Minuit status 16 June 2004

Fred James and Matthias Winkler

- Overview
- Performance comparison
- Numerical comparison
- Example
- Conclusion

Overview

- Project started 1 August 2002
- Re-implement Minuit in OO using C++
- Main concerns: computational + numerical performance
- http://www.cern.ch/minuit

Overview: continued

- 10k lines of code (wc -l)
- used by CMS and HippoDraw
- usable from ROOT
- fully documented
- SCRAM and autoconf/make (scripts provided by Paul Kunz)

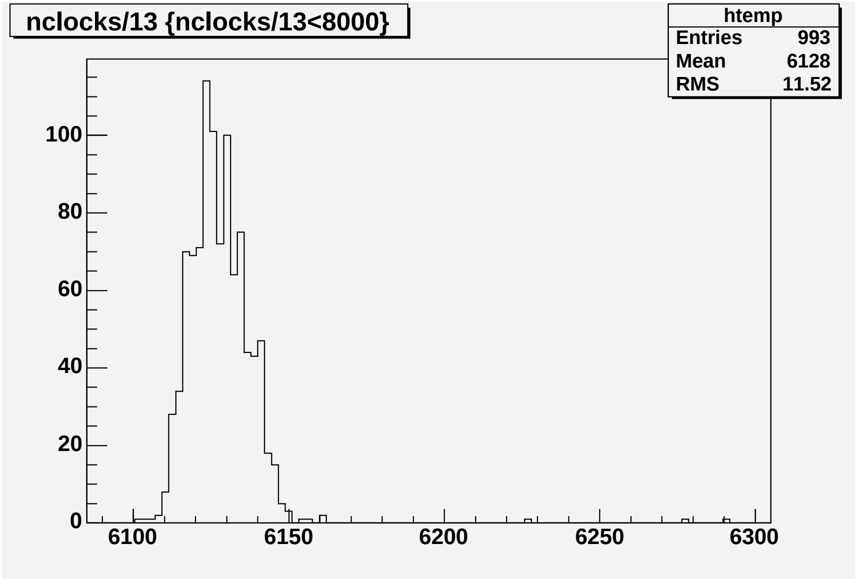
Overview: functionality

- Minimizers: Migrad, Simplex, Minimize, Scan
- Errors: Hesse, Minos, Contours
- Parameters: fix, release, put and remove limits, ...
- new: single sided limits on parameters (implemented by Lorenzo Moneta)

Computational performance

- directly compare with Fortran version
- Functions: x^2, ds/dq (physics example from Fortran)
- compare with ROOT's C++ version of Minuit
- performance test by Rene Brun

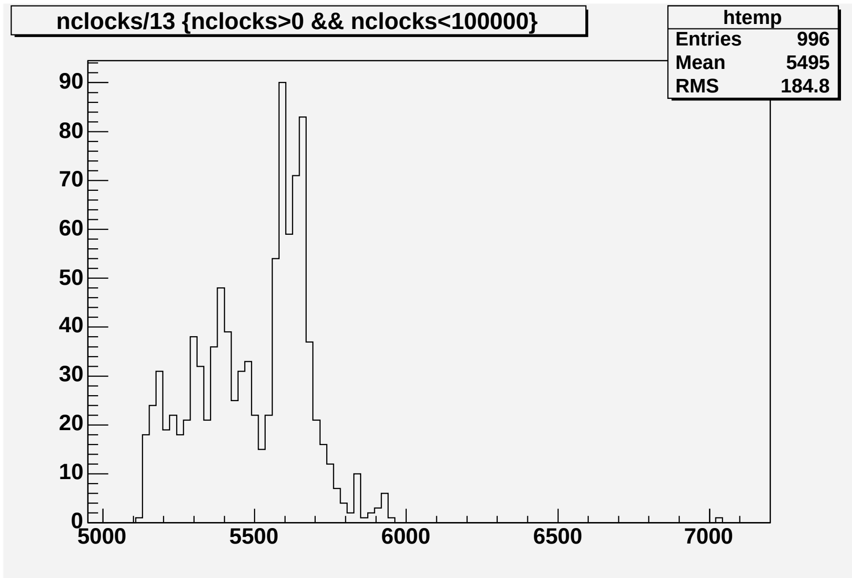
$y = x^2$



C++, nclocks/nfcn
(13 fcn calls)

