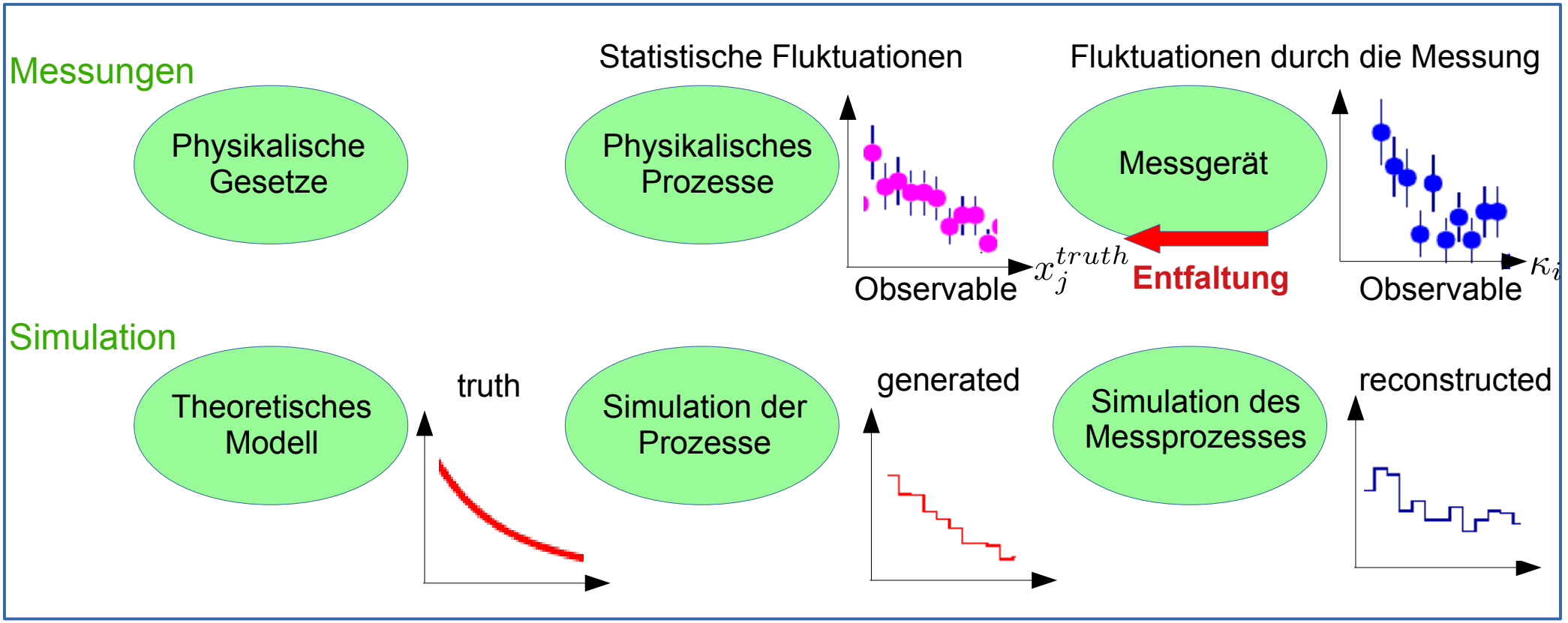
Bestimmung physikalischer Größen

Ziel von Messungen in der Physik ist die Bestimmung von fundamentalen Gesetzen. Die gemessenen Observablen beinhalten allerdings nicht nur die von fundamentalen Gesetzen bestimmten Werte, sondern auch die von Messmethoden und Messgeräten erzeugten Beiträge.



- Verwende Simulationen um eine Detektorantwort-Matrix A_{ij} zu bestimmen, die von dem i-ten gemessen Wert der Observablen κ_i zum j-ten Wert des physikalischen Prozesses x_j^{truth} zurückführt. → **Entfaltung**

$$\kappa_i = \sum_j A_{ij} x_j^{truth}$$

Details: S.Schmitt arXiv:1611.01927

Bestimmung physikalischer Größen

- Wenn die Detektorantwortfunktion A_{ij} signifikante Beiträge in den nicht diagonalen Elementen aufweist, kommt es zur Migration, d.h. die wahren i-ten Werte x_i^{truth} führen nicht nur zu Beiträgen im i-ten gemessenen Wert κ_i , sondern in auch in den benachbarten Werten.
  - Im Fall kleiner Migration werden mit Simulationsrechnungen Faktoren A_{ii} bestimmt, die dann als Korrekturfaktor in einer bin-to-bin Methode angewendet werden.

    Gemessene Werte i-ter Bin: $\kappa_i^{data} \pm \delta\kappa_i^{data}$
    Rekonstruierte simulierte Werte: $\kappa_i^{MC,rec}$
    Wahre simulierte Werte: $x_i^{MC,gen}$

    Korrigierte Werte: $x_i^{corr} = \frac{x_i^{MC,gen}}{\kappa_i^{MC,rec}} \kappa_i^{data}$ $\delta x_i^{corr} = \frac{x_i^{MC,gen}}{\kappa_i^{MC,rec}} \delta\kappa_i^{data}$

  - Im Fall von Migration muss mit Hilfe von Simulationsrechnungen die gesamte Detektor-antwort-Matrix A_{ij} bestimmt werden

    $$A_{ij} = \frac{\kappa_{ij}^{MC,rec}}{x_j^{MC,gen}}$$

    Für die Rekonstruktionseffizienz ergibt sich $\epsilon_j = \sum_i A_{ij}$

- Die gemessenen Werte $\kappa_i = \sum_j A_{ij} x_j^{truth}$ gehorchen einer Poisson Verteilung $\mathcal{P}(y_i; \kappa_i)$
- In ROOT sind verschiedene Algorithmen in der Klasse RooUnfold von T. Adye implementiert arXiv:1105.1160

Bestimmung physikalischer Größen

- Eine weitere Möglichkeit zur Bestimmung der physikalischen Größen ist das Modellieren der Messungen und die Anpassung und Ermittlung der Modellparameter (physikalische Größen) an die gemessen Daten.
  - Dazu muss die Detektorauflösung entweder in einer separaten Messung bestimmt werden oder im Fitmodell an die gemessenen Daten implementiert und aus der Anpassung ermittelt werden.
  - Ausserdem muss mit Hilfe von Pull Verteilungen z.B durch Toy Simulationsrechnungen gezeigt werden, dass die Bestimmung der Fit Parameter unbiased ist.

$$g_{pull} = \frac{y_i^{fit} - y^{true}}{\sigma_i^{fit}} \qquad \text{Gauss Verteilung} \quad \mathcal{P}(\mu = 0, \sigma = 1)$$

- Mit Hilfe von detaillierten Simulationsrechnungen müssen Effizienzen für Rekonstruktion und Selektion und die Akzeptanz der Messapparatur bestimmt werden.

- Im folgenden wird die Anpassung von Modellen, deren Simulation (Toy MC) und Verifikation mit Hilfe von rooFit diskutiert.

RooFit - Einleitung

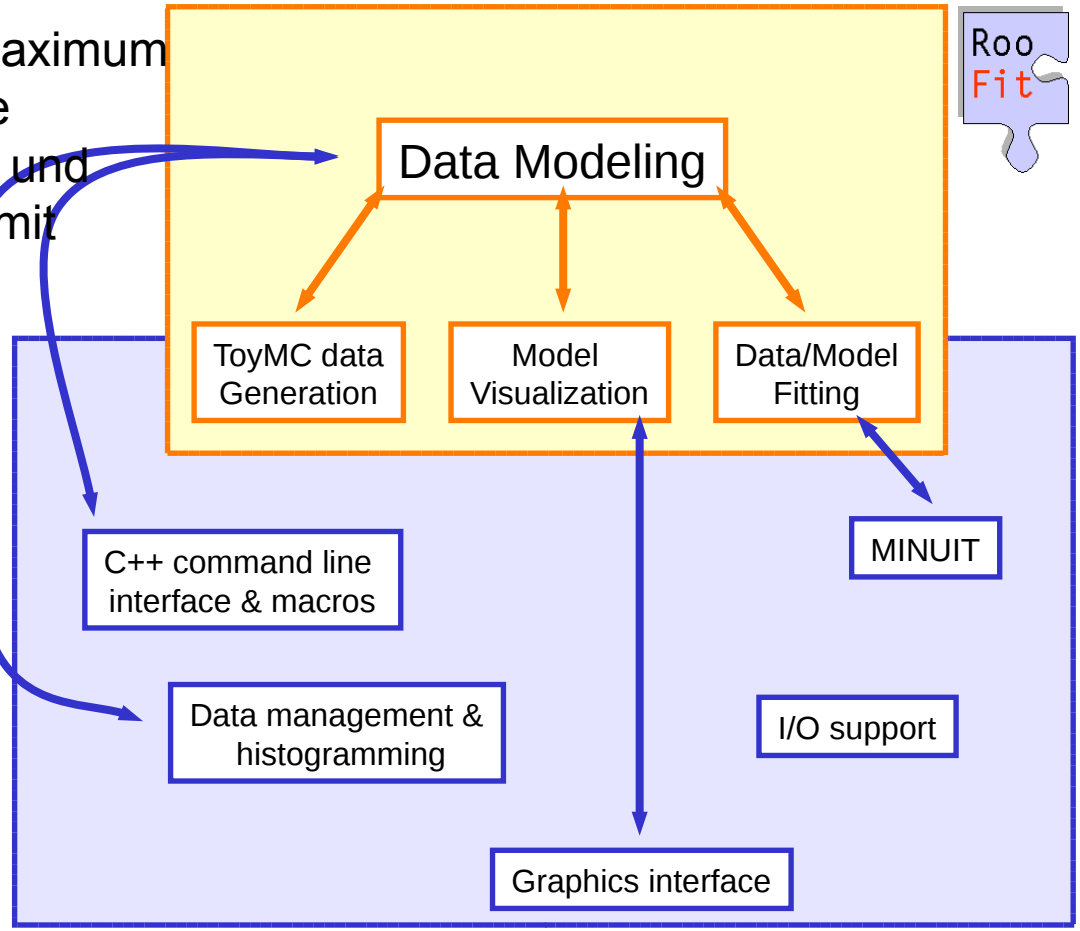
- RooFit ist ein in ROOT integriertes Toolkit zum Modellieren in Datenanalysen
  https://root.cern.ch/roofit-20-minutes

- Ziel: Modellieren von diskreten zeitlich geordneten Messungen einer oder mehrerer Observabler. Die Messungen gehorchen typischer Weise einer Poisson- oder Binomial-statistik.

