-1_Y_1_0_CONJ:_CSST3_h_2_-1_h_1_1;_CSST3_Y_2_-
1_Y_1_1_CONJ:_CSST3_h_2_-1_h_2_0;_CSST3_Y_2_-1_Y_2_0_CONJ:_CSST3_h_2_-1_h_2_1;_CSST3_Y_2_-
1_Y_2_1_CONJ:_CSST3_h_2_0_h_0_0;_CSST3_Y_2_0_Y_0_0_CONJ:_CSST3_h_2_0_h_1_-1;_CSST3_Y_2_0_Y_1_-
1_CONJ:_CSST3_h_2_0_h_1_0;_CSST3_Y_2_0_Y_1_0_CONJ:_CSST3_h_2_0_h_1_1;_CSST3_Y_2_0_Y_1_1_CONJ:_CSST3_h_2_0_h_2_-1;_CSST3_Y_2_0_Y_2_-
1_CONJ:_CSST3_h_2_0_h_2_1;_CSST3_Y_2_0_Y_2_1_CONJ:_CSST3_h_2_1_h_0_0;_CSST3_Y_2_1_Y_0_0_CONJ:_CSST3_h_2_1_h_1_-1;_CSST3_Y_2_1_Y_1_-
1_CONJ:_CSST3_h_2_1_h_1_0;_CSST3_Y_2_1_Y_1_0_CONJ:_CSST3_h_2_1_h_1_1;_CSST3_Y_2_1_Y_1_1_CONJ:_CSST3_h_2_1_h_2_-1;_CSST3_Y_2_1_Y_2_-
1_CONJ:_CSST3_h_2_1_h_2_0;_CSST3_Y_2_1_Y_2_0_CONJ)
```

Measuring Asymmetries

C. Mullen, Glasgow

