

# Recent Developments of the ROOT Mathematical and Statistical Software

**L. Moneta, I. Antcheva, R. Brun**

CERN, PH Department, 1211 Geneva, Switzerland

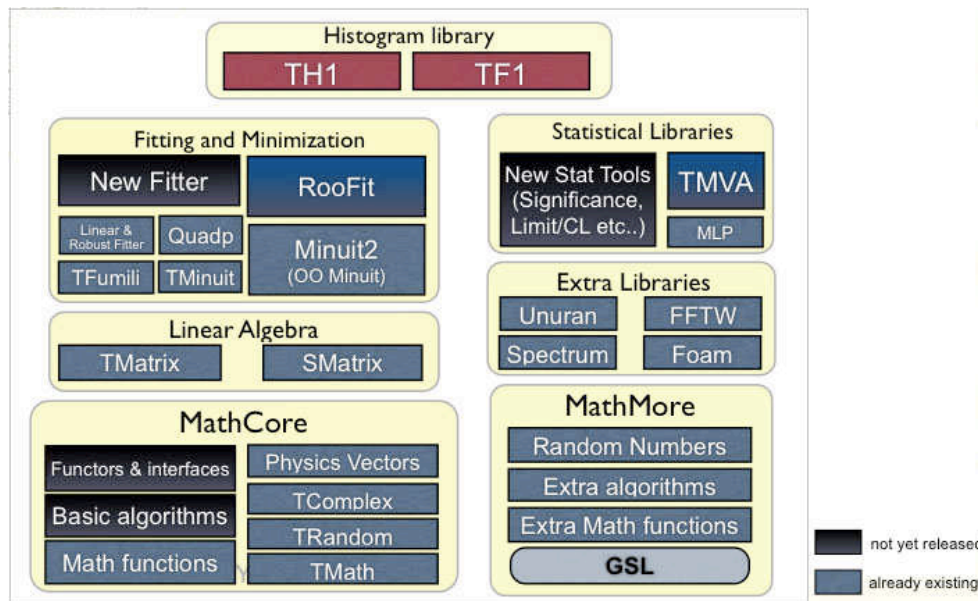
E-mail: [Lorenzo.Moneta@cern.ch](mailto:Lorenzo.Moneta@cern.ch)

**Abstract.** Advanced mathematical and statistical computational methods are required by the LHC experiments to analyze their data. These methods are provided by the Math work package of the ROOT project. An overview of the recent developments of this work package is presented by describing the restructuring of the core mathematical library in a coherent set of C++ classes and interfaces. The achieved improvements, in terms of performances and quality, of numerical methods present in ROOT are shown as well. New developments in the fitting and minimization packages are reviewed. A new graphics interface has been developed to drive the fitting process and new classes are being introduced to extend the fitting functionality. Furthermore, recent and planned developments of integrating in the ROOT environment new advanced statistical tools required for the analysis of the LHC data are presented.

## 1. Introduction

The ROOT MATH work package is responsible to provide and to support a coherent set of mathematical and statistical libraries required for simulation, reconstruction and analysis of high energy physics data. Existing libraries provided by ROOT are in the process of being re-organized with the aim to avoid duplication, increase modularity and to facilitate support in the long term. The main library components are the followings and shown in figure 1.

- **MathCore:** a self-consistent minimal set of mathematical tools and implemented as simple functions or C++ classes and required for basic HEP numerical computing.
- **MathMore:** a package incorporating advanced numerical functionality which might be needed for only specific applications (as opposed to MathCore which addresses the primary needs of users) and dependent on external libraries like the GNU Scientific Library [1].
- **Linear Algebra:** vector and matrix classes and linear algebra functions. Two libraries exist: a general matrix package and an optimized one (SMatrix) for small and fixed size matrices.
- **Fitting and minimization libraries:** classes and libraries implementing various types of fitting and function minimization methods.
- **Statistical libraries:** mainly packages for multivariate analysis methods and various statistical tools for computing confidence levels and discovery significances.
- **Histogram library:** classes for displaying and analyzing one or multidimensional binned data. Multidimensional unbinned data sets are handled by the tree library.



