New Developments of ROOT Mathematical Software Libraries

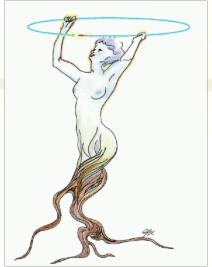
CHEP'07 VICTORIA, BC

International Conference on Computing in High Energy and Nuclear Physics

2-7 Sept 2007 Victoria BC Canada

Lorenzo Moneta, I. Antcheva , R. Brun, A. Kreshuk, M. Slawinska

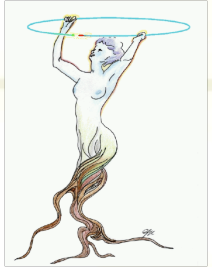
CERN/PH-SFT



Outline

- Recent developments of ROOT Math Libraries :
  - re-structuring of Math Libraries
    - new core library
  - random numbers and their performances
    - *UNU.RAN* package
  - *SMatrix* package and tests of matrix performances
- Fitting:
  - fitting GUI (new fit panel)
  - plans for new fitting and minimization classes
- Other recent developments:
  - histogram comparison, FFT
- Conclusions and future plans

New Structure of ROOT Mathematical Libraries



Histogram library
- TH1
- TF1

Fitting and Minimization
- New Fitter
- RooFit
- Linear & Robust Fitter
- Quadp
- Minuit2 (OO Minuit)
- TFumili
- TMinuit

Statistical Libraries
- New Stat Tools (Significance, Limit/CL etc..)
- TMVA
- MLP

Linear Algebra
- TMatrix
- SMatrix

Extra Libraries
- Unuran
- FFTW
- Spectrum
- Foam

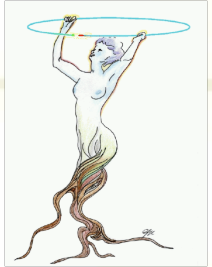
MathCore
- Functors & interfaces
- Physics Vectors
- Basic algorithms
- TComplex
- TRandom
- Math functions
- TMath

MathMore
- Random Numbers
- Extra algorithms
- Extra Math functions
- GSL

not yet released

already existing

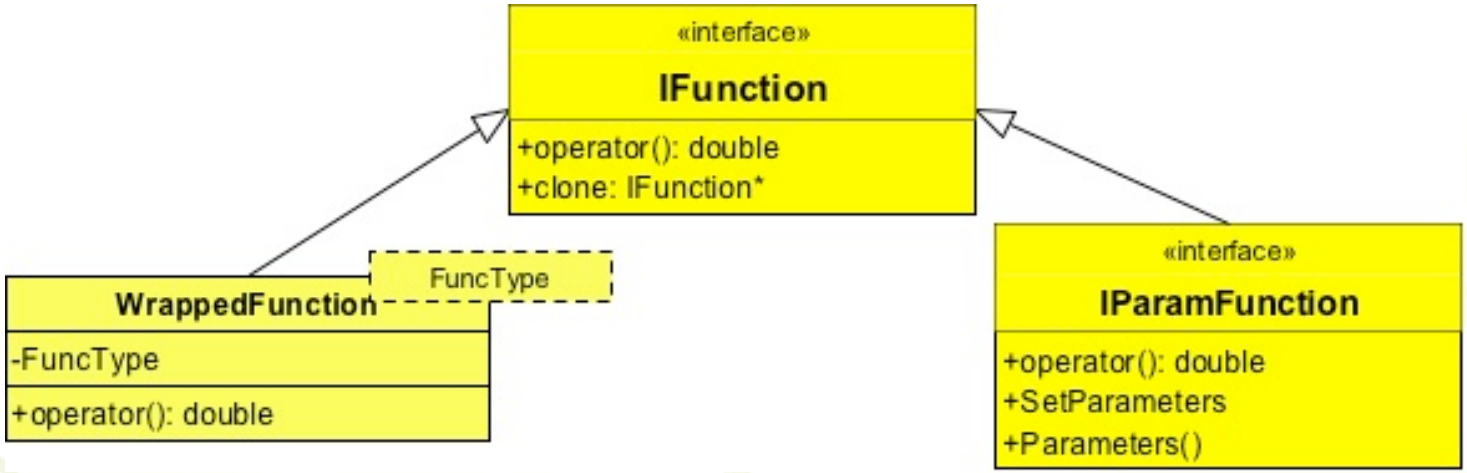
Re-structuring of ROOT MathLibs



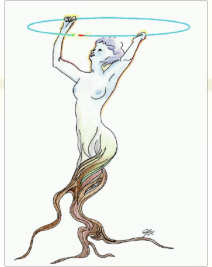
- Aim to improve modularity
- New core mathematical library containing
  - Math classes from *libCore*:
    - *TRandom* classes, *TComplex* , *TMath* (excluding *TMathBase*)
  - classes from current version of *MathCore*
    - basic mathematical and statistical functions
    - physics vector (2D, 3D and LorentzVector + transformations)
  - implementation of basic numerical algorithms
    - from *TF1* : derivation, numerical integration, Brent minimization
  - interfaces to functions and algorithms
    - use different implementations via the plug-in manager
  - functor and wrapper classes
    - create *TF1* classes from C++ callable objects
    - easy to apply algorithms to user functions

Function Interfaces

- Minimal interface used by all numerical algorithms:
- Abstract classes for 1D and multi-dimensional functions
  - interfaces for functions providing analytical derivatives
  - parametric function interfaces (used in fitting)
  - template classes (*WrappedFunction<FuncType>*) to wrap any C++ callable object (functors, C free function, etc..) in the desired interface



Functor classes

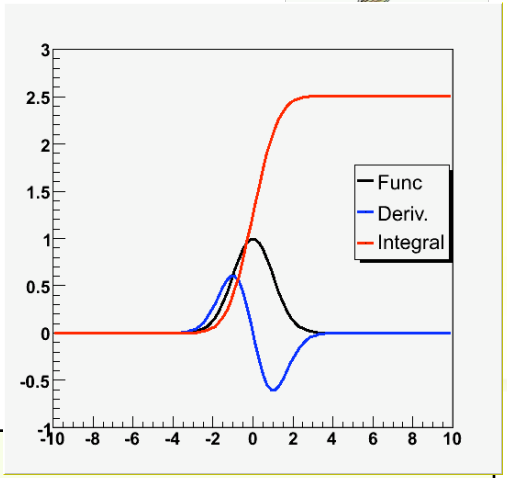


- Classes to wrap a C++ callable object with the right signature
  - can work with:
    - free C functions
    - C++ classes (or structs) implementing `operator()`
    - member functions of a class
  - Example: a generic parametric function (like *TF1* class)
    - `double F ( double *x, double *p)`
- Added a constructor of *TF1* using a functor type
  - can make *TF1* depending on other objects
- Powerful, easy to use and very flexible
  - user can customize callable object using its state
  - used extensively in STL algorithms

Example of Functors

- tutorials example:
  - *tutorials/math/exampleFunctor.C*

```
double MyFunc (double *x, double *p ) {
  return TMath::Gaus(x[0],p[0],p[1] );
}

struct MyDerivFunc {
  MyDerivFunc(TF1 * f): fFunc(f) {}
  double operator()(double *x,double *) const {
      return fFunc->Derivative(*x);
  }
  TF1 * fFunc;
};
struct MyIntegFunc {
  MyIntegFunc(TF1 * f): fFunc(f) {}
  double Eval(double *x,double *) const {
    double a = fFunc->GetXmin();
    return fFunc->Integral(a, *x);
  }
  TF1 * fFunc;
};
```

