Parameter Estimation Part C

The technicalities: MINUIT & etc.

Christoph Rosemann

DESY

February 23.,2015

MINUIT

MINUIT is a program to calculate numerically

- a function minimum of $F(\vec{a})$, of (max. 50) parameters a_i
- the covariance matrix of these parameters
- the (asymmetric or parabolic) errors of the parameters from $F_{\min} + \Delta$ for arbitrary Δ
- the contours of parameter pairs a_i, a_j

Author is Frank James

- originally in FORTRAN (which still shows)
- adapted in ROOT to C++

MINUIT

MINUIT contains algorithms to solve complex problems if presented correctly
Info and documentation:

- html
  `http://wwwasdoc.web.cern.ch/wwwasdoc/WWW/minuit/minmain/tableofcontents2_2.html`
- list of MINUIT commands
  `http://wwwasdoc.web.cern.ch/wwwasdoc/WWW/minuit/minmain/chapter2_8.html`

Usage of MINUIT

- include ROOT header
  `#include <TMinuit.h>`
- compile with root libraries
- typical order of commands:
  - read in data
  - define function to minimize
  - initialize MINUIT
  - steering commands for execution
  - obtain results

Function definition

The function to minimize has to be defined

- The name is passed in the initialization (here `fcn`)
  `minuit.SetFCN(fcn);`
- The arguments are predefined:

```
void fcn(int &npar, double *gin, double &f,
         double *par, int iflag)
```

`npar`	number of parameters
`gin`	partial derivatives (return values)
`f`	function value (return value)
`par`	parameter values
`iflag`	flag word

Usually only `f` and `par` are important

Initialization

- include the header
  `#include <TMinuit.h>`
- in the main() routine the following functions have to be called:
  1. number of parameters that are to be found (minimized):
     `TMinuit minuit(1);`
  2. function `fname`:
     `minuit.SetFCN(fname);`
  3. every parameter has to be defined the following way:
     `minuit.mnparm(num,name,start_value,step_size,`
     `l_bound,u_bound,err_flag);`

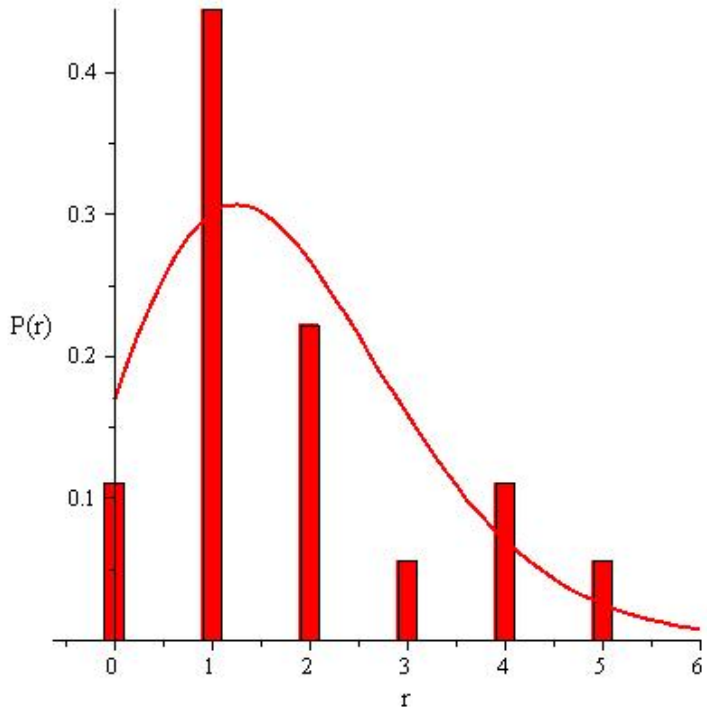
Repeat: Numerical example

A set of rate measurements at fixed intervals of a radioactive source yielded

$$r_i = [1, 1, 5, 4, 2, 0, 3, 2, 4, 1, 2, 1, 1, 0, 1, 1, 2, 1]$$

Assume a Poisson distribution

Better check: histogram the values and compare it with a Poisson. The estimated, best value for the mean is $\mu = \frac{1}{n}\sum_i^n r_i = 1.78$ the estimated uncertainty from this is $\sigma_\mu = \sqrt{\mu/n} = 0.31$



Looks OK!