**Figure 1.** New structure of the ROOT Mathematical Libraries, showing the components already existing and those which are in the process of being developed.

In the following sections a detailed description is given for some of these components which have been recently developed and released. A brief description will be given also for those components that are planned to be introduced in ROOT.

## 2. MathCore

MathCore provides the basic and most used mathematical functionality. It is an self-consistent component which can be released as an independent library and used outside of the ROOT framework. As shown in figure 1, MathCore consists up to now of commonly used mathematical functions; the physics and geometry vector package containing classes for specialized vectors in 2D, 3D and 4D and their operations; interfaces classes for functions and numerical algorithms.

It is planned to merge in MathCore classes and code currently present in the other ROOT libraries, like the random number (now in libCore) or implementations of some basic numerical algorithms (now in libHist). The goal of this restructuring is to remove duplications, improve the overall modularity of ROOT and the quality of the implementations in terms of CPU performances and numerical accuracy.

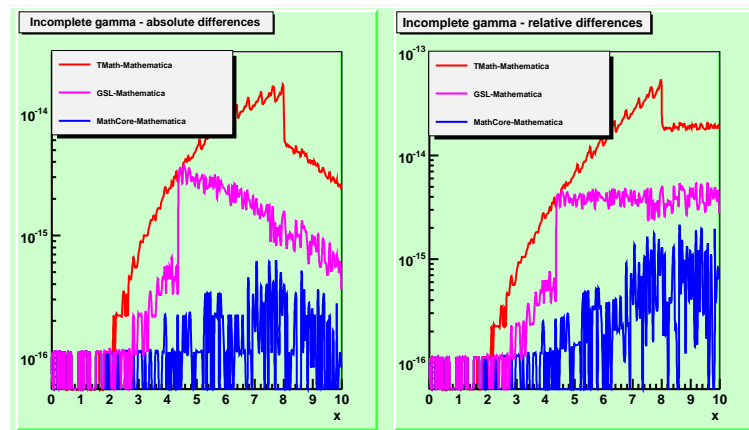
A more detailed description of the recent developments in MathCore is given in the following subsections. The physics vector package, which is currently used by the LHC experiments, has been presented in the previous CHEP conference [2] and it will not be described here.

### 2.1. Mathematical Function

These functions are defined as free function in the name-space `ROOT::Math`. The most used special functions, like the error function, gamma or beta are included in the MathCore library, while other special functions, such as Bessel functions are included instead in the MathMore library. The naming used is the one proposed as next extension of the C++ Standard Library [4]. Commonly used statistical distribution functions are also provided in a coherent naming scheme. For each different distributions there are available: the probability density function (with suffix `_pdf`: `normal_pdf` for the normal distribution); the cumulative distribution function (`_cdf`); the complement of the cdf (`_cdf_c`) and the quantile, inverse of the cdf, (`_quantile` and

\_quantile\_c). For a complete list of all the available mathematical functions see the user guide [3].

In the latest ROOT release (5.18), the function implementations have been improved, by using code imported from the Cephes [5] library. These new implementations are accurate at the required double numerical precision level and they are more efficient in term of CPU time than the previous ones implemented using GSL or in TMath. An example is given in figure 2. In addition this new code can be used within the terms of the ROOT license.



**Figure 2.** Absolute and relative difference in the values obtained by the incomplete gamma function (used for estimating the  $\chi^2$  probability) with *Mathematica* and the new MathCore implementation (blue) based on Cephes, TMath implementation (red) and GSL implementation (magenta).

## 2.2. Function interfaces and Functors

For decoupling the algorithm implementations from the function definition, abstract interfaces for function evaluation in one or multi dimensions have been introduced. They are used by all the ROOT Math classes implementing numerical algorithms, such as integration, derivation or minimization. Specialized interfaces are as well present for functions providing analytical derivatives and for parametric functions which are used for fitting and data modeling.

