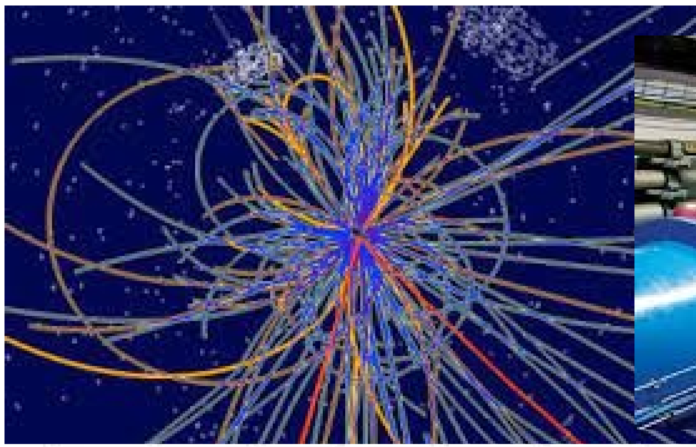
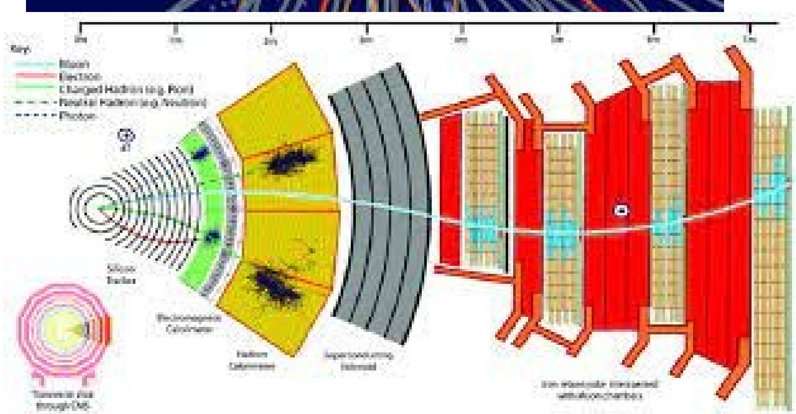
The framework ROOT and its use in particle physics

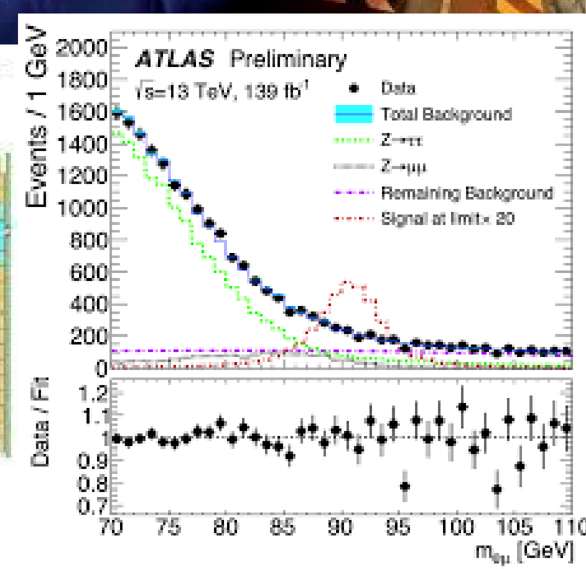
Session 3: Introduction to RooFit

Instructor: Gustavo Loachamin

What do we usually do in experimental particle physics (computationally)?





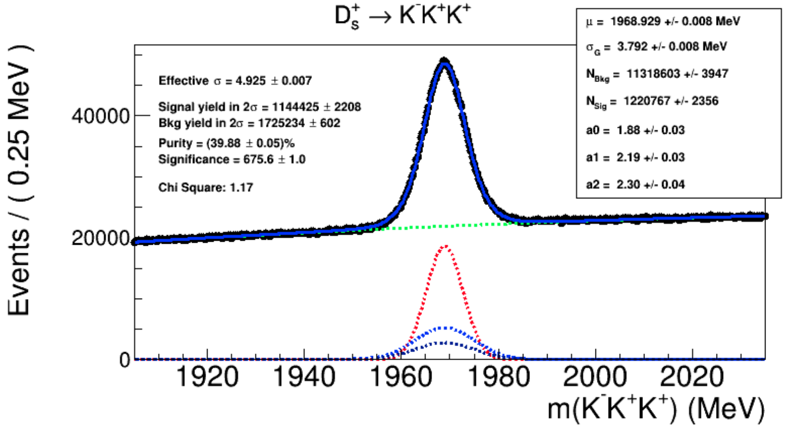


What do we usually do in experimental particle physics (computationally)?

