PROCEEDINGS OF SCIENCE

New Developments of ROOT Mathematical Software Libraries

L. Moneta*
CERN, Geneva, Switzerland
E-mail: Lorenzo.Moneta@cern.ch

I. Antcheva
CERN, Geneva, Switzerland
E-mail: Ilka.Antcheva@cern.ch

A. Kreshuk
CERN, Geneva, Switzerland
E-mail: Anna.Kreshuk@cern.ch

W. Brown, M. Fischler, J. Marraffino

FNAL, Batavia, IL 60510, USA

LHC experiments obtain needed mathematical and statistical computational methods via the coherent set of C++ libraries provided by the Math work package of the ROOT project. The new recent developments of this work package are presented. These developments include a new core library, MathCore, which has been developed as a self contained component encompassing basic mathematical functionality, MathMore providing a complementary and expanded set of C++ mathematical functions and algorithms based on a GNU Scientific Library, and SMatrix, a new liner algebra package optimized for small size matrices. An overview of the ROOT Mathematical libraries describing in greater detail the functionality and design of the packages recently introduced in ROOT is shown.

XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research
April 23-27 2007
Amsterdam, the Netherlands

*Speaker.

PoS(ACAT)042

1. Introduction

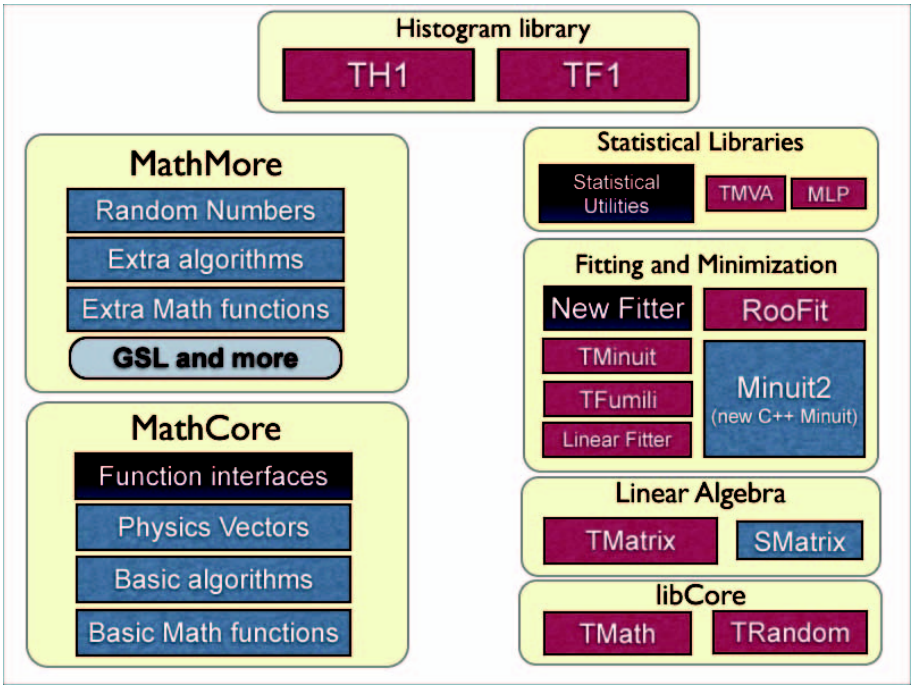

Figure 1: New ROOT Math Components

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

- **MathCore**: a self-consistent minimal set of mathematical functions and C++ classes for the basic needs of HEP numerical computing. It is released as an independent library.
- **MathMore**: a package incorporating functionality which might be needed for an advanced user (as opposed to MathCore which addresses the primary needs of users).
- **Linear Algebra**: classes describing vector and matrix in arbitrary dimensions and of various types. Two linear algebra libraries exist: a general matrix package completed with linear algebra algorithms and SMatrix, a dedicated package for small and fixed size matrices.
- **Fitting and minimization**: classes implementing various types of fitting methods, including the newly added linear and robust fitters and set of libraries for different function minimization algorithms like Minuit [1] and Fumili [2], which can be loaded at run time by using the plug-in manager system.
- **Histogram library**: classes for one, two and three dimensional histograms and profiles.