For user convenience, adapter classes and functors are available to wrap any C++ callable object with the right signature in the desired interface. In this way, user function classes can be easily created and plugged in the required numerical algorithm class. For example, one-dimensional function classes can be created from global functions like `double f(double x)`, from C++ classes implementing the `double operator()(double x)` or any member function of a class returning a `double` and taking a `double` as argument. This approach is very powerful, giving the user the possibility to customize the function objects using its constructor, and also very easy to use.

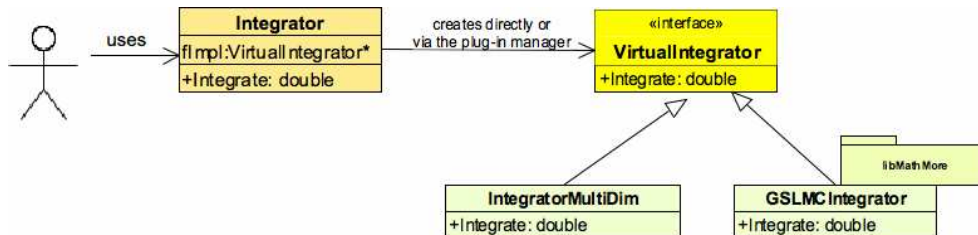
The existing TF1 class in ROOT has also been extended with the possibility to be created from a functor class with the signature `(double F (double * x, double * p))`. This allows to create TF1 instances from non-global objects and from class member functions, which was not possible before.

## 2.3. Numerical Algorithms

Various numerical algorithms such as derivation, adaptive and non-adaptive integration, interpolation, minimization and root finders are available in ROOT since some time in the

TF1 class of the Hist library. New algorithms have been implemented using the GNU Scientific Library (GSL) [1] in the MathMore library to complement and extend those of TF1.

In order to have a common entry point, interfaces classes for these numerical algorithms are being developed. These interfaces are then implemented by derived classes, which can be located in different ROOT libraries and they can be loaded automatically using the ROOT plug-in manager. For improving the modularity, the algorithm implementations present in the TF1 class are being moved in some of these derived classes.



**Figure 3.** Interfaces and implementation classes for multidimensional numerical integration

For example, see figure 3), in the case of multidimensional numerical integration, an implementation based on the adaptive quadrature algorithm is present in a MathCore (class `IntegratorMultiDim`) and GSL based Monte Carlo integration methods (available are available in MathMore (class `GSLMCIntegrator`). The user interacts with a common class, which at construction time, according to the given option, instantiates the corresponding implementation class. Using the plug-in manager we ensure then that libMathCore is independent at compile time from other ROOT libraries.

The complete list of numerical algorithms recently added in MathCore or MathMore is available from the reference ROOT documentation [6].

#### 2.4. Random Numbers

In ROOT pseudo-random numbers can be generated using the `TRandom` classes. A base class provides common methods for generating uniform and non-uniform numbers (according to specific distributions), while derived classes, such as `TRandom3`, implement pseudo-random number generators. These classes have been recently improved by replacing some obsolete generators. The following pseudo-random number generators are currently available in ROOT:

- Mersenne and Twister generator [7] implemented in the class `TRandom3`. This is the default generator in ROOT and the recommended one for its very good random proprieties and its speed. It can also be seeded automatically with a 128 bit UUID number in order to generate independent streams of random numbers which can be used in parallel jobs.
- RanLux generator [8] provided via the class `TRandom1`.
- Tausworhte generator [9] from L'Ecuyer implemented in the class `TRandom2`. This generator is fast and has the advantage to use only 3 words of 32 bits for its state.

The CPU time results for generating a pseudo-random number using the ROOT generators are shown in the table below.

The base class `TRandom` provides also a Linear Congruential Generator. This generator has a state of only 32 bits (one single word) and therefore a very short period ( $2^{31}$ ) and should not be used in any statistical application.