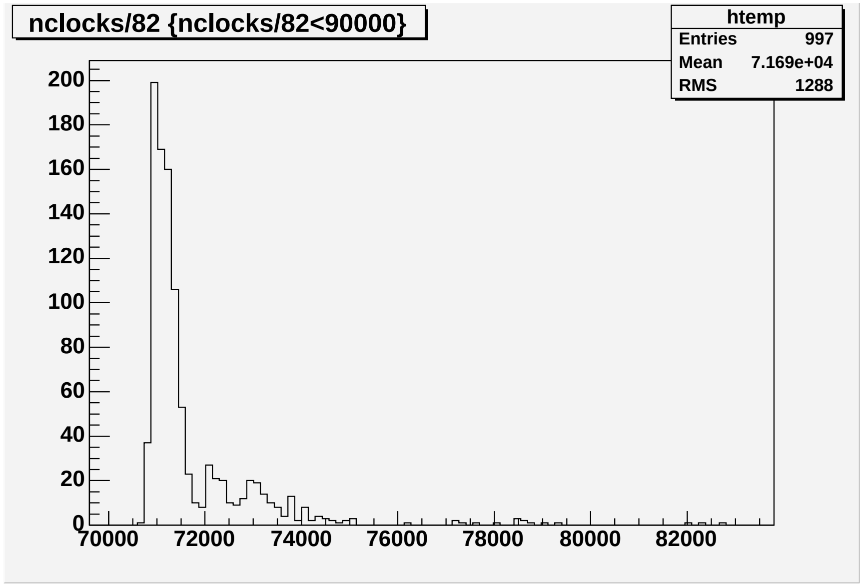
scram stand. flags

F77, nclocks/nfcn
(13 fcn calls)