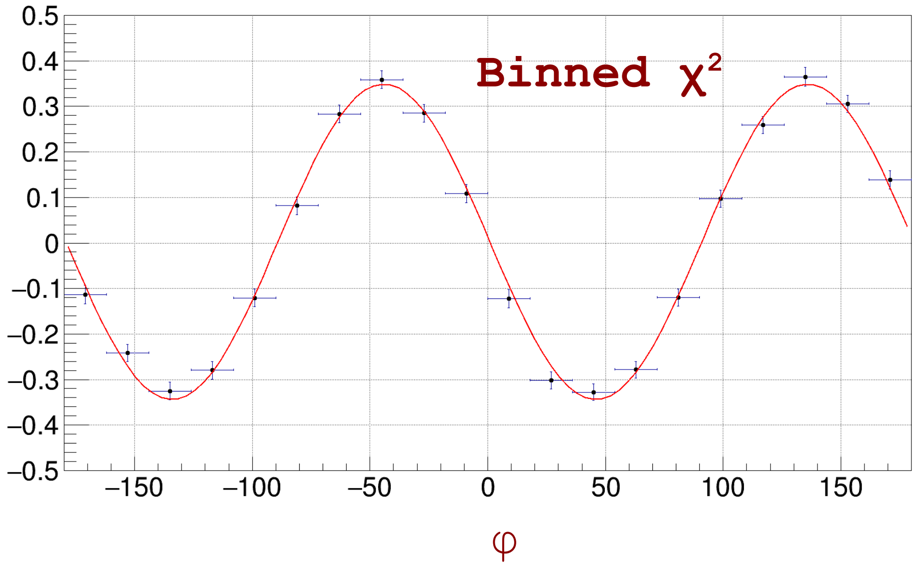
Example CB@MAMI nπ^0 photoproduction beam asymmetry

PDF :
$$f(\phi, P_\gamma, P=\pm 1) = (1 + A P_\gamma P \cos(2\phi))$$

Using histograms

$$\Sigma P_\gamma^{mean} \cos(2\phi) = \frac{N(\phi, P=+1) - N(\phi, P=-1)}{N(\phi, P=+1) + N(\phi, P=-1)}$$

Do not require acceptance correction (to first order)



Measuring Asymmetries with event based Likelihood

Normalisation Integral

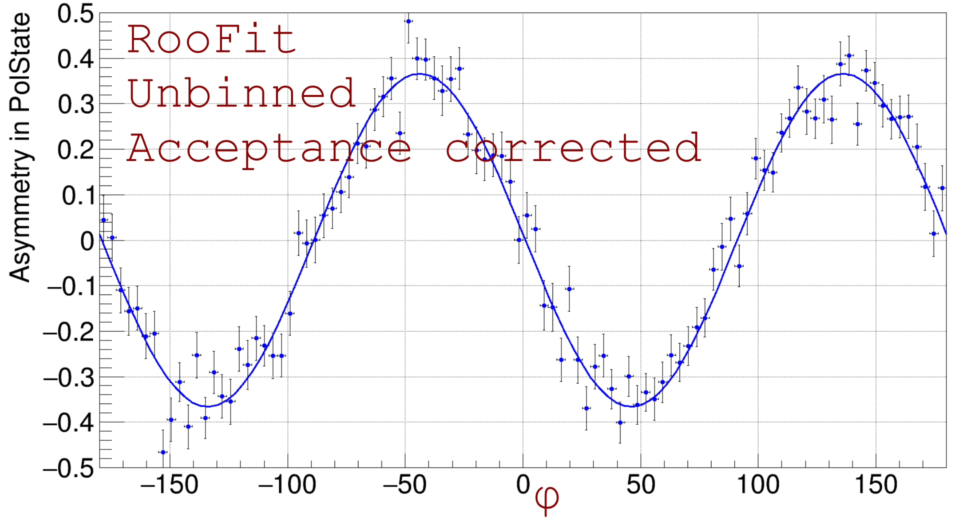
$$\sum_{i=0}^{Nacc} f(\tau_i, p, P=+1) + \sum_{i=0}^{Nacc} f(\tau_i, p, P=-1) = \sum_{i=0}^{Nacc} (1 + AP_{\gamma,i}\cos(2\phi_i)) + \sum_{i=0}^{Nacc} (1 - AP_{\gamma,i}\cos(2\phi_i)) = 2\,Nacc$$

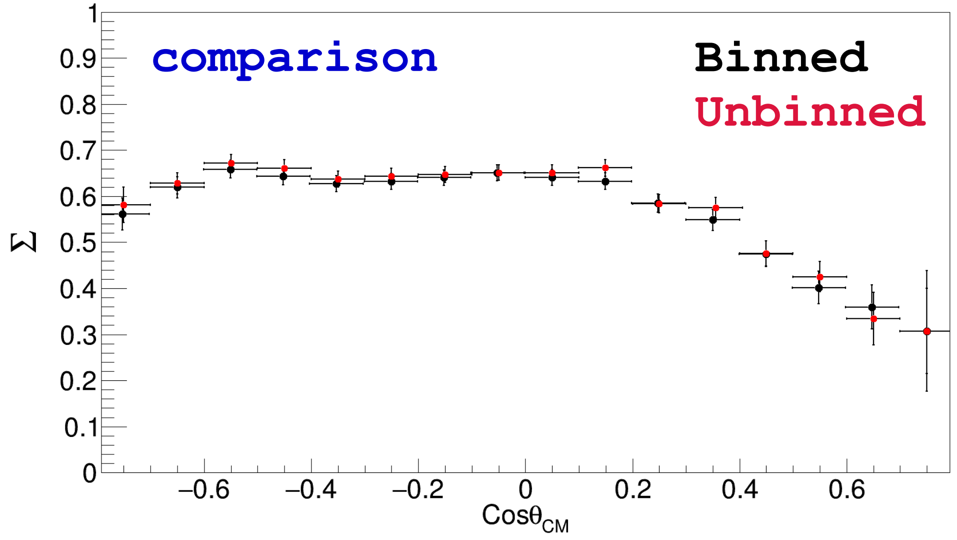
= constant, when N(P=+1)=N(P=-1) and $P_\gamma(P=+1)=P_\gamma(P=-1)$

