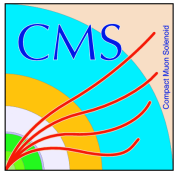
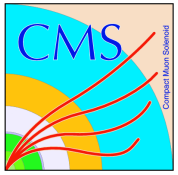


RooFit Tutorial

Jeff Haas
Florida State University
April 16, 2010

Outline

- Purpose
- Structure
  - Basic Classes
- Implementation
  - Toy Monte Carlo
  - Fitting data
  - Fitting options & results

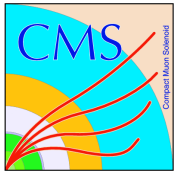

CMS
Compact Muon Solenoid


FLORIDA STATE UNIVERSITY
VIRES
ARTES
MORES
1851

Purpose

Purpose of RooFit

- Model distribution of observables with multiple parameters
  - Physical parameters
    - Position
    - Momentum
  - Detector effects
    - Resolution
    - Efficiency
- In RooFit this is done with a Probability Density Function, PDF


CMS
Compact Muon Solenoid


FLORIDA STATE UNIVERSITY
VIRES
ARTES
MORES
1851

Structure

Structure of RooFit

- Originally developed for the BaBar collaboration
  - Successor of RooFitTools (no longer maintained)
    - ~95% rewrite
- Now in all new CMSSW releases, an add-on to Root
- An object oriented data modeling language
  - Straightforward correlation between mathematics and code
    - Implements classes to represent variables, such as
      - Data points
      - Variables
      - Integrals
      - Likelihood
        - Uses MINUIT in Root
      - Probability Density Function, PDF
        - One can design one's own PDF
  - Regardless of complexity, all classes work the same way for every PDF

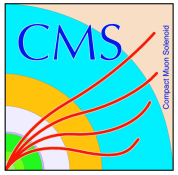
Structure of RooFit continued

- Code is split into two packages
  - RooFitCore
    - Core code
    - Base classes
    - Interface to MINUIT
      - MINUIT performs non-linear optimization with continuous parameters
        - A collection of fortran77 code
      - C-MIMUIT is the interface which enables one to use MINUIT with C++
    - Plotting logic
    - Integrators
    - Everything except PDFs
  - RooFitModels
    - PDF implementations
      - Gauss, Landau, Breit Wigner, ...
        - Written in a format such as RooGaussian
      - RooStats (not covered in presentation)

A Few RooFit Classes

Variable	x	RooRealVar
Space Point	$\vec{x}$	RooArgSet
List of space points		RooAbsData
Function	f(x)	RooAbsReal
Integral	$\int f(x)dx$	RooRealIntegral
PDF	$F(\vec{x}; \vec{p}, \vec{q})$	RooAbsPdf

- PDFs
  - Normalized over x, with respect to the parameters p and q
  - RooFit automatically normalizes for multiple dimensions
    - The user function need not be normalized
  - Limits are MINUIT bounds if variable is parameter