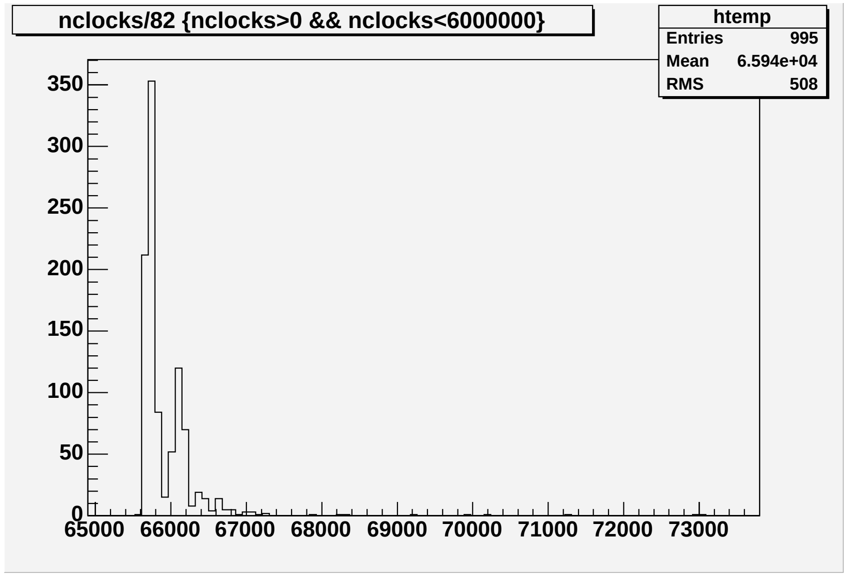
C++/F77 ~1.115

DsDq



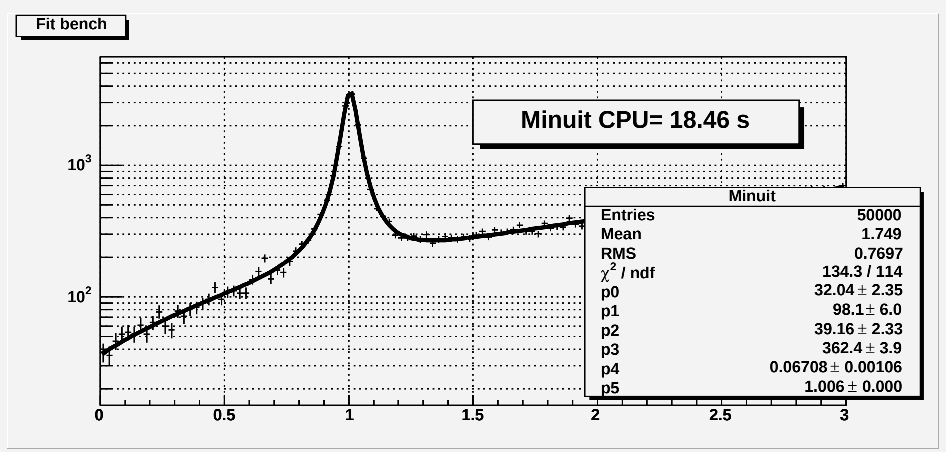
C++, nclocks/nfcn
(82 fcn calls)

F77, nclocks/nfcn
(82 fcn calls)

