GO4FITTER

Sergey Linev

M. Al_Turany
J. Adamczewski
D. Bertini
H.G. Essel

GSI, Darmstadt

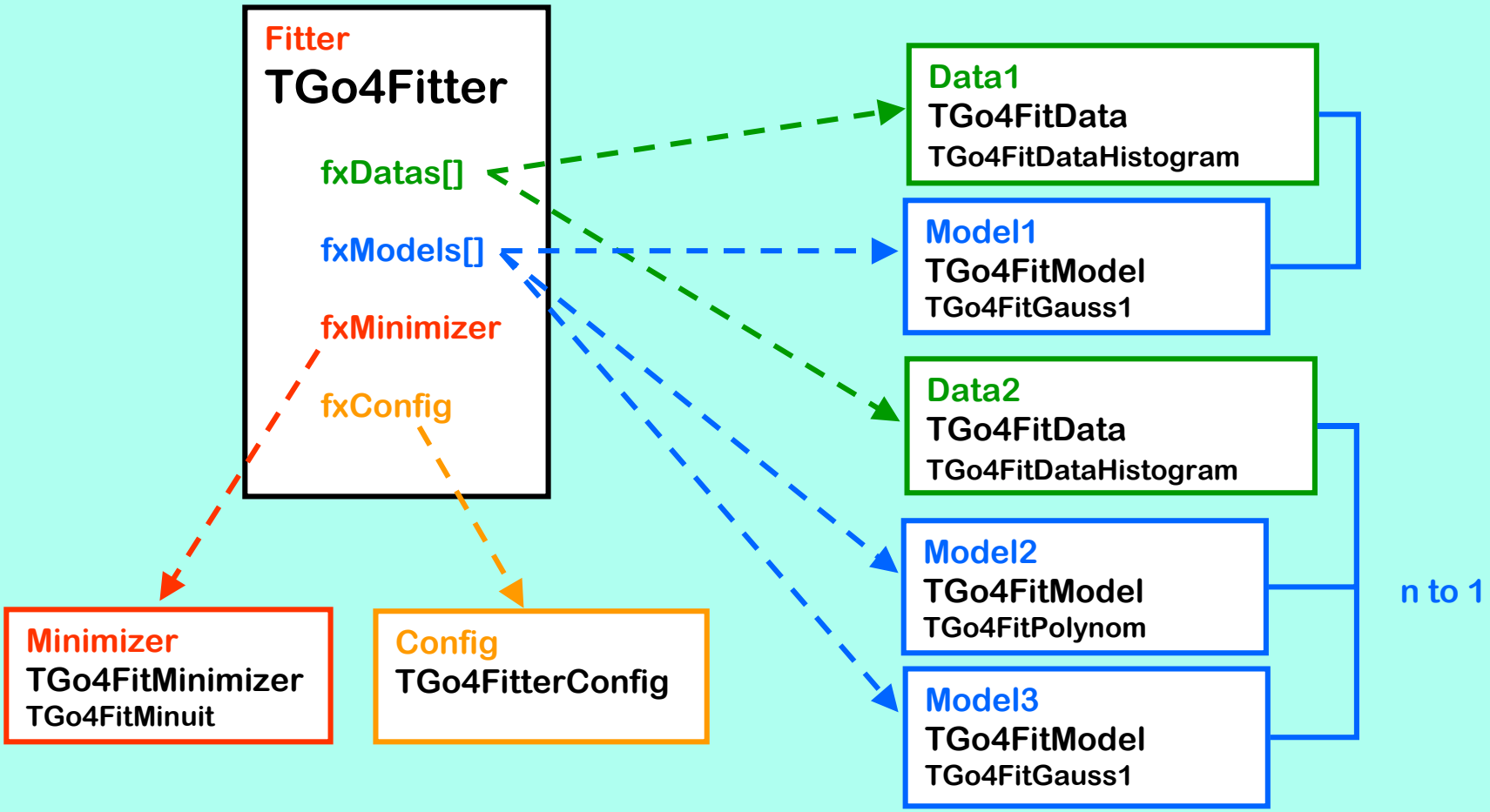
The Go4 Fit Package

- Design almost finished
- Implementation partially finished
- Standard fitting in CINT or compiled
- Independent part of Go4
- GUI to do