Continue numerical example

- Estimated mean is $\mu = \frac{1}{n}\sum_i^n r_i = 1.78$
- In the parabolic approximation the uncertainty is $\sigma_\mu = \sqrt{\mu/n} = 0.31$
- For finding the *true* parameter uncertainty, solve the actual Likelihood function for the intersection points with $\ell_{min} + \frac{1}{2}$:

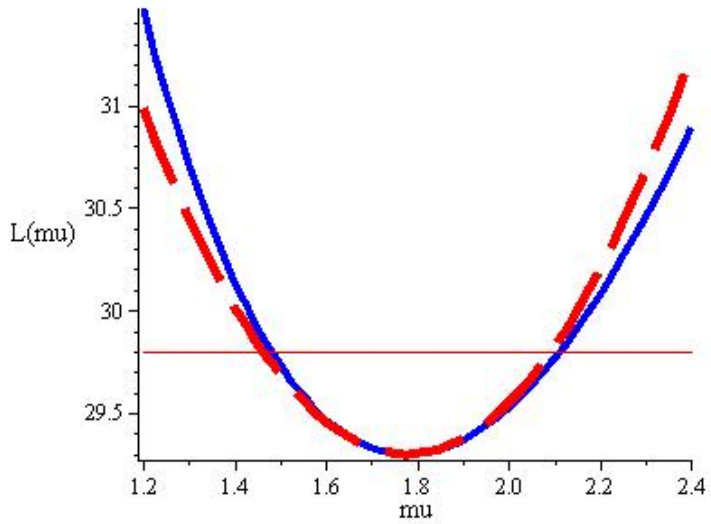
The result is

either

$$\mu = 1.78 \pm 0.31$$

or

$$\mu = 1.78^{+0.33}_{-0.30}$$



Now MINUIT

```
#define NDATA 18
int r[NDATA] = {1,1, 5, 4,2,0,3,2, 4,1,2,1,1,0,1,1,2,1};
int rfac[NDATA] = {1,1,120,24,2,1,6,2,24,1,2,1,1,1,1,1,2,1 };
```

```
void fcn(int &npar, double *gin, double &f, double *par,
    int iflag)
{
   int i;
   double mu, lnL;

   mu = par[0];
   lnL = 0.0;
   for(i=0;i<NDATA; i++)
    {
     lnL += r[i]*log(mu) - mu - log((double) rfac[i]);
    }
   f = -lnL;
}
```

Example: initialization

```
main()
{
   double arglist[10];
   int ierflg = 0;

   double start   = 1.0;
   double step = 0.1;
   double l_bnd = 0.1;
   double u_bnd = 10.;

   TMinuit minuit(1);
   minuit.SetFCN(fcn);
   minuit.mnparm(0,"Poisson mu", start, step, l_bnd,
    u_bnd, ierflg);
}
```

Parameter definition

```
minuit.mnparm(num,name,start_value,step_size,
    l_bound,u_bound,err_flag);
```

variable	type	meaning
num	int	parameter index ($\rightarrow$ Function fname)
name	char	parameter name
start_value	double	parameter start value
step_size	double	step width/approximate error
l_bound	double	lower bound for parameter
u_bound	double	upper bound
err_flg	int	return value: error flag

- choose good values for start_value and step_size
- l_bound =0.0 and u_bound=0.0 means unlimited parameter values never limit the value unless there is a good reason for it

Example: execution

```
[...]

   arglist[0] = 0.5;
   minuit.mnexcm("SET ERR",arglist,1,ierflg);

   minuit.mnexcm("MIGRAD",arglist,0,ierflg);

   minuit.mnexcm("MINOS",arglist,0,ierflg);

   arglist[0] = 2.0;
   minuit.mnexcm("SET ERR",arglist,1,ierflg);

   minuit.mnexcm("MINOS",arglist,0,ierflg);
```

MINUIT commands

after initialization MINUIT commands

- steer the minimization
- calculate contours
- fix parameters or make them variable again
- steer verbosity, etc.
- complete list:
  `http://wwwasdoc.web.cern.ch/wwwasdoc/hbook_html3/node125.html`
- most important commands are presented here
- mandatory arguments are designated by <...>, optional by [...]