Random Number Generator	Intel 32	Intel 64
MT (TRandom3)	22 ns	9 ns
TausWorthe (TRandom2)	17 ns	6 ns
RanLux (TRandom1)	120 ns	98 ns

**Table 1.** CPU time for generating one pseudo-random number on a Linux PC with the 32 or 64 bit architecture running CERN Scientific Linux 4 and using the GNU gcc version 3.4 compiler

TRandom implements as well methods for generating random numbers according to specific distributions. Recently a new faster algorithm for generating normal distributed random numbers, based on the acceptance-complement ratio method (ACR) [10], has been added to ROOT. This algorithm is much faster than the Box-Muller (polar) method used previously, since it avoids the evaluation of mathematical functions like `sqrt` or `log`. On a 64 Intel Linux box running ROOT compiled with gcc 3.4, the time for generating one random gaussian number has been decreased from 183 to 42 ns.

The latest ROOT releases, from version 5.14, contain an interface to UNU.RAN [11], a software package for generating non-uniform pseudo-random numbers. UNU.RAN provides universal (also called automatic or black-box) algorithms that can generate random numbers from large classes of continuous (in one or multi dimensions), discrete distributions, empirical distributions (like histograms) and from all standard distributions. Efficient methods based on Markov-Chain Monte Carlo are as well provided for multidimensional distributions.