=> does not effect position of likelihood maxmimum

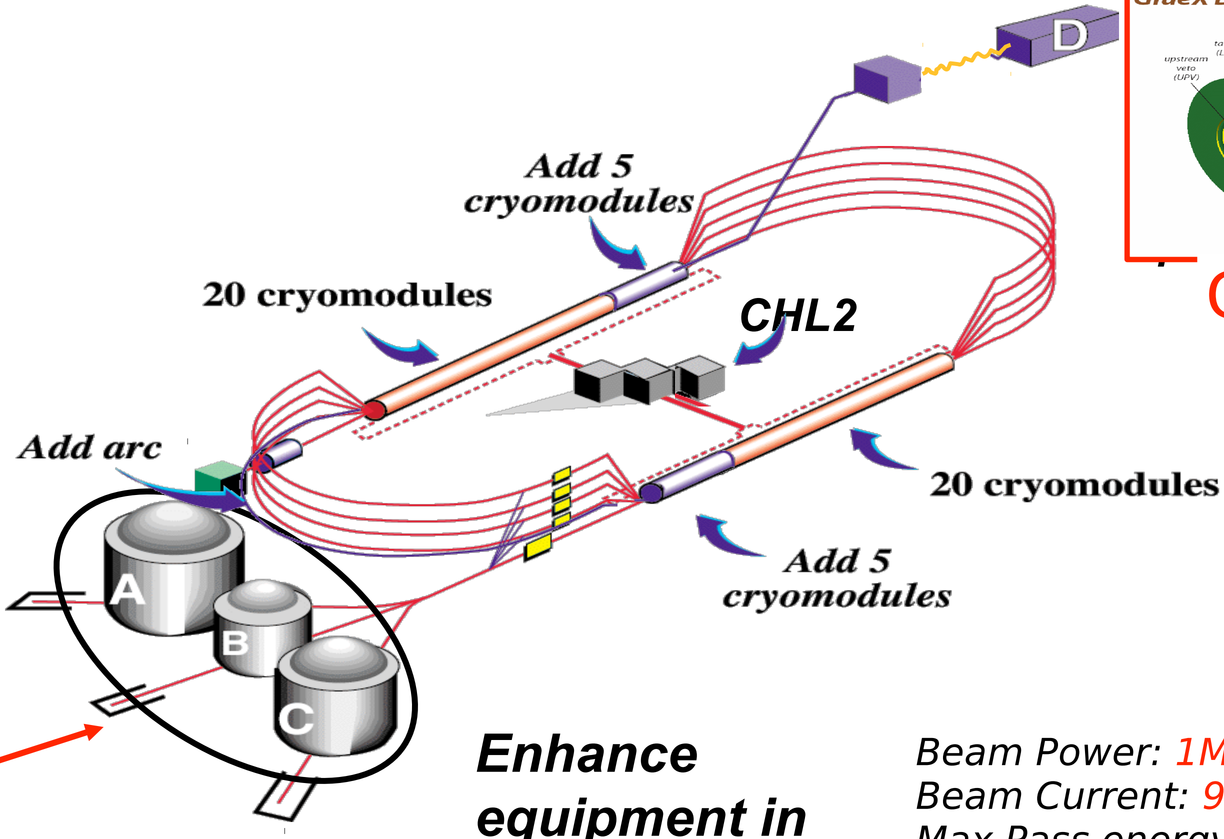
First order, ignore acceptance correction, do not need to calculate integral

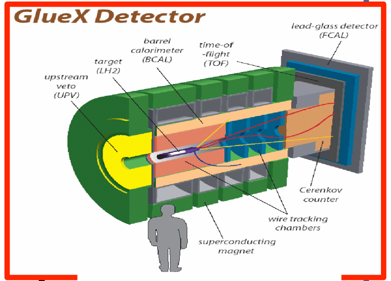
Second order, use acceptance, corrects for experimental polarisation and luminosity asymmetries



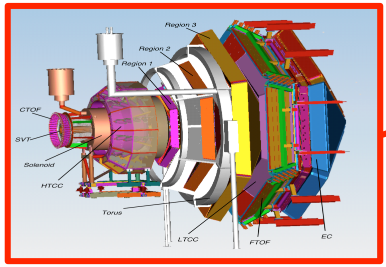


JLAB12





GLUEX



CLAS12

Enhance equipment in existing halls

Beam Power: 1MW
Beam Current: 90 μA
Max Pass energy: 2.2 GeV
Max Enery Hall A-C: 10.9 GeV
Max Energy Hall D: 12 GeV

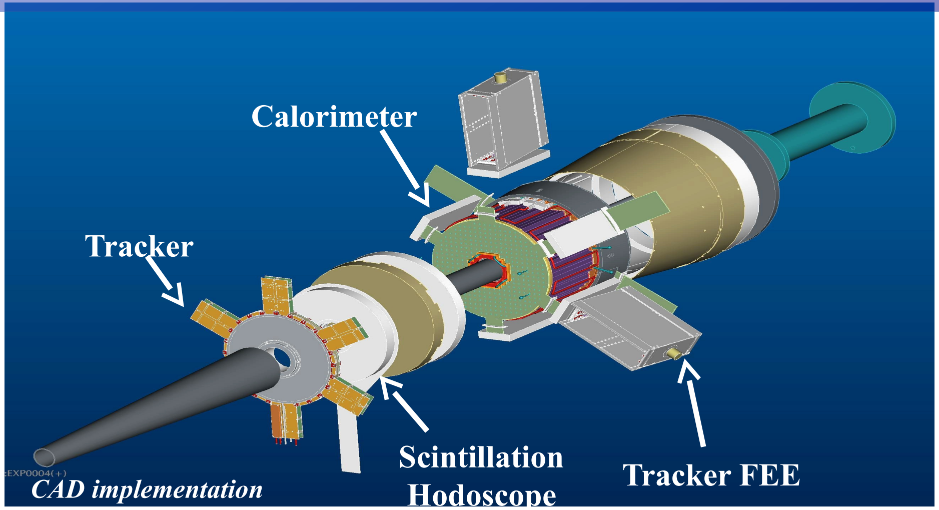
M

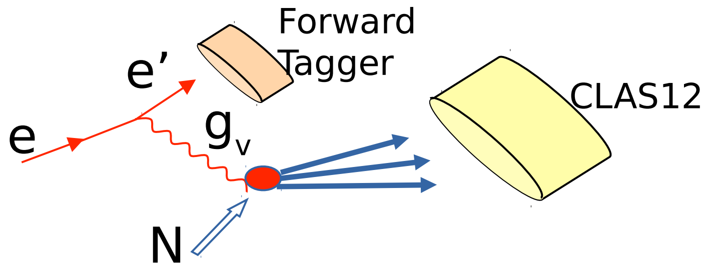
Detect electrons at small angle to perform quasi-real photo-production experiments.

