FITTING TUTORIAL

Using ROOT, TMinuit and RooFit for fitting.

Adrian Bevan
YETI January 2007

Uses ROOT 5.12.00
OVERVIEW

- 3 tutorials over the next two days:
  - Introduction:
    - Introduction to ROOT.
  - Multi Variate Analysis:
    - Training Neural Networks
    - Tools to calculate Fisher discriminants, train neutral networks and boosted decision trees.
  - Fitting:
    - Fitting in ROOT (1): writing your own PDF to fit to.
      - (2): Using TMinuit
      - (3): RooFit
Aims of this Tutorial

- Lean how to define a user function for fitting within ROOT
- Learn how to set up a fit using TMinuit directly.
- Learn how to set up a fit using RooFit.
  - There is insufficient time for this tutorial to cover the use of RooFit, however there is self study material on this subject.

Examples can be downloaded from these URLs or the YETI program page:

http://www.pp.rhul.ac.uk/~cowan/stat_yeti.html
http://www.ph.qmul.ac.uk/~bevan/yeti/index.html
Fitting in ROOT

- You saw how to use the ROOT Fit Panel to fit a pre-defined function to a histogram.
- What if the function you need to fit is not pre-defined within ROOT?
  - Define a function to use in ROOT.
  - Write the function to minimise and use TMinuit directly to minimise the function.
  - Use a higher level fitting package to build a more complicated fit model (e.g. RooFit).
Defining a function to fit in ROOT

Several steps involved

- Write the fit function.
- Instantiate a TF1 object for a 1D fit.
  - Note you can also define 2D (3D) fit function using a TF2 (TF3). If you need to perform a higher dimensional fit, then you need to consider using TMinuit, RooFit or an equivalent fit method.
- Set the initial parameter values, limits etc for the fit object
- Fit the data.
  - You can choose between a $\chi^2$ fit (default) or a $-\ln\mathcal{L}$ fit.

See Chapter 5 of the ROOT 5.12 User Guide for more details
Example: Triple Gaussian

- Sometimes a Gaussian function is insufficient to model both the core and outlier parts of a peaking distribution. The sum of more than one Gaussian might give a better $\chi^2$ than a single Gaussian model.

$$G3(x, N, \mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3) = N \left\{ f_1 e^{-(x-\mu_1)^2 / 2\sigma_1^2} + f_2 e^{-(x-\mu_2)^2 / 2\sigma_2^2} + (1 - f_1 - f_2) e^{-(x-\mu_3)^2 / 2\sigma_3^2} \right\}$$

- Variables to fit are:
  - $N$, the overall normalisation.
  - Three means and widths of the Gaussian functions ($\mu_i$, $\sigma_i$).
  - Two fractions: $f_1$ and $f_2$. 
Example: Triple Gaussian

- Write the fit function for

\[ G_3(x, N, \mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3) = N \left\{ f_1 e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + f_2 e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} + (1-f_1-f_2) e^{-\frac{(x-\mu_3)^2}{2\sigma_3^2}} \right\} \]

/* define the fit function to determine the parameters of the sum of three Gaussians */

/**
 * parameter name      description
 */

0 norm Normalisation. The value of this parameter is dependent on the fit range and choice of histogram binning.

1 mu1 mean of Gaussian 1

2 sig1 width of Gaussian 1

3 mu2 mean of Gaussian 2

4 sig2 width of Gaussian 2

5 mu3 mean of Gaussian 3

6 sig3 width of Gaussian 3

7 frac1 fraction of Gaussian 1

8 frac2 fraction of Gaussian 2

*/

Double_t fitFunc(Double_t * x, Double_t * par)
{
  Double_t PDF = 0.0;
  Double_t g1 = 0.0;
  Double_t g2 = 0.0;
  Double_t g3 = 0.0;

  // Calculate the exponents of the Gaussians
  Double_t arg1 = [par2] != 0.0 ? (x[0] - par[1])/(par[2]) : 0.0;
  Double_t arg2 = [par4] != 0.0 ? (x[0] - par[3])/(par[4]) : 0.0;
  Double_t arg3 = [par6] != 0.0 ? (x[0] - par[5])/(par[6]) : 0.0;

  // add each Gaussian contribution to the PDF
  g1 = exp(-0.5*arg2*arg2)/(par[2]*sqrt(2.0*par[2]));
  g2 = exp(-0.5*arg2*arg2)/(par[4]*sqrt(2.0*par[4]));
  g3 = exp(-0.5*arg3*arg3)/(par[6]*sqrt(2.0*par[6]));
  PDF = par[0] + par[1]*g1 + par[2]*g2 + (1-par[1]-par[2])*g3;

  return PDF;
}

It's a good idea to keep track of the parameter Names when writing the fit function.