- **Statistical libraries**: various libraries grouping the various statistical algorithms of ROOT like neural networks or boost decision trees for multivariate analysis which are provided by the TMVA library, or classes for computing confidence levels and significances for discoveries. Some of the algorithms are already present in ROOT, but we expect in the future to group together in a new set of coherent libraries.

In the following sections a detailed description is given for those components, which have been recently developed and are now integrated inside 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. MathCore consists up to now of:

- commonly used special functions like the Gamma, Beta and Error function
- mathematical functions used in statistics such as probability density functions for the major distributions (normal, poisson, binomial, breit-wigner, etc..)
- the physics and geometry vector package containing classes for specialized vectors in 3D and 4D and their operations

In the near future, it is planned to include in MathCore classes for random number generation (`TRandom`), some additional mathematical functions provided by the existing `TMath` and numerical algorithms such as numerical integrations which are currently provided by the `TF1` class.

2.1 Mathematical functions

The special functions in MathCore (and MathMore) are implemented following the same naming scheme proposed as next extension of the C++ Standard Library (see C++ extension proposal [3]). Extensive tests of these newly introduced mathematical functions have been performed by comparing the numerical results obtained with the functions from other packages like Mathematica or Nag [4]. Often an accuracy at the level of 10^{-16} (double numerical accuracy) is reached for functions such as the Gamma and the Error function, improving with respect to the functions previously present in ROOT `TMath`.

All the major statistical functions are also provided with a coherent naming scheme. For each statistical function, it is provided the probability density function (with suffix `_pdf`), the cumulative distribution function (with suffix `_cdf`), the complement of the cdf (with suffix `_cdf_c`), the quantile function (inverse of the cdf, with suffix `_quantile`) and the inverse of the complement of the cdf (with suffix `_quantile_c`).

2.2 Physics and Geometry Vector package

This new package, called GenVector, is intended to represent vectors and their operations and transformations, such as rotations and Lorentz transformations, in 2, 3 and 4 dimensions. The 2D and 3D space are used to describe the geometry vectors and points, while the 4D space-time is used

for physics vectors representing relativistic particles. It has been developed in collaboration with the Fermilab Computing group.

Class templates are provided for modeling the 3D and 4D vectors. There is a user-controlled freedom on how the vector is internally represented. This is expressed by a choice of coordinate system which is supplied as a template prameter when the vector is constructed. A coordinate system can be one of several choices (Cartesian, Polar, Cylindrical and so forth). There is a further degree of control: each coordinate system is itself a template so that the user can specify the underlying scalar type.

The transformations are modeled by simple (non-template) classes, using double as the scalar-type. For the purposes of understanding the classes available, the transformations are grouped in: Rotations (in two and three dimensions), Lorentz transformations, and Poincare transformations, which are Translation/Rotation combinations. Each group has several members, which may model physically equivalent transformations but with different internal representations For example, a Rotation may be kept as a 3x3 matrix (class `Rotation3D`) or as an axis and angle of rotation (`AxisAngle`), or as 3 Euler angles (`EulerAngles`), or as a Quaternion.

For the 3D vectors two different classes exist: the `DisplacementVector3D` template class to model an abstract 3-component direction-and-magnitude vector, not rooted at any particular point, and the `PositionVector3D` template class to model a point in the 3D space. The two classes behave differently to the transformations, for example the `PositionVector3D` rotates and translates while the `DisplacementVector3D` only rotates. Equivalent classes exist also for the 2D vectors.

In order to minimize any overhead in the run-time performances, as much as possible of all the functions are inlined and the classes don't have any virtual function and even virtual destructors. Furthermore, users can also obtain optimal run-time performances, by choosing the best coordinate system for basing the vectors. For example, an analysis which requires to evaluate the differences in the azimuthal anghle Φ and pseudo-rapidity η between vectors, will profit from basing the vectors on a Cylindrical-Eta based coordinate system. Examples of performances obtained using vector based on two different coordinate system are shown in figure 2.

