Minimization with ROOT using TMinuit

Regis Terrier

10/15/01

TMinuit

- TMinuit inherits from Tobject
- C++ translation of F. James's Minuit package
- See Documentation at

  http://root.cern.ch/root/html/TMinuit.html
- Can be used in a very close manner as initial package i.e. passing commands through a character string
- C++ version independent of ROOT exists:

  Midnight package (in pdrApp)

FCN

- TMinuit expects the function to minimize to be ALWAYS a static external function (no member functions). function pointer member of TMinuit is:

```
void (*)(Int_t&npar, Double_t*gin, Double_t&f,
  Double_t*par, Int_t flag) fFCN
```

- npar :number of free parameters involved in minimization
- gin : computed gradient values (optional)
- f : the function value itself
- par : vector of constant and variable parameters
- flag : to switch between several actions of FCN

FCN

- The general FCN function:

```
void FCN(Int_t&npar, Double_t*gin, Double_t&f, Double_t*par, Int_t flag)
{
    // select case using flag
    switch(flag)
    case 1:
       // Initialization
    case 2:
        // Compute derivatives
        // store them in gin
    case 3:
    // after the fit is finished
    else:
    // compute function itself
  f= …
}
```

Most of the time just forget about flags (unless if you compute the gradient yourself)

Accessing data in FCN

How to access your data in FCN?

- FCN is an external function so in order to access data used for computation, you have 2 options:
  - using static objects or variables.ex:

```
TSkyMapList* unmodG;
static void fcn_nosource(int &npar, double *gin, double &f, double *par, int
   iflag){
  // compute likelihood
  f=0;
  double p0=par[0];
  double p1=par[1];
  double norm = unmodG.Sum();
}
```

with an initialization function

```
void Like::Init(TSkyMapList& modelG){
   unmodG = &modelG;
...}
```

Accessing data in FCN

using SetObjectFit(TObect*) and GetFitObject()

- data must be in 1 TObject

(does not exist in Midnight) ex:

```
Tminuit* minuit=0;
static void fcn_nosource(int &npar, double *gin, double &f, double *par, int
   iflag){
     // compute likelihood
     f=0;
     double p0=par[0];
     double p1=par[1];
     TSkyMapList* unmodG = (TSkyMapList*) minuit->GetObjectFit();
     double norm = unmodG->Sum();
   ....}
```

And in the general fitting function:

```
void Like::Minimize(TSkyMapList& modelG){
// Define Minuit for 3 parameters
minuit = new TMinuit(3);
minuit->SetFitObject(&modelG);
...}
```

- All data must be stored in one TObject

Initialisation of minuit

First create minuit and define function to minimize

```
void Like::Minimize(double* flux)
{
 // Init static variables for the fit

  //Instantiate Minuit for 2 parameters
  TMinuit minuit(2);
  // Set fonction pointer
  minuit.SetFCN(fcn_nosource);

  // Vector of step, initial min and max value
  double vstrt[2];
  double stp[2];
  double bmin[2];
  double bmax[2];
  vstrt[0] = flux[0];  vstrt[1] = flux[1];
  stp[0]=0.01;  stp[1]= 0.01;
  bmin[0] = vstrt[0]-1.; bmin[1] = vstrt[1]-1.;
  bmax[0] = vstrt[0]+1.;  bmax[1] = vstrt[1]+1.;
```

Initialisation (cont)

Now define parameters using

void mnparm(Int_t k1, TString cnamj, Double_t uk, Double_t wk, Double_t a, Double_t b, Int_t &ierflg)

```
double arglist[10];
int ierflg = 0;
minuit.mnparm(0, "Flux galactique",     vstrt[0],stp[0],bmin[0],bmax[0],ierflg);
minuit.mnparm(1, "Flux extragalactique",vstrt[1],stp[1],bmin[1],bmax[1],ierflg);
```

Or use:

```
Int_t DefineParameter( Int_t parNo, const char *name, Double_t initVal, Double_t
    initErr, Double_t lowerLimit, Double_t upperLimit )
```

Set ouput

```
// Set Print Level
// -1 no output
// 1 standard output
minuit.SetPrintLevel(-1);
// No Warnings
minuit.mnexcm("SET NOW", arglist ,1,ierflg);
```

Defining strategy

```
// Set error Definition
  // 1 for Chi square
  // 0.5 for negative log likelihood
  minuit.SetErrorDef(0.5);
```

Minimization strategy

```
  // 1 standard
  // 2 try to improve minimum (slower)
  arglist[0]=2;
  minuit.mnexcm("SET STR",arglist,1,ierflg);
```

Fixing and releasing parameters (beware the numbering):

```
  // fix galactic flux
  arglist[0] = 1;
  minuit.mnexcm("FIX ", arglist ,1,ierflg);
```

or (beware the numbering):

```
minuit.FixParameter(0);
```

Minimization itself

Call Migrad with 500 iterations maximum

The MIGRAD algorithm is in general the best minimizer for nearly all functions. It is a variable-metric method with inexact line search, a stable metric updating scheme, and checks for positive definiteness. Its main weakness is that it depends heavily on knowledge of the first derivatives, and fails miserably if they are very inaccurate.

```
arglist[0] = 500;
minuit.mnexcm("MIGRAD", arglist ,1,ierflg);
```