Calculate the value of the function G3

Return the calculated value
Example: Triple Gaussian

- Set up the TF1 object for fitting

TF1 * myFitFunc = new TF1("fitFunc", fitFunc, min_range, max_range, npar);

Fit function name to use when you want to fit with this function.

The name of the function to call to evaluate G3

Double_t fitFunc(Double_t * x, Double_t * par)

Variables to fit: i.e. x

Array of parameters

The fit range in x

The number of fit parameters (9 for this example)
Example: Triple Gaussian

- Set up the TF1 object for fitting

TF1 * myFitFunc = new TF1("fitFunc", fitFunc, min_range, max_range, npar);

- Define the parameters, and give initial values

myFitFunc->SetParName(0, "norm");
myFitFunc->SetParName(1, "mul");
myFitFunc->SetParName(2, "sigma1");
myFitFunc->SetParName(3, "mu2");
myFitFunc->SetParName(4, "sigma2");
myFitFunc->SetParName(5, "mu3");
myFitFunc->SetParName(6, "sigma3");
myFitFunc->SetParName(7, "frac1");
myFitFunc->SetParName(8, "frac2");

```
myFitFunc->SetParameters(1500.0, 5, 0.5, 6, 1.5, 5, 5, 0.4, 0.3);
```

**Parameter number**

**Parameter name**

Ordered list of initial parameter values
Example: Triple Gaussian

- Set parameter limits

    myFitFunc->SetParLimits(0, 0.0, 2500.0);
    myFitFunc->SetParLimits(1, 0.01, 10.0);
    myFitFunc->SetParLimits(2, 0.0, 5);
    myFitFunc->SetParLimits(3, 0.01, 10.0);
    myFitFunc->SetParLimits(4, 0.0, 5);
    myFitFunc->SetParLimits(5, 0.01, 10.0);
    myFitFunc->SetParLimits(6, 0.0, 10.0);
    myFitFunc->SetParLimits(7, 0.0, 1.0);
    myFitFunc->SetParLimits(8, 0.0, 1.0);

- Fix any required parameters before fitting

    myFitFunc->FixParameter(5, 5.0);
    myFitFunc->FixParameter(6, 5.0);
**Example: Triple Gaussian**

- **Generate a dataset & fit**
  - A quick way to do this is to get ROOT to generate a data set by filling a histogram

\[
\text{TH1F data_hist("data_hist", ",", 50, min\_range, max\_range);} \\
data_hist.FillRandom(fitFunc, 1500);
\]

- Your fit function
- The number of entries to generate

- Then you can do a likelihood fit to the data

\[
data_hist.Fit("fitFunc", "L");
\]

- The name of your fit function
- Fit options

- You can also fit to a variable in a tree using this function.
Example: Triple Gaussian

- The full list of possible fit options is:

  W  Set all errors to 1
  I  Use integral of function in bin instead of value at bin center
  L  Use Loglikelihood method (default is chisquare method)
  LL Use Loglikelihood method and bin contents are not integers
  U  Use a User specified fitting algorithm (via SetFCN)
  Q  Quiet mode (minimum printing)
  V  Verbose mode (default is between Q and V)
  E  Perform better Errors estimation using Minos technique
  B  Use this option when you want to fix one or more parameters
     and the fitting function is like "gaus","expo","poln","landau".
  M  More. Improve fit results
  R  Use the Range specified in the function range
  N  Do not store the graphics function, do not draw
  O  Do not plot the result of the fit. By default the fitted function
     is drawn unless the option"N" above is specified.
  + Add this new fitted function to the list of fitted functions
     (by default, any previous function is deleted)
Exercises: Triple Gaussian

- Using the f_triple_gaussian.cc macro
  1. Run the macro and note the output parameter values.
  2. Change the fit option to be "" to perform a $\chi^2$ fit. What happens to the parameters and convergence status, and why do you think this happens?
  3. Do the same using MINOS (E) and the Improve solution (M) options & note differences. What option gives the best fit $\chi^2$/degree of freedom?
    - Hint:

$\chi^2$ can be calculated using \( \text{myFitFunc->GetChisquare();} \)
$v$ (#degrees of freedom) \( \text{myFitFunc->GetNDF();} \)
$P(\chi^2, v)$ can be calculated using \( \text{myFitFunc->GetProb();} \)
Exercises: Triple Gaussian

- Using the f_triple_gaussian.cc macro
  1. Run the macro and note the output parameter values.
  2. Change the fit option to be "" to perform a $\chi^2$ fit. What happens to the parameters and convergence status, and why do you think this happens?
  3. Do the same using MINOS (E) and the Improve solution (M) options & note differences. What option gives the best fit $\chi^2$/degree of freedom?
    - Hint:
      - $\chi^2$ can be calculated using $myFitFunc->GetChisquare();$
      - $\nu$ (#degrees of freedom) $myFitFunc->GetNDF();$
      - $P(\chi^2, \nu)$ can be calculated using $myFitFunc->GetProb();$