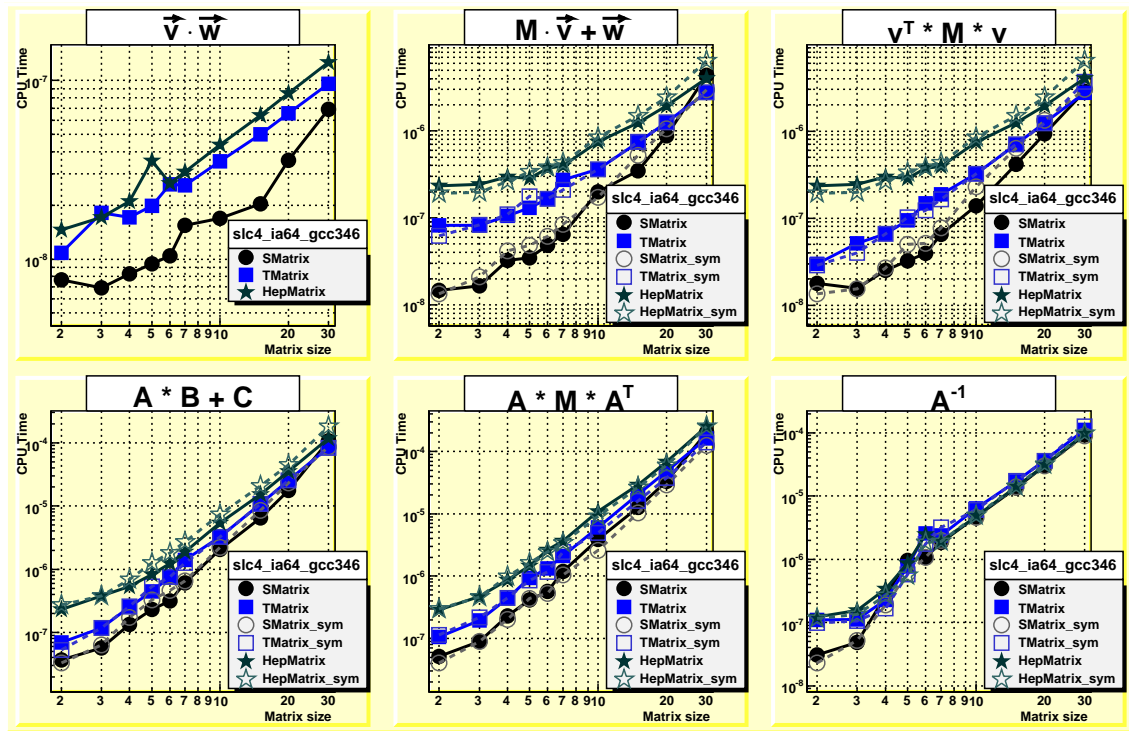
### 3. SMatrix Package

ROOT contains a general matrix package (TMatrix classes) for describing matrices and vectors and their linear algebra operations in arbitrary dimensions and of various types and a dedicated package, SMatrix, for fixed and small size matrices. SMatrix is based on C++ expression templates to achieve an high level optimization and minimize memory allocation in matrix operations. It is based on a package developed for HeraB [12] and it has been already presented in the CHEP06 conference [2]. The matrix and vector classes are templated on the scalar type and on the dimension, like number of rows and columns for a matrix. The matrix classes have in addition as template parameter, the storage representation. This extra parameter differentiates general and symmetric matrices. The package is designed for small size matrices, when maximum performances are achieved by avoiding temporaries with expression templates, and by having the functions inline. The disadvantages of this approach are large code size and long compilation time, which both increase when the matrices get bigger in size. It is therefore not recommended to use SMatrix for matrix (and vector) dimensions larger than 10.

Figure 4 shows the performances of SMatrix, comparing with TMatrix and CLHEP. The Table shows the performances of the three packages on different platform for a benchmark application, the Kalman filter update equation. These tests demonstrate the advantage of using SMatrix for fixed small sizes. SMatrix is used by the LHC experiments in their reconstructions for representing fixed size matrices such as covariance matrices obtained in track fits. Large improvements by using SMatrix instead of CLHEP have been observed as well by the experiments.

### 4. Fitting and Minimization

Fitting in ROOT is possible directly via the `Fit(...)` methods of the data object classes like histograms graphs and trees. Fit methods such as least-square or binned and un-binned likelihood are supported. An interface class, TVirtualFitter exists to perform more sophisticated fits and to interface the minimization packages, like Minuit [13], Fumili [14] or



**Figure 4.** Comparison in matrix operation between SMatrix, TMatrix and HepMatrix (CLHEP), for squared and symmetric matrices of various dimensions obtained on a Linux with an x86-64 processor running Scientific Linux 4. This is the main component of the current CERN cluster and it yields the best relative performances for SMatrix (see also table 3)

Architecture and compiler	Linux Intel x86-64 gcc 3.4	Mac OSX Intel x86-64 gcc 4.0	Linux Intel x86-32 gcc 3.2	Windows Intel x86-32 VS 7.1
Matrix sizes	<b>N1 = 5, N2 = 2</b>			
SMatrix	0.37 $\mu$ s	0.55 $\mu$ s	0.80 $\mu$ s	0.58 $\mu$ s
TMatrix	0.98 $\mu$ s	0.98 $\mu$ s	1.46 $\mu$ s	1.30 $\mu$ s
CLHEP matrix	3.13 $\mu$ s	4.88 $\mu$ s	5.62 $\mu$ s	-
Matrix sizes	<b>2 <math>\leq</math> N1 <math>\leq</math> 6, 2 <math>\leq</math> N2 <math>\leq</math> 6</b>			
SMatrix	23.4 $\mu$ s	27.4 $\mu$ s	40.4 $\mu$ s	29.4 $\mu$ s
TMatrix	53.1 $\mu$ s	33.1 $\mu$ s	64.5 $\mu$ s	48.7 $\mu$ s
CLHEP matrix	104.1 $\mu$ s	133. $\mu$ s	169.2 $\mu$ s	-
Matrix sizes	<b>6 &lt; N1 <math>\leq</math> 10, 6 <math>\leq</math> N2 <math>\leq</math> 8</b>			
SMatrix	163 $\mu$ s	162 $\mu$ s	308 $\mu$ s	182 $\mu$ s
TMatrix	305 $\mu$ s	175 $\mu$ s	354 $\mu$ s	338 $\mu$ s
CLHEP matrix	466 $\mu$ s	544 $\mu$ s	864 $\mu$ s	-

**Table 2.** CPU time results for running the Kalman benchmark test (update of the covariance matrix) on various machines and compilers and for different group of matrix sizes. First is the timing for the update using 2x2, 5x5 and 5x2 matrices (typical track fit use case). After, for  $\leq N1, N2 \leq 6$  and  $6 < N1, N2 \leq 10, 8$  the time is the inclusive results of all the possible 25 and 38 combinations.