Presentation

1. Package overview
2. Example with two Gauss peaks
3. Example adding more Gauss peaks
4. Example combining fits in two histograms
5. Example of user fit function

Object view


Fitter
TGo4Fitter
fxDatas[]
fxModels[]
fxMinimizer
fxConfig
Minimizer
TGo4FitMinimizer
TGo4FitMinuit
Config
TGo4FitterConfig
Data1
TGo4FitData
TGo4FitDataHistogram
Model1
TGo4FitModel
TGo4FitGauss1
Data2
TGo4FitData
TGo4FitDataHistogram
Model2
TGo4FitModel
TGo4FitPolynom
Model3
TGo4FitModel
TGo4FitGauss1
n to 1

Data to fit (TGo4FitData)

- provides method to get bins from data source
- supports N-dimensional case
- defines ranges where data should be fitted
- handle calibrations on each axis

Implementations

- TGo4FitDataHistogram for TH1, TH2, TH3
- TGo4FitDataGraph for TGraph, TGraphErrors

Model component (TGo4FitModel)

- provides method to build shape of single line
- supports N-dimensional case
- assigned to single data object
- handles a list of model parameters
- amplitude can be considered separately from the rest so-called “shape” parameters
- defines range where model is build

Implementations

- TGo4FitPolynom – N-dimensional polynom
- TGo4FitGauss1 – 1-dimensional Gaussian
- TGo4FitFormula – TFormula with up to 3 dimensions
- TGo4FitModelTemplate – external user function
- TGo4FitModelFromData – model bins fills from additional data

Configuration

- sets up ranges for the parameters in minimization
- some of the parameters can be fixed
- defines dependency between parameters like: Par0 = Par1*Par2 or Par3 = Par4
- defines initial values for some parameters

Implementation

- TGo4FitterConfig

Minimizer

- standard TMinuit ROOT class is used
- can be executed any list of Minuit commands
- special command added to store Minuit status and results in separate objects
- any other minimization method can be implemented

Implementation

- TGo4FitMinuit

TGo4Fitter

Central class of the Go4 Fit package

- handles set of data and set of models
- handles minimizer and configurations
- provides access to all objects via names
- provides general interface to set data objects (histograms and graphs) to be fitted
- can be stored with full setup together or without supplied data objects
- provides several fitting functions