- Vereinfacht das Fitten mit der Maximum Likelihood Methode, erlaubt eine schnelle graphische Darstellung und eine Simulation der Fitresultate mit Monte Carlo Methoden.

- RooFit wurde zur Datenanalyse für die BaBar Collaboration am SLAC entwickelt (Wouter Verkerke et al.).

- In RooFit können ungebinnte oder gebinnte fits durchgeführt werden.



RooFit – Warum wird das gebraucht?

- In Root können komplizierte Funktionen zwar einfach beschrieben werden, aber es werden relativ viel Code und weitere Klassen und Methoden gebraucht, um den Fit Prozess durchzuführen, z.B.
  - Behandlung von PDFs und Normierung der Integrale für multidimensionale Modelle
  - Berechnen des Signalanteils mit Fehlern
- Optimieren der Fit Performance durch einheitliche Behandlung und optimierten Code, z.B erfordern Summe, Produkte und Faltungen von PDFs einen Rahmen, der immer wieder verwendet werden sollte und nicht für jeden Fall erneut implementiert werden muss.
  → Unterstützung von komplizierten zusammengesetzten Modellen aus Standard Komponenten
- Minimieren von Programmierfehlern beim Fit Prozess
- Gleichzeitige Anpassung an mehrere Datensätze
- Das Fit Model steht auch zur Simulation im gleichen Framework zur Verfügung
- Speichern und wieder Einlesen des Fit Prozesses (Persistieren der Beschreibung des Likelihood Modells in Root Files)
- Unterstützung eines modularen, einheitlichen und flexiblen Aufbaus des Fit Prozesses
- Visualisierung der Daten und Fits

Maximum Likelihood - Methode

Die Fits in RooFit basieren auf der Maximum Likelihood Methode

- Maximum-Likelihood-Prinzip

Betrachten wir eine Stichprobe $x_1, x_2, ..., x_n$, die aus n Messungen besteht. Jedes x_i kann dabei für einen ganzen Satz von Variablen stehen.
Die Einzelmessungen x_i gehorchen dabei Verteilungen mit der Wahrscheinlichkeitsdichte $f(x_i)$, die von einem Parametersatz $\theta = \theta_1, \theta_2, ..., \theta_k$ abhängen, geschrieben $f(x, |\theta)$

Die Gesamtwahrscheinlichkeit für das Auftreten einer Stichprobe ist das Produkt der Einzelwahrscheinlichkeiten der einzelnen Elemente der Stichprobe.

$$L(x_1, ..., x_n) = \prod_{i=1}^{n} f(x_i|\theta) \qquad \text{mit} \qquad \int_\Omega L(x_1, ..., x_n) dx_1 ... dx_n = 1$$

wobei Ω der Definitionsbereich der Stichprobe $x_1, x_2, ..., x_n$ ist.
L heißt Likelihood-Funktion.

Der optimale Parametersatz $\hat{\theta}$ ist der, der die Likelihood-Funktion L maximiert.

Aus numerischen Gründen wird der Logarithmus der Likelihood-Funktion betrachtet

$$\mathcal{L}(x_1, ..., x_n|\theta) = log(L(x_1, ..., x_n|\theta)) = \sum_{i=1}^{n} log(f(x_i|\theta)) \qquad \text{und}$$

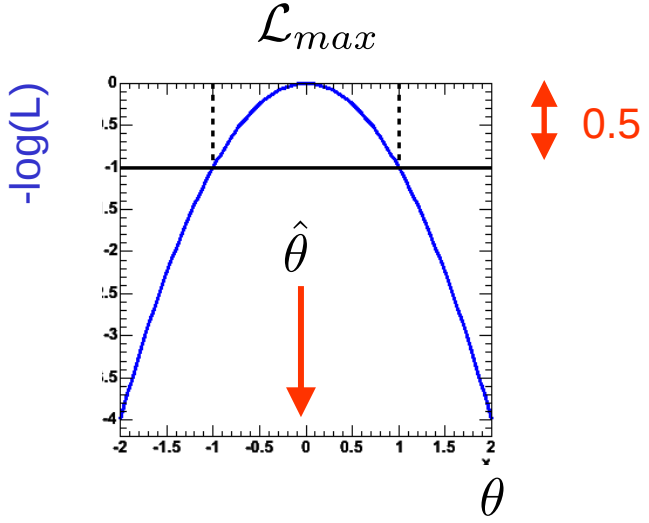
$-\mathcal{L}$ minimiert.

Maximum Likelihood - Methode

- Varianz der Maximum Likelihood Parameter

Im Minimum von $-\mathcal{L}$ ist $\dfrac{d\mathcal{L}(\vec{x}|\vec{\theta})}{d\vec{\theta}} = 0$

und die Varianz läßt sich aus der der 2ten Ableitung am Minimum bestimmen



$$\mathcal{L}(\theta \pm \sigma) = \mathcal{L}_{max} - \frac{1}{2}$$

Der Parameterfehler ist durch die Änderung der log Likelihood am Minimum um 0.5 gegeben.

Anschaulich:
Die 2te Ableitung ist ein Maß für die Krümmung der Likelihood-Hyperfläche im Parameterraum und damit für die Präzision.

Taylor-Entwicklung:

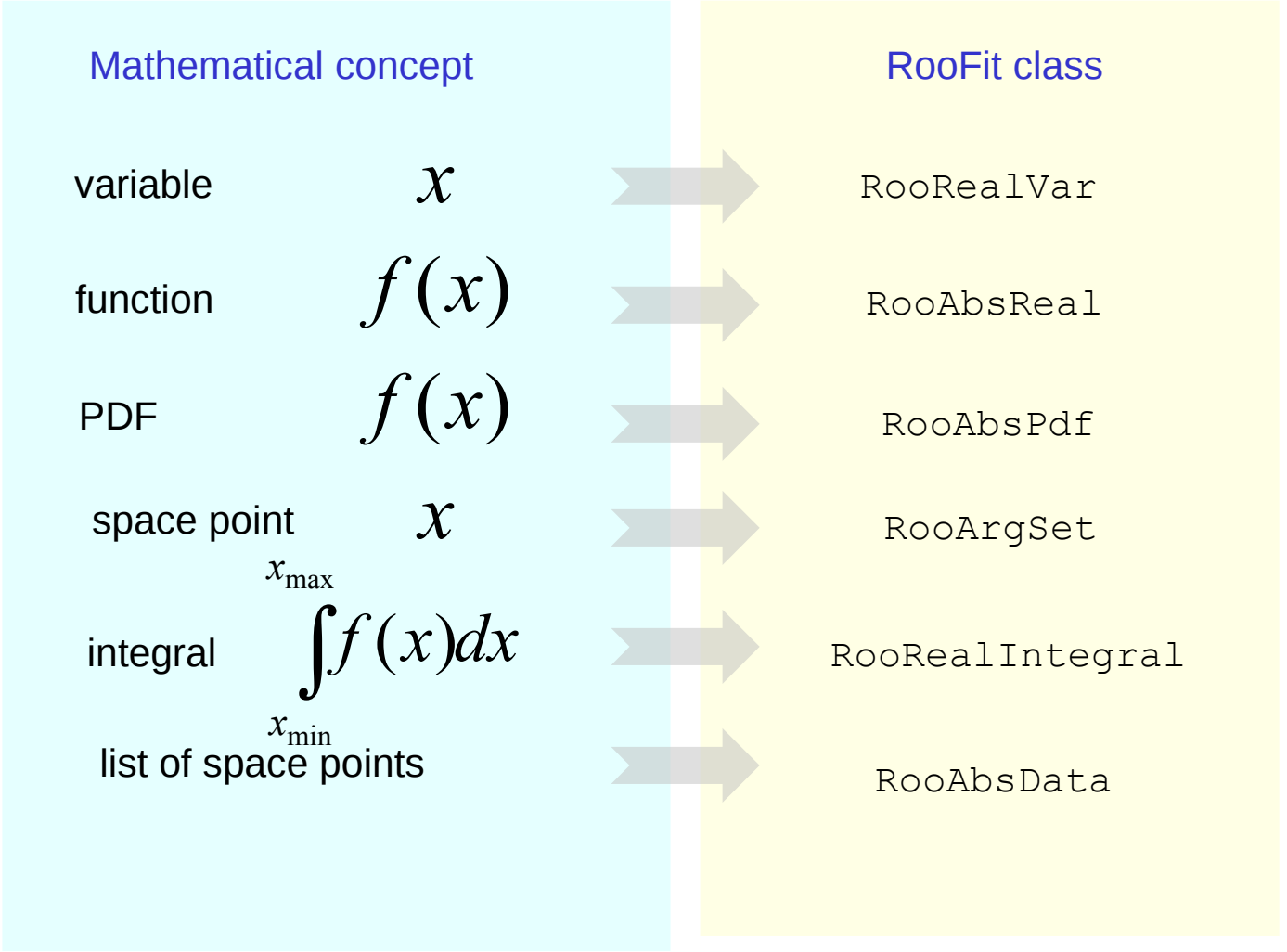
$$\ln L(p) = \ln L(\hat{p}) + \left.\frac{d \ln L}{dp}\right|_{p=\hat{p}} (p - \hat{p}) + \tfrac{1}{2} \left.\frac{d^2 \ln L}{d^2 p}\right|_{p=\hat{p}} (p - \hat{p})^2$$

$$= \ln L_{\max} + \left.\frac{d^2 \ln L}{d^2 p}\right|_{p=\hat{p}} \frac{(p - \hat{p})^2}{2}$$

$$= \ln L_{\max} + \frac{(p - \hat{p})^2}{2\hat{\sigma}_p^2} \quad \Rightarrow \quad \ln L(p \pm \sigma) = \ln L_{\max} - \frac{1}{2}$$

RooFit Design

Die mathematischen Bausteine eines Maximum Likelihood fits werden in der RooFit Klasse so nah wie möglich in Form vom Methoden implementiert. Dadurch werden Variable, Datenpunkte, Funktionen und PDFs durch C++ Objekte repräsentiert. Alle Normierungen werden automatisch (intern) ausgeführt. Alle Methoden funktionieren mit allen Objekten. Jedes durch Verkettung erzeugte Objekt ist voll funktionsfähig.