Online

- Reconstruction of tracks and measurement of physical, topological variables.
- Reduction of background noise and selection.
- Data collection and processing.
- This is usually done with machine learning techniques.

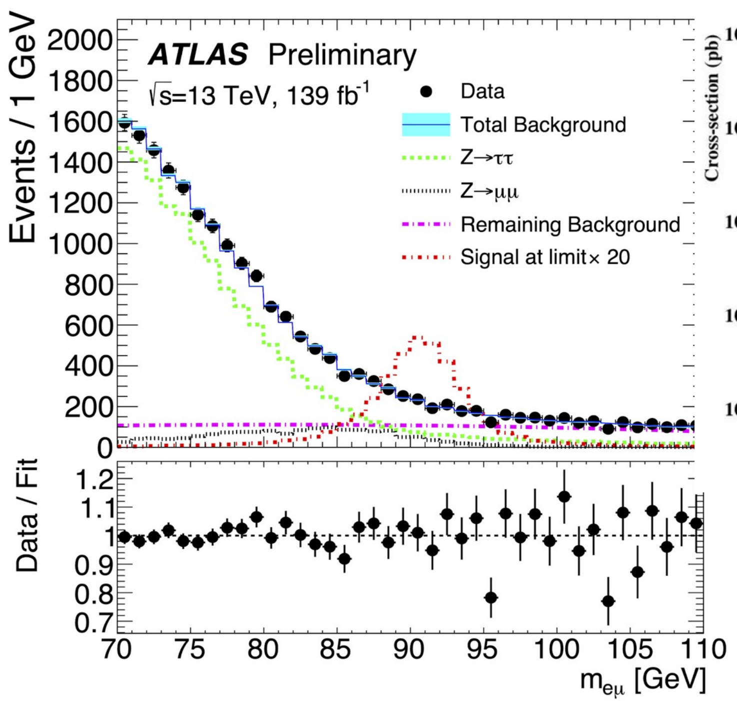
Offline

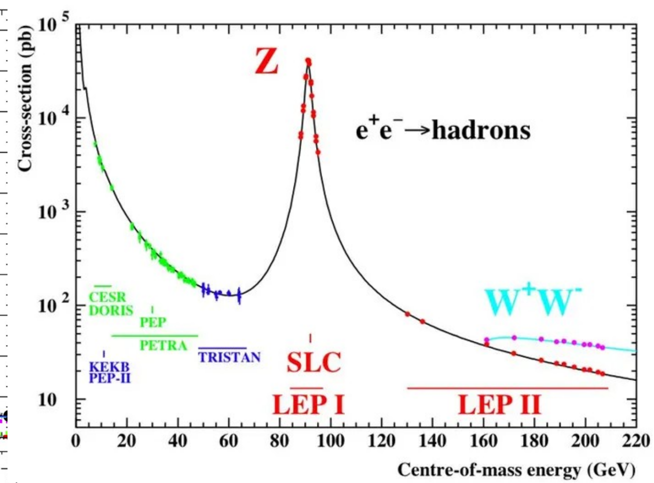


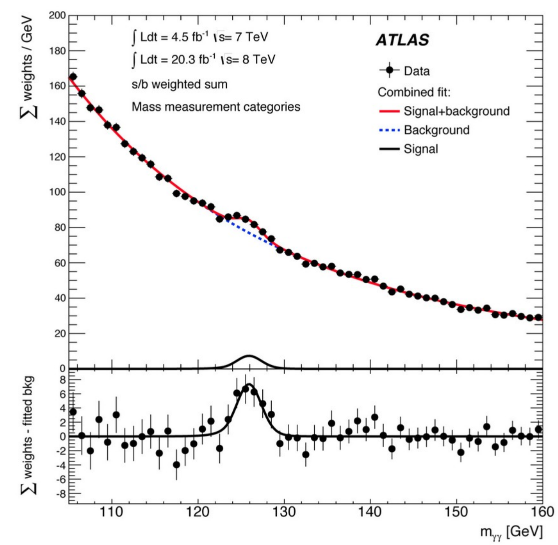
- Calculation of parameters.
- Reduction of background.
- Simulation of events.
- Comparison of model to data.

Fitting

- Model a data set to a theoretical function by finding the parameters that suit the model best.
- Technique used to compare data to models and can be combined with other techniques to reduce background, measure parameters.
- Fitting is a technique that appears somehow in every data analysis in particle physics.