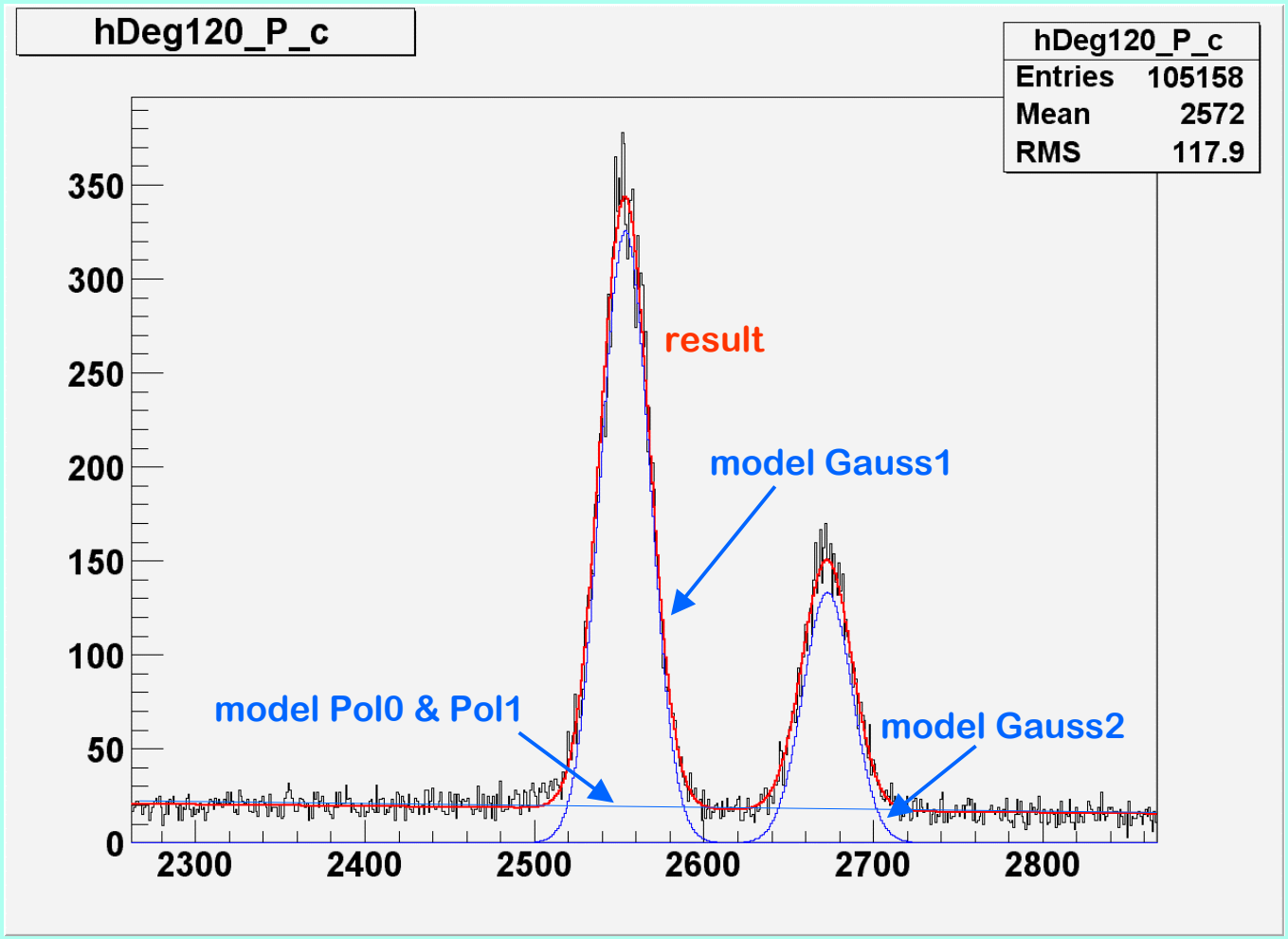
Fitted function

There are six fitting functions supported:

- `fcn_least_squares` $\sum_i (c_i - m_i)^2$
- `fcn_chi_square` $\sum_i \frac{(c_i - m_i)^2}{\sigma_i^2}$
- `fcn_chi_Pearson` $\sum_i \frac{(c_i - m_i)^2}{m_i}$
- `fcn_chi_Neyman` $\sum_i \frac{(c_i - m_i)^2}{\max(c_i, 1)}$
- `fcn_chi_gamma` $\sum_i \frac{(c_i + \min(c_i, 1) - m_i)^2}{c_i + 1}$
- `fcn_ML_Poisson` $2\sum_i (m_i - c_i \ln(m_i))^2$

where c_i – experimental points,
m_i – fitted model,
σ_i – standard deviation.

Example 1


hDeg120_P_c
hDeg120_P_c
Entries 105158
Mean 2572
RMS 117.9
result
model Gauss1
model Pol0 & Pol1
model Gauss2
350
300
250
200
150
100
50
0
2300
2400
2500
2600
2700
2800