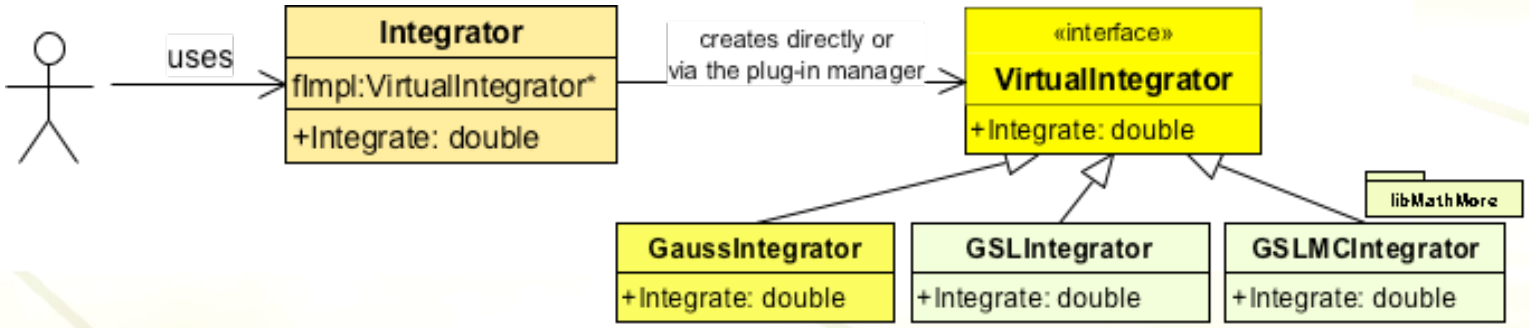



```
void exampleFunctor() {

   double xmin = -10;  double xmax = 10;
   TF1 * f1 = new TF1("f1",MyFunc,xmin,xmax,2);
   f1->SetParameters(0.,1.);
   f1->SetMaximum(3);   f1->SetMinimum(-1);
   f1->Draw();

   // create derivatives function using
   // the parameter of the parent function
   TF1 * f2 = new TF1("f2",MyDerivFunc(f1),
                      xmin, xmax, 0);

   f2->SetLineColor(kBlue);
   f2->Draw("same");

   // create integral function
   MyIntegFunc g(f1);
   TF1 * f3 = new TF1("f3",&g,&MyIntegFunc::Eval,
                      xmin, xmax, 0);
   f3->SetLineColor(kRed);
   f3->Draw("same");
}
```

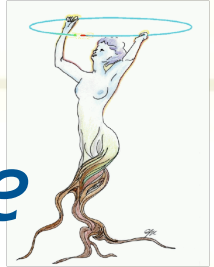
- before was possible only by using globals objects

Numerical Algorithms

- Collect in the core Math library all basic numerical algorithms
  - example: algorithms from *TF1* (derivation, integration, etc..)
- Provide a coherent interface for all these algorithms:
  - have a basic implementation using code from *TF1*
    - maintain the current methods for backward compatibility
  - use plug-in manager for alternative implementations (*GSL*)
- Algorithms could be used directly
  - no need to create a *TF1* object
  - user needs to provide right interface or a functor

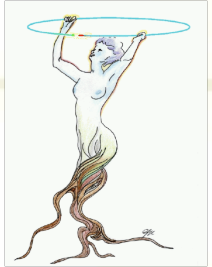


Numerical Algorithms in MathMore



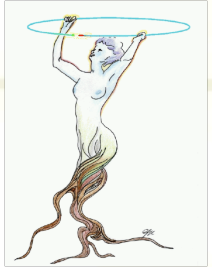
- *MathMore*: C++ interface to GNU Scientific Library (*GSL*)
- Numerical algorithms implemented:
  - **Numerical Derivation**
    - central evaluation (5 points rule) and forward/backward
  - **Numerical Integration**
    - adaptive integration for finite and infinite intervals, singular functions and with Cauchy principal value.
    - multidimensional MC integration based on VEGAS and MISER
  - **Root Finders**
    - bracketing and polishing algorithms using derivatives
  - **Minimization**
    - Golden section and Brent algorithm for 1D
    - conjugate gradient and BFGS algorithms for multi-dimensions
  - **Interpolation**
    - linear, polynomial, cubic and Akima spline
  - **Chebyshev polynomials** (for function approximation)
- Complement the various algorithms existing previously in ROOT (*TF1* class)

Improvements in pseudo-random numbers generation



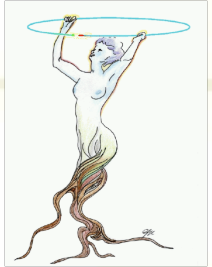
- *Mersenne-Twister* (*TRandom3*) is now the default generator
- Replaced some obsolete generators
  - *TRandom2* is now the *TausWorthe* generator from L'Ecuyer
- Generators can be seeded with an *UUID* (unique 128 bit number)
  - independent streams useful for parallel jobs
- New faster algorithm for generating Gaussian random number
  - acceptance-complement ratio method (W.Hörmann,G.Derflinger)
  - one of the fastest existing methods
    - no calls to mathematical functions (log, sqrt) as in polar method
- New algorithm for Poisson random-numbers
  - exact algorithm (rejection from a Lorentzian distribution)
- Introduced interface to UNU.RAN package

UNURAN



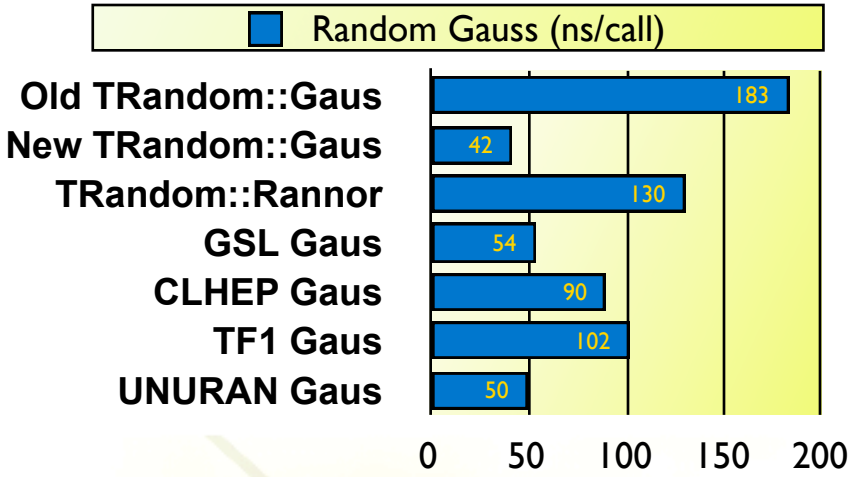
- Package for generating non-uniform pseudo-random numbers
  - developed (using C) by a group in Vienna (J. Leydold et al.)
  - added a ROOT interface and distribute (*libUnuran*)
- Universal methods based on inversion, rejection and composition
  - for 1D, multi-dimensional, discrete and empirical distributions (from a set of binned or un-binned data)
- Methods for standard distributions (like Gauss, Poisson, etc..)
- Efficient and exact methods (working for non truncated functions)
  - *TF1::GetRandom()* requires a range, it is approximate and is limited up to dimension 3
- Markov-Chain MC methods for multi-dimensional distributions