2.3 Random Numbers

In ROOT pseudo-random numbers can be generated using the `TRandom` classes. The classes have been recently improved by replacing some obsolete generators. Currently, the following four pseudo-random number generators are included in ROOT:

- Mersenne and Twister generator [5] provided by the class `TRandom3`. This is the default generator in ROOT and the recommended one for the very good random propriety and its speed. It can also be seeded automatically using a 128 bbit UUID number in order to generate independent and streams of random numbers.

- RanLux generator [6] provided by the class `TRandom1`.

- Tausworhte generator [7] from L'Ecuyer provided by the class `TRandom2`. This generatos has the advantage to use only 3 words of 32 bits for the state.

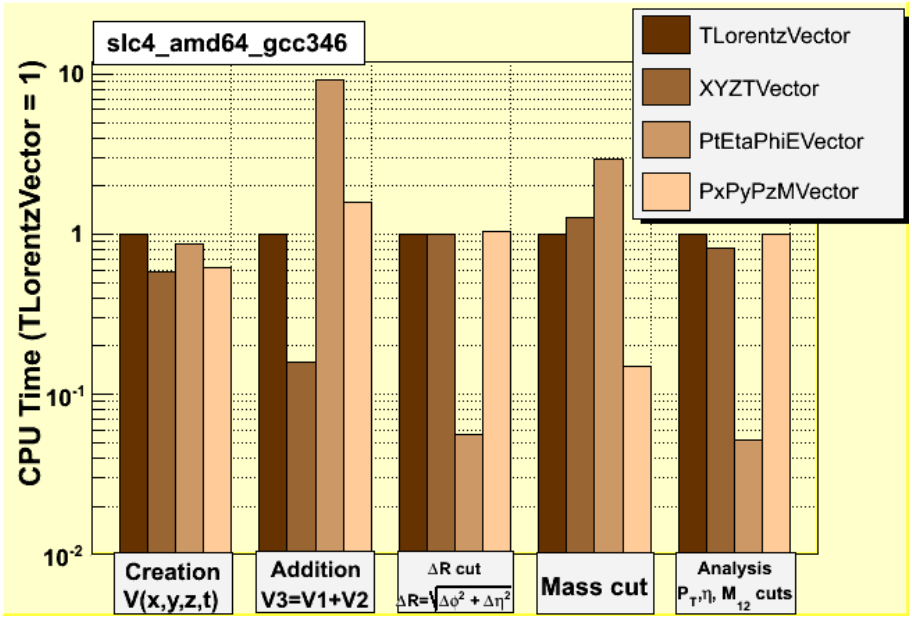

Figure 2: CPU time obtained using `TLorentzVector` and the new LorentzVector's based on Cartesian (`XYZVector`) and on Cylindrical-Eta (`PtEtaPhiVector` coordinates).

- Linear Congruential Generator provided directly in the base class `TRandom`. This generator has a state of only 32 bits and therefore a very short period 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
LCG (`TRandom`)	14 ns	5 ns

Table 1: CPU time (in nanoseconds) for generating one pseudo-random number on a Linux box with the 32 or 64 bit architecture running CERN Scientific Linux 4 and using the GNU gcc version 3.4 compiler

The CPU time for generating a pseudo-random number using the 4 generators are shown in table 2.3. The base class `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) [8], has been added to ROOT. This algorithm is much faster than the traditional Box-Muller method used previously in ROOT. For example, 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 releases of ROOT contains in addition an interface to UNU.RAN [9], a software package for generating non-uniform psedu-random numbers. It contains universal (also called automatic or black-box) algorithms that can generate random numbers from large classes of contin-

uous (in one or multi-dimensions), discrete distributions, empirical distributions (like histograms) and also from practically all standard distributions.