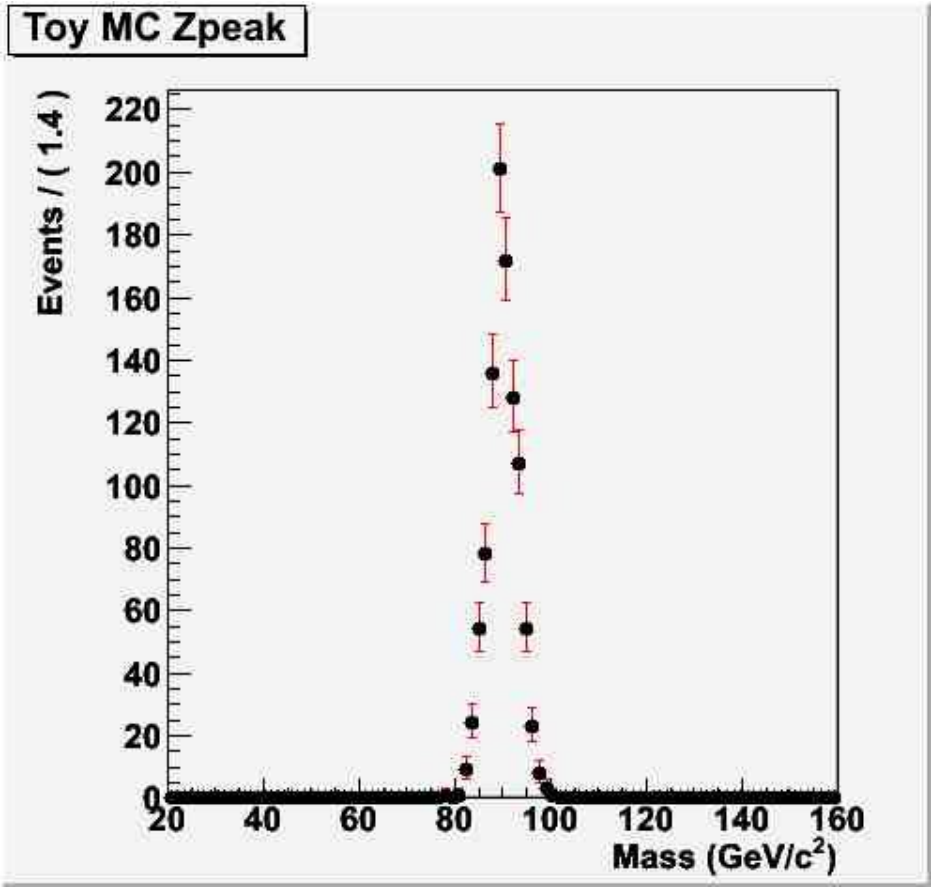
Implementation
Toy Monte Carlo

Toy MC

- Include namespace in code
  - using namespace RooFit;
  - Other headers are required to compile stand alone macro
- Objects representing a “real” value
  - RooRealVar object(“object”, “description”, range, optional);
  - RooRealVar x(“x”, “Mass (GeV/c^{2}”, 20, 160);
  - RooRealVar mean(“mean”, “mean”, 90.0, 20.0, 160.0);
  - RooRealVar sigma(“sigma”, “sigma”, 3.0, 0.1, 20.0);
- Use pre-made class of PDF object, RooGaussian
  - RooGaussian gauss(“gauss”, “gauss”, x, mean, sigma);
- Toy MC
  - RooDataSet* gauss_gen = gauss.generate(x, 1000);

