Exercise 11: Extended likelihood fits, limits

N. Berger (nberger@physi.uni-heidelberg.de)

7.1.2013

Please send your solutions to nberger@physi.uni-heidelberg.de until 21. 1. 2013, 12:00. Put your answers in an email (subject line *SMIPP:Exercise11*).

1. **Extended maximum likelihood** The usual maximum likelihood method does not allow for determining the absolute normalisation. This can be overcome by using the extended likelihood (for n events, coming from a Poisson distribution with mean ν):

$$L(\nu, \theta) = \frac{\nu^n}{n!} e^{-\nu} \prod_{i=1}^{n} f(x_i; \theta), \tag{1}$$

where θ are the model parameters and x_i the observed values. The logarithm of the likelihood then becomes

$$\ln L(\nu, \theta) = -\nu + \sum_{i=1}^{n} \ln(\nu f(x_i; \theta)) + C, \tag{2}$$

where ν can either be a free parameter or a function of θ. C is independent of ν and θ and can thus be ignored in an optimization.

Perform an extended maximum likelihood fit first fitting to a Gaussian distribution and then fitting to a broad (background) Gaussian distribution with a narrow (signal) Gaussian distribution on top. Use the standard minimizer MINUIT. This you can do in several ways, as MINUIT was included in ROOT in several different variants. The recommended way is to use the MINUIT2 C++ implementation, where you subclass ROOT::Minuit2::FCNBase (which is the objective function, i.e. $-\ln L$), see the following example:

```
#include "Minuit2/FCNBase.h"
#include "TFitterMinuit.h"

#include <vector>
#include <iostream>

class MyFCN : public ROOT::Minuit2::FCNBase {
public:
// Constructor - here you could e.g. pass in a histogram
  MyFCN(...) {...}
// The actual function, x contains the fit parameters
// - should return chi2 or likelihood
  double operator() (const std::vector<double> & x) const {
```

```
    ...
  }
  // Change in function corresponding to 1 sigma error
  // - 1.0 for chi2 / 0.5 for lieklihood
  virtual double Up() const;
};

int testMinimize() {
  TFitterMinuit * minuit = new TFitterMinuit();

  MyFCN fcn;
  minuit->SetMinuitFCN(&fcn);
  // starting values
  double startX = -1.2;
  double startY = 1.0;
  // if not limited (vhigh <= vlow)
  minuit->SetParameter(0,"x",startX,0.1,0,0);
  minuit->SetParameter(1,"y",startY,0.1,0,0);
  minuit->SetPrintLevel(3);
  // create Minimizer (default is Migrad)
  minuit->CreateMinimizer();
  int iret = minuit->Minimize();
  if (iret != 0) {
     return iret;
  }
 }
```

Ensure that the fit reproduces your chosen imput values.
(Attach code)

2. **The CL_s method** Read *Modified frequentist analysis of search reslts* by A.L. Read, available at
`http://cdsweb.cern.ch/record/451614/files/p81.ps.gz?version=1`.
This is the method used to present limits in searches by both the LEP and the LHC experiments. Try to answer the following questions:

   - What is *coverage*?
   - What is *flip-flopping*?
   - In this context, what is *conservative*?
   - What happens if the background estimate is too large?
   - What if is too small?
   - Would you consider yourself a Bayesian or a Frequentist? Why?

3. **The CL_s method in root** The CL_s method is implemented in the root class `TLimit`. Assume you do a search in a simple counting experiment, where you expect 100 background events. Using `TLimit`, answer the following questions:

   - If you expect 10 signal events and observe a total of 90, 100, 110 events, what are your expected and observed confidence levels for the signal plus background hypothesis using the CL_s method?
   - If you observe 100 events with the same background expectation, what signal size can you exclude at the 95% confidence level?
   - How much would you have to reduce the background to be able to exclude a 10 event signal at the 95% confidence level?

   (Attach code)