3. MathMore

This package incorporate more advanced mathematical functionality to extend MathCore. The need of separating the functionality is twofold. In order to keep the size of the core of ROOT reasonable, only the most used mathematical functionality is included in it. Secondly, there are licensing issues concerning some of the more advanced functionality which uses the GNU Scientific Library (GSL) [10]. One of the design goals is to hide the implementation and presently the mathematical functionality from GSL is used underneath. It would be very easy to shift to use another numerical package and being completely transparent to the user and straightforward for the developer. As for now MathMore is composed of the following parts:

- special functions like Bessel functions of various types and fractional order, elliptic integrals, Laguerre and Legendre polynomials, hypergeometric functions
- cumulative distribution functions and their inverse for chi-squared, gamma, f and t-distributions and their inverses. There are also the inverses of the CDF's of the Breit-Wigner, exponential, Gaussian, lognormal and uniform distributions.
- classes for numerical algorithms like derivation, various types of adaptive and non-adaptive numerical integration, interpolation and root finding algorithms for one dimensional functions

It is foreseen to extend MathMore with C++ binding to GSL numerical algorithms for multidimensional functions such as Monte Carlo integration, root finders and minimization.

Fast Fourier Transforms are provided via an interface to the FFTW [11] package, that has been recently introduced in ROOT.

4. Linear Algebra

ROOT contains a general matrix package for describing matrices and vectors and their linear algebra operations in arbitrary dimensions and of various types. Classes exist to model general matrices, symmetric and sparse matrices. Recently a template parameter has been introduced in the `TMatrix` classes for describing the underlying type.

4.1 SMatrix

Following requests from the LHC experiments a new package, SMatrix, has been introduced. SMatrix is a C++ package for high performance vector and matrix computations. It can be used only in problems when the size of the matrices is known at compile time, like in the tracking reconstruction of HEP experiments. It is based on a C++ technique, called expression templates, to achieve an high level optimization. The C++ templates can be used to implement vector and matrix expressions such that these expressions can be transformed at compile time to code which is equivalent to hand optimized code in a low-level language like Fortran or C (see for example [12])

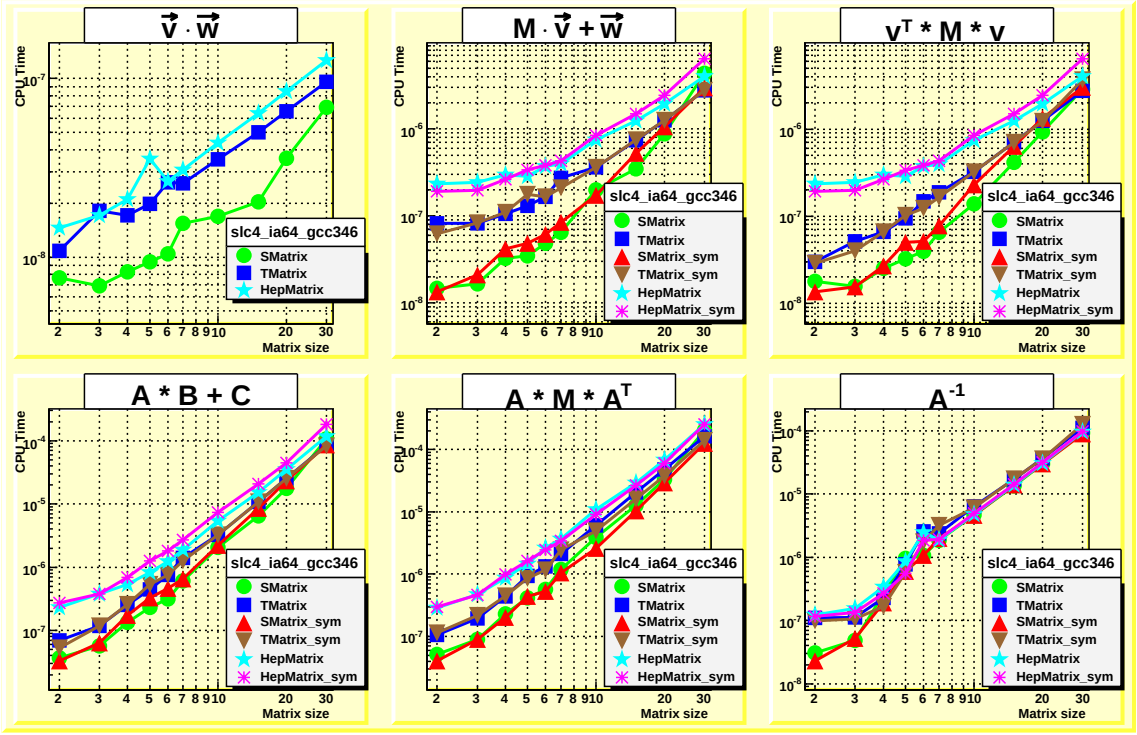

Figure 3: Comparison in matrix operations between SMatrix, TMatrix from ROOT and HepMatrix from CLHEP, for the general squared and symmetric matrices of various dimensions.