  4. You can print the covariance and correlation matrices by adding the following line at the end of the macro:

    $gMinuit->mnmatu(1);$

    Make this alteration and re-run the fit. What does this tell you about the fit parameters?
  5. Is there a better way to write the PDF?
Using TMinuit

- Many logical steps involved:
  - Write the function to minimise.
  - Define the global data to fit to.
  - Set up a TMinuit object for fitting.
    - Need to specify parameters (and ranges) to use in the fit.
      - Are parameters allowed to vary in the fit?
      - Are parameters fixed to a constant value in the fit?
    - Need to specify what type of fit to do.
      - MIGRAD?
      - HESSE?
      - MINOS?
  - Recall the results (and test statistic at the fit minimum) for interpretation. [can be read from file or screen]

See Chapter 5 of the ROOT 5.12 User Guide for more details
Example: Unitarity Triangle

- Define the problem to solve.
  - Write down the equations to solve and calculate the function to minimise (e.g. a $\chi^2$):

$$
\chi^2 = \sum_{\text{observables}} \frac{(x_i - \hat{x})^2}{\sigma_i^2}
$$

- Then code into a function with a specific prototype.

- Measured values should be declared with global scope (i.e. outside any function & before they are used).

Measured value
Calculated value (e.g. from model)
Error on measurement
Example: Unitarity Triangle

- Two independent variables define the shape of the Unitarity Triangle: $\bar{\rho}, \bar{\eta}$

\[
\begin{align*}
\rho & = \frac{V_{td} V_{tb}^*}{V_{cd} V_{cb}^*} \\
\alpha & = \tan^{-1} \left( \frac{\rho}{\eta} \right) + \tan^{-1} \left( \frac{1-\rho}{\eta} \right) \\
\beta & = \tan^{-1} \left( \frac{\eta}{1-\rho} \right)
\end{align*}
\]

- And the B-factories have produced measurements of the angles (and sides of this triangle). e.g.

Measurements

$\alpha = (92 \pm 7) ^\circ$

$\beta = (21.2 \pm 1) ^\circ$
Example: Unitarity Triangle

- So we can write the $\chi^2$ to calculate for a given value of the apex co-ordinates:

$$\chi^2 = \frac{(\alpha_{\text{meas}} - \alpha(\bar{\rho}, \bar{\eta}))^2}{\sigma_{\alpha_{\text{meas}}}^2} + \frac{(\beta_{\text{meas}} - \beta(\bar{\rho}, \bar{\eta}))^2}{\sigma_{\beta_{\text{meas}}}^2}$$

- Need to write a function that calculates the $\chi^2$ for a given set of values of the parameters $\bar{\rho}$ and $\bar{\eta}$.

- This fit is coded up in the file yet07/TMinuit/FitUT.cc.

- To compile & run the program:
  
  ```
  cd yeti07/TMinuit/
  gmake
  ./FitUT
  ```
Example: Unitarity Triangle

- **Declare the global data: measurements of** $\beta$ **and** $\alpha$ **(with their uncertainties)**

```c
double deg2rad = M_PI/180.0;
double beta_meas = 21.2*deg2rad;
double dbeta_meas = 1.0*deg2rad;

double alpha_meas = 95.8*deg2rad;
double dalpha_meas = 7.0*deg2rad;
```

- **Write the function to minimise**

```c
extern void chi2(Int_t &npar, Double_t *gin, Double_t *f, Double_t *par, Int_t iflag)
{
    double thechi2 = 0.0;

    // calculate the alpha chi2 contribution for a given set of rhobar and etabar
    double alpha_arg1(0), alpha_arg2(0);
    if(par[1] != 0.0) { alpha_arg1 = par[0]/par[1];
        alpha_arg2 = (1-par[0])/par[1];
    }
    double alpha_chi2 = (alpha_meas - ( atan(alpha_arg1) + atan(alpha_arg2) ))/dalpa_meas;

double beta_arg = 0.0;
    if( (1-par[0]) != 0.0) beta_arg = par[1]/(1-par[0]);
    double beta_chi2 = (beta_meas - atan(beta_arg))/dbeta_meas;

    // add the chi2 contributions together and set the return value
    thechi2 += alpha_chi2*alpha_chi2;
    thechi2 += beta_chi2*beta_chi2;
    f = thechi2;
}
```

An array of the parameters defined in the fit $(\rho, \eta)$

Calculate the $\chi^2$

Equate the variable $f$ with the value of the $\chi^2$ before returning from the function
Example: Unitarity Triangle

- Set up a TMinuit object in the main function

```c++
TMinit *gMinuit = new TMinit(2);
// set the function to minimise with minuit.
gMinuit->SetFCN(chi2);
Double_t arglist[10];
arglist[0] = 1;
gMinuit->mnexcm("SET ERR", arglist, 1, iflag);
```