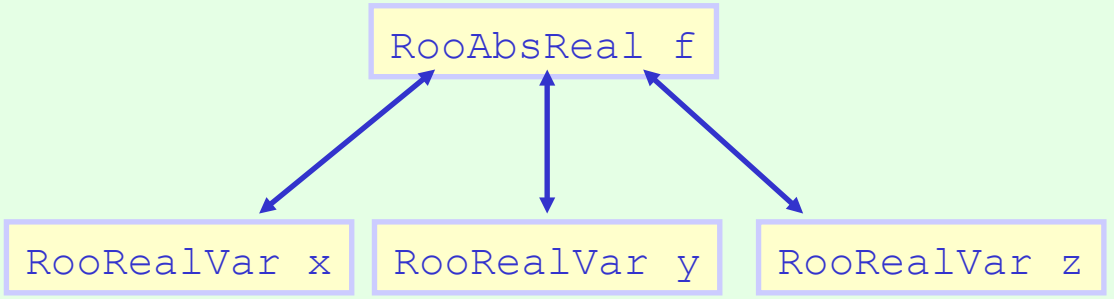
RooFit Design

- Represent relations between variables and functions as client/server links between objects

Math

$$f(x,y,z)$$

RooFit diagram



RooFit code

```
RooRealVar x("x","x",5) ;
RooRealVar y("y","y",5) ;
RooRealVar z("z","z",5) ;
RooBogusFunction f("f","f",x,y,z) ;
```

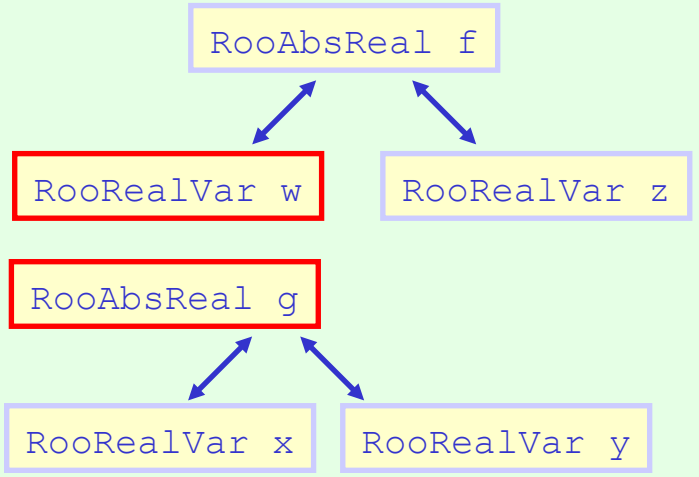
RooFit Design

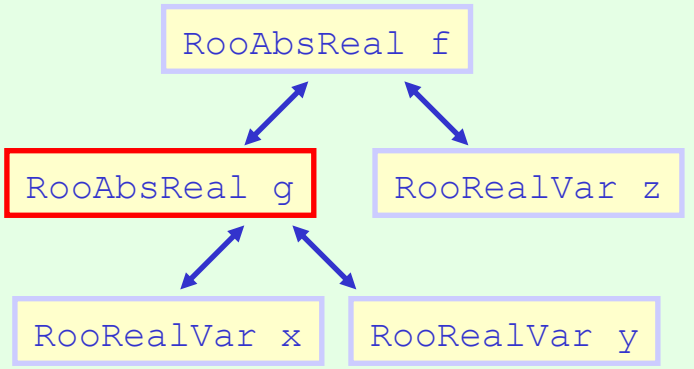
- Composite Funktion → Composite Objects

Math

$f(w,z)$ $g(x,y)$ → $f(g(x,y),z) = f(x,y,z)$

RooFit diagram





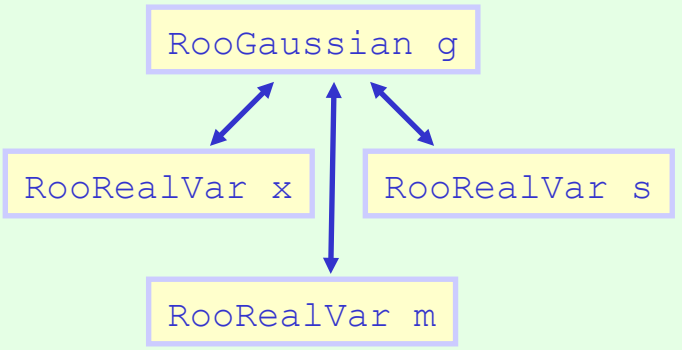
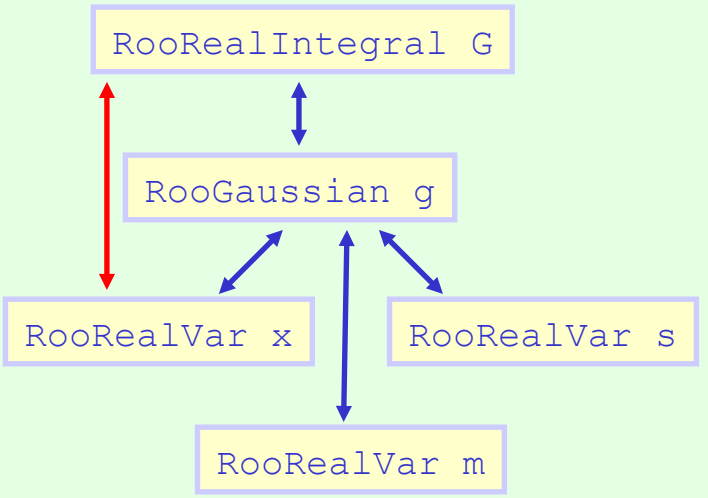
RooFit code

```
RooRealVar x("x","x",2) ;
RooRealVar y("y","y",3) ;
RooGooFunc g("g","g",x,y) ;

RooRealVar w("w","w",0) ;
RooRealVar z("z","z",5) ;
RooFooFunc f("f","f",w,z) ;
```

```
RooRealVar x("x","x",2) ;
RooRealVar y("y","y",3) ;
RooGooFunc g("g","g",x,y) ;

RooRealVar z("z","z",5) ;
RooFooFunc f("f","f",g,z) ;
```

RooFit Design

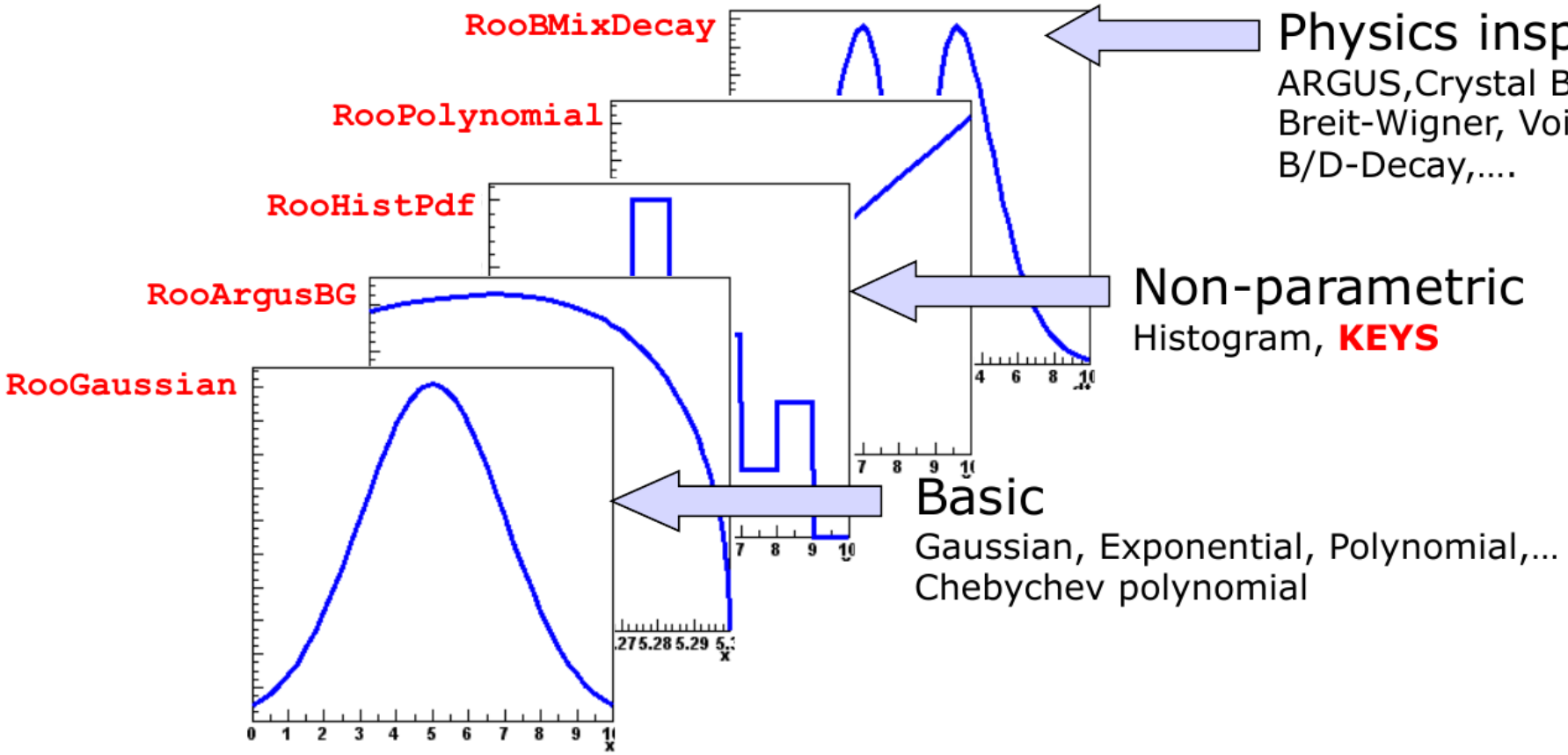
- Represent integral as an object, instead of representing as an action

Math	$g(x,m,s)$	$\int_{x_{min}}^{x_{max}} g(x,m,s)dx = G(m,s,x_{min},x_{max})$
RooFit diagram		

RooFit code	`RooRealVar x("x","x",2,-10,10)`
	`RooRealVar s("s","s",3) ;`
	`RooRealVar m("m","m",0) ;`
	`RooGaussian g("g","g",x,m,s)`
		`RooAbsReal *G =`
		`  g.createIntegral(x) ;`

RooFit Design

- RooFit provides a collection of compiled standard PDF classes



Easy to extend the library: each p.d.f. is a separate C++ class

Extended Maximum Likelihood

- Aufgrund der Definition der Maximum Likelihood Methode über normierte Wahrscheinlichkeitsdichtefunktionen (PDF) läßt sich neben den Parametern nur der Anteil der beitragenden Funktionen bestimmen, nicht die Ereignisanzahl.