MINUIT commands

One way to call commands is by function call, syntax:

```
minuit.mnexcm(command, argument list, argument number,
    error flag);
```

- *command* is a character array holding the command
- *argument list* a `double` array as the arguments
- *argument number* is the number of arguments
- *error flag* return value ($\neq 0$)

Example: 1-σ error definition of a Log-Likelihood-function:

```
double arglist[10];
int ierflg;
arglist[0] = 0.5;
minuit.mnexcm("SET ERR",arglist,1,ierflg);
```

Important MINUIT commands

- SET ERRordef <*value*>
  defines function value above minimum value for error and contour calculation
- MIGrad [*maxcalls*] [*tolerance*]
  searches for minimum, computes covariance matrix and from this the parabolic errors of all parameters (most efficient algorithm)
  maxcalls limits the number of function calls
  tolerance changes the convergence criterion of the minimization
- MINOs [*maxcalls*] [*parameter*] . . . [*parameter*]
  calculates exact, asymmetric errors for all variable parameters (or the ones given by index)
  is called after the calculation of minimum and covariance matrix
- SCAn [*parameter*] [*points*] [*FROM*] [*TO*]
  calculates and plots *points* in the given interval of the parameter

Further MINUIT commands

- MNContour <*parameter*> <*parameter*> [points]
  calculates contour of function in the plane of the two parameters
  the function is minimized w.r.t all other parameters
  the function value of the contour is the minimal value plus the last given value of SET ERRordef
- FIX <*parameter*> [*parameter*] . . . [*parameter*]
  fixes the listed parameters to the current value
- RELease <*parameter*> [*parameter*] . . . [*parameter*]
  make the listed parameters variable again
- HESse [*maxcalls*]
  calculates the covariance matrix from the inverse matrix of second derivatives (Hesse matrix) $V = G^{-1}$ with $G_{ij} = \frac{\partial^2 F(\vec{a})}{\partial a_i \partial a_j}$
  is called internally by MIGrad

And even more MINUIT commands

- IMProve [*maxcalls*]
  searches for further local minima of the given function
  useful, when the found minimum seems suspicious
- SIMplex [*maxcalls*] [*tolerance*]
  different (faster) algorithm than MIGrad
- MINImize [*maxcalls*] [*tolerance*]
  tries MIGrad, in case this doesn't converge changes to SIMplex

Two examples

Example Likelihood

likelihood_example.cpp

Example Forward-Backward Asymmetry

likelihood_example.cpp
Introduction next pages

Introduction

The L3 experiment was an experiment at the large electron positron collider at CERN near Geneva. Until 2000 it recorded the decay products of e^+e^- collisions at center of mass energies up to 208 GeV.
An example is the muon pair production

$$e^+e^- \longrightarrow \mu^+\mu^-$$

Both muons are recorded mainly in the tracking system. From the measurements the curvature, the charge and the momentum are determined. The z-axis is defined as the direction of the incoming electrons.
The file `L3.dat` contains recorded muon pair events. Every line is an event. The first three columns contain the momentum components p_x, p_y and p_z of μ^+. The other three columns contain the information on μ^-. Units are in Gev/c.

Forward-Backward Asymmetry

The task

An important parameter is the forward-backward asymmetry A:

$$A = (N_{\mathrm{F}} - N_{\mathrm{B}})/(N_{\mathrm{F}} + N_{\mathrm{B}})$$

N_F and N_B are the events, in which the μ^- flies forwards ($\cos\theta_{\mu^-} > 0$), respectively backwards. The value including the statistical error is to be determined.

Simple treatment

- The statistical error can be written as

$$\sigma_A = \sqrt{\frac{1 - A^2}{N}}$$

with $N = N_F + N_B$.
This can be deduced from taking N_F and N_B as independent Gaussian random variables.
BTW: the same result can be deduced from taking every event as Bernoulli experiment (e.g. every forward muon is a success).

- This is the first way: simply count the forward events.

Using all information

- The formula for the angular distribution $\cos(\theta_{\mu^-})$ is

$$\frac{3}{8}\left(1+\cos^2\theta_{\mu^-}\right)+A\cos\theta_{\mu^-}$$

- It can be shown, that this is correctly normalized and equivalent to the above definition of A.
  Remember: always check! Plot the values!
- The second way: calculate the asymmetry including the statistical error with a maximum likelihood fit.