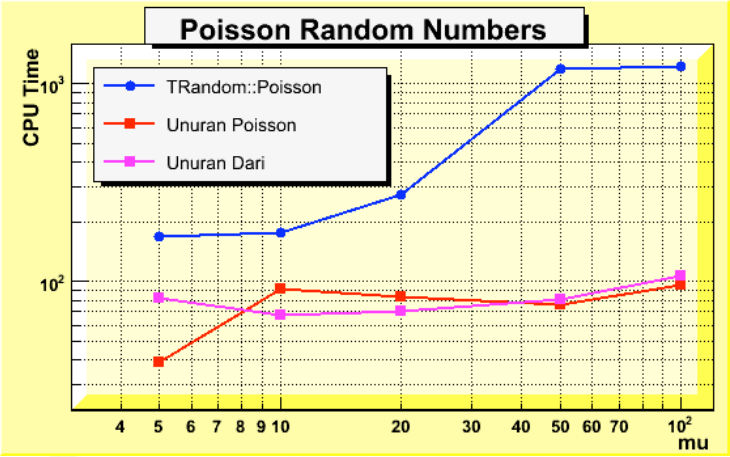
Performances of Random Number



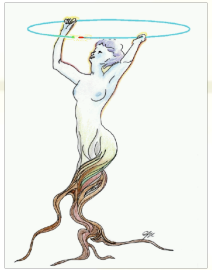
- Performances tests
  - lxplus, gcc 3.4
    - Intel 32 and 64 bits
- Uniform generation
- Gaussian
- Poisson number generation

Random Number Uniform Generators	Intel32 (ns/call)	Intel64 (ns/call)
MT (*TRandom3*)	22	9
TausWorthe (*TRandom2*)	17	6
RanLux (*TRandom1*)	120	98
LCG (*TRandom*)	14	5

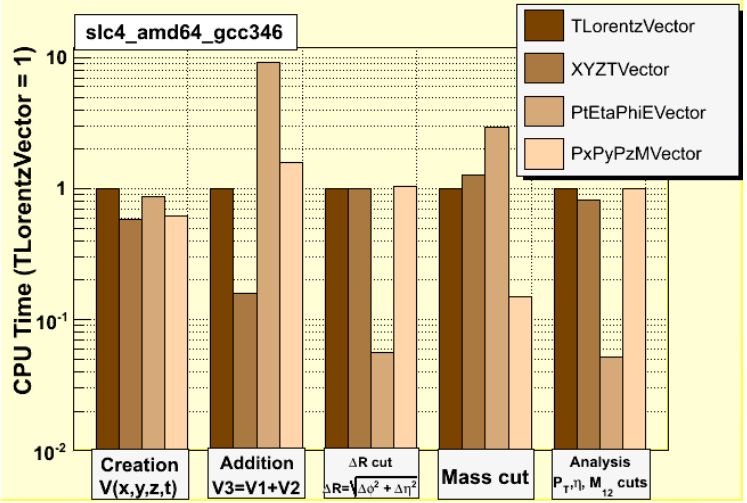




Physics and Geometry Vectors

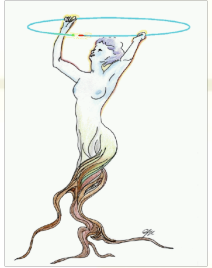


- Classes for 2D, 3D and 4D vectors with their operations and transformations (rotations)
  - functionality as in CLHEP *Vector* and *Geometry* packages
- Work done in collaboration with Fermilab computing group *(M. Fischler, W. Brown and J. Marraffino)*
- Main features of the new classes:
  - generic scalar contained type
    - i.e. single, double precision or *Double32_t*
  - generic coordinate system concept
    - i.e. cartesian, polar and cylindrical



- **Used by CMS and LHCb**
  - in reconstruction, event model
- Classes do not inherit from *TObject*
  - cannot be used in ROOT collections (like *TClonesArray*)
- Plan in the future to re-implement *TLorentzVector* using the new classes

SMatrix Package



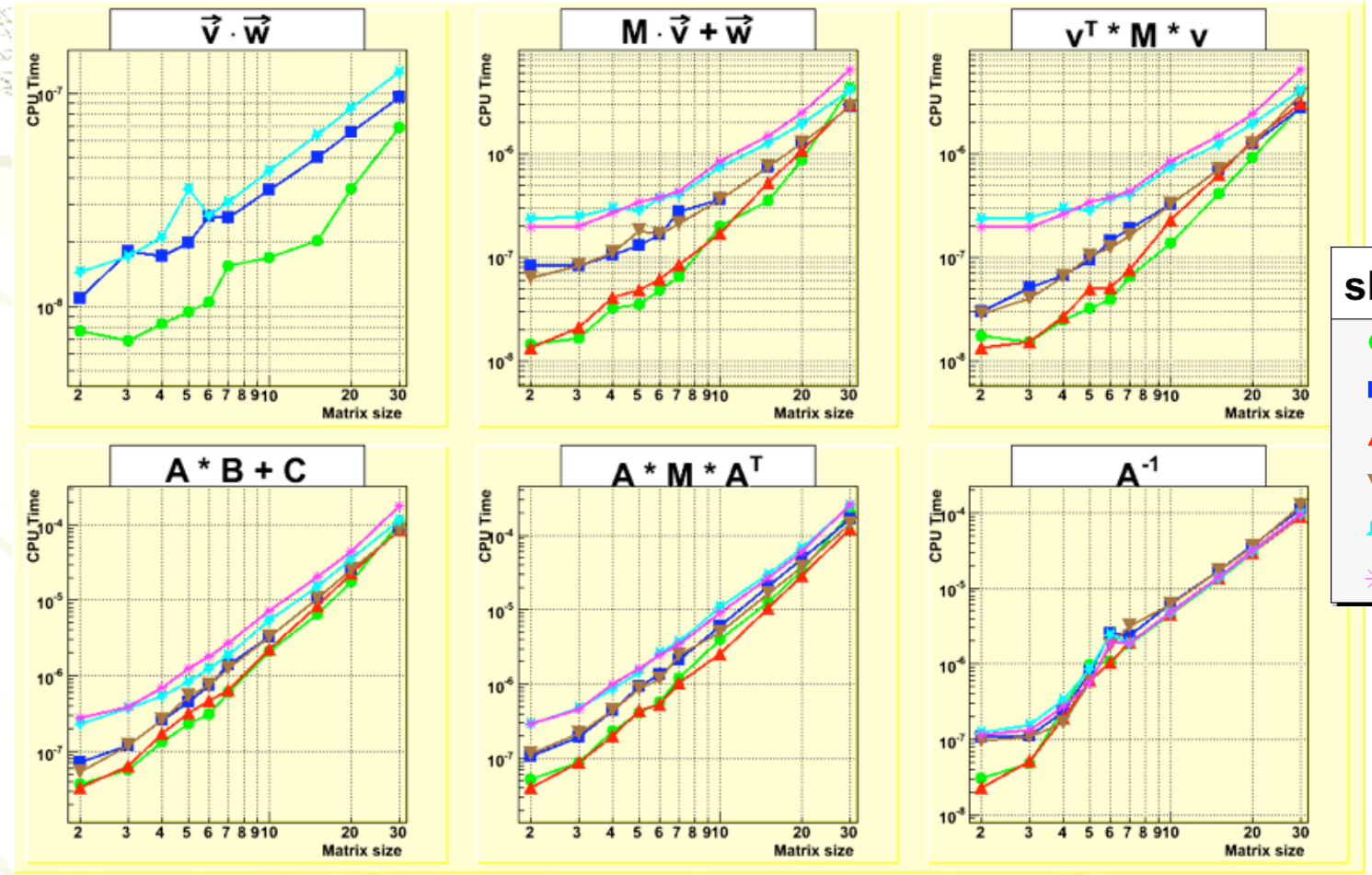
- Package initially developed (by T. Glebe for HeraB)
- For **fixed** (not dynamic) matrix and vector sizes :
  - size must be knows at compile time

```
SMatrix< double, 2 , 5>
SVector< double, 5 >
```