Die Ereigniszahl N_{obs} gehorcht häufig einer Poissonverteilung mit einem Mittelwert der erwarteten Ereigniszahl N_{exp}

$$P(N_{exp}, N_{obs}) = \frac{N_{exp}^{N_{obs}}}{N_{obs}!} \cdot e^{-N_{exp}}$$

Aus der kombinierten Likelihood kann dann die Anzahl der Ereignisse bestimmt werden. Für den Fall dass N_{exp} unabhängig vom Parametersatz θ ist, ist im Maximum N_{exp} der optimale Wert.

Um die absolute Anzahl von Ereignissen zu bestimmen, müssen weitere Terme zur Likelihood Funktion hinzugefügt werden.

→ Extended Maximum Likelihood Methode:

$$\mathcal{L}(x_1, ..., x_n|\theta) = log(L(x_1, ..., x_n|\theta)) = \sum_{i=1}^{N_{obs}} log(f(x_i|\theta)) - N_{exp} + N_{obs} log(N_{exp})$$

Diese Methode steht auch in RooFit zur Verfügung

RooFit – Start

Komplizierte Datenanpassung mit 10 Zeilen C++ code: Elemente, die für einen Fit mit Minuit innerhalb von RooFit gebraucht werden (funktionsfähiges Beispiel).

```
#include <RooRealVar.h>
#include <RooGaussian.h>
#include <RooDataSet.h>
#include <RooPlot.h>

using namespace RooFit ;

void rooSimple() {
//define variables
RooRealVar mass("mass","D^0 mass [MeV]",1790,1920);
RooRealVar Mean ("Mean","mean of D^0 mass",1864.4,1839.5, 1889.5);
RooRealVar Sigma("Sigma","D^0 mass width [MeV]", 7.08, 0., 20.);
// Define the Gaussian PDF
RooGaussian Signal ("Signal","Mass distribution",mass,Mean,Sigma);
// Generate the nGen events, store the values in a RooFit Dataset
RooDataSet *data = Signal.generate(mass,5000);
// Fit the gaussian PDF to the dataset
Signal.fitTo(*data,NumCPU(2),Timer(kTRUE)) ;
// Plot the Data, Fit result
RooPlot *xframe0 = mass.frame() ;
data->plotOn(xframe0);
Signal.plotOn(xframe0) ;
xframe0->Draw();}
```

RooFit Funktionalität

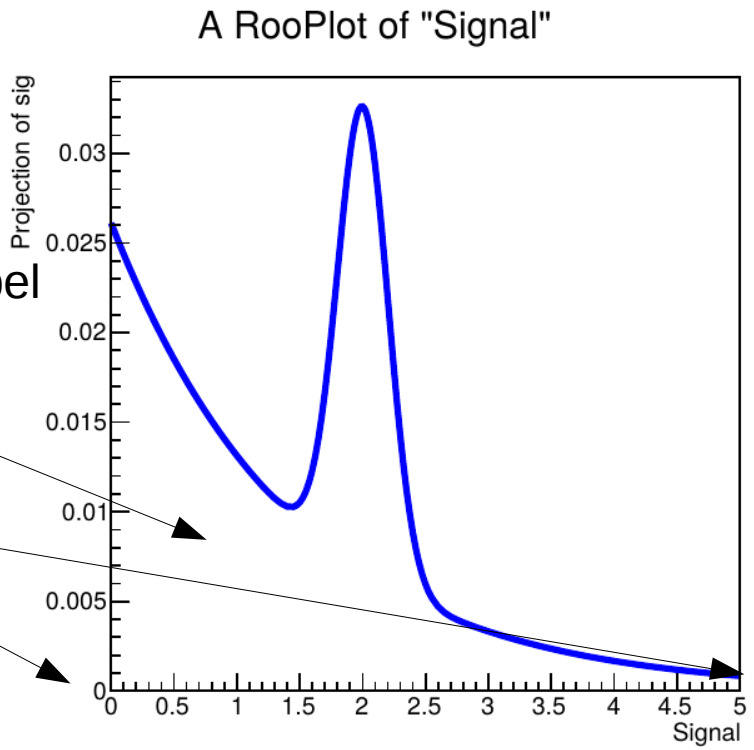
- PDF Visualisierung

```
RooPlot *xframe = Signal.frame() ;
sig->plotOn(xframe);
xframe→Draw();
```

axis label taken from PDF label

nomalized

plot range taken from “Signal”



`RooPlot` is an empty frame holding everything plotted to it.

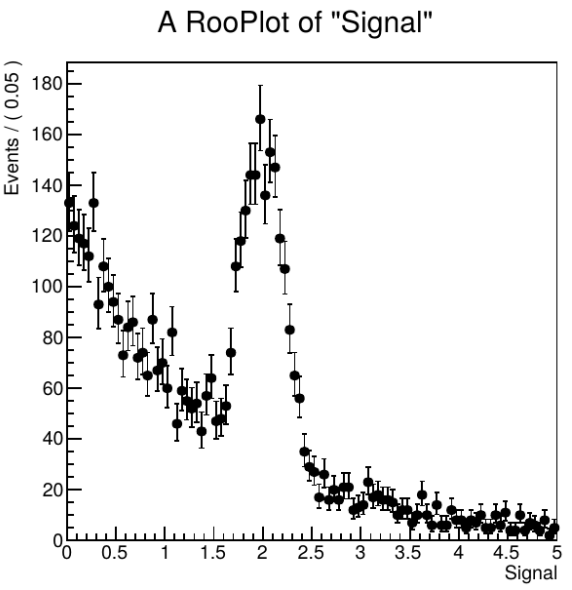
- Data Generation aus der PDF `sig`

```
RooDataSet * Data = sig→generate(*signal,1000);
```

- Daten Visualisierung

```
RooPlot *xframe = Signal.frame() ;
Data->plotOn(xframe);
xframe→Draw();
```

Daten in *Data sind ungebinnt!



RooFit Funktionalität

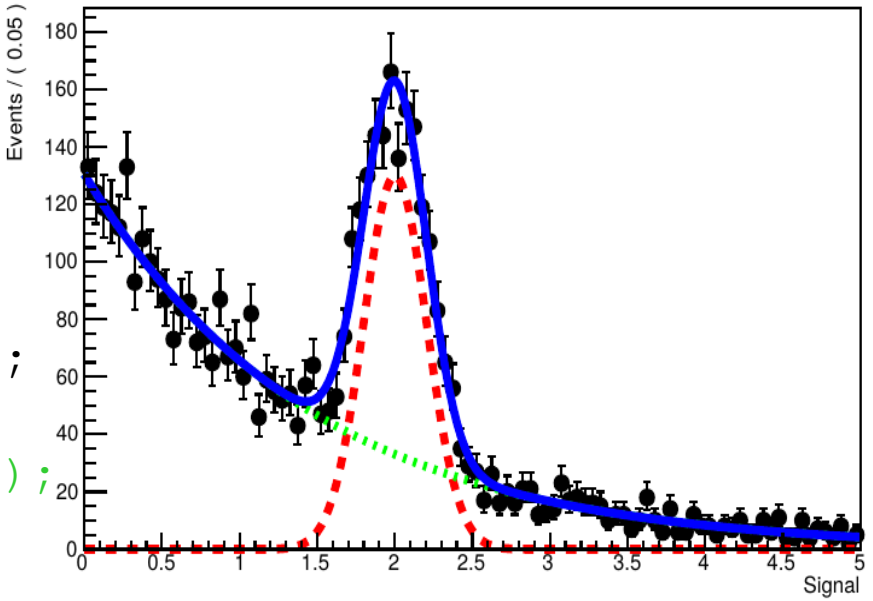
- Fit eines Models an die Daten

```
sig→fitTo(Data);
```

unbinned maximum likelihood fit des Modells `sig` an die generierten Daten `Data`

- Daten und PDF Visualisierung

```
RooPlot *xframe = Signal.frame();
sig->plotOn(xframe);
Data→plotOn(xframe);
Data→plotOn(xframe,Components("g"),
   LineStyle(kDashed),LineColor(kRed));
Data→plotOn(xframe,Components("e"),
  LineStyle(kDotted),LineColor(kGreen));
xframe→Draw();
```



- Daten aus einem ROOT Tree importieren

```
TFile* tf=new TFile("FitTestTTree.root","OPEN");
TTree* myTree  = (TTree*) gDirectory->Get("FitTest");
RooDataSet data("data","data",signal,Import(*myTree));
```

- Daten aus einem text File und aus einem Histogramm importieren

```
RooDataSet * data = RooDataSet::read("data.txt",RooArgList(x,y)));
RooDataHist data ("data","data",signal,Import(*myTH1));
```

RooFit Funktionalität

- Range mit Namen "center" bezüglich der Variablen mass definieren, der im Fit oder in der Integration benutzt werden kann

```
mass.setRange("center",1850.,1870.);
```

- Integrieren eines Modells eSignal, bezüglich der Variablen mass im Bereich center

```
RooAbsReal* sigIntegral =
        eSignal.createIntegral(mass,NormSet(mass),Range("center"));

sigIntegral→ getVal();
```

liefert eine double , die weiter verwendet werden kann

- Fit einer Extended PDF

```
RooGaussian sigMass("sigMass","mass Distr",mass,MeanMass,SigmaMass);
```

Definition einer PDF

```
RooRealVar nSig("nSig","n signal events",2000,0,100000);
```

Definition einer Anzahl von Ereignissen

```
RooExtendPdf eSig("eSig","extended signal D^0",sigMass,nSig);
```

Definition einer Extended PDF

```
RooFitResult* r = eSig.fitTo(*data,Extended(kTRUE),Save()) ;
```

Extended fit der PDF eSig an den Datensatz mit dem pointer *data

RooFit Funktionalität

- Zugang zu den Fit Resultaten

```
RooFitResult* result = Signal→fitTo(Data);
```

https://root.cern.ch/doc/master/classRooFitResult.html

```
result→edm();
result→minNll();
```

-log(L) at minimum

```
result→correlation(par1,par2);
```

correlation between 2 parameters

```
result→Print();
const TMatrixDSym& cor = result->correlationMatrix();
const TMatrixDSym& cov = result→covarianceMatrix();
cov.Print();
```

print covariance matrix

- Zugriff auf Werte und Fehler von rooFit Objekten im Programm

```
double x = MeanMass.getVal();
```

double Wert , der im Pgm weiter verwendet werden kann

```
double dx = MeanMass.getError();
```

double Wert des Fehlers

RooFit Funktionalität

- Zugang zur Likelihood and profile likelihood (RooProfileLL)