Or:

```
minuit.SetMaxIteration(500);
minuit.Migrad();
```

Output

```
FCN=26299.8 FROM MIGRAD    STATUS=CONVERGED      49 CALLS         50 TOTAL
                 EDM=5.14224e-09    STRATEGY= 2      ERROR MATRIX ACCURATE
 EXT PARAMETER                                  STEP         FIRST
 NO.   NAME        VALUE            ERROR          SIZE      DERIVATIVE
  1  Flux galactique   1.02412e+00   6.04935e-02   2.94882e-03  -2.45383e-03
  2  Flux extragalactique   9.93560e-01   4.44450e-02   2.16529e-03  -3.62470e-03
```

Sometimes use Melder & Nelde Simplex method

```
minuit.mnsimp();
```

```
.                      .
     26950 ...                             .                           .
            .                           .                            .
       26900 ...        *                    .                           .
            .                           .                            .
       26850 ...                            .                           .
            .                           .                            .
       26800 ...         *                   .                           *.
            .                           .
       26750 ...                            .                           .
            .           *                .                         *  .
       26700 ...                            .                           .
            .                           .                        *    .
       26650 ...           *                 .                           .
            .                           .                      *      .
       26600 ...                            .                     *      .
            .             *              .                            .
       26550 ...                            .                   *        .
            .              *             .                  *         .
       26500 ...                            .                *           .
            .                *           .                            .
       26450 ...                            .               *            .
            .                 *          .              *             .
       26400 ...                  *         .            *               .
            .                    *       .           *                .
       26350 ...                     *      .        * *                 .
            .                       *    .       *                    .
       26300 ......................................*.*.*.&.*.*.*.*................................
            .                           .                            .
       26250 ...................................................................................
            /      /      /      /      /      /      /      /      /
          -0.025    0.225    0.475    0.725    0.975    1.225    1.475    1.725    1.975
```

Scan is usefull to check FCN variation over the selected range

```
arglist[0] = 0;
  minuit.mnexcm("SCAN", arglist
,1,ierflg);
```

Getting fit result

Now fill fitted variables E0,G0 and associated errors

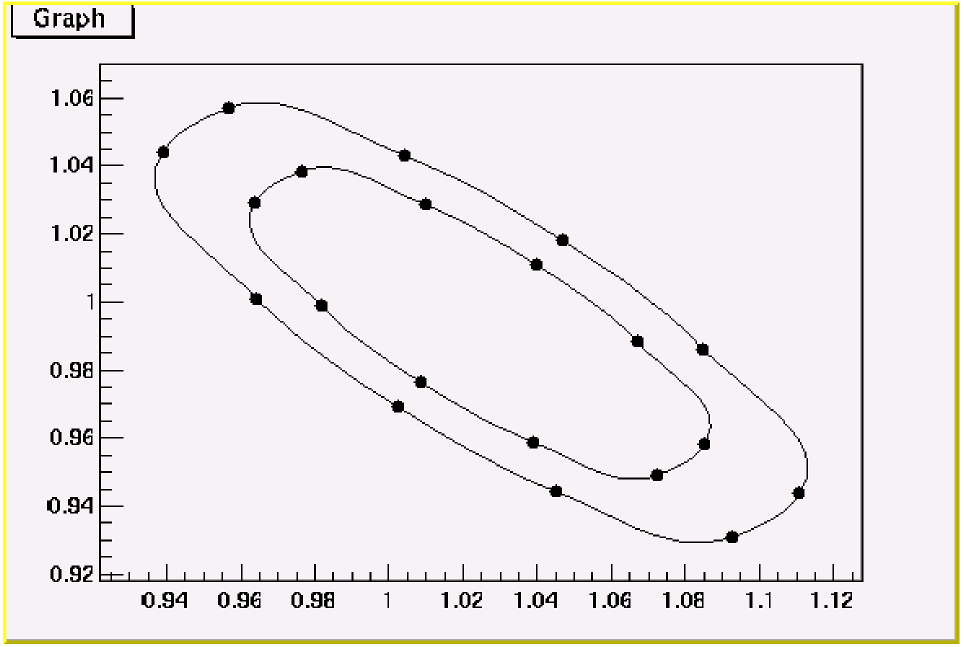
```
minuit.GetParameter(0,G0,errG0);
minuit.GetParameter(1,E0,errE0);
```

Get minimum log likelihood

```
double ln0,edm,errdef;
int nvpar,nparx,icstat;
minuit.mnstat(ln0,edm,errdef,nvpar,nparx,icstat);
```

Get 1 and 2 sigma contour

```
//1 sigma
 Tgraph* graph1 =
 (TGraph*) minuit.Contour(10,0,1);
// 2 sigma
 minuit.SetErrorDef(2);
 Tgraph*  graph2 =
(TGraph*) minuit.Contour(10,0,1);
```



And much more

- Many more functions

see http://root.cern.ch/root/html/TMinuit.html

and http://wwwinfo.cern.ch/asdoc/minuit/minmain.html

See also in TH1.cxx

```
void H1FitLikelihood(Int_t &npar, Double_t *gin, Double_t &f,
Double_t *u, Int_t flag)
```

In pdrApp, see CsIClustersAlg::Profile()