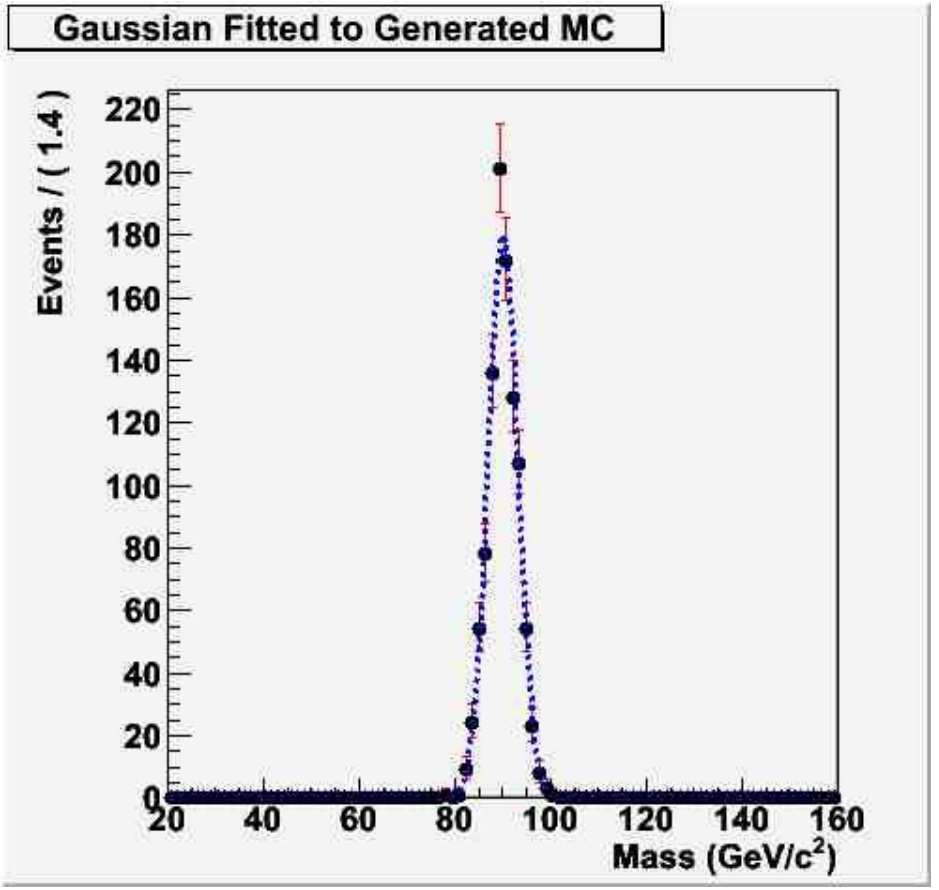
Toy MC continued

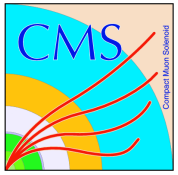
- Plotting
  - RooPlot* xframe = x.frame();
  - gauss_gen->plotOn(xframe, LineColor(kRed));
  - xframe->Draw();
  - TCanvas* c = new TCanvas("c", "", 500, 500);
  - c->cd();
  - gPad->SetLeftMargin(0.18);
  - xframe->GetYaxis()->SetTitleOffset(1.9);
  - xframe->Draw();



Gaussian Fitted to Generated MC

- Fit
  - gauss.fitTo(*gauss_gen);
- Add line to plotting section
  - gauss.plotOn(xframe, LineStyle(2), LineColor(kBlue));





Implementation
Fitting Data

A Couple of Handy PDFs

Gaussian	$\exp\left(-0.5\left(\frac{x-m}{s}\right)^2\right)$	`RooGaussian(name,title,x,m,s)`
Exponential	$\exp(a\cdot x)$	`RooExponential(name,title,x,a)`
Breit-Wigner	$\frac{1}{(x-m)^2+\frac{1}{4}g^2}$	`RooBreigWigner(name,title,x,m,g)`
Crystal Ball	$\left.\frac{(\frac{n}{\lvert a\rvert})^n e^{-\frac{1}{2}a^2}}{(\frac{n}{\lvert a\rvert}-\lvert a\rvert-x)^n}\right\rvert_{x<-\lvert a\rvert}$, $\left.\exp\left(-\frac{1}{2}\left(\frac{x-m}{s}\right)^2\right)\right\rvert_{x>-\lvert a\rvert}$	`RooCBShape(name,title,x,m,s,a,n)`

- Crystal Ball
  - “x” invariant mass
  - “m” mass mean value
  - “s” mass resolution
  - “a” Gaussian tail
  - “n” normalization