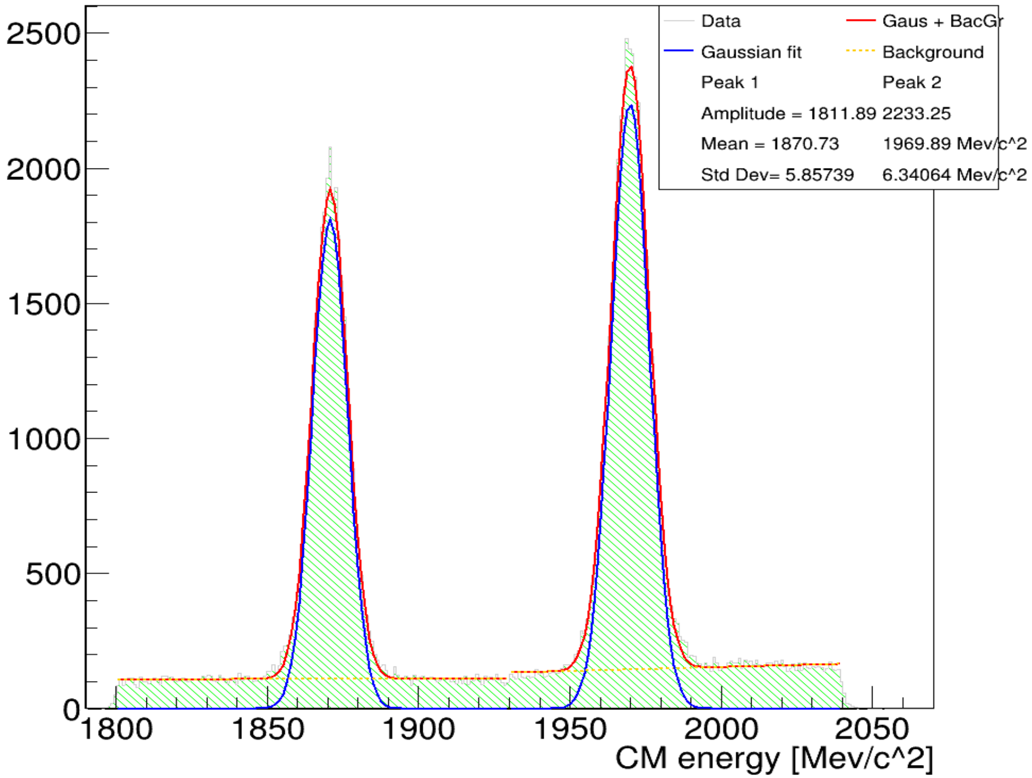


Fitting in ROOT

- Fitting in ROOT can be performed using method Fit with TH classes.

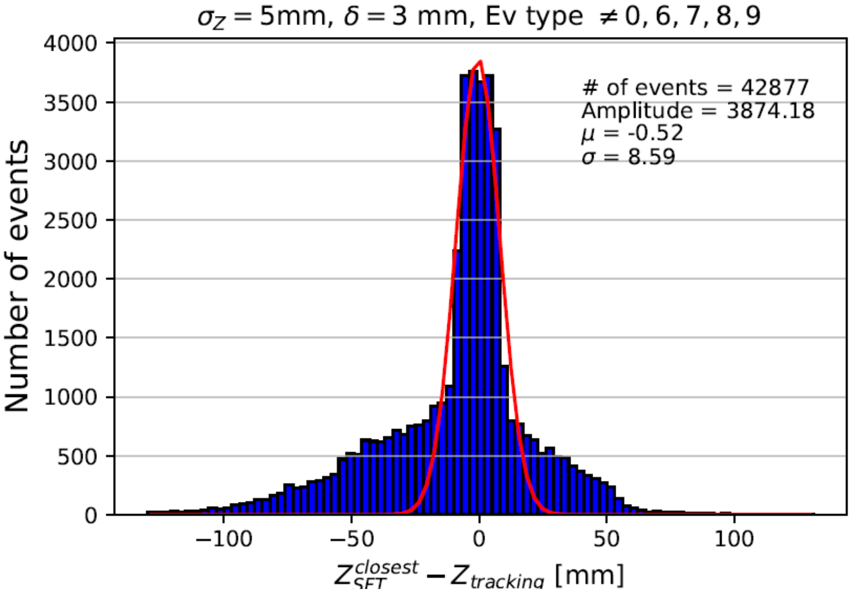
- For a TH1 h and a TF1 f:

  h->Fit(f)



Fitting in ROOT

- Theoretical models are constructed in the form of PDFs
- For ‘simple’ functions (gauss, polynomial), ROOT built-in models are sufficient.



- When introducing non-trivial functions, multidimensional functions and other complex methods, other packages are needed

The package RooFit

- It is a library (as many others created to perform fits) included in ROOT that is based on the maximum likelihood method.