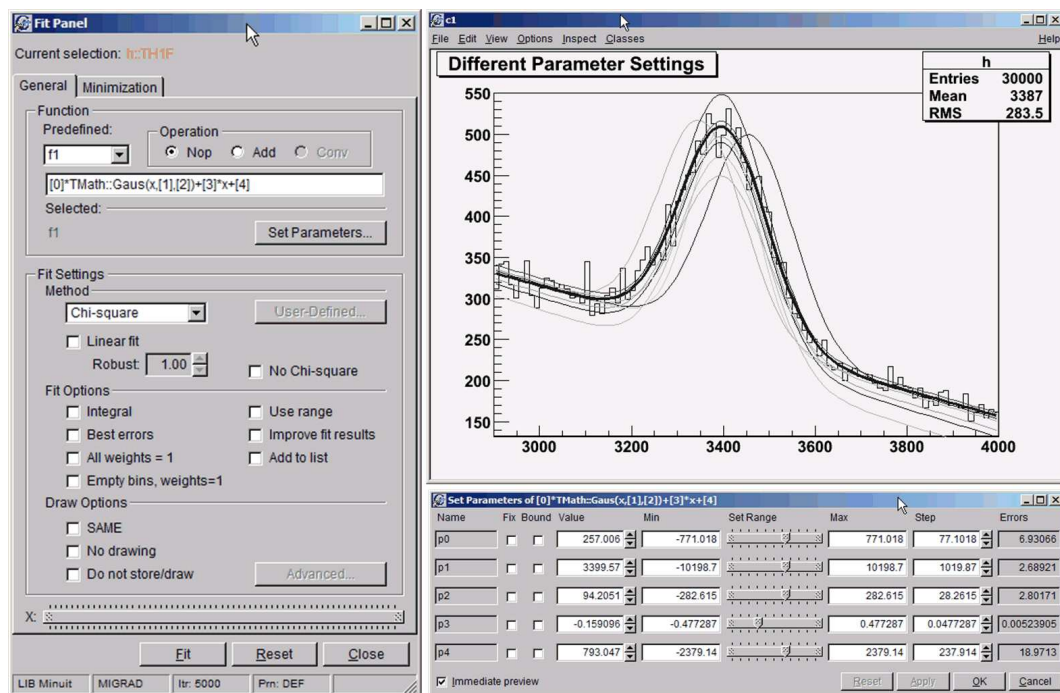


Minuit2 [15], the new objected oriented version of Minuit. In the case of linear fits a dedicated class, `TLinearFitter`, exists. An extension to the linear fitter (robust fitter) for removing bad observations, outliers, based on the approximate Fast Least Trimmed Squares (LTS) regression algorithm for large data sets [16] exists as well. More complex fits can be performed by using the RooFit package [17], distributed now within ROOT.

Fits can be controlled by using the ROOT Graphics User Interface (GUI), which has been re-designed recently and improved by adding new functionality.

#### 4.1. The Fit Panel

The new fit panel Graphical User Interface (GUI) became a part of the ROOT version 5.14. Its goal is to provide a more user friendly way for fitting directly the ROOT data objects with the various available options. It allows an easy selection of a data set, built-in function, fit method and model. The fit panel is a mode-less dialog, when opened, it does not prevent users from interacting with other windows. When the fit panel is active, users can select an object for fitting in the usual way for selecting an object in the ROOT canvas, i.e. by left-mouse click on it. If the selected object is suitable for fitting, the fit panel is connected with this object and users can perform fits by setting different options, parameters values, etc. The fit panel can be activated via the context menu of any ROOT object suitable for fitting. Furthermore, the fit panel implementation includes methods, which allow users to embed this interface in their applications.