Read Histogram & Fit

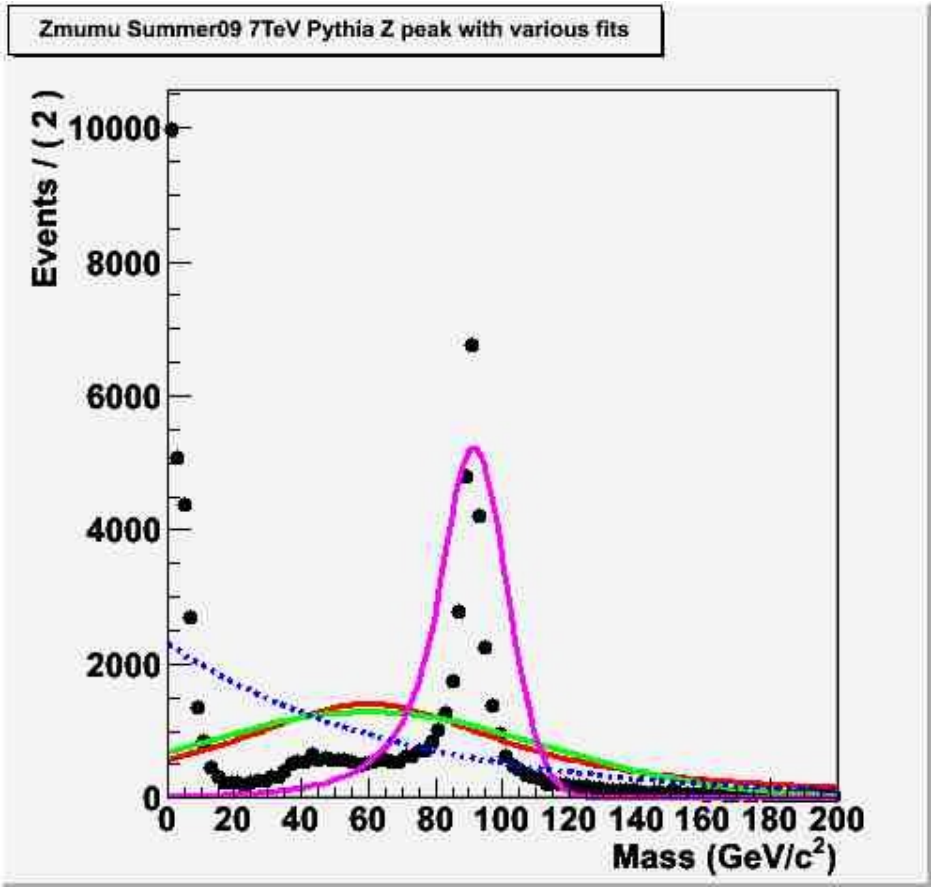
- Read histogram
  - TFile f = new TFile (“myZpeak.root”, “READ”);
  - TH1* h = (TH1*)f → Get(“hist_name”);
- Assign histogram dataset to x object
  - RooDataHist data("data", "dataset", x, h);
- Build PDFs
  - New variable object for exponential
    - RooRealVar a("a", "a", -2.0, 2.0);
  - Exponential PDF
    - RooExponential expo("expo", "exponential", x, a);

Read Histogram & Fit continued

- PDFs continued
  - New objects for Breit Wigner
    - RooRealVar bwmean("bwmean", "bwmean" , 90.0, 20, 180);
    - RooRealVar bwsigma("bwsigma", "bwsigma" , 4.12, 0.1, 100.0);
    - RooRealVar bwsig("bwsig", "signal", 10, 0, 1000000);
  - Breit Wigner PDF
    - RooBreitWigner bwgauss("bwgauss","bwgauss", x, bwmean, bwsigma);
  - New objects for Crystal Ball function
    - RooRealVar cbmean("cbmean", "cbmean" , 90.0, 20, 180.0) ;
    - RooRealVar cbsigma("cbsigma", "cbsigma" , 2.0, 1.0, 40.0) ;
    - RooRealVar cbsig("cbsig", "cbsignal", 10, 0, 1000000);
    - RooRealVar n("n","", 5.1);
    - RooRealVar alpha("alpha","", 1.3);
  - Crystal Ball PDF
    - RooCBShape cball("cball", "crystal ball", x, cbmean, cbsigma, alpha, n);

Fitting Functions Separately

- Fitting
  - bwgauss.fitTo(data);
  - gauss.fitTo(data);
  - expo.fitTo(data);
  - cball.fitTo(data);
- Plotting
  - bwgauss.plotOn(xframe, LineColor(kRed));
  - gauss.plotOn(xframe, LineColor(kGreen));
  - cball.plotOn(xframe, LineColor(kMagenta));
  - expo.plotOn(xframe, LineStyle(kDashed), LineColor(kBlue));