- Suppose we have a data set with $\vec{x}_i$ that are distributed according to the PDF F, then the likelihood is define as:

$$L(\vec{p}) = \prod_i F(\vec{x}_i;\vec{p}), \quad \text{i.e.} \quad L(\vec{p}) = F(x_0;\vec{p}) \cdot F(x_1;\vec{p}) \cdot F(x_2;\vec{p}) \ldots$$

$$-\ln L(\vec{p}) = -\sum_i \ln F(\vec{x}_i;\vec{p})$$

- The objective is to minimize $-\ln(L)$

- Documentation can be found in https://root.cern.ch/download/doc/RooFit_Users_Manual_2.91-33.pdf

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`

RooFit

- All objects in RooFit are initialized with name and title.

- **Name** - Unique identifier of object
- **Title** – More elaborate description of object

Initial range

Objects representing a ‘real’ value.

```
RooRealVar mass("mass","Invariant mass",5.20,5.30) ;
RooRealVar width("width","B0 mass width",0.00027,"GeV");
RooRealVar mb0("mb0","B0 mass",5.2794,"GeV") ;
```

Initial value Optional unit

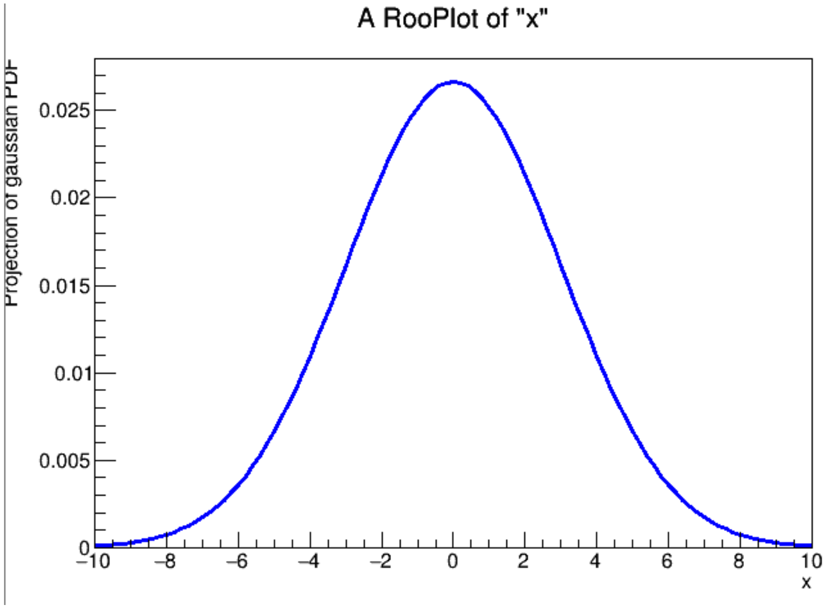
PDF object

```
RooGaussian b0sig("b0sig","B0 sig PDF",mass,mb0,width);
```

References to variables

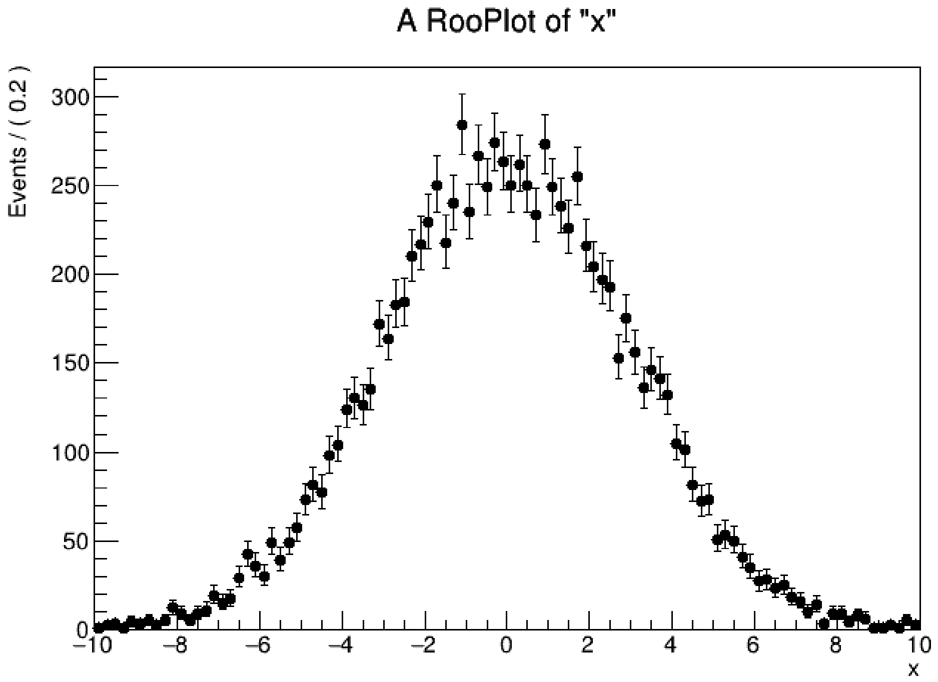
Wouter Verkerke, NIKHEF

Plotting a Gaussian




```
// Build Gaussian PDF
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",0,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",3) ;
RooGaussian gauss("gauss","gaussian PDF",x,mean,sigma) ;
// Plot PDF
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
xframe->Draw() ;
```