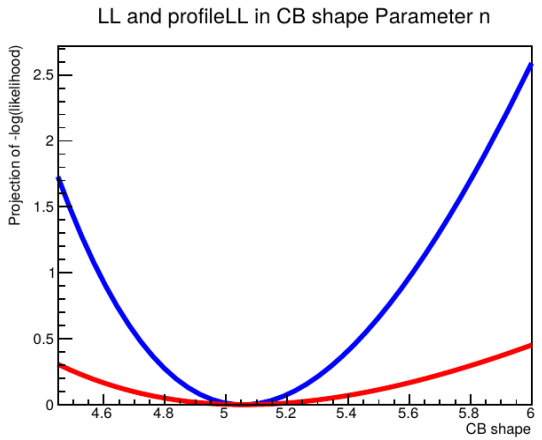
```
RooAbsReal* nll = myModel.createNLL(*data);
```
obtain likelihood of Signal PDF

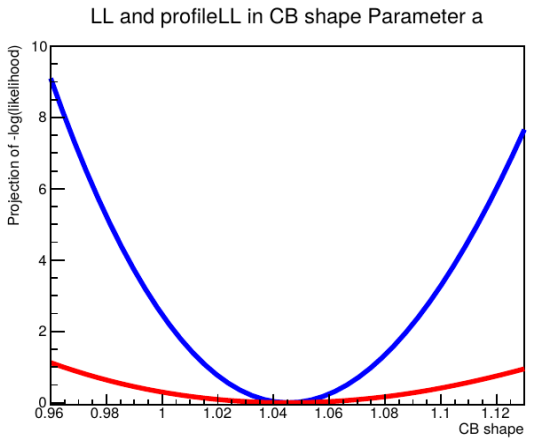
```
RooAbsReal* pll_p = nll→createProfile(Parameter);
```
create a profile likelihood of `nll` with respect to all parameters except for `Parameter`

→ Verwendung näherungsweise Bestimmung von Konfidenz-intervallen über Wilks theorem

- Plot likelihood and profile likelihood

```
RooPlot *LikeFrame0 = Parameter.frame(Bins(10),Range(4.5,6.0),
                                      Title("LL and profileLL in Parameter"));

nll→plotOn(LikeFrame0,ShiftToZero());

LikeFrame0->SetMinimum(0);  LikeFrame0->SetMaximum(10) ;

// Prepare plot for the profile likelihood in Parameter
pll_p->plotOn(LikeFrame0,LineColor(kRed)) ;
```





Rückgabe

$$-ln\frac{L(x|\theta)}{L(x|\hat{\theta})}$$

RooFit Funktionalität

- Minuit Komandos in rooFit ausführen

```
RooAbsReal* nll = myModel.createNLL(*data);          create likelihood model
RooMinimizer m(*nll);                                create Minuit interface
m.setVerbose(kTRUE) ;                                activate control print out
m.migrad();                                          perform Migrad fitting
m.hesse() ;                                          calculate Hesse error from 2nd derivative of nll
par.Print() ;                                        print parameter par
m.minos(par) ;                                       run Minos on parameter par
par.setConstant(kTRUE) ;                             set a parameter constant
RooFitResult* r = m.save() ;                         save current fit status
```

Die Fit Komandos können in beliebiger Konfiguration erneut ausgeführt werden

- rooFit Konfiguration kann mit rooMinimizer verändert werden

https://root.cern.ch/doc/v612/classRooMinimizer.html

RooFit – Fit Resultate

Letzter Schritt des Minuit Fits an eine simulierte D^0 Massenverteilung mit einem Gauss Model

Algorithmus

Ergebnis

Fit result correct

```
[#1] INFO:Minization -- RooMinimizer::optimizeConst: activating const optimization
[#1] INFO:Eval -- RooAbsTestStatistic::initMPMode: started 2 remote server process
…...
 **********
 **    9 **HESSE        1500
 **********
 COVARIANCE MATRIX CALCULATED SUCCESSFULLY
 FCN=-20727.2 FROM HESSE     STATUS=OK             16 CALLS          73 TOTAL
                     EDM=7.372e-05    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   INTERNAL      INTERNAL
  NO.   NAME      VALUE            ERROR          STEP SIZE       VALUE
   1  nSignal      4.99970e+03   7.07075e+01   1.99267e-05  -1.42926e+00
   2  signalMeanMass    1.86462e+03   9.96806e-02   7.92994e-05   4.74751e-03
   3  signalSigmaMass    7.04850e+00   7.04853e-02   2.93520e-05  -2.99613e-01
                                ERR DEF= 0.5
 EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  3    ERR DEF=0.5
  5.000e+03  0.000e+00  0.000e+00
  0.000e+00  9.936e-03  3.586e-07
  0.000e+00  3.586e-07  4.968e-03
 PARAMETER  CORRELATION COEFFICIENTS
       NO.  GLOBAL      1      2      3
        1  0.00000   1.000  0.000  0.000
        2  0.00005   0.000  1.000  0.000
        3  0.00005   0.000  0.000  1.000
```

Aus der 2ten Ableitung im Minimum

RooFit – Fit Optionen und PDFs

- Fit Optionen lassen sich in Form von Parametern der fitTo Methode übergeben

Pointer to Fit Result | PDF | Pointer to Data | Fit Options

```
RooFitResult* result = eSignal.fitTo (data , Optionen);
```

Optionen werden Komma separiert:

`Range("center")`	Fit wird im definierten Bereich mit dem Namen center durchgeführt
`Extended(kTRUE)`	Es wird ein extended Maximum Likelihood fit durchgeführt
`Save()`	Resultate werden in RooFitResult gespeichert
`Timer(kTRUE)`	Timing information wird angezeigt
`NumCPU(Anzahl)`	Einschalten von Multiprozessor fitting
`Minos(kTRUE)`	Minos Error Estimation, links und rechts unterschiedliche Fehler bei asymetrischem Minimum der Likelihood Funktion

- RooFit hat viele vordefinierte PDFs https://root.cern.ch/root/html/ROOFIT_ROOFIT_Index.html

"Gauss" Verteilung mit vermehrten Einträgen bei kleinen Werten:

```
RooCBShape CB("CB","Cystal Ball Function",mass,Mean,Sigma,a,n);
```

Gauss Verteilung mit links und rechts unterschiedlichem Sigma:

```
RooBifurGauss BiFu("BiFu","Bifurcated G.",mass,Mean,Sigma_L,Sigma_R)
```

RooFit – Fit Qualität

Wie gut beschreibt unser Fit mit Minuit innerhalb von RooFit unsere gemessenen Daten?

- PDF Beschreibung muss zu den Daten passen. Ein schlecht beschreibendes Model führt zu einem schlechten fit.
  (PDFs in RooFit werden automatisch normiert)
- Fit Output genau ansehen!
- Qualitätssicherung über die graphische Darstellung von Daten und Fit.

$$residual(N_{sig}) = N_{sig}^{fit} - N_{sig}^{meas} \qquad pull(N_{sig}) = \frac{N_{sig}^{fit} - N_{sig}^{meas}}{\sigma_N^{fit}}$$

```
// Plot the Data, Fit result, residual and pull distribution
RooPlot *xframe0 = mass.frame() ;
data->plotOn(xframe0);
Signal.plotOn(xframe0) ;
xframe0→Draw();

RooHist* massResid    = xframe0->residHist();
RooPlot* xframe1      = mass.frame(Title("Residual Distribution")) ;
xframe1->addPlotable(massResid,"P") ;
xframe1→Draw();

RooHist* massPull    = xframe0->pullHist();
RooPlot* xframe2      = mass.frame(Title("Pull Distribution")) ;
xframe2->addPlotable(massPull,"P") ;
xframe2->Draw();
```

rooSimple.C

Keinen Fit ohne Pull Verteilung zeigen!

Modelierung einer Messung mit RooFit – erste Schritte

In dem File D0Mass.txt und D0CBMass.txt finden Sie wiederholte Messungen der invarianten Masse des D^0 Zerfalls $D^0 \to K^+\pi^-$

Arbeitsvorschlag:

- Schreiben Sie ein RooFit Macro, das die gemessene invariante Masse aus jeweils einem File liest und ein Modell mit einer Gauss PDF an die Daten anpasst.

- Beschreibt der Fit die Daten, fertigen Sie Residual und Pull Verteilung an.

- Probieren Sie unterschiedliche Fit Optionen aus, z.B. Minos

- Definieren Sie einen Signalbereich und bestimmen Sie das Integral

- Verändern Sie ihren Fit, z.B. Bereichseinschränkungen.

- Verwenden Sie andere PDFs, Crystal Ball Funktion, Bifurcated Gauss.

- Stellen Sie die Likelihood bezüglich eines Fit Parameters dar.

- Fertigen Sie einen Likelihood Profile für die Crystal Ball Funktion an und scannen die beiden shape Parameter.