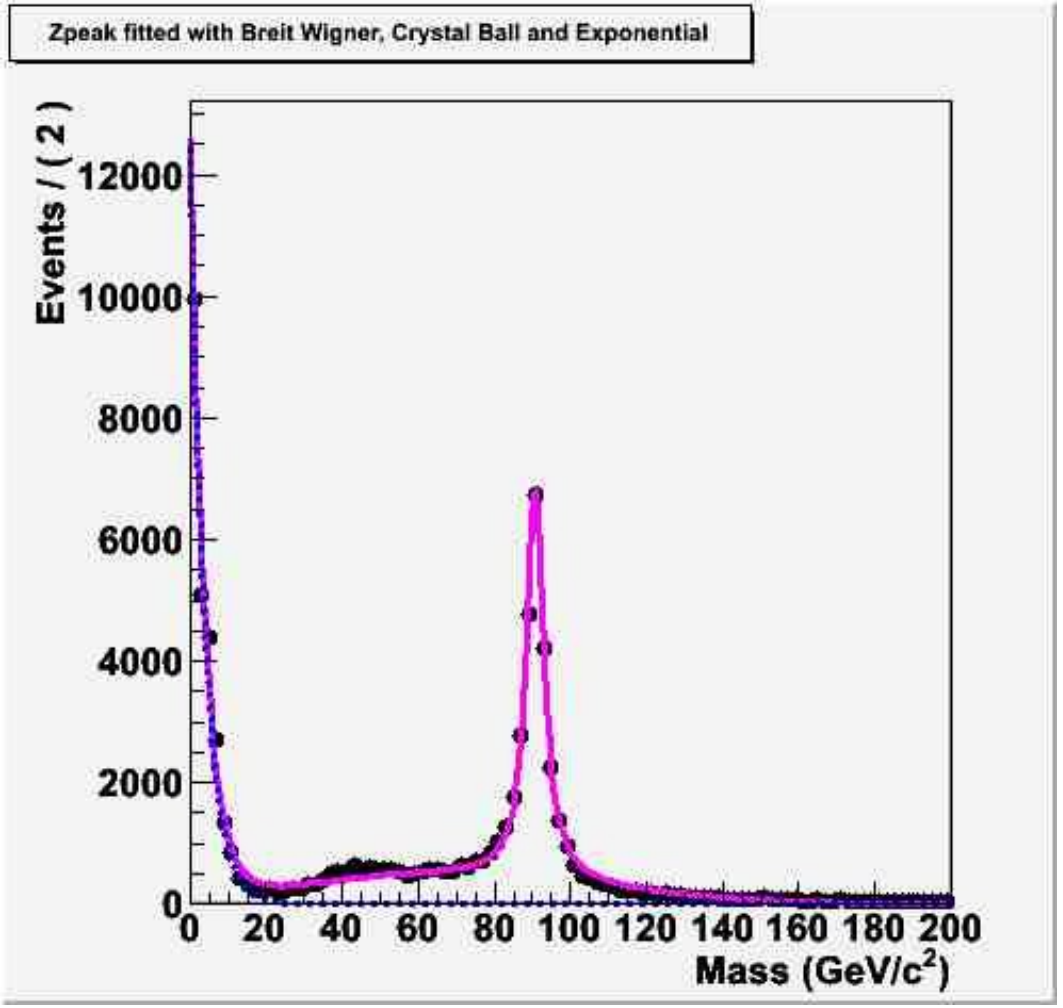
PDFs

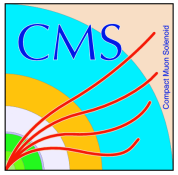
- Methods to combine PDFs
  - Multiplication - RooProdPdf()
  - Convolution – RooNumConvPdf()
  - Composition
    - Substituting a PDF parameter with a function that depends on other observables
  - Addition - RooAddPdf()
- Addition was used in this presentation
  - RooAddPdf sum( );
    - "sum", "", RooArgList(bwgauss, cball, expo), RooArgList(bwsig, cbsig, b)
  - sum.fitTo(data);
  - sum.plotOn(xframe, LineColor(kMagenta));
  - sum.plotOn( );
    - xframe, RooFit::Components(expo), RooFit::LineStyle(kDashed)