**Figure 5.** The General tab (left) and the Set Parameter dialog (right) of the fit panel. Several functions for different parameters values are plotted together with the result of the fit. The function parameters can be changed and viewed immediately using the dialog slider.

By design, the user interface is separated in two tabs: *General* and *Minimization* (see figure 5). The *General* tab provides user interface elements for setting the fit function, fit method and different options for fitting. The combo box *Predefined* contains by default a list of predefined functions in ROOT like polynomials or Gaussian functions. The user-defined fit functions are

recognized and included in this list. Also, users are free to enter the function expression into the text entry field below the *Predefined* combo box.

The parameters settings are very flexible and allow users control via the dialog that shows up via the button *Set Parameters* (see Fig 5). Users can set the initial parameter values or they can change the values after the fitting and having observed the result. In addition, they interactively can fix parameters or set boundary limits. The combo box *Method* provides fit model choices depending on the data set. For example for histograms, the Chi-square or binned likelihood methods are provided. In case of linear fits, the check button *Linear Fit* sets the use of the linear fitter to find directly the solution without using a numerical minimization method. The *Minimization* tab offers an easy choice of minimizer, minimization methods and possibility for specifying the minimization parameters such as error definition, maximum tolerance, the maximum number of iterations, and print options.

The fit panel interface is flexible and allows extensions. It is planned to include developments related to advanced drawing options (including contours and confidence levels), more built-in fit functions, interface to ROOT trees using unbinned maximum likelihood fits, fitting multi-graphs, multi-histograms and support of user-defined fit methods. In long term extensions are planned for building more complex fit function models and to interface to PDF classes of the RooFit toolkit [17].

#### 4.2. Developments of ROOT Fitting Extensions

It is planned to improve the existing ROOT fitting classes, by extending the functionality of the `TVirtualFitter` class and by providing additional fitting and minimization methods, support for fits running on parallel architectures and easier integration with RooFit.

The new ROOT fitting extensions will consist of new separate classes for the fitting and minimization process. The role of the fitter class is, given the data and a model function, to build the appropriate objective function to be minimized via the minimizer class. The fitter drives as well the minimization process, by setting all the required control parameters. With this design, it is possible to use with the same fitting class different minimization algorithms, which can then be part in various libraries and instantiated via the plug-in manager.

Another important characteristics of this extension, is the decoupling between the data source and the fitter class. In this way the same fitter class can be re-used on many different types of data sets, not only on ROOT data objects like the histogram classes. Furthermore, the results of the fit are represented within a separate object which can be stored and retrieved. The minimal function interfaces defined in MathCore are then used for the fitting (model) function and for the minimization (objective) function. It will then be possible to use different function classes, such as `TF1` or pdf classes defined in RooFit. Another important capabilities will be the support for parallel fits using multiple threads for time consuming tasks in order to achieve optimal scalable performances on multi-core CPU's.

#### 4.3. New Statistical Tools

For multivariate analysis and signal-background discrimination a new package, TMVA [18], has been integrated recently in ROOT. It provides various algorithms, like automatic cuts optimizations, likelihood estimators, neural networks and boosted decision trees with common interfaces to use all these multivariate methods easily together.

A new package is also currently being developed to extend and improve the functionality of estimating confidence levels to satisfy the LHC requirements and focusing in particular on estimating discovery significances. It will both include frequentists and bayesian methods and it will be based on the RooFit data modeling framework [19]. Tools for easy statistical combinations of results will be as well provided by this new package.



Improvements in the comparison of histograms have been recently introduced by using a new algorithm for the  $\chi^2$  test. The new algorithms [20] provides the possibility to compare as well weighted histograms, histograms with different scales and produces also normalized residuals.