rooReadD0Signal.C

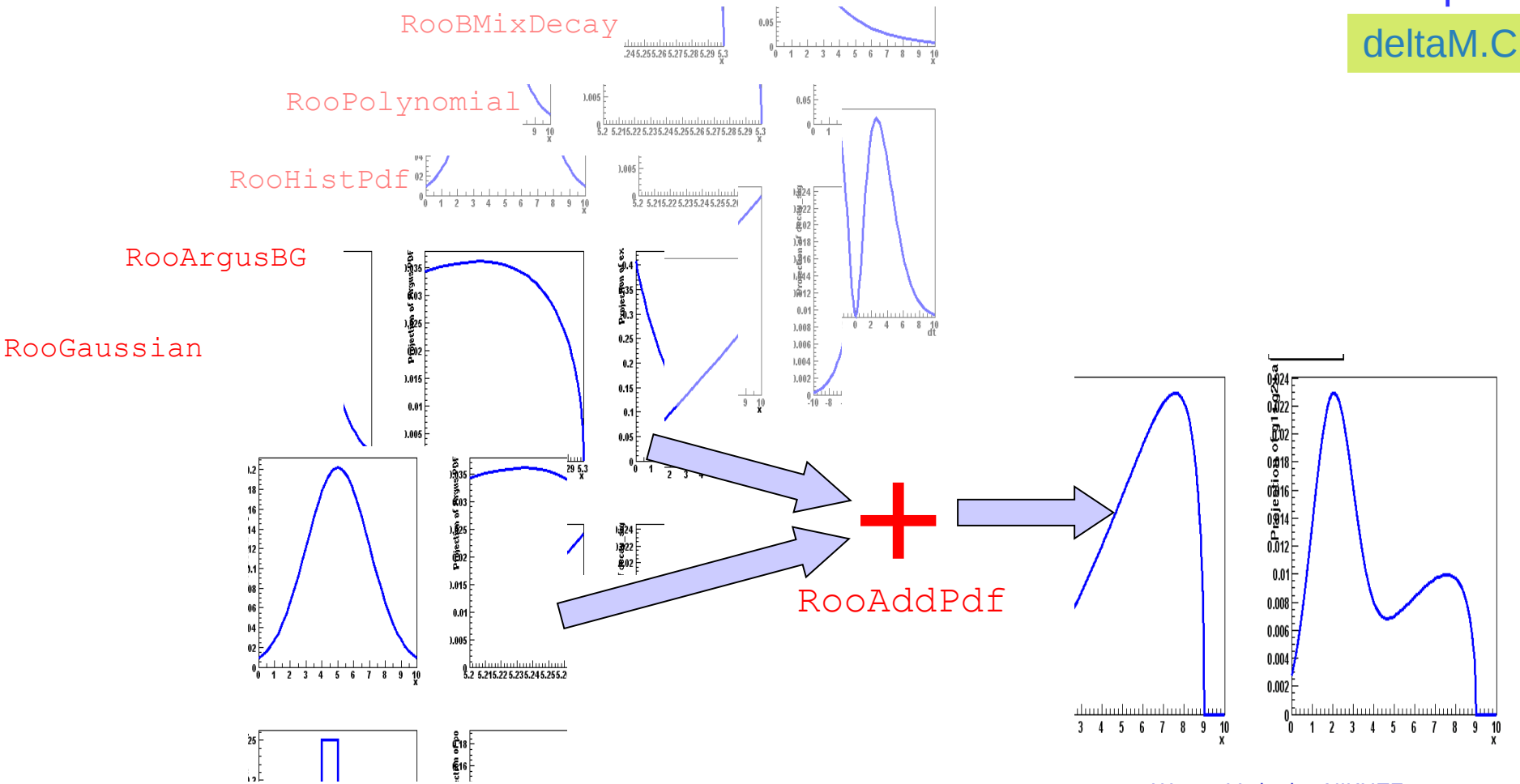
RooFit – RooAddPdf

- Wir betrachten hier ein Model, indem eine Signalverteilung auf einer Untergrundverteilung liegt. In rooFit gibt es neben Signalverteilungen auch vorgefertigte Untergrundverteilungen. Wir summieren Untergrund und Signal PDF

  - Facilitated through operator p.d.f **`RooAddPdf`**

- Beispiel:

  deltaM.C



RooFit – Workspace

- Ist über die Klasse `RooWorkspace` implementiert und ist ein “persistent container”, der alle erzeugten Objekte einer rooFit Anwendung enthalten kann, wie
  - Modellbeschreibung und Konfiguration
  - Datensätze
- Enthält die vollständige Beschreibung des Modells einschliesslich der reflexion Daten
  - Speichern der rooFit Anwendung in einem ROOT File in einer Zeile Quellcode mit von rooFit organisiertem und optimierten streaming der Objekte
  - Einlesen der gespeicherten rooFit Anwendung mit einfachen Methoden für
    - PDFs → `pdf()`
    - rooFit Objekte, z.B. rooRealVar → `var()`
    - Daten → `data()`

    Zugriff auf den workspace über Objektnamen (keine doppelten Objektnamen!)
- Einheitliches Format um Physikresultate zu kombinieren und weiterzugeben
- Quellcode eigener Klassen kann ebenfalls in den Workspace importiert werden
- Stellt die Methode `factory` zur Verfügung, die baumartige Strukturen von rooFit Objekten mit einer eigenen Syntax erzeugt. In einer Zeile werden Variable (`RooRealVar`) mit Gültigkeitsbereichen und Summen, Produkte und Verkettungen von PDFs instanziert.
  - aus 4 Zeilen rooFit code zur Beschreibung einer PDF wird eine Zeile

```
RooRealVar x("x","x",2,-10,10)
RooRealVar s("s","s",3) ;
RooRealVar m("m","m",0) ;
RooGaussian g("g","g",x,m,s)
```

→

```
RooWorkspace w;
w.factory("Gaussian::g(x[2,-10,10],m[0],s[3])")
```

RooFit – Workspace

- `RooWorkspace` und `factory` Methodπe im rooFit Programm

```
#include <RooWorkspace.h>

using namespace RooFit ;
……...
RooWorkspace w;
w.factory("Gaussian::g(x[2,-10,10],m[0],s[3])");
```

Instanzieren des workspace

PDF Definition mit der factory methode

- Erzeugung eines Workspace

```
RooWorkspace w("w","Analyse D->Kππ");
```

- Beim Import einer PDF innerhalb einer rooFit Anwendung werden alle notwendigen Komponenten automatisch importiert

```
w.import(pdf);
```

- Import von Daten

```
w.import(data);
```

- Variable, Funktionen und Daten aus dem Workspace `w` im ROOT File extrahieren

```
TFile* f = new TFile("GausExpModel.root","READ") ;
RooWorkspace* w = (RooWorkspace*) f->Get("w") ;
RooRealVar *x = w->var("x");
RooAbsPdf *pdf = w->pdf("pdf");
RooAbsData* data = w->data("data");
```

Namen der rooFit Objekte

RooFit Workspace

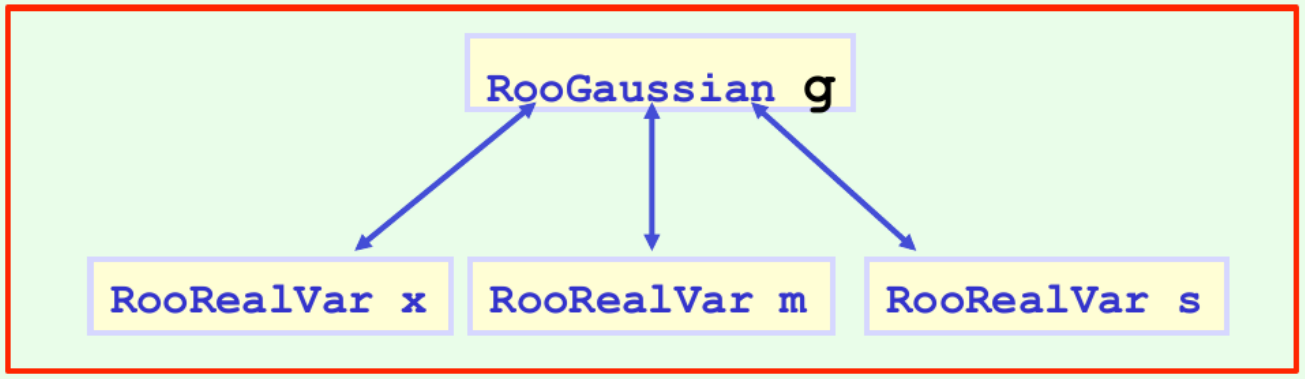
- A special container class owns all objects that together build a likelihood function

Math

$$\text{Gauss}(x,\mu,\sigma)$$

RooFit diagram

RooWorkspace (keeps all parts together)



RooFit code

```
RooRealVar x("x","x",-10,10) ;
RooRealVar m("m","y",0,-10,10) ;
RooRealVar s("s","z",3,0.1,10) ;
RooGaussian g("g","g",x,m,s) ;
RooWorkspace w("w") ;
w.import(g) ;
```

RooFit Factory

- The factory allows to fill a workspace with pdfs and variables using a simplified scripting language

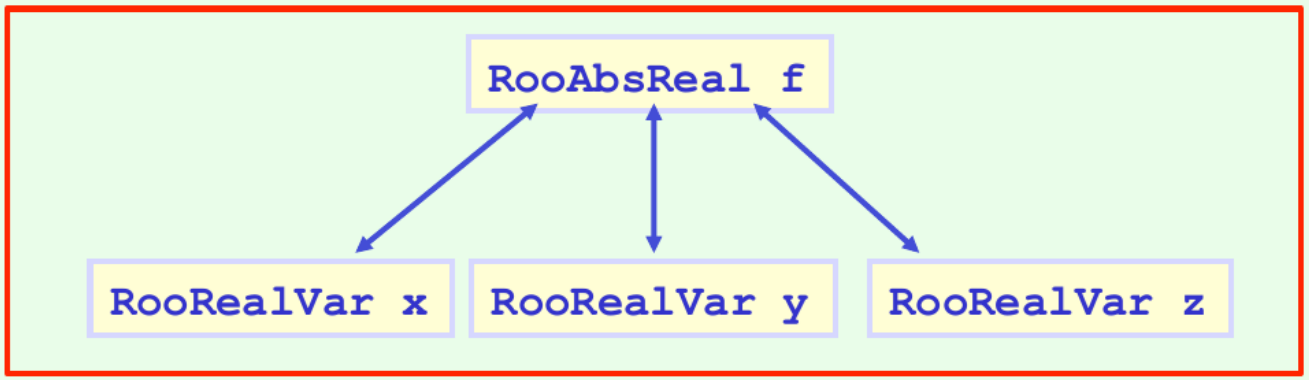
Math

$$\text{Gauss}(x,\mu,\sigma)$$

New feature for LHC

RooFit diagram

RooWorkspace



RooFit code

```
RooWorkspace w("w") ;
w.factory("Gaussian::g(x[-10,10],m[-10,10],z[3,0.1,10])");
```

RooFit – Workspace

- Anzeigen des Workspace Inhalt

```
w.Print();
```

- Workspace `w` in ein File schreiben

```
w.writeToFile("workspace.root");
```

- Synthax der `factory` Methode

```
x[-10,10]          Variable mit Bereich
x[5,-10,10]        Variable mit Wert und Bereich
x[5]               Variable mit Startwert 5
```

- Erzeugen eines PDF Objektes
  - + `Roo` nicht notwendig
  - + Argumente sind meist existierende Objektnamen
  - + Objekte mit Namen brauchen den richtigen Typ → Fehler
  - + Listen und Sets definieren mit { … }

```
w.factory(" …..");
```

```
ClassName::Objectname(arg1,[arg2],…)

Gaussian::myG(x,mean,sigma)
Gaussian::myG(x[-10,10],mean[-10,10],sigma[3])
Gaussian::myG(x[-10,10],0,3)
SUM:model(0.5*Gaussian(x[-10,10],0,3),Uniform(x))
```

Variablenerzeugung ohne vorherige Definition

Variablen brauchen keinen Namen

Jeder prozessierte Ausdruck (Schritt) gibt den Namen des erzeugten Objektes zurück, daher können Argumente ohne vorherige Definition erzeugt werden.