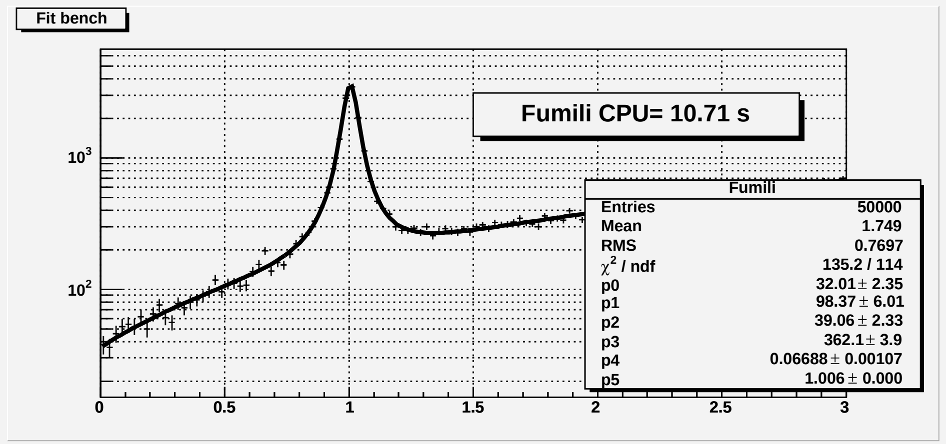


C++/F77 ~1.087

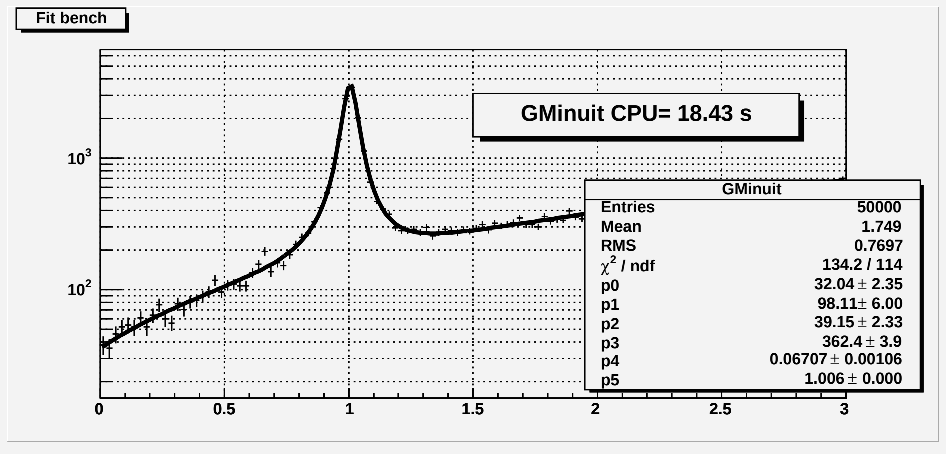
ROOT Minuit



ROOT Fumili



Fred/Matthias



Computational performance: technicalities

- minimization: number of function calls
- memory: allocation/deallocation cost expensive
- memory: use dedicated StackAllocator (provided by Teddy Todorov)
- temporaries: avoid unnecessary creation

Numerical performance

- compare with Fortran version
- fully controlled simulation environment: pulls, chi2

Minuit State of the Art: numerical performance

standard example: ds/dq

```
 FCN=   33.43457     FROM MIGRAD    STATUS=CONVERGED     81 CALLS       82 TOTAL
                 EDM=  0.32E-06    STRATEGY= 1      ERROR MATRIX ACCURATE

      EXT PARAMETER                                   STEP         FIRST
 NO.   NAME         VALUE          ERROR          SIZE       DERIVATIVE
  1   Re(X)        0.55653E-01    0.88802E-01   0.17203E-03  -0.22100E-02
  2   Im(X)        0.27634E-01    0.12524        0.23591E-03  -0.75792E-02
  5   Delta M      0.32672        0.24295        0.64857E-03   0.16586E-03
 10    T Kshort    0.89200        constant
 11    T Klong     518.30         constant
```

F77 output

```
# of function calls: 82
function value: 33.4346
expected distance to the minimum (edm): 1.60772e-07
external parameters:
# ext. ||   name    ||  type  ||   value   ||  error +/-

   0   ||Re(X)      ||  free    ||   0.05565 ||  0.0888
   1   ||Im(X)      ||  free    ||   0.02763 ||  0.1252
   4   ||Delta M    ||  free    ||    0.3267  ||  0.2429
   9   ||T Kshort   ||  const  ||     0.892   ||
  10   ||T Klong    ||  const  ||     518.3   ||
```

C++ output

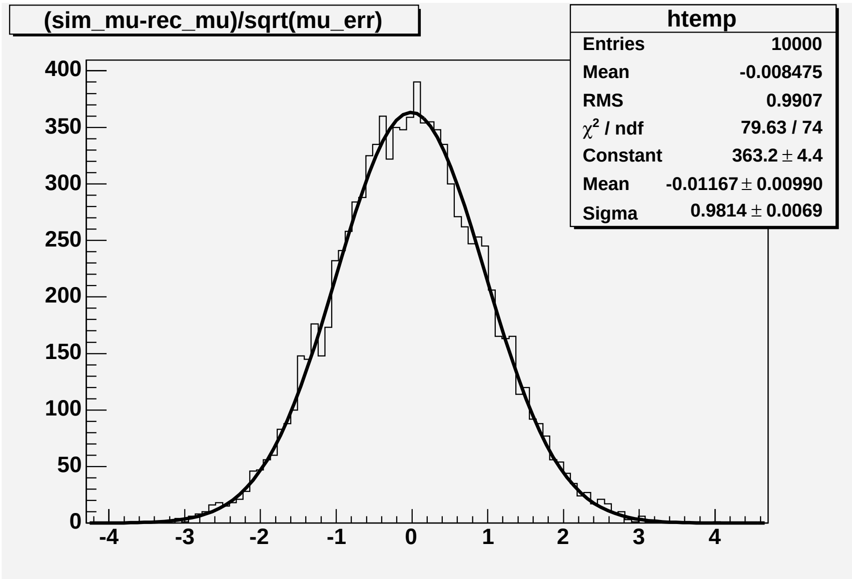
...identical...