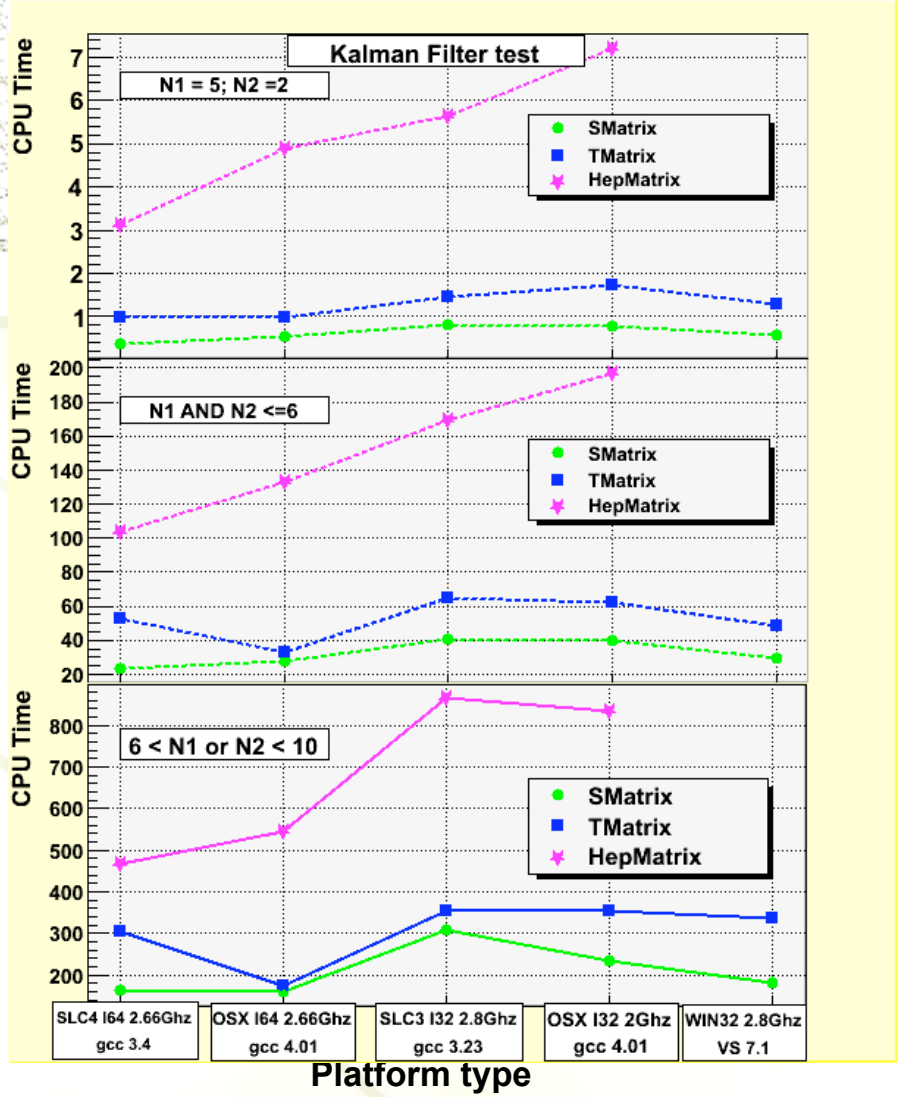
- Complementary and **NOT** a replacement of *TMatrix*
- Optimized for small matrix sizes (dim <= 6):
  - use expression templates to avoid temporaries
- Support for basic operations and matrix inversion
  - not full linear algebra functionality
- **Support now for symmetric matrices** (thanks to J.Palacios, LHCb)
  - reduced memory and I/O storage of only `n*(n+1)/2` elements
- Used by LHCb, CMS and (in some part in ATLAS) to replace CLHEP
  - significant improvement observed in their reconstruction code
    - CMS tracking CPU time reduce by 2
  - mainly due to the fixed memory allocation

Matrix Operations Performances

- Comparison ROOT (*TMatrix/SMatrix*) and CLHEP (*HepMatrix*)
  - lxplus (new Intel dual-core 64 bits) running slc4 with gcc 3.4.6

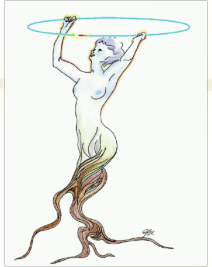


Kalman Filter Tests

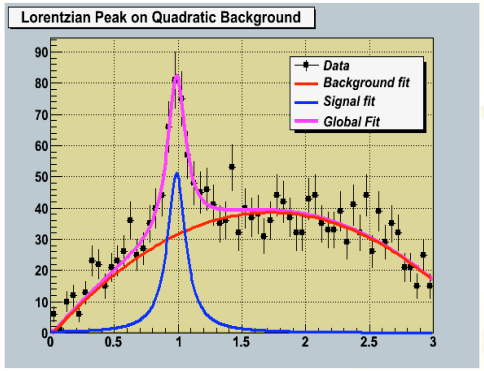


- CPU performances in the Kalman filter update equations
  - addition,multiplication and inversion
  - test varying matrix sizes
    - N1 = 2, N2 = 5
    - 2 <N1,N2 ≤ 6
    - 6 < max(N1,N2) ≤ 10
  - test various platforms with different C++ compilers
    - Linux 32 and 64 bits (g++ 3.2 and 3.4)
    - MacOS (g++ 4.0)
    - Windows (VS 7.1)
- Strong performance gains for both *SMatrix* and *TMatrix* compared to the CLHEP matrix

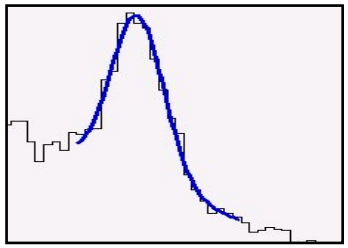
Fitting in ROOT



- Possible to Fit directly ROOT data classes (Histograms, Trees, Graphs) with various options:
  - least square/likelihood fits
  - linear/robust fits, etc..
- GUI has been developed to drive the fits
- Interface exist (*TVirtualFitter*) for custom fits
  - user defined objective functions
  - various minimization methods
    - *Minuit*, *Fumili*, *Minuit2*, *Fumili2*
  - not very flexible (designed for Minuit) and limited functionality, difficult to implement complex fits
    - many dimensions and multiple data objects
- *RooFit* for complex fitting and data modeling