- Initialise Minuit with a maximum of 2 parameters to minimise
- Set the function to minimise
- Interprets command
- Set the ‘1σ’ tolerance for the change in FCN
- That determines when a function has been minimised
Example: Unitarity Triangle

- Set up a TMinuit object in the main function

```c
TMinit *gMinuit = new TMinit(2);

// set the function to minimise with minuit.
gMinuit->SetFCN(chi2);

Double_t arglist[10];
arglist[0] = 1;
gMinuit->mnexcm("SET ERR", arglist, 1, iflag);

Set the '1\sigma' tolerance for the change in FCN
That determines when a function has been minimised

gMinuit->mnparm(0, "rhobar", 0.2, .1, 0.0, 0.5, iflag);
gMinuit->mnparm(1, "etabar", 0.3, .1, 0.0, 0.5, iflag);
```

```
Interprets command
Set the function to minimise
Set the parameters used in the fit
```

```
Parameter number (start counting from zero)
Parameter name
Start value for the parameter
Step size for parameter
Allowed limits
OK if zero
```

9th January 2007
Adrian Bevan (a.j.bevan@qmul.ac.uk)
Example: Unitarity Triangle

- Set up a TMinuit object in the main function

```c
TMinuit *gMinuit = new TMinuit(2);

// set the function to minimise with minuit.
gMinuit->SetFCN(chi2);

Double_t arglist[10];
arglist[0] = 1;
gMinuit->mnexcm("SET ERR", arglist, 1, iflag);

Set the '1σ' tolerance for the change in FCN
That determines when a function has been minimised

gMinuit->mparm(0, "rhobar", 0.2, .1, 0.0, 0.5, iflag);
gMinuit->mparm(1, "etabar", 0.3, .1, 0.0, 0.5, iflag);

gMinuit->mnexcm("CALL FCN", arglist, 1, iflag);
```

Call the user defined function, to calculate the value FCN, and print the result out to the screen.
Example: Unitarity Triangle

- Set up a TMinuit object in the main function

```cpp
TMinuit *gMinuit = new TMinuit(2);
Initialise Minuit with a maximum of 2 parameters to minimise

// set the function to minimise with minuit.
gMinuit->SetFCN(chi2);
Set the function to minimise

Double_t arglist[10];
arglist[0] = 1;
gMinuit->mnexcm("SET ERR", arglist, 1, iflag);
Interprets command

Set the ’1σ’ tolerance for the change in FCN
That determines when a function has been minimised

gMinuit->mnparm(0, "rhobar", 0.2, .1, 0.0, 0.5, iflag);
gMinuit->mnparm(1, "etabar", 0.3, .1, 0.0, 0.5, iflag);
Set the parameters used in the fit

gMinuit->mnexcm("CALL FCN", arglist, 1, iflag);

Run the minimisation Using MIGRAD

gMinuit->mnexcm("MIGRAD", arglist,2, iflag);
```
MIGRAD vs. HESSE vs. MINOS

- **MIGRAD**

- **HESSE**
  - Calculates the full second-derivative matrix of the user function FCN using a finite difference method. This is often used to improve upon the result obtained by MIGRAD.

- **MINOS**
  - Performs a MINOS error analysis. This can result in different errors than obtained using MIGRAD or HESSE methods. In general one obtains asymmetry errors from MINOS.

  You should use (at least) HESSE after MIGRAD to obtain reliable errors for a given fit result. MINOS will give the best estimate of the errors of a given set of parameters.
Exercises: Unitarity Triangle

- Using the FitUT program
  1. Run the program and note the output values of the apex coordinates and errors.
  2. Run with the –hesse and –minuit options and note how the fitted values and errors change, and what the different options are doing.
  3. What is the significance of the correlation between the two fit parameters?
  4. What do you do when the fit fails to converge?

\[ \chi^2 = (\gamma, \rho, \eta) \]
Exercises: Unitarity Triangle

- Using the FitUT program
  1. Run the program and note the output values of the apex coordinates and errors.
  2. Run with the –hesse and –minuit options and note how the fitted values and errors change, and what the different options are doing.
  3. What is the significance of the correlation between the two fit parameters?
  4. What do you do when the fit fails to converge?
- Try extending the program to include $\gamma = (82 \pm 20) ^\circ$. 

Exercises: Unitarity Triangle

- Using the FitUT program
  1. Run the program and note the output values of the apex coordinates and errors.
  2. Run with the –hesse and –minos options and note how the fitted values and errors change, and what the different options are doing.
  3. What is the significance of the correlation between the two fit parameters?
  4. What do you do when a fit fails to converge?