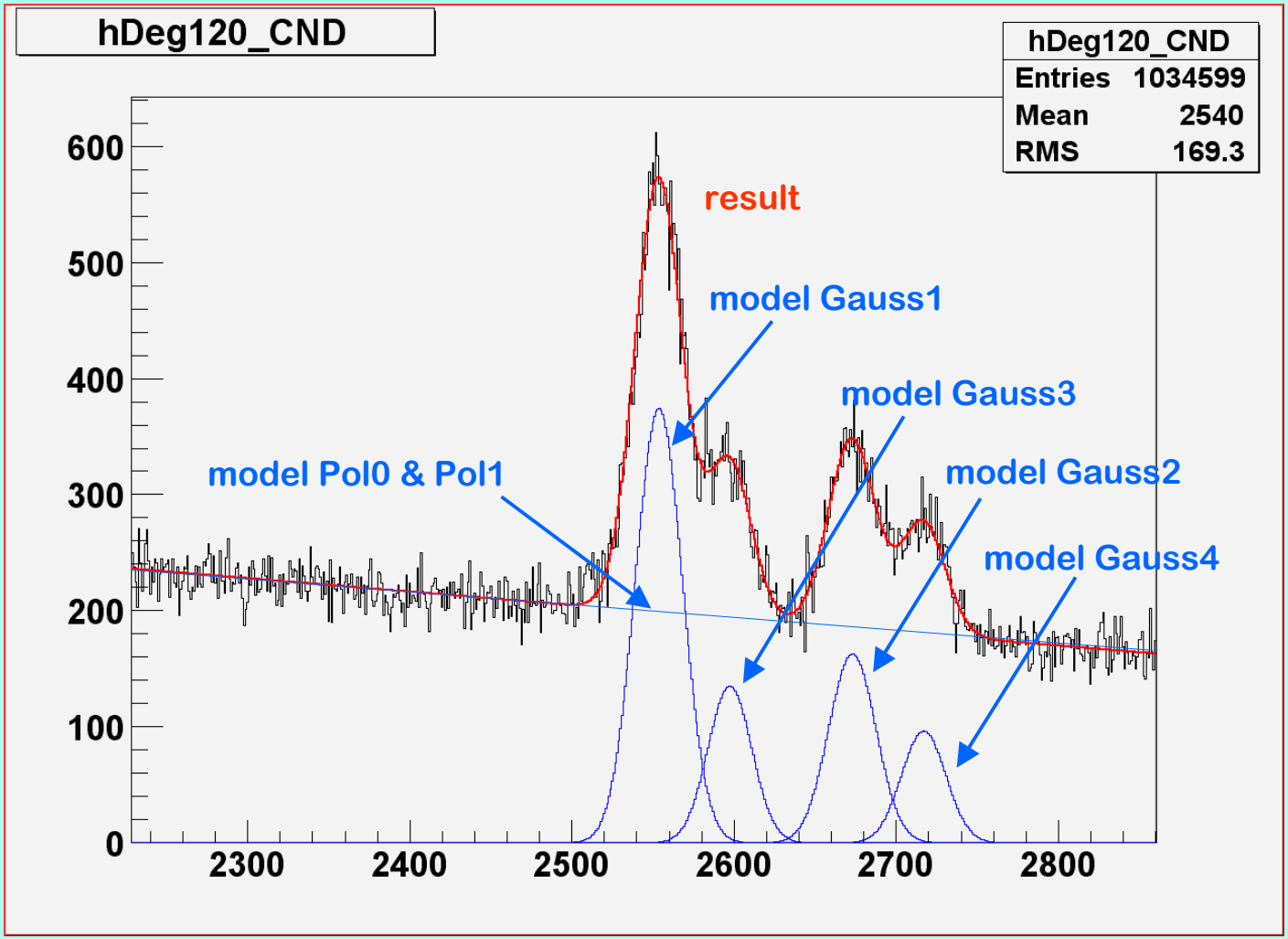
Example 1

```
void Example1() { // Fit two Gauss peaks with background

// create fitter and select method:
   TGo4Fitter* fitter = new TGo4Fitter("Fitter","Example fitter object");
   fitter->SetFCNType(TGo4Fitter::fcn_ML_Poisson); // maximum likelihood
// create minimizer:
   TGo4FitMinuit *fMinuit = new TGo4FitMinuit("Minuit","Minimization object");
   fMinuit->AddCommand("MIGRAD 500 1");            // Command for Minuit
   fitter->SetMinimizer(fMinuit);
// create object to fit (histogram):
   TH1D* histo = GetHistogram("hDeg120_P_c");      // get a histogram
   TGo4FitDataHistogram *data = new TGo4FitDataHistogram("data1",histo);
   data->SetUseBinScale(kTRUE);                    // X-values are bin number
   data->SetRange(0,2200.,2900.);                  // Fit range
// add data object to fitter:
   fitter->AddData(data);
// create four models and add to fitter:
   fitter->AddModel("data1", new TGo4FitPolynom("Pol0",0) ); // for background
   fitter->AddModel("data1", new TGo4FitPolynom("Pol1",1) ); // for background
   fitter->AddModel("data1", new TGo4FitGauss1("Gauss1",350.,2552.8,14.3) );
   fitter->AddModel("data1", new TGo4FitGauss1("Gauss2",150.,2671.6,11.2) );
// Prepare fitter:
   fitter->Initialize();
// Optional optimization:
   fitter->CalculateAmplitudes();
// Fit:
   fitter->DoMinimization();
// Get the result as same type as data object (histogram):
   TH1* result = (TH1*) fitter->CreateResult("data1");
// Close fit (CreateResult off):
   fitter->Finalize();
// Draw data histogram and fit curve(s):
   DrawTwoHistograms("Example",histo,result);

   delete fitter;
}
```