The SMatrix has been deeveloped initially as part of the HeraB analysis framework [13]. A subset of the original package has been now incorporated in ROOT, with the aim to provide to the LHC experiments a stand-alone and high performant matrix package for reconstruction. The package has now substantially evolved and the API differs from the original one.

SMatrix contains the generic `SMatrix` and `SVector` classes to describe matrix and vector of arbitrary dimensions and of arbitrary type. The 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. It has a default instantiation, valid for a general matrix, and based on a class containing a C array of size $N \times M$, where N is the number of rows and M is the number of columns. The storage for a symmetric $N \times N$ matrix is instead based on an array of reduced size $N * (N+1)/2$, the independent parameters of a symmetric matrix. A static structure provides in addition the values of the offsets, enabling to translate indices from the matrix positions to the storage positions. This structure avoids therefore to calculate these offsets every time a matrix element is requested.

This package it is not intended to be a replacement of the `TMatrix` classes and it does not provide complete linear algebra functionality. What is provided are operations such as the matrix-matrix, matrix-vector, and vector-vector operations, plus some extra functionality for square matrices, like inversion and determinant calculation. An optimized inversion is provided for matrices of size up to 5x5.

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 large matrices ($N, M > 10$). Figure 3 shows the performances of SMatrix, comparing with TMatrix and CLHEP.

5. Minimization and Fitting

ROOT contains 2 general purpose minimization packages: Minuit[1] and Fumili[2] and a smaller class `TLinearFitter`, specific for fitting functions linear in parameters. In the linear case fitting requires only one pass over the data and the user doesn't have to set initial parameter values any more. The computation time decreased substantially, which makes it very convenient for large datasets. 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 [14], has been added.

The RooFit package[15], is now distributed within ROOT. This toolkit contains a collection of "standard" probability distribution functions and allows to easily construct new complicated models. It provides also functionality for normalization and automatic pre-fit normalization of PDFs.

A new version of Minuit, developed inside the SEAL project [16], it is now integrated inside ROOT as a new package, called Minuit2.

A new GUI for fitting has been introduced in order to drive the fitting process, by selecting the fitting function, initial parameter values, fitting and minimization options. The lay-out of the new main fitting panel is shown in figure 4. Additional panels exist for setting the initial values of the function parameters or to set the minimizer type and options.

In the future it is planned to improve the existing ROOT fitting classes, by extending the functionality of the `TVirtualFitter` class, by providing support for parallel fits, various fitting and minimization methods and easier integration with RooFit.

5.1 Minuit2

The algorithm of the original fortran version of Minuit have been re-designed and re-implemented in the C++ language, under the directed supervised by the original author [1]. Minuit2 provides and enhances all the functionality of the original Fortran version. The profits from basing on an object oriented design are an increased flexibility, easy maintainability in the long term and opening to extensions such as integration of new algorithms, new functionality, changes in user interfaces. For example, the Fumili algorithm has been integrated directly inside the minimization framework provided by Minuit2.

Various extensive tests have been performed to study and validate the numerical quality, convergence power and computational performances of this new version. Some of these tests are described in details in this document [17].

Minuit2 has been now integrated inside ROOT as an additional implementation of the ROOT `TVirtualFitter` class. In the future it is expected to improve the functionality by adding the possibility of supplying constraints on the parameters.