- Try extending the program to include $\gamma=(82\pm20)\degree$.
  - So add the $\chi^2$ term:
    
    $$\chi^2 = \frac{(\gamma_{\text{meas}} - \gamma(\rho, \eta))^2}{\sigma_{\gamma_{\text{meas}}}^2}$$

    where $\gamma = \text{atan}\left(\frac{\eta}{\rho}\right)$
### Possible commands for mnexcm

**void mnexcm(const char *command, Double_t *plist, Int_t list, Int_t &ierflg)**

<table>
<thead>
<tr>
<th>mnexcm command</th>
<th>equivalent TMinuit function call</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIGrad</td>
<td>Migrad()</td>
</tr>
<tr>
<td>HESse</td>
<td>Migrad()</td>
</tr>
<tr>
<td>MINOs</td>
<td>Mnseek()</td>
</tr>
<tr>
<td>MINimize</td>
<td>Migrad()</td>
</tr>
<tr>
<td>SEEk</td>
<td>Mnseek()</td>
</tr>
<tr>
<td>SIMplex</td>
<td>Mnset()</td>
</tr>
<tr>
<td>SET xxx</td>
<td>FixParameter(Int_t iPar)</td>
</tr>
<tr>
<td>SHOW xxx</td>
<td>Release(Int_t iPar)</td>
</tr>
<tr>
<td>FIX</td>
<td>Mnsimp()</td>
</tr>
<tr>
<td>REStore</td>
<td>Mnsimp()</td>
</tr>
<tr>
<td>RELease</td>
<td>Mnset()</td>
</tr>
<tr>
<td>SCAN</td>
<td>Mnset()</td>
</tr>
<tr>
<td>CONtour</td>
<td>Mnset()</td>
</tr>
<tr>
<td>SAVE</td>
<td>Mnset()</td>
</tr>
<tr>
<td>TOP of pag</td>
<td>Mnscan()</td>
</tr>
<tr>
<td>IMProve</td>
<td>Contour(Int_t npoints=10, Int_t pa1=0, Int_t pa2=1)</td>
</tr>
<tr>
<td>CALI fcn</td>
<td>Mnimpr()</td>
</tr>
<tr>
<td>STAndard</td>
<td>Eval(Int_t npar,Double_t *grad,Double_t &amp;val,Double_t &amp;fval,Double_t *par,Int_t flag)</td>
</tr>
<tr>
<td>END</td>
<td>Mncler()</td>
</tr>
<tr>
<td>EXIT</td>
<td>Mncler()</td>
</tr>
<tr>
<td>RETURN</td>
<td>Mnhelp(), Mnhelp(TString command)</td>
</tr>
<tr>
<td>CLEar</td>
<td>Contour(Int_t npoints=10, Int_t pa1=0, Int_t pa2=1)</td>
</tr>
<tr>
<td>HELP</td>
<td>Mnhel()</td>
</tr>
<tr>
<td>MNContour</td>
<td>Mnhel()</td>
</tr>
<tr>
<td>STOp</td>
<td>Mnimpr()</td>
</tr>
<tr>
<td>JUMp</td>
<td>Mnimpr()</td>
</tr>
<tr>
<td>COVARIANCE</td>
<td>Obsolete</td>
</tr>
<tr>
<td>PRINTOUT</td>
<td>Obsolete, please use SET PRInt</td>
</tr>
<tr>
<td>GRADIENT</td>
<td>Obsolete, please use SET GRAdient</td>
</tr>
<tr>
<td>MATOUT</td>
<td>Obsolete, please use SHOW COVar</td>
</tr>
<tr>
<td>ERROR DEF</td>
<td>Obsolete, please use SET ERRordef</td>
</tr>
<tr>
<td>LIMITS</td>
<td>Obsolete, please use SET LI Más</td>
</tr>
<tr>
<td>PUNCH</td>
<td>Obsolete please use SAVe</td>
</tr>
</tbody>
</table>
Synopsis: A fitting package to facilitate complex likelihood fits within the HEP community. This is an extremely flexible and an extendible set of tools.

RooFit

The RooFit Toolkit for Data Modeling

The RooFit packages provide a toolkit for modeling the expected distribution of events in a physics analysis. Models can be used to perform likelihood fits, produce plots, and generate "toy Monte Carlo" samples for various studies. The RooFit tools are integrated with the object-oriented and interactive ROOT graphical environment.

RooFit has been developed for the BaBar collaboration, a high energy physics experiment at the Stanford Linear Accelerator Center, and is primarily targeted to the high-energy physicists using the ROOT analysis environment, but the general nature of the package make it suitable for adoption in different disciplines as well.

Quick Tour

For a quick overview of what RooFit can do, have a look at the PHYSTAT2005 write-up on RooFit (here), or download the Users Manual (here), or have a look at our (now slightly outdated) 10 page RooFit web slide show.

More information is available at http://roofit.sourceforge.net

9th January 2007 Adrian Bevan (a.j.bevan@qmul.ac.uk)
Basic Types

- **RooRealVar**
  - Used for discriminating variables (x, y, ...).
  - Used for shape parameters (Gaussian width and mean etc.).