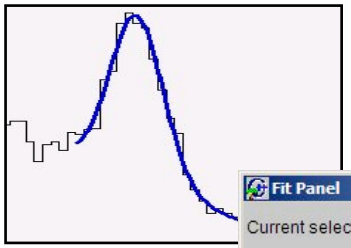


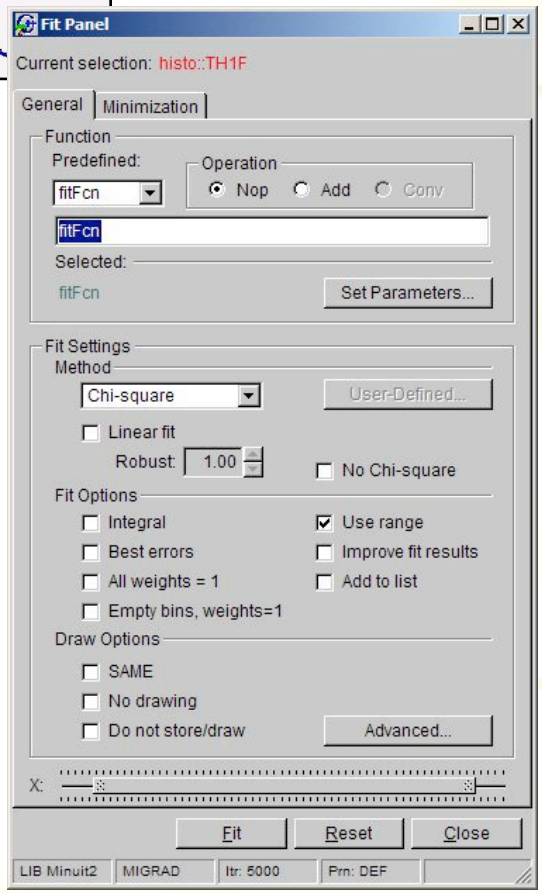
Fit Panel



- Easy data set selection, built-in function, fit method, model
- Available via context menu
- Recognize and accept user-defined functions
- Allow use of different minimizers
- Flexible parameters' settings and control
- Same interface for all ROOT data objects (binned and unbinned data)

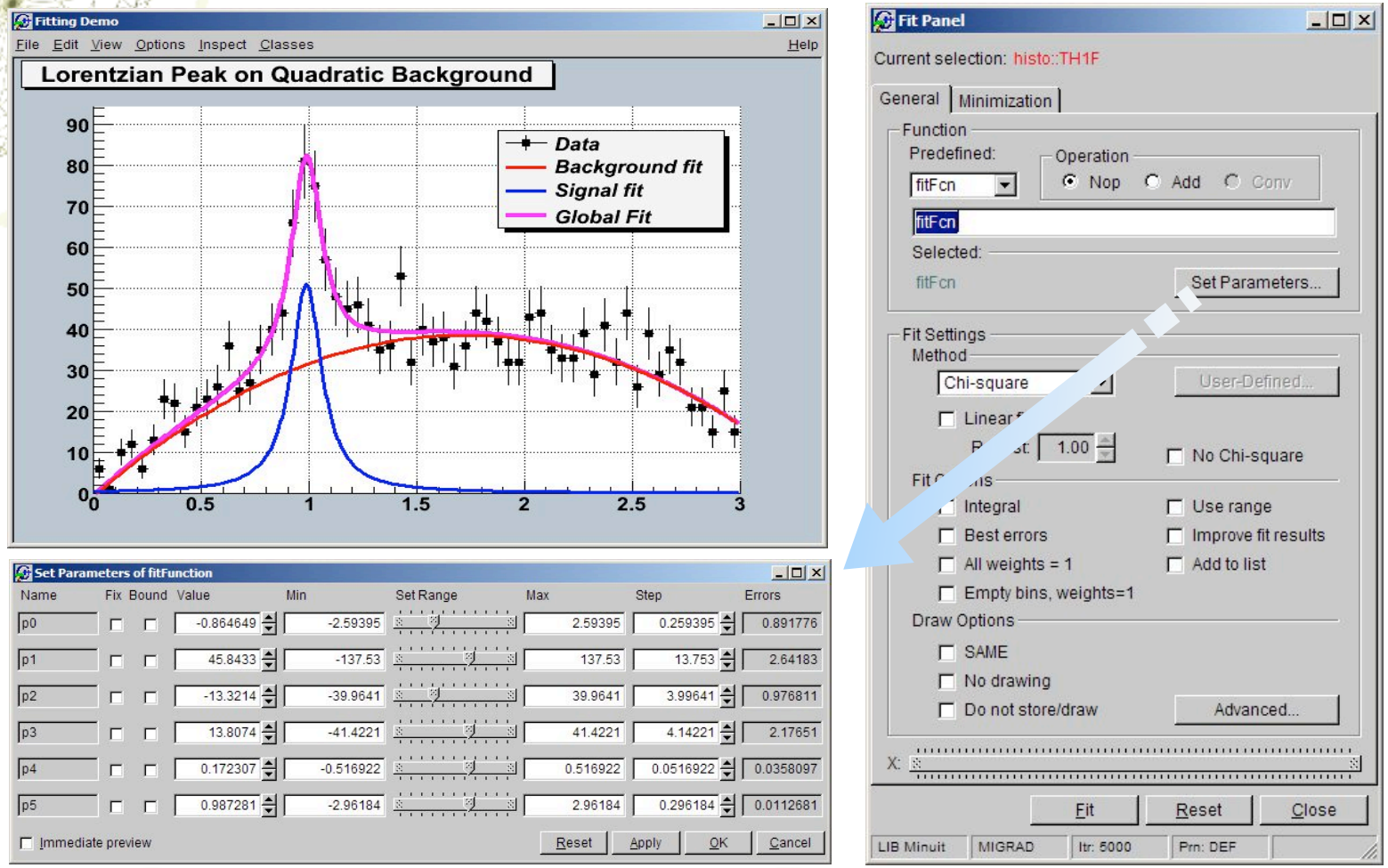
Fit Panel



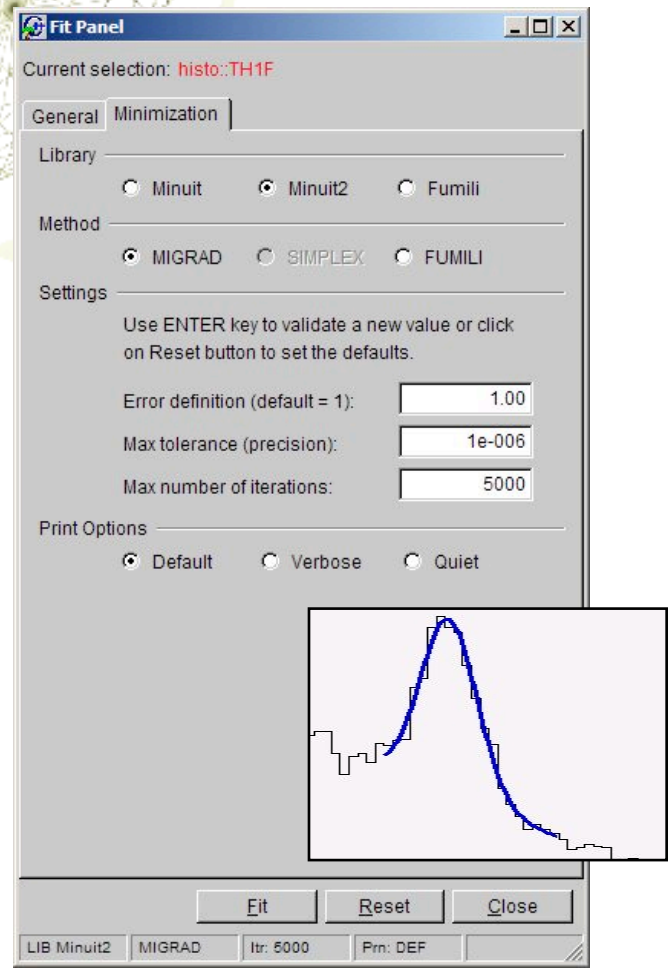


- Easy data set selection, built-in function, fit method, model
- Available via context menu
- Recognize and accept user-defined functions
- Allow use of different minimizers
- Flexible parameters' settings and control
- Same interface for all ROOT data objects (binned and unbinned data)