Example 2


hDeg120_CND
hDeg120_CND
Entries 1034599
Mean 2540
RMS 169.3
result
model Gauss1
model Gauss3
model Pol0 & Pol1
model Gauss2
model Gauss4
600
500
400
300
200
100
0
2300
2400
2500
2600
2700
2800

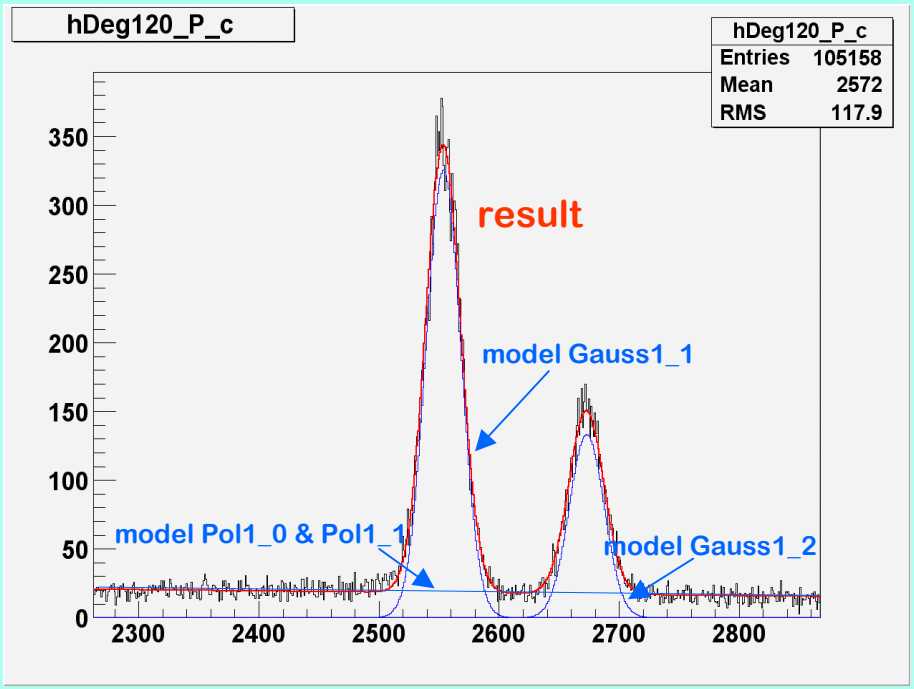
Example 2

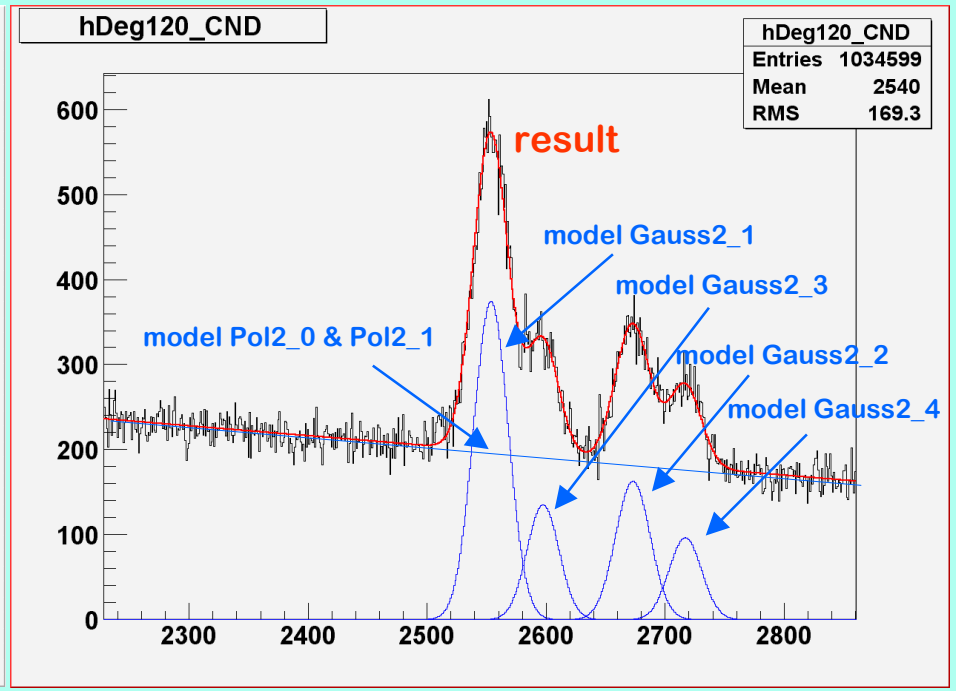
```
void Example2() { // Do a fit, modify it, and fit again
// same as example 1 :
   TGo4Fitter* fitter = new TGo4Fitter("Fitter","Example fitter object");
   fitter->SetFCNType(TGo4Fitter::fcn_ML_Poisson); // maximum likelihood
   TGo4FitMinuit *fMinuit = new TGo4FitMinuit("Minuit","Minimization object");
   fMinuit->AddCommand("MIGRAD 500 1");            // Command for Minuit
   fitter->SetMinimizer(fMinuit);
   TH1D* histo = GetHistogram("hDeg120_P_c");      // get a histogram
   TGo4FitDataHistogram *data = new TGo4FitDataHistogram("data1",histo);
   data->SetUseBinScale(kTRUE);
   data->SetRange(0,2200.,2900.);
   fitter->AddData(data);
   fitter->AddModel("data1", new TGo4FitPolynom("Pol0",0) ); // for background
   fitter->AddModel("data1", new TGo4FitPolynom("Pol1",1) ); // for background