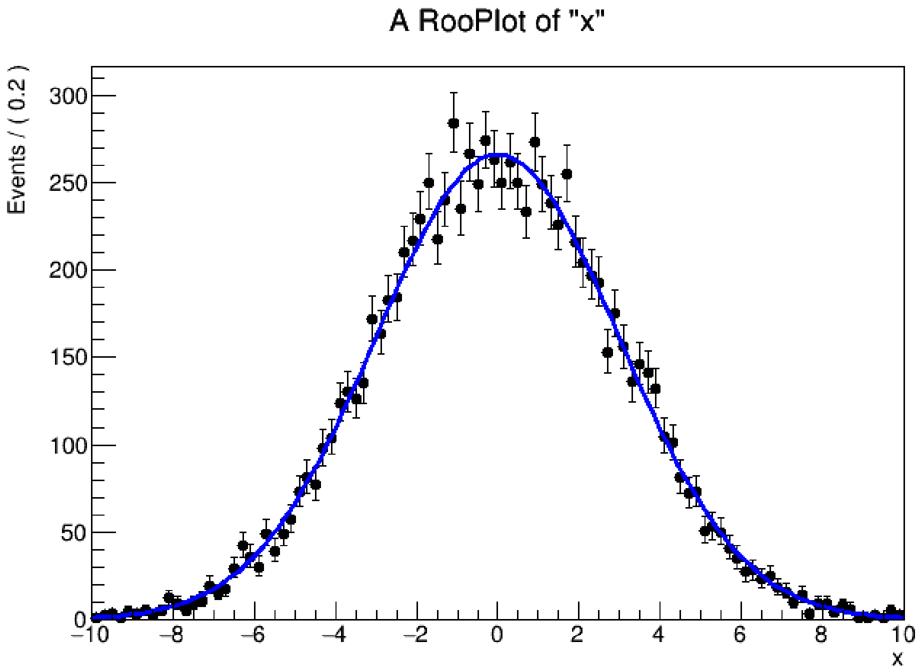
Generating a sample




```
// Generate a toy MC set
RooDataSet* data = gauss.generate(x,10000) ;
// Plot PDF
RooPlot* xframe = x.frame() ;
data->plotOn(xframe) ;
xframe->Draw() ;
```

Fit

```
// ML fit of gauss to data
gauss.fitTo(*data) ;
// Parameters if gauss
//now reflect fitted values
mean.Print()
sigma.Print()
// Plot fitted PDF and toy data overlaid
RooPlot* xframe2 = x.frame() ;
data->plotOn(xframe2) ;
gauss.plotOn(xframe2) ;
xframe2->Draw() ;
```