Calorimeter: electron energy/momentum
Photon energy (ν=E-E')
Polarization $\varepsilon^{-1} \approx 1 + \nu^2/2EE'$
$PbWO_4$ crystals with APD/SiPM readout

Scintillation Hodoscope: veto for photons
Scintillator tiles with WLS readout

Tracker: electron angles, polarization plane
MicroMegas detectors





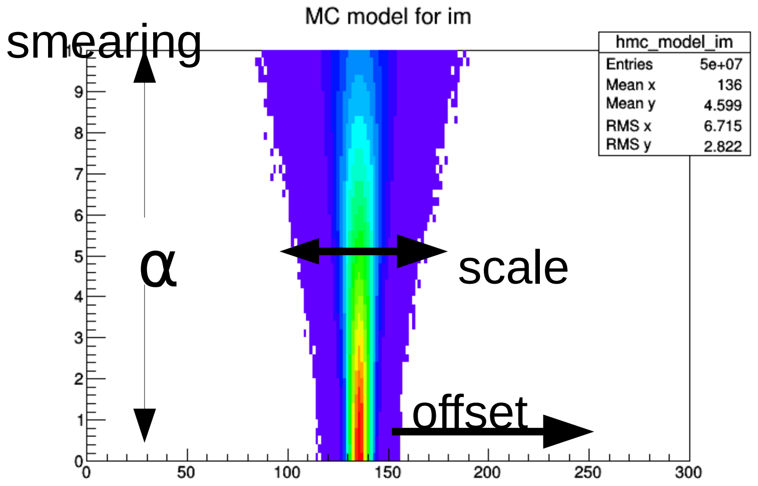
$E_{scattered}$	0.5 - 4.5 GeV
θ	2.5^o - 4.5^o
ϕ	0^o - 360^o
ν	6.5 - 10.5 GeV
Q^2	0.01 - 0.3 GeV^2 ($< Q^2 >$ 0.1 GeV^2)
W	3.6 - 4.5 GeV

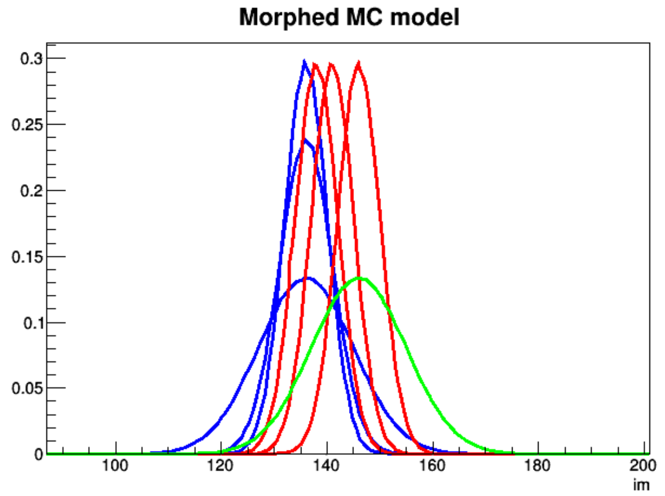
Fitting discriminatory variables

Signal shapes are not always well described by parameteric functions

⇒Simulated PDFs

Systematic uncertainty in shape accounted for via morphing with additional nuisance parameters


MC model for im
smearing
α
scale
offset
hmc_model_im
Entries 5e+07
Mean x 136
Mean y 4.599
RMS x 6.715
RMS y 2.822


Morphed MC model
im