   fitter->AddModel("data1", new TGo4FitGauss1("Gauss1",350.,2552.8,14.3) );
   fitter->AddModel("data1", new TGo4FitGauss1("Gauss2",150.,2671.6,11.2) );
   fitter->Initialize();
   fitter->CalculateAmplitudes();
   fitter->DoMinimization();
   TH1* result = (TH1*) fitter->CreateResult("data1");
   fitter->Finalize();
// Up to here same as example 1
// Now we modify the fitter and the object to be fit
// Add two more models (peaks) to the fitter
   fitter->AddModel("data1", new TGo4FitGauss1("Gauss3", 130.,2596.5,14.8) );
   fitter->AddModel("data1", new TGo4FitGauss1("Gauss4", 100.,2725.0,10.0) );
   TH1D* histo2 = GetHistogram("hDeg120_CND"); // get another histogram
   data->SetHistogram(histo2);                 // fit object is now histo2
   fitter->Initialize();
   fitter->CalculateAmplitudes();
   fitter->DoMinimization();
   TH1* result2 = (TH1*) fitter->CreateResult("data1");
   fitter->Finalize();
   DrawTwoHistograms("Example",histo,result);   // first  fit
   DrawTwoHistograms("Example",histo2,result2); // second fit

   delete fitter;
}
```

Example 3

Combined fit of all models!