- **Probability Density Function types (e.g. RooPolynomial and RooGaussian)**
  - See the RooFit web site for examples and a user guide listing other available PDF types.

- **RooDataSet**
  - Data container that stores information on an entry by entry basis.
  - Can get a TTree from a RooDataSet.
Example: Peak finding

- Fit for a Gaussian peak of known mass on a polynomial background.

Need the following:

- Signal
  - Shape (PDF)
  - Yield

- Background
  - Shape (PDF)
  - Yield

- Data or Monte Carlo simulated data (to fit and plot result of)

The ROOT macro for this exercise is yeti07/fitting/rf_fit_for_peak.cc
Peak finding: Signal PDF

- Need to define the discriminating variable:
  \[
  \text{RooRealVar mass("mass", "Mass", 1, 10, "GeV/c^\{2\}")};
  \]
  Allowed range for the parameter mass

- Then define the signal PDF
  \[
  \text{RooRealVar sig_mu("sig_mu", "mean", 3.15)}; \\
  \text{RooRealVar sig_sigma("sig_sigma", "width", 0.05, 0.0, 0.1)}; \\
  \text{RooGaussian sig_pdf("sig_pdf", "signal pdf", mass, sig_mu, sig_sigma)};
  \]
  Parameter fixed to 3.15
  Parameter set to 0.05
  Parameter range

\[
G(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]
Peak finding: Background PDF

- Use the same discriminating variable as the signal

- Define the background PDF

\[ P = ax + bx^2 + C \]

RooRealVar bg_poll("bg_poll", "linear coefficient", 0.01, -10, 10);
RooRealVar bg_pol2("bg_pol2", "quadratic coefficient", 0.05, -10, 10);
RooArgList bgCoefList;
bgCoefList.add(bg_poll);
bgCoefList.add(bg_pol2);
RooPolynomial bg_pdf("bg_pdf", "background pdf", mass, bgCoefList);

- RooFit defines a polynomial of order \( n \) (\( P_n \)) as

\[ P_n = \sum_{i=1}^{n} a_i x^i + C \]
Peak finding: Total PDF

- Need to define signal and background yields

RooRealVar signalYield("signalYield", ",", 50.0, -10.0, 150.0);
RooRealVar backgroundYield("backgroundYield", ",", 500.0, 0.0, 15000.0);

- And make the total PDF that corresponds to the extended likelihood that we want to fit

RooArgList pdfList;
pdfList.add(sig_pdf);
pdfList.add(bg_pdf);
RooArgList coefList;
coefList.add(signalYield);
coefList.add(backgroundYield);

RooAddPdf TotalPdf("TotalPdf", "Extended PDF", pdfList, coefList);

\[
TotalPdf = N_{signal} \left[ \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \right] + N_{background} \left[ ax + bx^2 + C \right]
\]
Peak finding: Generate & fit a toy data sample

- **Use** *TotalPdf* to generate a data set

  ```cpp
  Int_t nToGen = (Int_t)(signalYield.getVal() + backgroundYield.getVal());
  RooDataSet * data = (RooDataSet*)TotalPdf.generate(RooArgSet(mass), nToGen);
  ```

- **Then fit...**

  ```cpp
  RooFitResult * result = TotalPdf.fitTo(*data, "etr");
  ```

**AVAILABLE FIT OPTIONS:**

- "m" = MIGRAD only (no MINOS)
- "s" = Estimate step size with HESSE
- "h" = Run HESSE after MIGRAD
- "e" = Perform extended \(-\ln L\) fit
- "0" = Run MIGRAD with strategy MINUIT 0 (don’t calculate correlation matrix – not valid if running HESSE or MINOS)
- "q" = Switch off verbose mode
- "l" = Save log file with values at each MINUIT step
- "v" = Show change in parameters at each step
- "t" = Time the fit
- "r" = Save the fit output in a RooFitResult object.
Peak finding: Looking at the data and fit result

- The fit result obtained is printed to the screen:

```text
**********
** 23 **MINOS  2500
**********
FCN=-1817.09 FROM MINOS STATUS=SUCCESSFUL 511 CALLS 682 TOTAL
EDM=3.55819e-06 STRATEGY= 1 ERROR MATRIX ACCURATE
EXT PARAMETER PARABOLIC MINOS ERRORS
NO.  NAME                 VALUE ERROR NEGATIVE  POSITIVE
1  backgroundYield   5.02392e+02   2.25756e+01  -2.22056e+01   2.29511e+01
2  bg_pol1               -1.37144e-01    1.00305e-01   -8.06255e-02   1.34265e-01
3  bg_pol2                4.53396e-02    7.60813e-03   -7.33347e-03   8.40973e-03
4  sig_sigma             4.32667e-02    5.39951e-03   -5.07501e-03   6.83116e-03
5  signalYield           4.75675e+01    7.39504e+00  -7.07024e+00   7.75175e+00
```