RooFit – Workspace

- Regeln für Expressions

  - interpretierte Ausdrücke zur Definition von PDFs EXPR::pdf_name(‘ausdruck’,variablen)

```
w.factory("EXPR::pdf_name('sqrt(a*x)+b',x,a,b)");
```

  - compilierte Ausdrücke, die während der Ausführung gelinkt werden
    CEXPR::pdf_name(‘ausdruck’,variablen)

```
w.factory("CEXPR::pdf_name('sqrt(a*x)+b',x,a,b)");
```

  - (erneute) Parametrisierung von Ausdrücken expr::w(‘ausdruck’,variable[Bereich])

```
w.factory("expr::w('1-D',D[0,1])");
```

  `expr` erzeugt eine rooFit Function (RooAbsReal)
  `EXPR` erzeugt eine PDF (RooAbsPdf)
  Diese Regeln gelten auch für die factory Ausdrücke SUM, PROD, FCONV, ...

- Extended PDF: Beschreibt ein Model, mit dem ein “extended Fit” ausgeführt wird

```
SUM::model(nSig[0,1000]*Gaussian(x[10,10],0,3),nBck[0,100]*Uniform(x))
```

RooFit – Workspace Extended Fit

- Im extended maximum likelihood fit wird ein Poisson Term zur Likelihood Funktion hinzugefügt. Im Fall einer Signal und Untergrundkomponente ist die Summe der PDFs

  $F(x) = f \cdot S(x) + (1 - f) \cdot B(x)$ mit N als Normierung und f als Signalanteil

- Im extended Fall gehen f, N in $N = N_S + N_B$ über

  $$F(x) = \frac{N_s}{N_s + N_B} \cdot S(x) + \frac{N_B}{N_S + N_B} \cdot B(x)$$

  und die likelihood enthält einen Poisson Term, der von der Gesamtzahl der erwarteten Events abhängt.

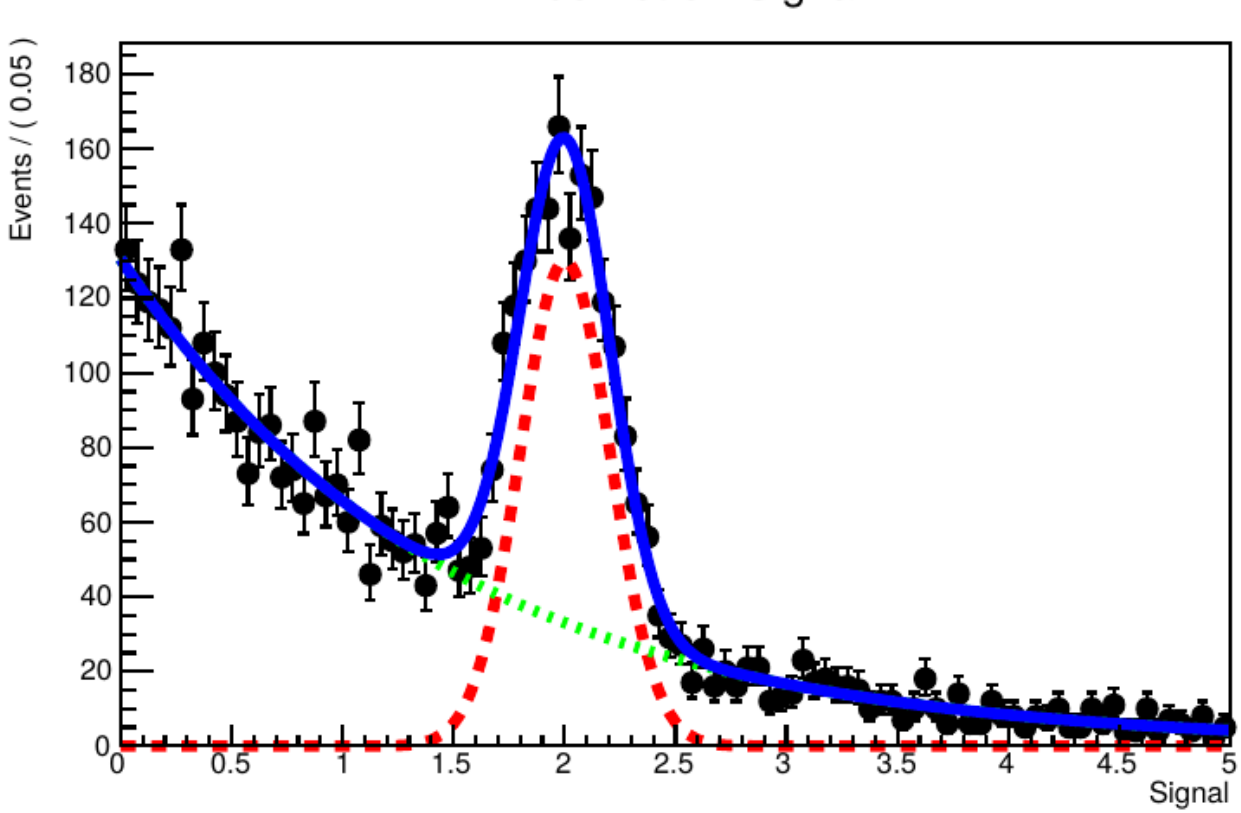
- In workspace Notation schreiben wir einfach

```
SUM::model (Nsig * S, Nbck * B);
```

RooFit – Workspace

- Die Signal PDF `signal` besteht aus 2 Komponeten, die in folgender Weise dargestellt werden

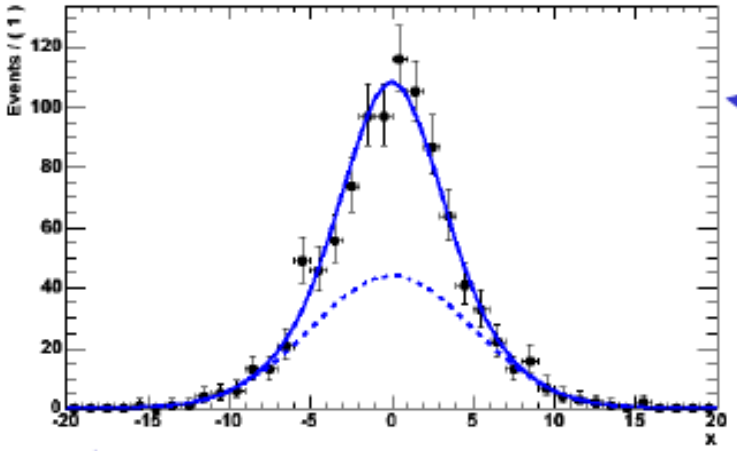
```
workspace->pdf("signal")->plotOn(xframe,Components("gaus"),
                                 LineStyle(kDashed),LineColor(kRed));
workspace->pdf("signal")->plotOn(xframe,Components("exponential"),
                                 LineStyle(kDashed),LineColor(kGreen));
workspace->pdf("signal")->plotOn(xframe);
xframe->Draw();
```



Mitigating fit stability problems

- Strategy I – More orthogonal choice of parameters
  - Example: fitting sum of 2 Gaussians of similar width

$$F(x; f, m, s_1, s_2) = fG_1(x; s_1, m) + (1-f)G_2(x; s_2, m)$$



HESSE correlation matrix

PARAMETER	CORRELATION COEFFICIENTS				
NO.	GLOBAL	[f]	[m]	[s1]	[s2]
[f]	0.96973	1.000	-0.135	0.918	0.915
[m]	0.14407	-0.135	1.000	-0.144	-0.114
[s1]	0.92762	0.918	-0.144	1.000	0.786
[s2]	0.92486	0.915	-0.114	0.786	1.000

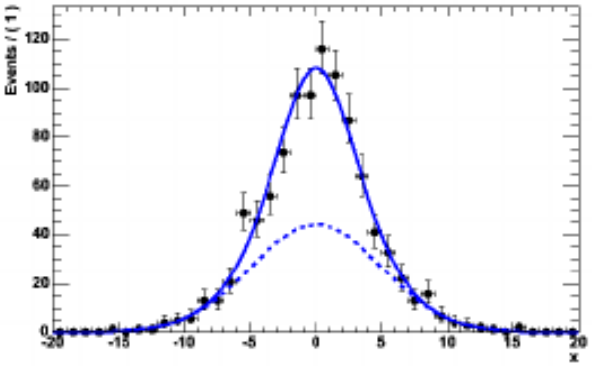
Widths s_1,s_2 strongly correlated fraction f

Mitigating fit stability problems

- Different parameterization:

$$fG_1(x;s_1,m_1)+(1-f)G_2(x;s_1\cdot s_2,m_2)$$

```
PARAMETER  CORRELATION COEFFICIENTS
      NO.   GLOBAL      [f]     [m]    [s1]    [s2]
 [ f]  0.96951    1.000 -0.134  0.917 -0.681
 [ m]  0.14312   -0.134  1.000 -0.143  0.127
 [s1]  0.98879    0.917 -0.143  1.000 -0.895
 [s2]  0.96156   -0.681  0.127 -0.895  1.000
```



- Correlation of width s2 and fraction f reduced from 0.92 to 0.68
- Choice of parameterization matters!

- Strategy II – Fix all but one of the correlated parameters
  - If floating parameters are highly correlated, some of them may be redundant and not contribute to additional degrees of freedom in your model

Modelierung einer Signal + Untergrund Verteilung mit workspace Commands

Schreiben Sie eine rootFit Anwendung unter Verwendung des workspace, das den Tree “FitTest” aus dem ROOT File FitTestTTree.root liest und ein Signal mit Untergrund Modell anpasst.

Arbeitsvorschlag:

- Stellen Sie die Daten, den Fit und die Komponenten der PDF dar und fertigen Sie Residual und Pull Verteilungen an. Ist das Minimum symmetrisch.

- Bestimmen Sie die Anzahl der Signal Ereignisse und den Fehler. rooFitTest.C

- Speichern Sie das Projekt in einem ROOT File und lesen Sie es wieder ein.

rooFitTestreadWsp.C

Toy Monte Carlo Simulation

- Wie können wir sicherstellen, das die bestimmten Fitparameter keinen Bias haben und die Fehlerabdeckung vollständig ist?

  → Benötige einen Rahmen, in dem statistisch unabhängige ‘Pseudo Experimente’ mit einer Bestimmung der Fit-Parameter durchgeführt werden können.