It is planned to introduce new tools for density estimation and smoothing for multidimensional functions and cluster analysis algorithms. An effort will be spent also on reorganizing the existing statistical tools by grouping them in a common library and remove duplications. The priority will be however in developing the statistical tools required by the LHC experiments to analyze their data.

## 5. Conclusions

ROOT contains already a large variety of mathematical and statistical functionality required for the analysis of LHC data. An effort is on-going to consolidate the existing libraries by improving the algorithms, by making them easier to use and by increasing their modularity to gain in long term maintainability. The needs and the feedback received from users working on data analysis and reconstruction of the experiment data are as well taken into account in this consolidation process. Many of the tools currently present in ROOT have been developed by various contributors from the high energy physics community. It is therefore important to ensure a continuation of these user contributions and to provide an easy way for the users to plug-in their developed tools. In addition, this consolidation effort should aim to remove duplications and provide implementations which are considered standard by the community.

## References

- [1] M. Galassi et al., *The GNU Scientific Library Reference Manual* - Second Edition, ISBN = 0954161734 (paperback). See also the url <http://www.gnu.org/software/gsl>.
- [2] L. Moneta et al., *New Developments of ROOT Mathematical Libraries*, proceeding to CHEP-2006, Vol. I, 367.
- [3] R. Brun et al., *Math Libraries in ROOT*, ROOT User's Guide, Chapter 13, available online at the url <ftp://root.cern.ch/root/doc/13MathLibraries.pdf>.
- [4] W. Brown and M. Paterno, *A proposal to Add Mathematical Special Functions to the C++ Standard Library*, WG21/N1542, available at <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2003/n1542.pdf>.
- [5] S. L. Moshier, Cephes library from <http://www.netlib.org/cephes>
- [6] MathMore reference doc, see the url <http://www.cern.ch/mathlibs/sw/html/MathMore.html>.
- [7] M. Matsumoto and T. Nishimura, *Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator*, ACM Trans. on Modeling and Computer Simulations **8**, 1, 3-20 (1998).
- [8] F. James, *RANLUX: A Fortran implementation of the high quality pseudo-random number generator of Lüscher*, Computer Physics Communication **79**, 111 (1994).
- [9] P. L'Ecuyer, *Maximally Equidistributed Combined Tausworthe Generators*, Mathematics of Computation **65**, 203-213 (1996).
- [10] W. Hoermann and G. Derflinger, *The ACR Method for generating normal random variables*, OR Spektrum **12** (1990), 181-185.
- [11] J. Leydold et al., *UNU.RAN - A Library for Non-Uniform Universal Random Variate Generation*. Institut für Statistik, WU Wien, Austria, see the url <http://statistik.wu-wien.ac.at/unuran>.
- [12] T. Glebe, *SMatrix - A high performance library for Vector/Matrix calculation and Vertexing*, HERA-B Software Note 01-134, December 2, 2003.
- [13] F. James, *MINUIT Reference Manual*, CERN Program Library Writeup D506.
- [14] S. Yashchenko, *New method for minimizing regular functions with constraints on parameter region*, Proceedings of CHEP'97 (1997).
- [15] M. Hatlo et al., *IEEE Transactions on Nuclear Science* **52-6**, 2818 (2005)
- [16] P.J. Rousseeuw and K. Van Driessen, *Computing LTS Regression for Large Datasets*, Estadística **54**, 163 (2002).
- [17] see the url <http://roofit.sourceforge.net>.
- [18] F. Tegenfeld et al., *TMVA - Toolkit for multivariate data analysis with ROOT*, proceedings to the PHYSTAT-LHC 2007 Workshop, see also the url <http://tmva.sourceforge.net>.
- [19] W. Verkerke, *Statistical software tools for LHC analysis*, proceedings to the PHYSTAT-LHC 2007 Workshop.
- [20] N. Gagunashvili, *Comparison of weighted and unweighted histograms*, preprint arXiv:physics/0605123, available online at the url <http://arxiv.org/abs/physics/0605123>.