Fit worked and errors are sensible

- Get a RooPlot
  - Add the data to the plot
    ```cpp
    RooPlot * frame = mass.frame(50);
data->plotOn(frame);
    ```
  - Add the fitted PDF to the plot with correct normalisation
    ```cpp
    TotalPdf.plotOn(frame);
    ```
  - Display the result on-screen
    ```cpp
    frame->Draw();
    ```
Peak finding: Validating the fit

- \( \chi^2 \) and likelihood fits are intrinsically biased. So the fit needs to be validated before you can trust the result.
  - The first step in validating a fit is to run ensembles of toy MC simulations and compare the results with the fit to data.
  - Then check the pull distribution.

\[
\text{Pull} = \frac{\text{fitted} - \text{input}}{\text{error}}
\]

![Graph showing pull distribution](image)
Peak finding: Validating the fit

- If the fit works then you can compare the value of $-\ln \mathcal{L}$ you obtain in data with that of the ensemble of toy MC experiments that you run.
  - $-\ln \mathcal{L}$ from data should be compatible with the toys or you have a problem!

$$\mathcal{L} = e^{-\left(N_{\text{signal}} + N_{\text{background}}\right)} \prod_{i=1}^{n \text{ events}} N_{\text{signal}} \text{PDF}_{\text{signal}} + N_{\text{background}} \text{PDF}_{\text{background}}$$

Fit or toy has a problem if the data $-\ln \mathcal{L}$ is outside the distribution from toy.
Exercises: Peak Finding

1. Run the macro `yeti07/fitting/rf_fit_for_peak.cc` and look at the resulting plot and fit result.
2. Change the signal yield from 50 events and see what happens.
3. Increase the background yield by a factor of 10, 20, 100 and see what happens to the error on the signal yield.
Exercises: Peak Finding

1. Run the macro `yeti07/fitting/rf_fit_for_peak.cc` and look at the resulting plot and fit result.
2. Change the signal yield from 50 events and see what happens.
3. Increase the background yield by a factor of 10, 20, 100 and see what happens to the error on the signal yield.
4. Run the macro `yeti07/fitting/rf_toy_mc.cc`
   1. What are the mean ($\mu$) and width ($\sigma$) of the pull distribution when you fit a Gaussian to them?
   2. If you increase the number of toy MC studies run to 1000 toys what happens to $\mu$ and $\sigma$?
Summary

- You have seen
  - How to do simple fits using ROOT.
    - Fitting a user defined PDF to a histogram (or data from a TTree).
  - How to write a program that uses TMinuit to solve a problem.
    - Fitting for model parameters given the results of measurements from experiment.
  - How to write a fit using RooFit, and how to use ensembles of toy Monte Carlo simulated data to check that the fit is unbiased.
    - Hunting for a particle of known mass, sitting on a slowly varying background.
- More information on fitting using ROOT and RooFit is available on the web:

  http://root.cern.ch
  http://roofit.sourceforge.net
  http://www.pp.rhul.ac.uk/~cowan/stat_yeti.html
  http://www.ph.qmul.ac.uk/~bevan/yeti/index.html
Answers
Exercises: Triple Gaussian

1. The fit output is: (likelihood fit using MINOS)

   FCN=279.841 FROM MINOS  STATUS=SUCCESSFUL  1633 CALLS  2654 TOTAL
   EDM=1.03679e-07  STRATEGY= 1  ERROR MATRIX ACCURATE

   EXT PARAMETER                  PARABOLIC         MINOS ERRORS
   NO.  NAME      VALUE            ERROR      NEGATIVE     POSITIVE
   1  norm  2.99283e+02  1.21244e+01  -1.30126e+01  1.17130e+01
   2  mu1   5.05660e+00  3.13761e-02  -3.16178e-02  3.12311e-02
   3  sigma1 5.47804e-01  3.52334e-02  -3.45777e-02  3.61277e-02
   4  mu2   5.96774e+00  1.88659e-01  -1.68086e-01  2.22287e-01
   5  sigma2 1.83507e+00  2.55664e-01  -2.55416e-01  2.55664e-01
   6  mu3   5.00000e+00  fixed
   7  sigma3 5.00000e+00  fixed
   8  frac1  4.48193e-01  3.40959e-02  -3.48849e-02  3.48849e-02
   9  frac2  3.01842e-01  8.24768e-02  -7.44924e-02  7.44924e-02

   FCN=56.4376 FROM MIGRAD  STATUS=CONVERGED  421 CALLS  422 TOTAL
   EDM=1.00695e-07  STRATEGY= 1  ERROR MATRIX ACCURATE

   EXT PARAMETER                                   STEP         FIRST
   NO.  NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  norm  2.92910e+02  9.93126e+00  3.54802e-05  -1.74127e-02
   2  mu1   5.04437e+00  3.59062e-02  2.11292e-05  1.36630e-02
   3  sigma1 5.26065e-01  4.85833e-02  6.13261e-05  -9.31252e-03
   4  mu2   5.70132e+00  1.94817e-01  7.92342e-05  8.87536e-03
   5  sigma2 1.40108e+00  2.50419e-01  1.62533e-04  5.24887e-03
   6  mu3   5.00000e+00  fixed
   7  sigma3 5.00000e+00  fixed
   8  frac1  4.21881e-01  6.18272e-02  1.31500e-04  -7.82614e-03
   9  frac2  2.74470e-01  5.47831e-02  2.14242e-04  -5.61992e-03

2. The corresponding $\chi^2$ fit result is

   where most of the parameter error have increased.