Minuit State of the Art: numerical performance (continued)

simulate 10k Gaussians

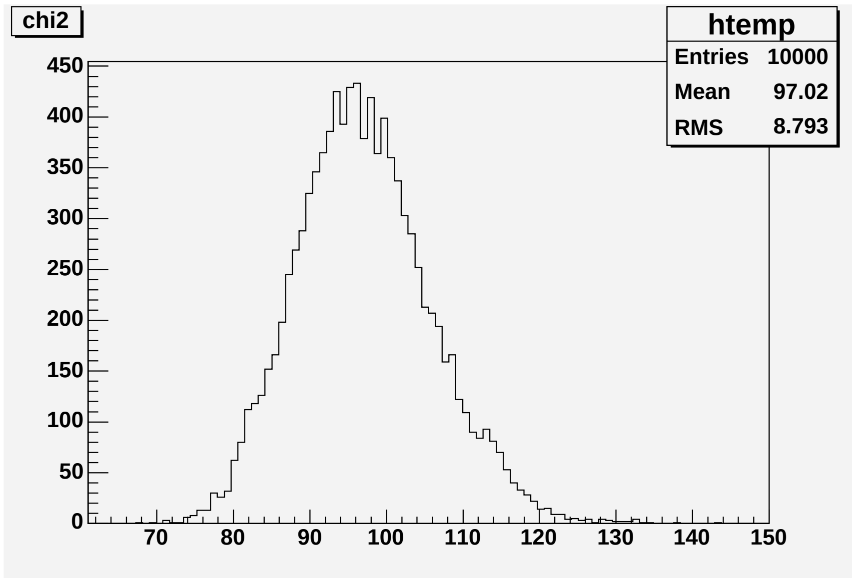
+ generate 100 measurements for each Gaussian

+ fit mean, sigma and constant



show pulls of mean values

Chi2 distribution



Example I: minimal required interface

```
{
  // demonstrate minimal required interface for minimization
  // create Minuit parameters without names

  // starting values for parameters
  std::vector<double> init_par;
  init_par.push_back(mean);
  init_par.push_back(rms);
  init_par.push_back(area);

  // starting values for initial uncertainties
  std::vector<double> init_err(3, 0.1);

  // create minimizer (default constructor)
  VariableMetricMinimizer theMinimizer;

  // minimize
  FunctionMinimum min = theMinimizer.minimize(theFCN, init_par, init_err);

  // output
  std::cout<<"minimum: "<<min<<std::endl;
}
```

Example II: parameter interaction

```
{
  // create Minuit parameters with names
  MnUserParameters upar;
  upar.add("mean", mean, 0.1);
  upar.add("sigma", rms, 0.1);
  upar.add("area", area, 0.1);

  // access parameter by name to set limits...
  upar.setLimits("mean", mean-0.01, mean+0.01);

  // create Migrad minimizer
  MnMigrad migrad(theFCN, upar);

  // fix a parameter...
  migrad.fix("mean");

  // ... and minimize
  FunctionMinimum min = migrad();

  // output
  std::cout<<"minimum: "<<min<<std::endl;

  // release a parameter...
  migrad.release("mean");

  // ... and fix another one
  migrad.fix(1);

  // and minimize again
  FunctionMinimum min1 = migrad();
```

Documentation (etc...)

- http://www.cern.ch/minuit
- forum-minuit@cern.ch
- Quick start instructions
- Part I: User's guide
- Part II: Tutorial on function minimization
- Part III: Error interpretation

Conclusion

- main functionality available
  - missing: interactivity, verbosity, debugging
- computational performance under control
- numerical performance ok within tested examples
  - more tests + testers (users) are necessary
- extendable
  - single sided limits (already available)
  - planned: Fumili, linear constraints