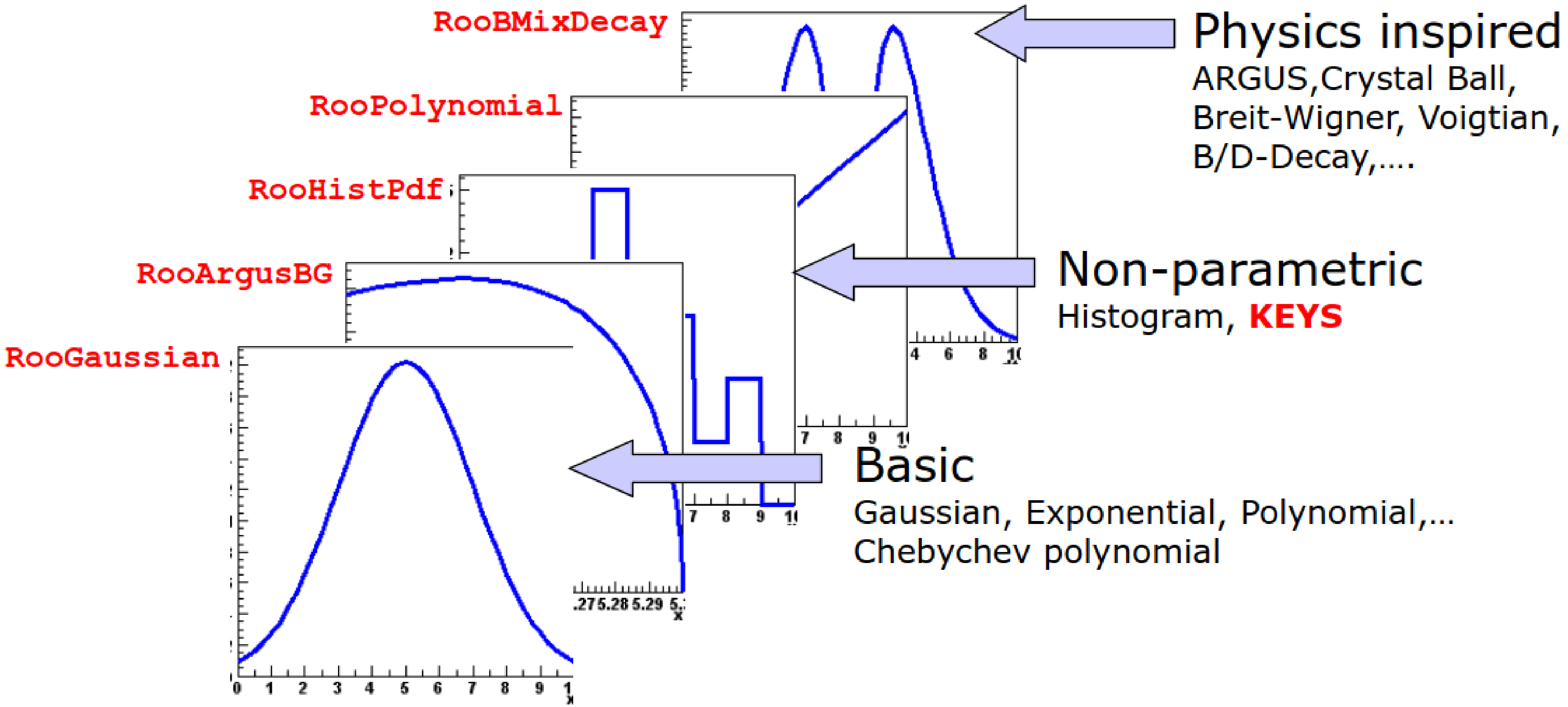


Data can also be imported

```
TH1* hh = (TH1*) gDirectory->Get("ahisto") ;
RooRealVar x("x","x",-10,10) ;
RooDataHist data("data","dataset with x",x,hh) ;
```

Model building

- RooFit contains many PDF classes



Model building

- If PDF classes are not included, one can define PDFs.
- Write down the expression as a C++ formula.

```
// PDF variables
RooRealVar x("x","x",-10,10) ;
RooRealVar y("y","y",0,5) ;
RooRealVar a("a","a",3.0) ;
RooRealVar b("b","b",-2.0) ;

// Generic PDF
RooGenericPdf gp("gp","Generic PDF","exp(x*y+a)-b*x",
                          RooArgSet(x,y,a,b)) ;
```

Model building

- If PDF classes are not included, one can define PDFs.
- One can also use the RooClassFactory method

```
RooClassFactory::makePdf("RooMyPdf","x,alpha") ;
```

- The expression can be compiled and linked using:

```
root>.L RooMyPdf.cxx+
```

Exercise: Model Building

- Intermediate states are related to relativistic Breit-Wigner lineshapes.

$$R(m) = \frac{1}{(m_0^2 - m^2) - i\, m_0 \Gamma(m)} \,,$$

ϕ

- In the Isobar Model, the width is proportional to the momentum of the resonance

$$\Gamma(m) = \Gamma_0 \left(\frac{q}{q_0}\right)^{2L+1} \left(\frac{m_0}{m}\right) X^2(q\, r_{\mathrm{BW}}^R) \,,$$