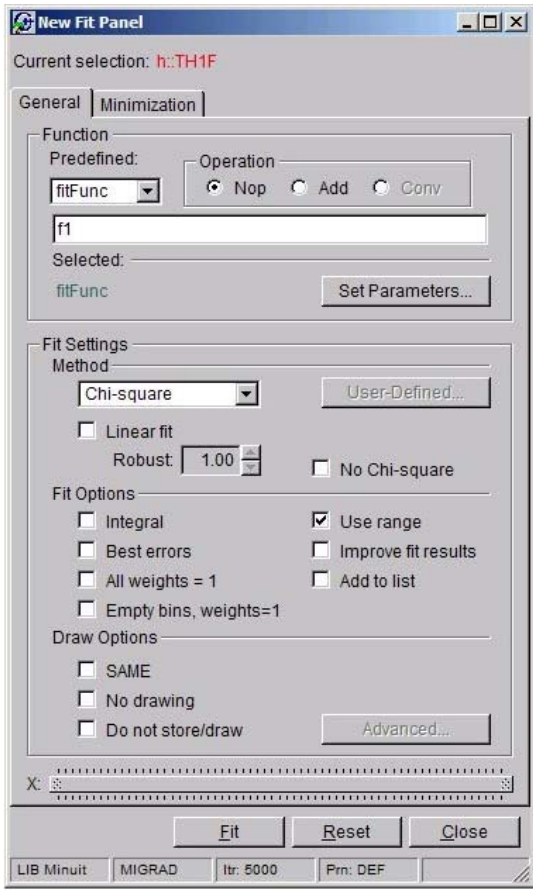

Figure 4: The new panel used for fitting ROOT data objects like the histogram classes.

6. Conclusions

Various new developments in the ROOT Mathematical libraries have been performed since the last ACAT conference. A large fraction of the work is driven by the needs of the LHC experiments for the reconstruction and analysis of their data. In the future it is expected to consolidate the new developed libraries taking into account the needs and feedback received from the users. Already some of the LHC experiments, like LHCb and CMS, are using these new libraries in their software. Furthermore, it is planned to improve the existing ROOT analysis classes, such as the histograms and function classes to use the mathematical functionality provided by these new libraries.

References

[1] F. James, *MINUIT Reference Manual*, CERN Program Library Writeup D506.

[2] S. Yashchenko, *New method for minimizing regular functions with constraints on parameter region*, Proceedings of CHEP'97 (1997).

[3] W. Brown and M. Paterno, *A proposal to Add Mathematical Special Functions to the C++ Standard Library*, WG21/N1422 = J16/03-0004.

[4] The Numerical Algorithm Group (Nag) C Library, see also http://www.nag.co.uk/numeric/cl/CLdescription.asp

[5] M. Matsumoto and T. Nishimura, *Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generato*, ACM Trans. on Modeling and Computer Simulations, 8, 1, (1998), 3-20

[6] F. James, *RANLUX: A Fortran implementation of the high quality pseudo-random number generator of Lüscher*, Computer Physics Communication, 79 (1994) 111.

[7] P. L'Ecuyer, *Maximally Equidistributed Combined Tausworthe Generators*, Mathematics of Computation, 65, 213 (1996), 203-213

[8] W. Hoermann and G. Derflinger, *The ACR Method for generating normal random variables*, OR Spektrum 12 (1990), 181-185.

[9] http://statistik.wu-wien.ac.at/unuran.

[10] M. Galassi et al, *The GNU Scientific Library Reference Manual* - Second Edition, ISBN = 0954161734 (paperback). See also http://www.gnu.org/software/gsl

[11] http://www.fftw.org.

[12] T. Veldhuizen, *Expression Templates*, C++ Report, Vol. 7 No. 5 June 1995, pp. 26-31.

[13] T. Glebe, *SMatrix - A high performance library for Vector/Matrix calculation and Vertexing*, HERA-B Software Note 01-134, December 2, 2003.

[14] P.J. Rousseeuw and K.Van Driessen, *Computing LTS Regressio n for Large Datasets*, *Estadistica* **54**, 163 (2002).

[15] http://roofit.sourceforge.net.

[16] M. Hatlo *et al.*, *IEEE Transactions on Nuc lear Science* **52-6**, 2818 (2005)

[17] A. McLennan,*Function Minimization*, CERN-LCGAPP-2005-07