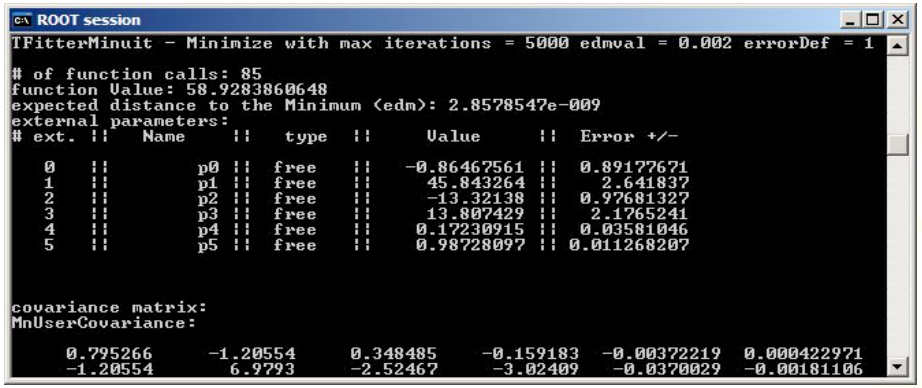
General Tab


Fitting Demo
File Edit View Options Inspect Classes Help
Lorentzian Peak on Quadratic Background
Data
Background fit
Signal fit
Global Fit
Set Parameters of fitFunction
Name Fix Bound Value Min Set Range Max Step Errors
p0 -0.864649 -2.59395 2.59395 0.259395 0.891776
p1 45.8433 -137.53 137.53 13.753 2.64183
p2 -13.3214 -39.9641 39.9641 3.99641 0.976811
p3 13.8074 -41.4221 41.4221 4.14221 2.17651
p4 0.172307 -0.516922 0.516922 0.0516922 0.0358097
p5 0.987281 -2.96184 2.96184 0.296184 0.0112681
Immediate preview
Reset Apply OK Cancel
Fit Panel
Current selection: histo::TH1F
General Minimization
Function
Predefined:
fitFcn
Operation
Nop Add Conv
fitFcn
Selected:
fitFcn
Set Parameters...
Fit Settings
Method
Chi-square
User-Defined...
No Chi-square
Integral
Use range
Best errors
Improve fit results
All weights = 1
Add to list
Empty bins, weights=1
Draw Options
SAME
No drawing
Do not store/draw
Advanced...
Fit Reset Close
LIB Minuit MIGRAD Itr: 5000 Prn: DEF

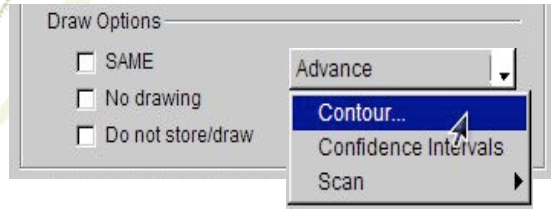
Minimization Tab

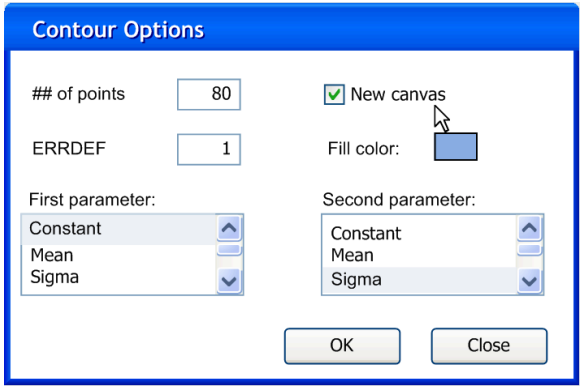


- Interactive library selection
- Easy choice of minimization method
- Users can specify
  - Error definition
  - Maximum tolerance
  - Maximum number of iterations
- Print Options
  - Default, Verbose, Quite

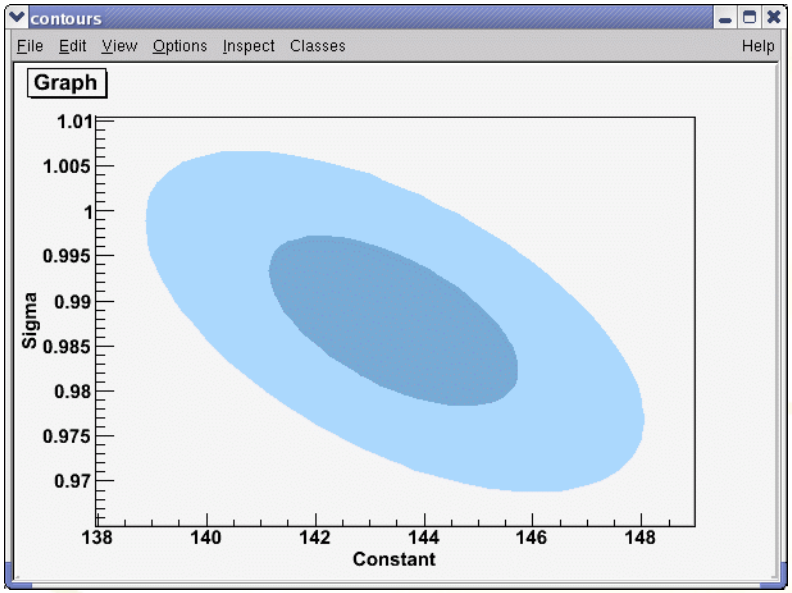


New Coming Features

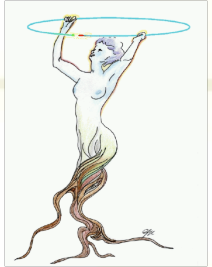




- Advanced draw options
  - contours, confidence intervals
- Implement more built-in functions
- Fitting of trees, multi-graphs and multi-histograms
- Interface for unbinned fits (*TTree*)
- Support of user-defined fit methods
- Interface to pdf classes of RooFit
- Straightforward way to build complicated models: add, multiply and convolute