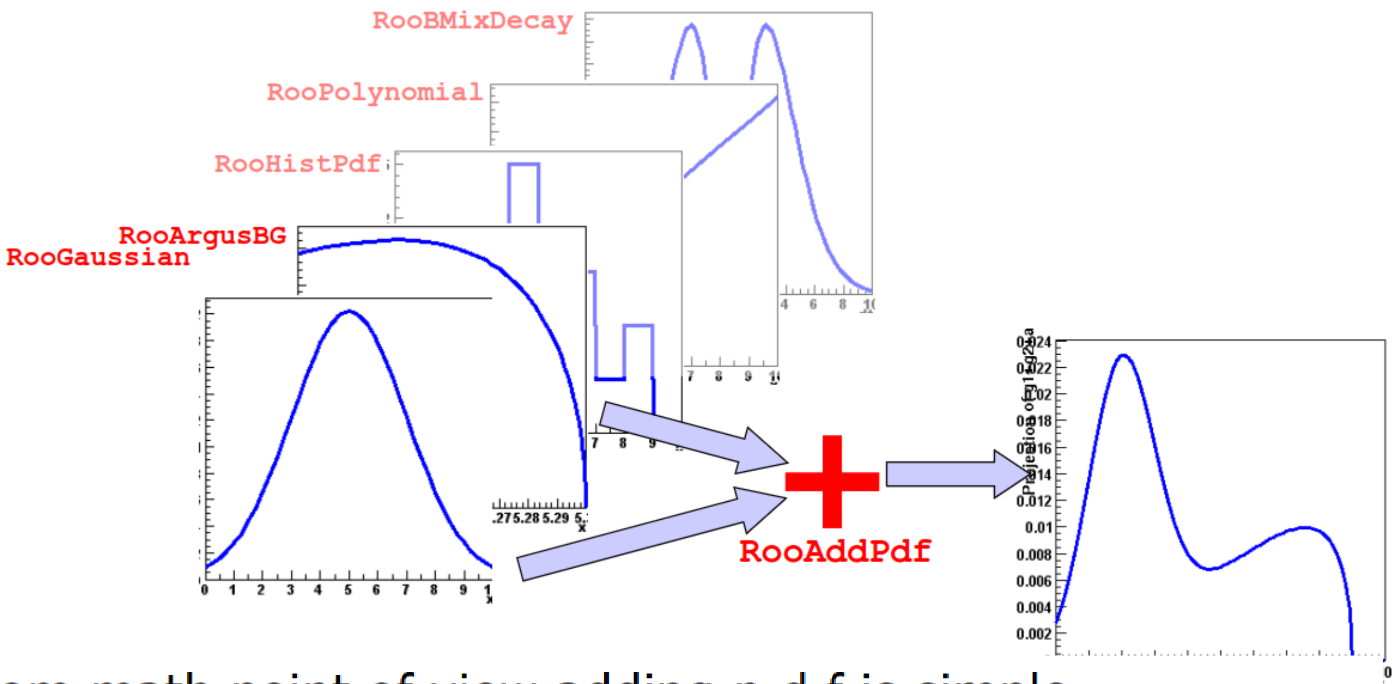
$$| = \frac{\lambda^{1/2}(s_{12}, m_2^2, m_1^2)}{2\sqrt{s_{12}}}$$

$$\lambda(x, y, z) = x^2 + y^2 + z^2 - 2xy - 2yz - 2xz$$

We are going to build this PDF for the resonance phi (1020) : m0 = 1019,46 MeV, witdh= 4,26 MeV

PDF sum

- It is performed using the class RooAddPdf



From math point of view adding p.d.f is simple

- Two components *F, G*

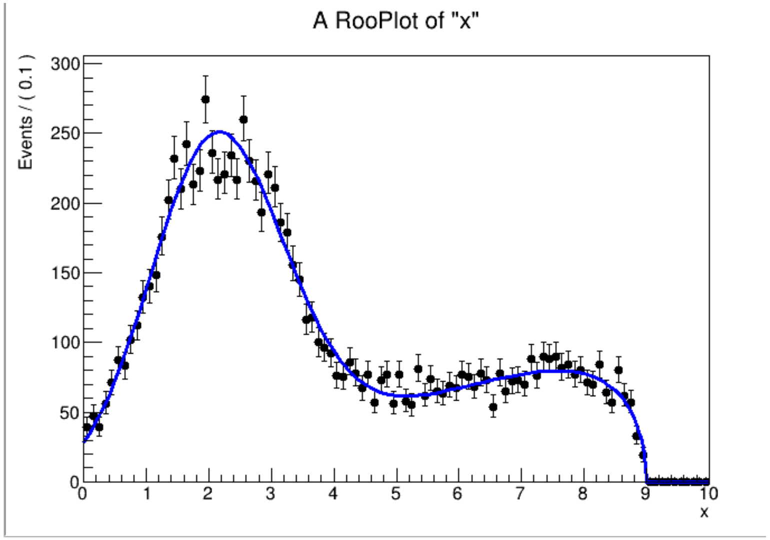
$$S(x) = fF(x) + (1-f)G(x)$$

- Generically for *N* components P_0-P_N

$$S(x) = c_0 P_0(x) + c_1 P_1(x) + \ldots + c_{n-1} P_{n-1}(x) + \left(1 - \sum_{i=0,n-1} c_i\right) P_n(x)$$