3. The results don’t significantly change, but the error is marginally better using these options.
Exercises: Triple Gaussian

4. The correlation matrix is:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLOBAL</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.63008</td>
<td>-0.238</td>
<td>-0.186</td>
<td>-0.157</td>
<td>-0.499</td>
<td>-0.340</td>
<td>-0.224</td>
</tr>
<tr>
<td>2</td>
<td>0.60735</td>
<td></td>
<td>1.000</td>
<td>0.305</td>
<td>0.151</td>
<td>0.539</td>
<td>0.424</td>
</tr>
<tr>
<td>3</td>
<td>0.85179</td>
<td>-0.186</td>
<td>0.305</td>
<td>1.000</td>
<td>0.711</td>
<td>0.575</td>
<td>0.832</td>
</tr>
<tr>
<td>4</td>
<td>0.83930</td>
<td>-0.157</td>
<td>0.151</td>
<td>0.711</td>
<td>1.000</td>
<td>0.545</td>
<td>0.792</td>
</tr>
<tr>
<td>5</td>
<td>0.91961</td>
<td>-0.499</td>
<td>0.539</td>
<td>0.575</td>
<td>0.545</td>
<td>1.000</td>
<td>0.794</td>
</tr>
<tr>
<td>8</td>
<td>0.95900</td>
<td>-0.340</td>
<td>0.424</td>
<td>0.832</td>
<td>0.792</td>
<td>0.794</td>
<td>1.000</td>
</tr>
<tr>
<td>9</td>
<td>0.88240</td>
<td>-0.224</td>
<td>-0.106</td>
<td>-0.641</td>
<td>-0.638</td>
<td>-0.118</td>
<td>-0.595</td>
</tr>
</tbody>
</table>

All of the parameters are correlated with each other. Change one parameter, and the best fit for the all remaining parameters change.

5. Can you write the PDF in a better way? Yes.

\[
G3 = N\left\{f_1 e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + f_2 e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} + (1-f_1-f_2) e^{-\frac{(x-\mu_3)^2}{2\sigma_3^2}}\right\}
\]

The second form of the sum of three Gaussians ranks the contributions from the three Gaussians, which can help the fitter converge to the minimum more quickly than the original implementation of the function. Such a finesse is usually worth making when a function is very heavily used.
Exercises: Unitarity Triangle

- Using the FitUT program
  1. $\bar{\rho} = 0.17 \pm 0.04$, $\bar{\eta} = 0.32 \pm 0.02$
  2. HESSE and MINOS give the same errors as MIGRAD to 2d.p.
  3. The fit parameter correlation is -0.597, so the parameters are highly correlated.

- What do you do when a fit fails to converge?
  a) Try a different start value (in case you are too far from the minimum to converge)
  b) Try to fix a parameter: If parameters are highly correlated, fixing one to a reasonable value can help improve the convergence property of a fit. Do you need to re-write the fit so that it is simpler (less parameters)?

- Include $\gamma = (82 \pm 20)^\circ$, you should fit

$$
\bar{\rho} = 0.15 \pm 0.04, \quad \bar{\eta} = 0.32 \pm 0.02
$$

And the correlation changes to -0.554

- N.B. $180 - \alpha - \beta = 63$ degrees, so the shift in fitted parameters comes from a mild discrepancy ($1\sigma$) between the three angles.
Exercises: Peak Finding

- If you increase the background yield, the error on the signal yield increases
  - For 10K background, the error on signal yield is $\pm 13.0$ events.
  - For 500 background events, the error on signal yield is $\pm 7.4$ events for 500.

- Toys:
  - 100 Toys $\mu_{pull} = -0.14 \pm 0.10$, $\sigma_{pull} = 0.96 \pm 0.07$
  - 1000 Toys $\mu_{pull} = -0.06 \pm 0.03$, $\sigma_{pull} = 1.03 \pm 0.02$
  - 10000 Toys $\mu_{pull} = -0.04 \pm 0.01$, $\sigma_{pull} = 1.03 \pm 0.007$

- Looks biased.
  - The pull distribution is asymmetric because of the Poisson nature of the signal yield. If you increase the signal yield and repeat the toy experiment this bias will be reduced.