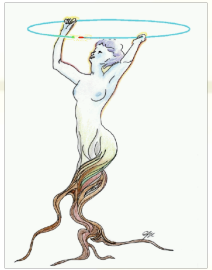


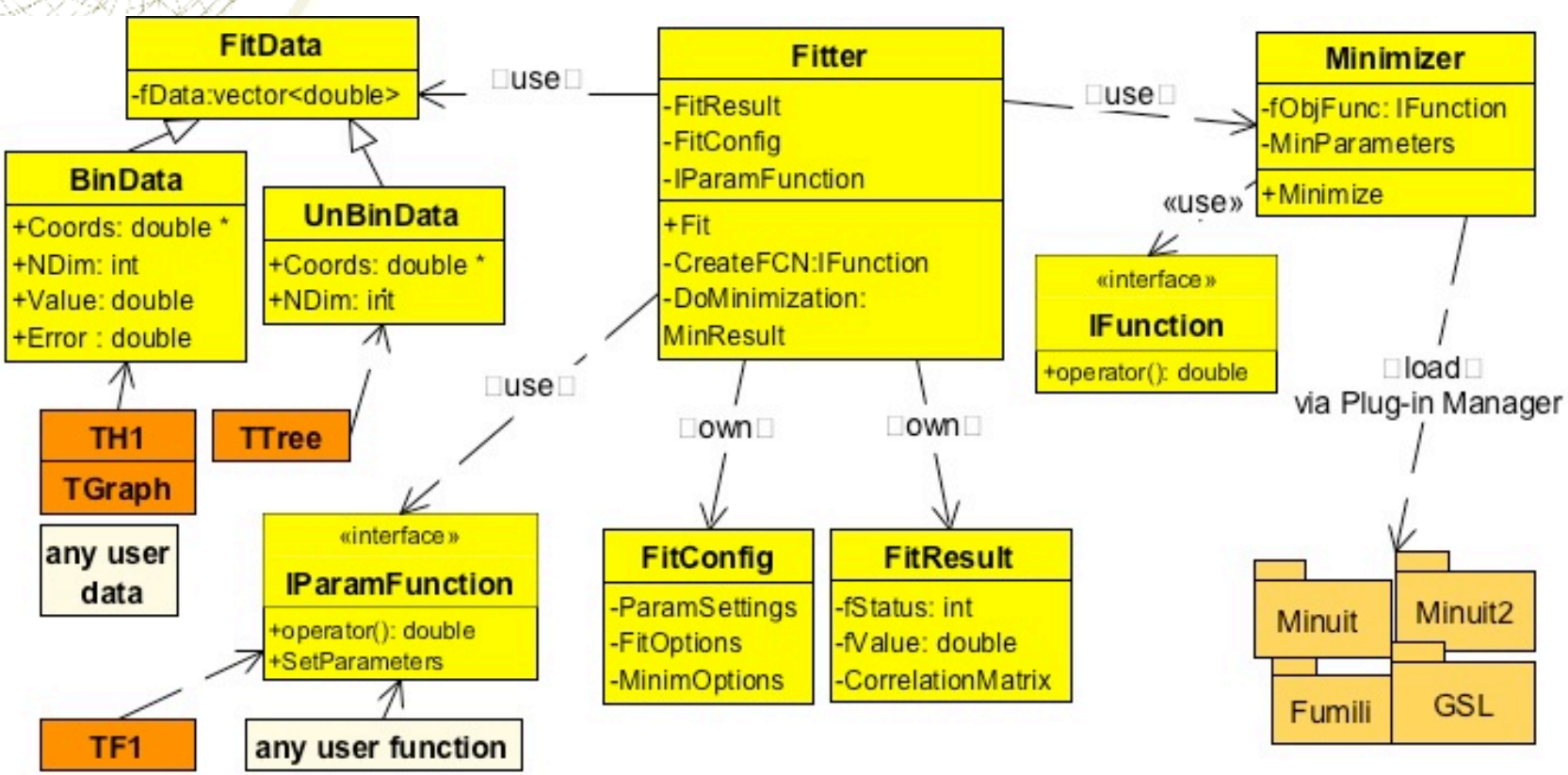
ROOT Fitting Extensions



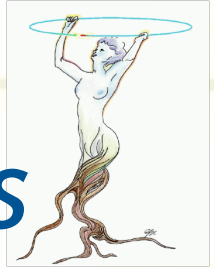
- Plan to introduce in ROOT new fitting classes
  - improve and extend functionality of *TVirtualFitter*
- Main features
  - Separate Fitter and Minimization interfaces
    - possible to use (and mix) various minimization engines
  - Decouple fitting from data source
    - fitter can be used on different data
    - fit results can be stored and retrieved
  - Minimal and generic function interface for model functions (pdf) and objective (minimization) functions
    - easier for user to plug-in their functions
    - easy to wrap and use pdf classes from *RooFit*
    - support for parallel fits (use it in a multi-threads environment)

Preliminary Fitter Design Proposal





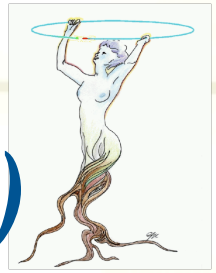
Status of ROOT Fit Extensions



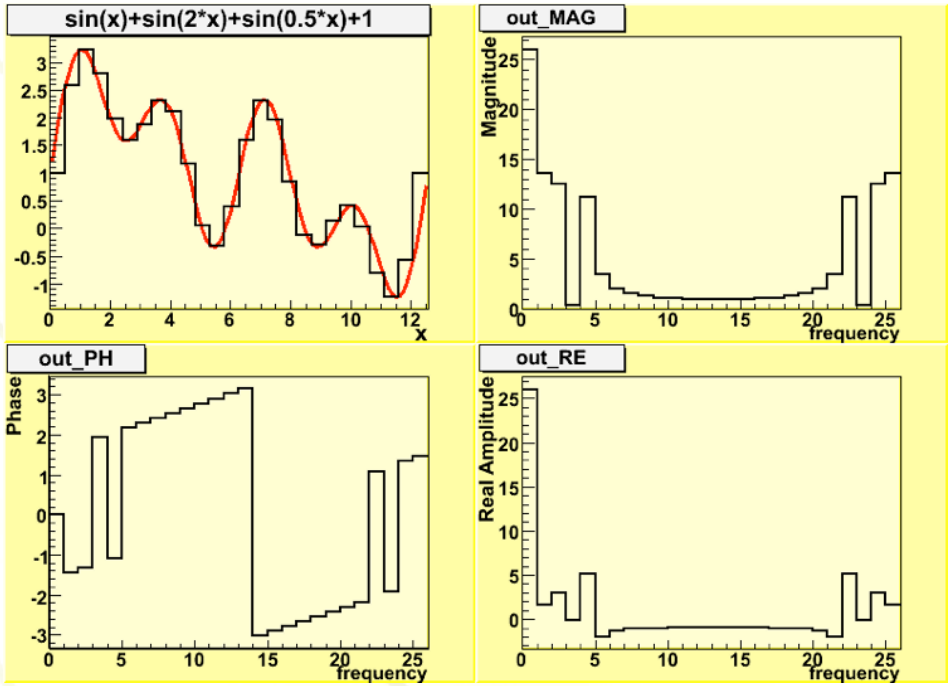
- Prototype Fitter working for Least Square and Unbinned Likelihood
- Developed Minimizer interface
  - implementations based on *Minuit*, *Minuit2* and *GSL*
  - loading using the ROOT plug-in manager
- First version soon in CVS
  - use for Fit Panel (GUI)
  - re-implement *Fit(...)* methods in ROOT data classes
  - re-implement *TVirtualFitter* methods to maintain backward compatibility

```
// fit inputs
TH1 * h1   = .....
TF1 * func = .....

ROOT::Fit::BinData d;
// fill the data set from the histogram
ROOT::Fit::FillData(d,h1);

// create wrapped parametric function
ROOT::Math::WrappedTF1 f(*func);

// create fitter
ROOT::Fit::Fitter

// set minimizer and configuration
fitter.Config().SetMinimizer("Minuit2");

//perform the fit
bool ret = fitter.Fit(d,f);

//retrieve optionally the fit results
if (ret) fitter.Result().Print();
```