  → Toy Monte Carlo Simulation

  → Pull Verteilung von mehrfach statistisch unabhängigen generierten Daten-sätzen ist eine standard Gauss-Verteilung

- Die Klassen `RooStudyManager` und `RooGenFitStudy` unterstützen Anwendungen bei denen wiederholt Operationen wie Generieren von Daten bezüglich eines Modells und Anpassen der Parameter durchgeführt werden.
  - Logistik → `RooStudyManager`
  - Anpassen an die gewünschte Funktionalität → `RooGenFitStudy`
  - Es können auch andere study Module geschrieben werden, die von `RooAbsStudy` erben
  - Unterstützung von PROOF - The Parallel ROOT Facility - sorgt für das parallele Bearbeiten von ROOT Files auf Clustern oder Viel-Kern-CPUs

- Meist sind eine große Anzahl von Toy Simulationen in einer HEP Analyse notwendig

Run a Toy Monte Carlo Simulation

- Funktion von `RooStudyManager` und `RooGenFitStudy`:

```
RooWorkspace* workspace = new RooWorkspace("w");
workspace->factory("Exponential::sig(signal,s_alpha[-1,0])") ;
```

- Setup simulation based on the model stored in the workspace

```
RooGenFitStudy gfs ;
//configure
gfs.setGenConfig("sig","signal",NumEvents(nEvents)) ;
gfs.setFitConfig("sig","signal",PrintLevel(-1)) ;
gfs.storeDetailedOutput(kTRUE) ;

RooStudyManager mgr(*workspace,gfs);
mgr.run(nToys); // execute nToy toys inline
//mgr.runProof(nToys,"") ; // execute nToys through PROOF-lite

gfs.summaryData()->Print() ;
```

- Summary data of the toy simulation sets is a `RooDataSet` with a column for each fitted par.

```
RooDataSet * myMC = gfs.summaryData();
```

- Pull distribution: demonstrate how to access the toy fit results of the model stored in “w”

```
for (int j = 0 ; j< nToys ; j++ ){
  const RooArgSet* row = myMC->get() ;
  RooRealVar*  xrow = (RooRealVar*)row->find("s_mean");
 //RooRealVar*  xrow = (RooRealVar*) (myMC->get(j)->find("s_mean"));
  PullMeanSignal=(xrow->getVal()-SignalMeanTrue ) / xrow->getError();
}
```

Modelierung einer Signal + Untergrund Verteilung mit workspace Commands

Schreiben Sie eine rootFit Anwendung unter Verwendung des workspace, das den Tree “FitTest” aus dem ROOT File FitTestTTree.root liest und ein Signal mit Untergrund Modell anpasst.

Arbeitsvorschlag:

- Stellen Sie die Daten, den Fit und die Komponenten der PDF dar und fertigen Sie Residual und Pull Verteilungen an. Ist das Minimum symmetrisch.

- Bestimmen Sie die Anzahl der Signal Ereignisse und den Fehler. rooFitTest.C

- Speichern Sie das Projekt in einem ROOT File und lesen Sie es wieder ein.

rooFitTestreadWsp.C

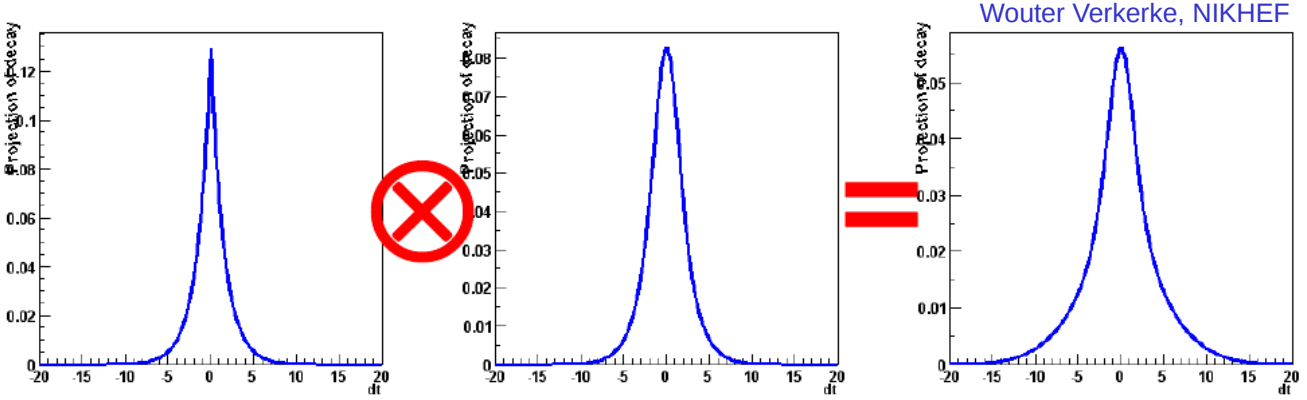
- Generieren Sie 500 toy Experimente mit der Statistik des Datensatzes und stellen Sie mean und sigma der Fits an die Gauss Verteilungen (Signal fit) dar. Erzeugen Sie Pull Verteilungen von beiden Größen. Können Sie Aussagen über Ihren Fit machen?

rooFitTestToySim.C

Faltung von PDFs

- Eine gemessene Observable x enthält von Messgeräten und Messmethoden erzeugte Beiträge. Eine PDF hängt also nicht nur von x ab, sondern auch von einer PDF, die die Auflösung repräsentiert, typischer weise eine oder die Summer mehrerer Gauss-Verteilungen. Die PDF der Messung ist eine Faltung der Observablen und der PDF der Messung.

Wouter Verkerke, NIKHEF



$$O(x;\ \vec{\theta}) \bigotimes R(x;\ \vec{\mu}) \quad = \quad C(x;\ \vec{\theta}, \vec{\mu})$$

$$\frac{1}{N} \int_{-\infty}^{\infty} O(x;\ \vec{\theta}) \cdot R(x - x';\ \vec{\mu}) dx' \quad = \quad C(x;\ \vec{\theta}, \vec{\mu})$$

$$N = \int_{x_{min}}^{x_{max}} \int_{-\infty}^{\infty} O(x;\ \vec{\theta}) \cdot R(x - x';\ \vec{\mu}) dx'$$

- 3 verschiedene Algorithmen sind implementiert
  - Analytische Ausdrücke für einige B Physik PDFs
  - Numerische Faltung mit Hilfe von Fast Fourier Transforms `FCONV` im workspace framew.
  - Numerische Integration des Faltungsintegrals `NCONV`

FFT Vortrag N. Holzwarth

Faltung von PDFs

- Beispiel code für die Faltung einer Landau Verteilung mit einer Gauss-Verteilung. Es wird ein FFT Algorithmus verwendet

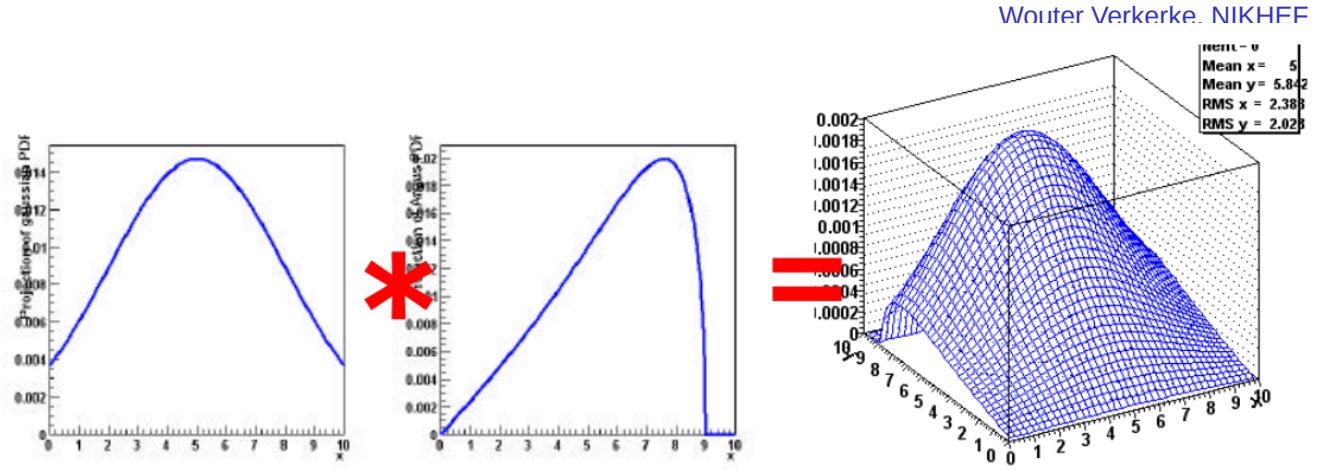
```
…...
RooWorkspace* workspace = new RooWorkspace("w",”Convolution”);

workspace→factory("Landau::lan(t[-10,30],ml[5,-20,20],sl[1,0.1,10])");
workspace→factory("Gaussian::gau(t,mg[1,5],ms[0.5,4.5])");
workspace→factory("FCONV::lxg(t,lan,gau)");
…..
// alternativ ohne workspace, variable und PDFs sind aus dem workspace
// extrahiert
auto t = workspace→var("t");
auto landau = workspace→pdf("lan");
auto gauss = workspace→pdf("gau");
RooFFTConvPdf lxg("lxg", "landau (X) gauss", *t, *landau, *gauss);
```

- Beispiel zur Faltung von PDFs: siehe Lösung der Hausaufgabe Blatt 8 Aufgabe 2

Mehrdimensionale PDFs

- Mehrdimensionale Modelle lassen sich durch Produkte von PDFs mit unabhänigen Variablen konstruieren. → `RooProdPdf`

$$R(x;\ \vec{\theta}) \cdot S(y;\ \vec{\mu}) \cdot ..... \cdot S(z;\ \vec{\nu}) \quad = \quad C(x, y, ..., z;\ \vec{\theta}, \vec{\mu}, ..., \vec{\nu})$$



- Ausserdem können auch Korrelationen berücksichtigt werden $R(x|y) \cdot S(y)$, wobei $R(x|y)$ eine Verteilung in x für einen gegebenen Wert von y beschreibt und die Verteilung von y durch $S(y)$ beschrieben wird.
- Verteilungen der Form $R(x, y) \cdot S(x, z)$ mit einer abhängigen Variablen in beiden PDFs sind nicht von `RooProdPdf` aufgrund einer anderen Normierung unterstützt. Dies kann allerdings mit `RooGenericPdf` realisiert werden.
- Für graphische Darstellungen ist zu beachten, dass `RooPlot* xframe = x.frame() ; Prod→PlotOn(xframe) ;`

$$f(x) = \int pdf(x, y)dy$$

Zusammenfassung

- Acknowledgement: Folien sind partiell aus Vorträgen und Einführungen von Wouter Verkerke (NIKHEF) übernommen.
- Diese kurze Einführung sollte Sie in die Lage versetzen weitere Beispiele der RooFit Tutorial Programme auszuprobieren und zu verstehen.

  https://root.cern.ch/root/html/tutorials/roofit/index.html