# NEW DEVELOPMENTS OF ROOT MATHEMATICAL LIBRARIES

L. Moneta*, A. Kreshuk, A. Zsenei, CERN, Geneva, Switzerland
W. Brown, M. Fischler, J. Marraffino, FNAL, Batavia, IL 60510, USA

*Abstract*

The new developments of the ROOT Math work package, formed from the merge of the ROOT and SEAL activities, are presented. An overview of the ROOT Mathematical libraries is given, describing in detail the functionality and design of the packages recently introduced.

## INTRODUCTION

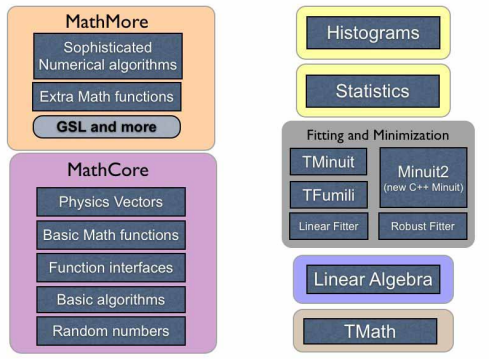


Figure 1: New ROOT Math Components

The ROOT project has been restructured following the merge with the LCG SEAL project [1]. A MATH work package has been formed with the aim to provide and to support a coherent set of mathematical and statical libraries. Existing Mathematical library provided by ROOT and by the SEAL project [2] are merged in new 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 [3] and Fumili [4], 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 library**: package grouping the various statistical algorithms of ROOT like neural network for multivariate analysis or classes for computing confidence levels. The algorithms are presently spread out in various ROOT libraries, but we expect in the future to group together in a single package.

In the following sections a detailed description is given for those components, which have been recently developed and are now integrated inside ROOT.

## 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 future, it is planned to include in MathCore a random number generator package, which combined with the statistical functions, will provide functionality to generate random numbers according to common used statistical distributions.

* Lorenzo.Moneta@cern.ch

### *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 [5]). 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 [6]. 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`.

### *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 3 and 4 dimensions. The 3D space is used to describe the geometry vectors and points, while the 4D space-time is used for physics vectors representing relativistic particles.

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) in 3, or 4 dimensions. 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.

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 $\Phi$ and pseudo-rapidity $\eta$ 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.

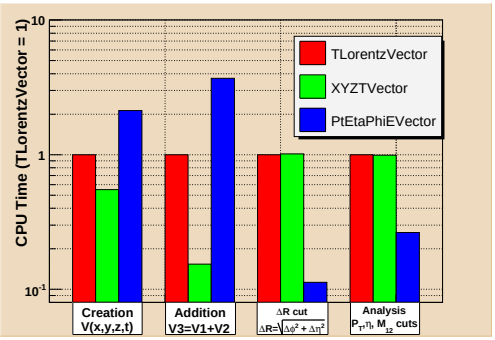


Figure 2: CPU time obtained using `TLorentzVector` and the new LorentzVector's based on Cartesian (`XYZVector`) and on Cylindrical-Eta (`PtEtaPhiVector` coordinates).

## 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) [7]. 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 numerical algorithms for multidimensional functions, Fast Fourier transforms and for linear algebra algorithms.