Fast Fourier Transforms (FFT)



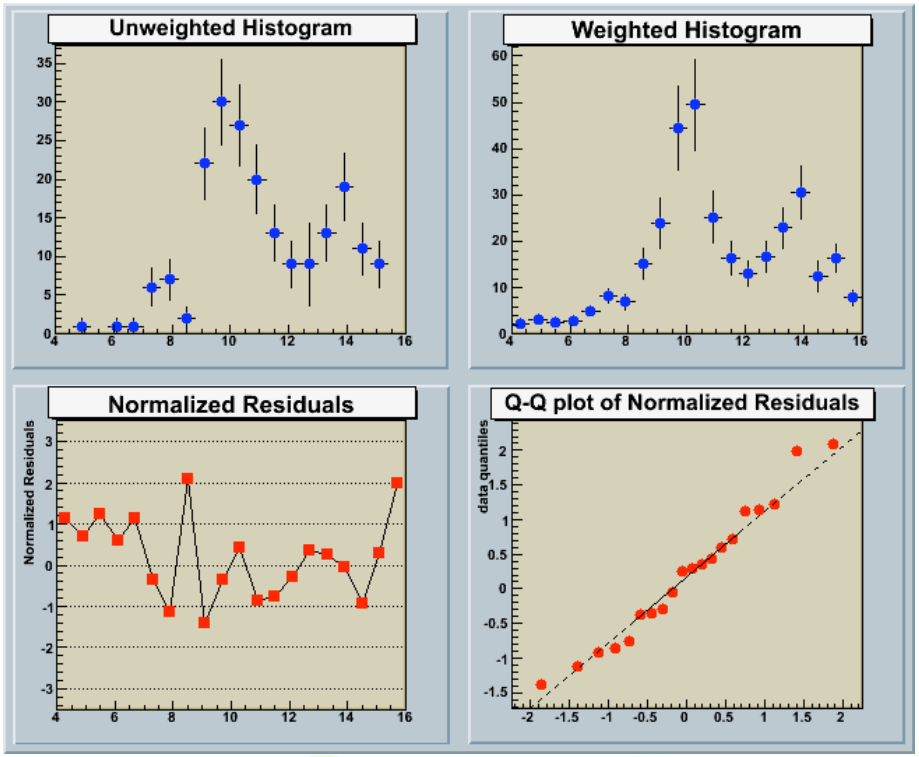
- Included in ROOT a common base class (*TVirtualFFT*)
  - add a functions to use it from TH1 (*TH1::FFT*)
- Implemented an interface to the popular *FFTW* package
  - support for one and multi-dimensional transforms
  - support for complex and real transformations



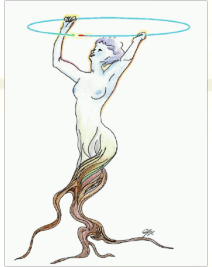
- *TFFTComplex* for complex input/ complex output transforms
- *TFFTRealComplex* for real input/ complex output
- *TFFTComplexReal* for complex input/real output
- *TFFTReal* for real input/output

Histogram Comparison

- Improvements in Chi2 test for comparing histograms
  - algorithm from *N. Gagunashvili* and implemented in C++ by *D. Haertl*
    - add possibility to use weighted histograms
    - comparison of histograms with different scales
    - produce normalized residuals

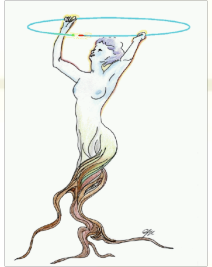


New Statistical Tools



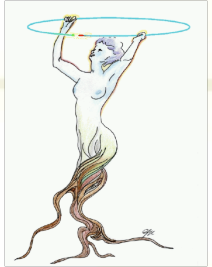
- Package for multi-variate analysis: *TMVA* (see J. Stelzer's talk)
  - various tools for multi-variate signal/background discrimination
    - likelihood estimator, neural networks, decision trees, etc..
    - common interface to all the methods
- Evaluate other MVA packages (*StatPatternRecognition*)
  - ROOT interface has been already developed
- Re-organization of statistical tools in ROOT
  - driven by requirements from LHC experiments
  - on-going developments for a statistical framework for evaluating discovery significances and confidence levels
    - based on *RooFit*
    - support for Frequentists and Bayesian techniques
    - easy way to share and combine results between experiments
- Planning to introduce new statistical algorithms
  - multi-dim smoothing, density estimators, cluster analysis

Future Work



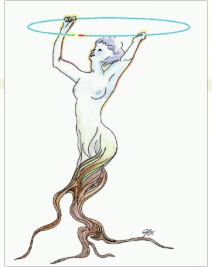
- Complete integration with existing ROOT Math functionality
  - add common interfaces to various implementations
  - remove duplication (obsolete algorithms or functions)
- Complete improvements in fitting and minimization classes
  - improve as well goodness of fit tests
  - add constraint minimization
- Developments of new statistical tools for LHC data analysis
  - collaborating with LHC experiments
    - facilitate user contributions with new tools
  - tools required for visualization of multi-dimensional data-sets
- Work on requests and feedback from LHC experiments

Conclusions



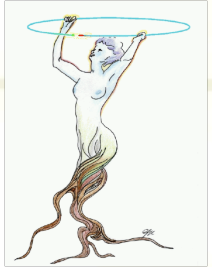
- Large collection of math and statistical tools available
  - working on improving them for better usability and for easier integrations of new tools
- Considerable efforts from external contributors in developing tools for LHC analysis
  - complex fitting (*RooFit*)
  - multivariate analysis (*TMVA*)
  - new statistical tools for discovery, confidence levels and result combination
- Important to ensure the correctness of tools we are using
  - need to improve validation and test suites
  - good documentation for reference and maintainability
- Need continuously the feedback from users and experts

Documentation



- Online reference documentation
  - class description with *THtml* and *Doxygen*
    - for example (*MathCore*):
      - http://root.cern.ch/root/htmldoc/MATHCORE_Index.html
- Maintain inventory of all Math functions and algorithms with links to ROOT online doc and CERNLIB
    - http://www.cern.ch/mathlibs/mathTable.html
- Written a new Math chapter in the ROOT User Guide *(chapter 13, 227-248)*
  - describe random numbers, *MathCore* (*GenVector*), mathematical functions, *SMatrix*
  - see ftp://root.cern.ch/root/doc/13MathLibraries.pdf
- Separate user guides exist also for other packages
  - *Minuit2, RooFit, TMVA*

References



- *ROOT* online doc :
  - Class doc: http://root.cern.ch/root/html/ClassIndex.html
  - *MathCore* online doc: http://www.cern.ch/mathlibs/sw/MathCore/html/index.html
  - *MathMore* online doc: http://www.cern.ch/mathlibs/sw/MathMore/html/index.html
  - *SMatrix* online doc: http://www.cern.ch/mathlibs/sw/SMatrix/html/index.html
  - *Minuit2* online doc: http://www.cern.ch/mathlibs/sw/Minuit2/html/index.html
- *RooFit* homepage: http://roofit.sourceforge.net/
- *TMVA* homepage: http://tmva.sourceforge.net/
- *FFTW* homepage: http://www.fftw.org/
- Histogram comparison paper: http://arxiv.org/abs/physics/0605123
- *UNURAN* homepage: http://statmath.wu-wien.ac.at/unuran/