A Few Extras

- Filling with color
  - sum → plotOn(xframe, Components(RooArgSet(pdf1)), DrawOption("F"), FillColor(kGreen));
- Plot line between PDFs
  - sum → plotOn(xframe, Components(RooArgSet(pdf1, pdf2)), LineStyle(kDashed));
- Writing to ASCII files from RooArgList()
  - Helpful to load initial values of fit parameters
    - set.writeToFile("config.txt");
    - set.readFromFile("config.txt");

Fitted Zpeak

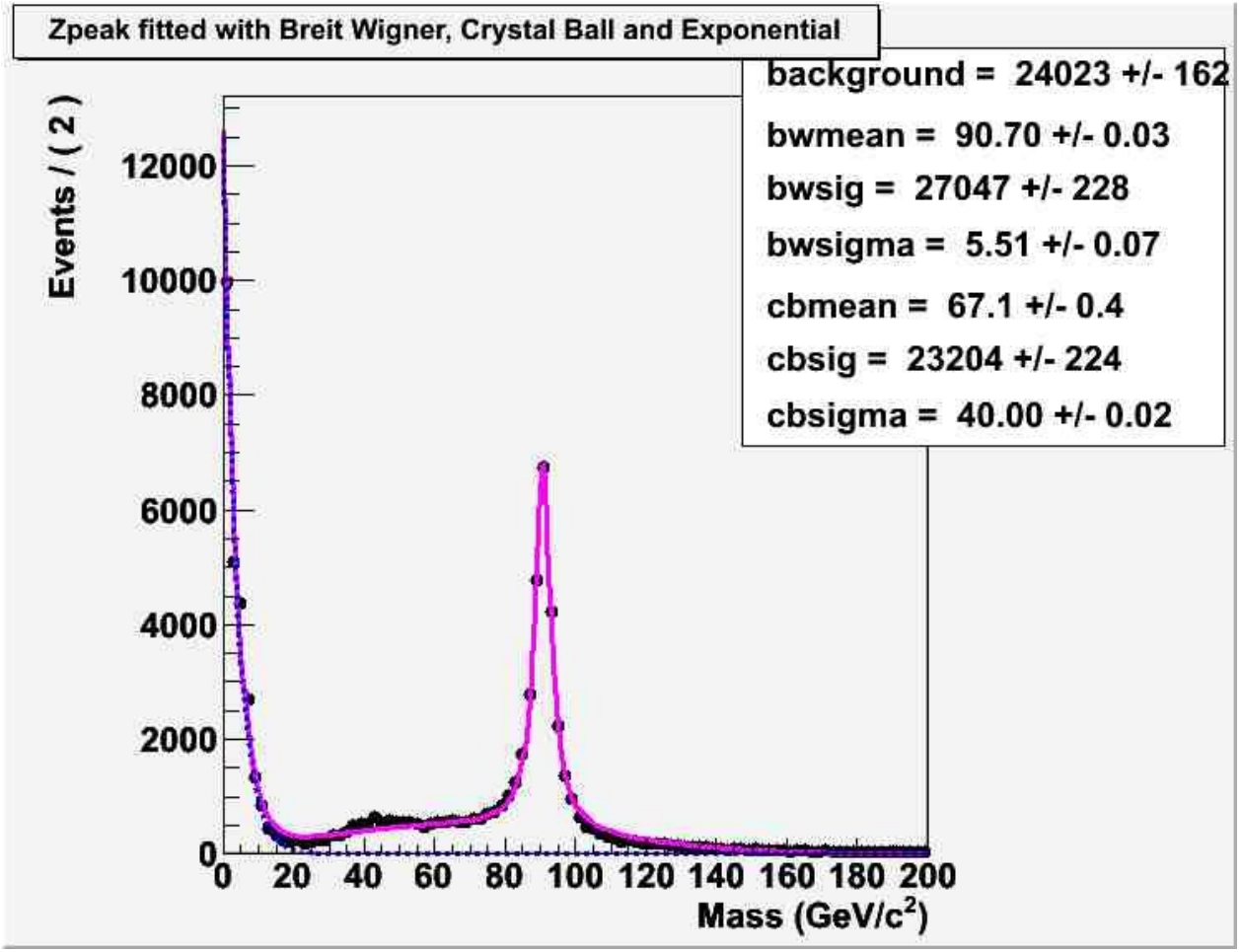


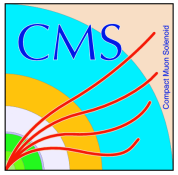


Implementation
Labels

Labels

- RooRealVar b("background", "background yield", 10, 0, 1000000)
- sum.paramOn(xframe, Layout(0.6), Format("NEU", AutoPrecision(1)), Parameters(RooArgList(bwmean, bwsigma, cbmean, cbsigma, bwsig, cbsig, b)));
- Not shown on next plot, but might be helpful
  - data.statOn(xframe);
  - TPaveText* tbox = new TPaveText(0.3, 0.1, 0.6, 0.2, “BRNDC”);
  - tbox → AddText(“Whatever label needs to say”); xframe → addObject(tbox);
  - TArrow* arrow = new TArrow(0, 40, 3, 100); xframe → addObject(arrow);


Zpeak fitted with Breit Wigner, Crystal Ball and Exponential
background = 24023 +/- 162
bwmean = 90.70 +/- 0.03
bwsig = 27047 +/- 228
bwsigma = 5.51 +/- 0.07
cbmean = 67.1 +/- 0.4
cbsig = 23204 +/- 224
cbsigma = 40.00 +/- 0.02
Events / (2)
0
2000
4000
6000
8000
10000
12000
0 20 40 60 80 100 120 140 160 180 200
Mass (GeV/c²)



Implementation
Fitting Options & Results

Fitting Options

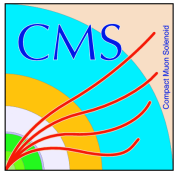
- Binned vs unbinned fit depends on type of dataset
  - Binned
    - RooAbsData* data;