Example 3

```
TGo4Fitter * BuildFitter() { // Fit two peaks in two histograms and two more in one of them
   TGo4Fitter* fitter = new TGo4Fitter("Fitter","Example fitter object");
// create first object to fit, but specify histogram later:
   TGo4FitDataHistogram     *data1 = new TGo4FitDataHistogram("data1",0);
   data1->SetRange(0,2.2,2.9);
// Use a linear calibration for data1:
   TGo4FitLinearCalibration *cali1 = new TGo4FitLinearCalibration("cali1","axis calibration");
   cali1->SetCalibrationByRange(3800,0.,3.8);
   data1->SetCalibration(0,cali1,kTRUE);
   fitter->AddData(data1);
   fitter->AddModel("data1", new TGo4FitPolynom("Pol1_0",0) ); // for background
   fitter->AddModel("data1", new TGo4FitPolynom("Pol1_1",1) ); // for background
   fitter->AddModel("data1", new TGo4FitGauss1("Gauss1_1",350.,2552.8,14.3) );
   fitter->AddModel("data1", new TGo4FitGauss1("Gauss1_2",150.,2671.6,11.2) );
// create second object to fit, but specify histogram later:
   TGo4FitDataHistogram     *data2 = new TGo4FitDataHistogram("data2",0);
   data2->SetRange(0,2.2,2.9);
   TGo4FitLinearCalibration *cali2 = new TGo4FitLinearCalibration("cali2","axis calibration");
   cali2->SetCalibrationByRange(3800,0.,3.8);
   data2->SetCalibration(0,cali2,kTRUE);
   fitter->AddData(data2);
   fitter->AddModel("data2", new TGo4FitPolynom("Pol2_0",0) ); // for background
   fitter->AddModel("data2", new TGo4FitPolynom("Pol2_1",1) ); // for background
   fitter->AddModel("data2", new TGo4FitGauss1("Gauss2_1",350.,2552.8,14.3) );
   fitter->AddModel("data2", new TGo4FitGauss1("Gauss2_2",150.,2671.6,11.2) );
   fitter->AddModel("data2", new TGo4FitGauss1("Gauss2_3",350.,2552.8,14.3) );
   fitter->AddModel("data2", new TGo4FitGauss1("Gauss2_4",150.,2671.6,11.2) );
// specify initial values and dependencies between fit parameters:
   TGo4FitterConfig* config = new TGo4FitterConfig("config","fitter configuration");
   config->AddInitialization("Pol1_0.Ampl","20" );
   config->AddInitialization("Pol1_1.Ampl","0"  );
   config->AddInitialization("Pol2_0.Ampl","430");
   config->AddInitialization("Pol2_1.Ampl","0"  );
   config->AddDependence("Gauss2_1.Pos",  "Gauss1_1.Pos"  );
   config->AddDependence("Gauss2_1.Width","Gauss1_1.Width");
   config->AddDependence("Gauss2_2.Pos",  "Gauss1_2.Pos"  );
   config->AddDependence("Gauss2_2.Width","Gauss1_2.Width");
   fitter->SetConfig(config,kTRUE);
   return fitter;}
```

Example 3

```
void Example3() {// Fit two peaks in two histograms and two more in one of them

// create fitter and select method and minimizer:
   TGo4Fitter* fitter = BuildFitter();              // from last slide
   fitter->SetFCNType(TGo4Fitter::fcn_ML_Poisson); // maximum likelihood
// create minimizer:
   TGo4FitMinuit *fMinuit = new TGo4FitMinuit("Minuit","Minimization object");
   fMinuit->AddCommand("MIGRAD 500 1");             // Command for Minuit
   fitter->SetMinimizer(fMinuit);

// create object to fit (histogram):
   TH1D* histo1 = GetHistogram("hDeg120_P_c"); // get a histogram
   TH1D* histo2 = GetHistogram("hDeg120_CND"); // get a histogram
   fitter->SetObject(histo1, "data1");
   fitter->SetObject(histo2, "data2");

// Prepare fitter:
   fitter->Initialize();
// Optional optimization:
   fitter->CalculateAmplitudes();
// Fit:
   fitter->DoMinimization();

// Get the result as same type as data object (histogram):
   TH1* result1 = (TH1*) fitter->CreateResult("data1");
   TH1* result2 = (TH1*) fitter->CreateResult("data2");
// Close fit:
   fitter->Finalize();

// Draw data histogram and fit curve(s):
   DrawTwoHistograms("Example",histo1,result1);
   DrawTwoHistograms("Example",histo2,result2);
// print results
   fitter->Print();

   delete fitter;
}
```