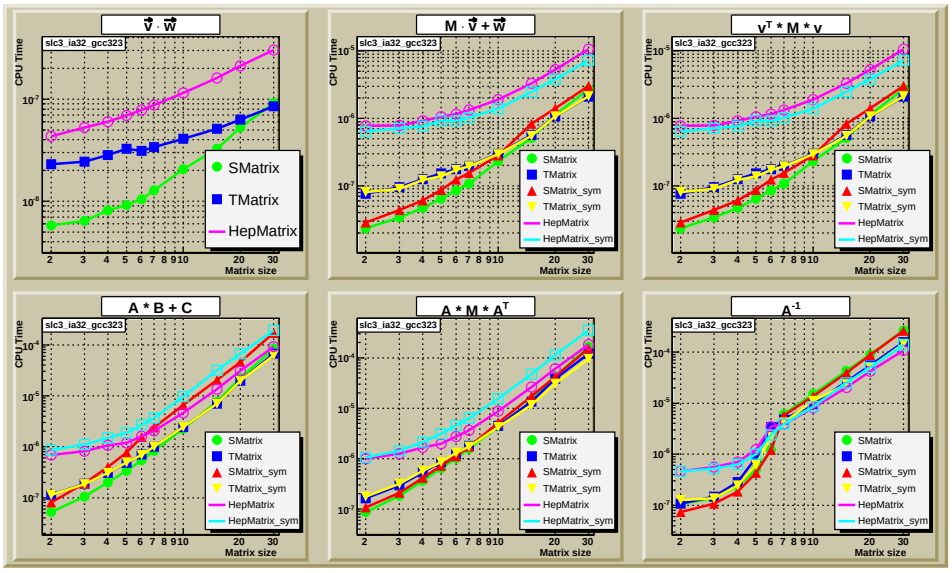


Figure 3: Comparison in matrix operations between SMatrix, TMatrix from ROOT and HepMatrix from CLHEP, for the general squared and symmetric matrices of various dimensions.

## 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.

### *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 [8]) The SMatrix has been deeveloped initially as part of the HeraB analysis framework [9]. 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 6x6.

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.

## MINIMIZATION AND FITTING

ROOT contains 2 general purpose minimization packages: Minuit[3] and Fumili[4] 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 for removing bad observations, outliers, has been added.

The RooFit package[12], 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, has been developed inside the SEAL project [2] and it is now integrated inside ROOT as a new package, called Minuit2.

### *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 [3]. 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 [10].

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.

### *Robust Fitting*

The classical least squares fitting procedures are known to be very sensitive to bad observations. Even one very bad outlier can make it produce results arbitrarily far from the true parameter values. To fit such "contaminated" datasets, an extension for robust fitting was added to the ROOT linear fitter. It is based on the approximate Fast Least Trimmed Squares (LTS) regression algorithm for large data sets[11]. The algorithm tries to find a subset of *h* points (out of *n*) that have the smallest sum of squared residuals. The parameter *h*, number of good points in the dataset, should lie between *n/2* and *n*, default value is around *n/2*. The algorithm is highly robust, with breakdown point *(n-h)/n*. An example is shown in figure 4.

## CONCLUSIONS AND OUTLOOK

Various new developments in the ROOT Mathematical libraries have been performed since the last CHEP 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, MathCore, MathMore and SMatrix, taking into account the needs and feedback received from the users. Already some of the experiments (LHCb and CMS) are starting to integrate these new libraries in their software. Furthermore, it is planned to improve the existing ROOT analysis classes, such as the Histograms and Functions to use the mathematical functionality provided by the new libraries.

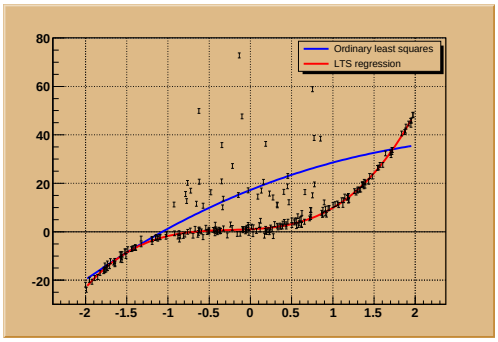


Figure 4: LTS fit compared to ordinary least squares

## REFERENCES


[1] R. Chytracek *at al.*, *Nuclear Instruments And Methods* **A534**, 115 (2004). See also http://www.cern.ch/seal.

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

[3] F. James, "MINUIT Reference Manual", CERN Program Library Writeup D506.

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

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

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

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

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

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

[10] A. McLennan,"Function Minimization", CERN-LCGAPP-2005-07

[11] P.J. Rousseeuw and K.Van Driessen, "Computing LTS Regression for Large Datasets", *Estadistica* **54**, 163 (2002).

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