  - Unbinned
    - RooAbsSet* data;
- In order to use MINUIT control options
  - RooAbsPdf* pdf;
  - RooFitResult* fit_resolution;
  - fit_resolution = pdf → fitTo(*data, “<options>”);

Fitting Options

- MINUIT
  - Control Options
    - “m” MIGRAD only, no MINOS
    - “s” estimate step size with HESSE before starting MIGRAD
    - “h” run HESSE after MIGRAD
    - “e” Perform extended MLL fit
    - “0” Run MIGRAD with strategy MINUIT 0
      - Runs faster but with no correction matrix at end
      - Does not apply to HESSE or MINOS, if run afterwards
  - Output Options
    - “q” Switch off verbose mode
    - “l” Save log file with parameter values at each MINUIT step
    - “v” Show changed parameters at each MINUIT step
    - “t” Time fit
    - “r” Save fit output in RooFitResult object
      - Return value is object RFR pointer

Fitting Results

- RooFitResult (A good place to check for fitting issues)
  - Contains
    - Errors, issues
    - Initial and final values of floating parameters and constant parameters
    - Global correlations & full correlation matrix
  - For compact output
    - fit_results → Print();
  - Verbose output
    - fit_results → Print("v");
  - Checking for specific elements
    - fit_results → correlation(variable1, variable2) → Print("v")
  - Output results to a Root file
    - RooPlot* f = new RooPlot(variable1, variable2, x1, x2, y1, y2);
    - fit_results → plotOn(f, variable1, variable2, "plot options"); f → Draw();


CMS
Compact Muon Solenoid


FLORIDA STATE UNIVERSITY
VIRES
ARTES
MORES
1851

Thank You