Exercise 10: χ^2 and likelihood fits

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

12.12.2011

Please send your solutions to nberger@physi.uni-heidelberg.de until 9. 1. 2011, 12:00. Put your answers in an email (subject line *SMIPP:Exercise10*).

1. **χ^2 distribution** Fill n histograms with m bins between 0 and 1 with k events. Then calculate the χ^2 for each histogram with regards to a flat PDF with an expectation of k/m entries in each bin. Assume Gaussian errors ($\sqrt{N_{entries}}$) for the bin contents (i.e. choose $k/m >> 1$). For your ensemble of n histograms, plot χ^2 and χ^2/NDF as a function of m (In this case, the number of degrees of freedom NDF is $m-1$).
(Attach code and suitable plots)

2. **χ^2 minimization** Again fill a histogram with m bins between 0 and 1 with k events. Now try to extract $a = k/m$ by minimizing the χ^2 in a scan of a. Also give errors (look for a $\Delta\chi^2$ of 1).
(Attach code and suitable plots)

3. **Using a minimizer** Repeat exercise 2, but this time using 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`, 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
  // - 1.0 for chi2 / 0.5 for likelihood
};
```

```
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;
  }
 }
```

4. **Lifetime fit: χ^2**: Generate a histogram with N events in an exponential (lifetime) distribution with a lifetime τ of 1 ns. Write a `ROOT::Minuit2::FCNBase` derived class calculating a χ^2 from this histogram with regards to an exponential distribution with two free parameters, namely normalisation and lifetime. Use Minuit to determine these parameters.

5. **Lifetime fit: Binned likelihood**: Again generate a histogram with N events in an exponential (lifetime) distribution with a lifetime τ of 1 ns. Write a `ROOT::Minuit2::FCNBase` derived class calculating the negative (because Minuit searches minima, not maxima) logarithm of the likelihood from this histogram with regards to a normalised exponential distribution with one free parameter, namely the lifetime. The binned log likelihood is $\sum_{b=1}^{M} n_b \log f_b$, where the sum is over the bins, n_b are the bin contents and f_b is the function value for the bin (which you can approximate by the value at the bin center). Use Minuit to determine the lifetime parameter.

6. **Lifetime fit: Unbinned likelihood**: Generate N events in an exponential (lifetime) distribution with a lifetime τ of 1 ns. Write a `ROOT::Minuit2::FCNBase` derived class calculating the negative (because Minuit searches minima, not maxima) logarithm of the likelihood from these events with regards to a normalised exponential distribution with one free parameter, namely the lifetime. Use Minuit to determine the lifetime parameter.

7. **Comparison**: Compare the performance of the methods described above for different N and different bin sizes.