Plot pdf of a sum

```
// Build two Gaussian PDFs
RooRealVar x("x","x",-50, 500) ;
RooRealVar mean1("mean1","mean of gaussian 1",2) ;
RooRealVar mean2("mean2","mean of gaussian 2",3) ;
RooRealVar sigma("sigma","width of gaussians",1) ;
RooGaussian gauss1("gauss1","gaussian PDF",x,mean1,sigma) ;
RooGaussian gauss2("gauss2","gaussian PDF",x,mean2,sigma) ;
// Build Argus background PDF
RooRealVar argpar("argpar","argus shape parameter",-1.0) ;
RooRealVar cutoff("cutoff","argus cutoff",9.0) ;
RooArgusBG argus("argus","Argus PDF",x,cutoff,argpar) ;
// Add the components
RooRealVar g1frac("g1frac","fraction of gauss1",0.5, 0.2, 0.3) ;
RooRealVar g2frac("g2frac","fraction of gauss2",0.1, 0.05, 0.4) ;
RooAddPdf sum("sum","g1+g2+a",RooArgList(gauss1,gauss2,argus),RooArgList(g1frac,g2frac));
//Generate sample with distribution
RooDataSet *data =sum.generate(x,10000);
// Plot data and PDF overlaid
RooPlot* xframe = x.frame() ;
data->plotOn(xframe) ;
sum.plotOn(xframe) ;
xframe->Draw() ;
```



Chi square

- cout << "chi^2 = " << xframe->chiSquare() << endl;
- RooHist *hresid = xframe->residHist();

Residuals

- RooHist *hresid = xframe->residHist();
- RooPlot *frame2 = x.frame();
- frame2->addPlotable(hresid,"P");
- frame2->Draw();

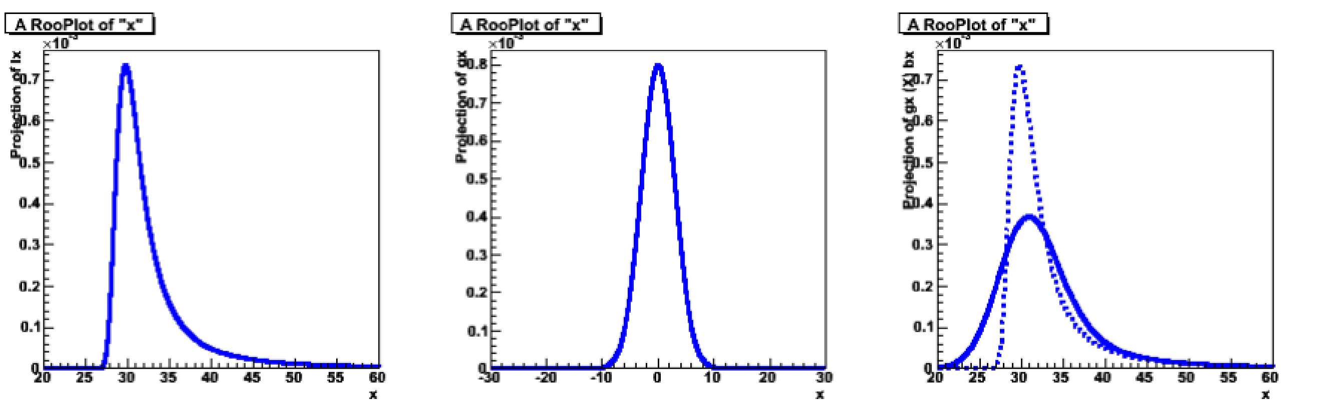
Convolutions

Properties of **RooNumConvPdf**

- Can convolve *any* two input p.d.f.s
- Uses special numeric integrator that can compute integrals in [-∞,+∞] domain
- Slow (very!) especially if requiring sufficient numeric precision to allow use in MINUIT (requires ~10^{-7} estimated precision). *Converge problems in MINUIT if precision is insufficient*

```
// Construct landau (x) gauss
RooNumConvPdf lxg("lxg","landau (X) gauss",t,landau,gauss) ;
```

Landau *Gauss* *Landau ⊗ Gauss*



Wouter Verkerke, NIKHEF

Convolutions

$$P(dt,\ldots) = \sum_k c_k(\ldots)\left(f_k(dt,\ldots) \otimes R(dt,\ldots)\right)$$

coefficient

'basis function'

resolution function

Example: B^0 decay with mixing

$$c_0 = 1 \pm \Delta w, \qquad f_0 = e^{-|t|/\tau}$$

$$c_1 = \pm(1-2w), \qquad f_1 = e^{-|t|/\tau}\cos(\Delta m \cdot t)$$

Multidimensional models

- Not for today.